UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MODÉLISATION DE L'APPRENANT : APPLICATION D'UN MODÈLE COGNITIF AU DÉVELOPPEMENT D'UN SYSTÈME D'APPRENTISSAGE

THÈSE
PRESENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR
ABDERRAHIM DANINE

JUILLET 2010

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de cette thèse se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.01-2006). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

Cette thèse n'aurait pu aboutir sans le soutien et la contribution directe ou indirecte de plusieurs personnes. Je voudrais, en quelques lignes, leur témoigner ma reconnaissance.

Mes remerciements vont tout d'abord à mon directeur de thèse, Monsieur Bernard Lefebvre, professeur au Département d'informatique à l'UQAM, et mon codirecteur de thèse, Monsieur André Mayers, professeur au département d'informatique à l'Université de Sherbrooke. Je leur suis particulièrement reconnaissant pour leur patience, leur soutien tout au long de ce travail de thèse, leurs différentes remarques enrichissantes et leurs nombreux conseils judicieux.

Je tiens également à remercier les membres du jury : Monsieur Daniel Memmi, professeur au Département d'informatique à l'UQAM, Madame Jacqueline Bourdeau, professeure à la Télé-Université, et Monsieur Michel C. Desmarais, professeur au Département de génie informatique à l'École Polytechnique de Montréal, de m'avoir fait l'honneur d'accepter de rapporter cette thèse, ainsi que de l'intérêt qu'ils ont manifesté à l'égard de mon travail et pour leurs remarques intéressantes et leurs questions pertinentes.

Je tiens à remercier également Monsieur Roger Nkambou directeur du programme de doctorat en informatique cognitive à l'UQAM, Monsieur Mounir Boukadoum et Monsieur Ghislain Lévesque (ex-directeurs), pour leur coopération et leurs précieux conseils.

Je tiens également à exprimer mes plus sincères remerciements à Monsieur Jean-Guy Meunier, professeur au département de philosophie à l'UQAM, pour son soutien et ses conseils. Il m'a soutenu et encouragé en de nombreuses occasions.

À mes amis, Abdallah Kourri, Karim Bouzouba et Adil Kabbaj du groupe Amine Platforme, vont toutes ma considération et ma reconnaissance pour leur aide et leur collaboration. Mes amis de travail, depuis 25 ans d'enseignement.

Enfin, toute ma gratitude va à tous mes proches, dont cette thèse porte l'empreinte. Mes remerciements vont tout particulièrement à la mémoire de ma mère, décédée trop tôt, de qui je tiens l'amour de la sagesse et la passion pour la science. Sans leur amour et leurs sacrifices, je n'aurais pas pu faire ce parcours et à mes frères et mes sœurs qui m'ont toujours soutenu et encouragé.

TABLE DES MATIÈRES

LISTE DES FIGURES

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 6

LISTE DES TABLEAUX

LISTE DES TABLES

Chapitre 6

LISTE DES ANNEXES

RÉSUMÉ

Bien que le diagnostic des erreurs des apprenants soit central à toute stratégie d'intervention correctrice relevant au mode d'évaluation dans un système d'apprentissage, trop souvent, la prise d'information qui l'accompagne est incomplète ou incertaine. Ajoutons aussi le problème de la modélisation dans un contexte d'apprentissage où on ne peut observer directement ce qui se passe dans la tête d'un apprenant, ni de savoir avec certitude son plan de raisonnement, ni le but qu'il cherche à accomplir. Il s'ensuit une réduction de l'efficacité des interventions pédagogiques qui limite les apprentissages scolaires.

Cette thèse apporte des solutions à cette problématique. Elle consiste en la conception et le développement d'un Système Tutoriel Intelligent pour le Diagnostic des Erreurs en Soustraction (TIDES). Elle s'inscrit dans une perspective d'évaluation diagnostique des compétences et connaissances arithmétiques en utilisant une approche originale qui vise à modéliser l'apprenant dans une situation d'apprentissage où les informations sur cet apprenant sont potentiellement incomplètes ou incertaines.

Dans cette thèse, nous présentons la conception, le développement et une mise à l'essai du système TIDES. Le design de ce système est basé sur un modèle cognitif, la théorie d'apprentissage ACT-R d'Anderson, capable d'analyser le comportement d'un apprenant et de savoir son état cognitif. Le choix de ce design est discuté et justifié aussi.

L'architecture du système TIDES comporte au moins trois modules : un module qui permet de spécifier des tâches à l'apprenant, un module d'analyse qui permet d'analyser les actions de l'apprenant et un module de diagnostic qui permet

d'inférer les informations sur l'apprenant, d'évaluer ses compétences impliquées dans une tâche d'apprentissage, de détecter sa stratégie mise en œuvre, en s'appuyant sur une méthode de reconnaissance de plan, de prédire sa prochaine action la plus probable et de savoir avec exactitude les causes réelles de ses erreurs. Les caractéristiques du système TIDES sont décrites en détail dans la thèse.

La méthodologie d'une mise à l'essai du système avec une vingtaine d'élèves est présentée et les données recueillies dans cette mise à l'essai sont regroupées et analysées.

L'ensemble des résultats obtenus indique que le système TIDES offre le potentiel d'analyser et de diagnostiquer les erreurs des apprenants de façon plus précise, et donne effectivement lieu à un apprentissage conforme à celui qui était prévu en se basant sur la méthode originale adoptée.

Enfin, nous proposerons des améliorations possibles (extension du système TIDES à l'aide des réseaux bayésiens) que nous présenterons comme explorées mais non encore complètement intégrées dans l'état actuel du système TIDES et aussi non évaluées. Il s'agit en fait de déterminer à quelles conditions le modèle bayésien peut être intégré à un système d'apprentissage, en tant que système tutoriel intelligent et dont le domaine d'apprentissage est l'arithmétique.

Mots-clés : Intelligence artificielle, environnement interactif pour l'apprentissage humain, système tutoriel intelligent, théories d'apprentissage, Modèle d'Anderson ACT-R, modélisation d'un apprenant, analyse des erreurs, diagnostic des erreurs, modélisation statistique et réseaux bayésiens.

INTRODUCTION GÉNÉRALE

1 Position de problème

L'histoire de l'IA en éducation, montre que les efforts des chercheurs dans ce domaine ont abouti à la production d'un ensemble de systèmes à caractère pédagogique très élaborés (Ohlsson, 1992 ; Bruillard, 1997). Dans un premier temps, on voit apparaître un bon nombre de systèmes qui se sont contentés de traduire sous une forme informatique les supports traditionnels de l'enseignement : cours, recueils d'exercices, ...etc. Ensuite, les recherches se sont accentuées pour pousser plus loin la capacité de ces systèmes pédagogiques. D'où l'apparition, depuis les années 50, des systèmes fondés explicitement sur la notion d'enseignement et sont nés de la recherche en Enseignement Assisté par Ordinateur (EAO, ou CAI pour Computer Assisted Instruction) (Nkambou, 1996).

Dans les systèmes de type EAO, un ensemble de relations stimuli-réponses sont anticipées par le concepteur du programme et permettent au système de réagir à partir d'informations qui sont prévues de façon exhaustive dans la base de données. Les systèmes servaient alors essentiellement à l'évaluation ou à l'acquisition de connaissance basée sur un enseignement behavioriste.

Malgré que les premiers systèmes d'EAO ont fait les premiers pas des technologies éducatives actuelles, ces systèmes ont connu certaines faiblesses sérieuses. D'une part, ils laissent peu de possibilités à l'apprenant d'orienter le déroulement d'une session d'apprentissage, l'apprenant a un rôle passif quant au contrôle, et d'autre part, leurs interventions tutorielles sont peu adaptées aux

besoins et aux intérêts de l'apprenant et ne tiennent pas compte du contexte global de la situation.

Si les premiers systèmes d'EAO se sont révélés extrêmement primitifs, se contentant d'attribuer une valeur « vrai » ou « faux » à la réponse fournie, dès les années 70, et grâce à l'évolution des travaux sur les applications des techniques de l'intelligence artificielle à l'éducation, de nouveaux modèles se sont développés qui montrent une tendance à afficher un comportement plus proche de celui d'un tuteur humain (Carbonell, 1970). Ces modèles, développés sous le thème Enseignement Intelligent Assisté par Ordinateur (EIAO), également appelé Environnement Interactif d'Apprentissage avec Ordinateur, sont plus souples, plus interactifs, s'adaptant mieux à leurs utilisateurs, ce sont les Systèmes Tutoriels Intelligents (STI) (O'Shea & Self, 1983). Ces systèmes sont caractérisés par une représentation explicite des connaissances du domaine et des mécanismes de raisonnement, ainsi qu'une explicitation de stratégies tutorielles pour permettre aux systèmes de générer dynamiquement leurs interventions en fonction du modèle de l'apprenant et des objectifs pédagogiques spécifiques.

Ces systèmes ont dépassé la phase de la production des réponses toutes faites aux interrogations de l'apprenant, mais ils peuvent faire face à des questions non prévues, comme ils peuvent utiliser et traiter des connaissances accumulées au cours d'une session d'apprentissage. En outre, ces systèmes peuvent aussi conduire des dialogues en langue naturelle.

La conception des systèmes tutoriels reposait au début sur des méthodes d'enseignement sans préoccupation de validation expérimentale, par conséquent, les effets de ces systèmes sur le processus d'apprentissage ne sont pas connus. Par la suite, l'environnement de l'apprentissage domine les recherches sur les systèmes tutoriels (Dillenbourg, 1993 ; Paquette, 1999), ainsi, leur conception, reposant sur des théories de l'apprentissage, comme les tuteurs d'Anderson (Anderson et al., 1985), est recentrée sur des problèmes plus directement liés à l'apprentissage qu'à l'enseignement.

D'un point de vue historique, on peut constater que les recherches sur les systèmes tutoriels intelligents se sont orientées vers deux modèles : modèle de l'enseignement et modèle de l'apprentissage (Dillenbourg, 1993 ; Paquette, 1999).

1.1 Modèle de l'enseignement et ses problèmes

La majorité des systèmes classiques d'EIAO concernent le modèle de l'enseignement. Ils sont conçus, d'une part, pour enseigner les connaissances d'un domaine à l'apprenant (comme par exemple les systèmes SCHOLAR et SOPHIE (Carbonell, 1970 ; Brown, Burton et Bell, 1975)) et d'autre part d'être capables de résoudre des problèmes que l'apprenant doit résoudre. Ces systèmes sont aussi compétents dans la matière qu'ils enseignent. Ils peuvent accompagner l'apprenant en lui présentant des cours, des rappels, des problèmes ou des conseils. Les échanges avec l'apprenant se limitaient généralement à la saisie des données et à la présentations des résultats.

Une des principales caractéristiques de ce modèle est que les systèmes peuvent expliquer leurs démarches aux apprenants en générant un texte à partir de la liste des règles utilisées (Mendelsohn & Dillenbourg, 1991).

Cette rapide présentation de ce modèle montre le potentiel de recherche offert par l'IA au domaine de l'éducation, mais elle fait apparaître aussi les limitations inhérentes aux systèmes liés à ce modèle. En effet, les performances de ces systèmes sont insatisfaisantes et anticipent mal le niveau de connaissance de l'apprenant. De plus, les stratégies tutorielles pour la gestion du comportement de l'apprenant ne sont pas clairement définies. Ajoutons aussi que l'interactivité avec l'apprenant est souvent dominée par le système (Mendelsohn & Dillenbourg, 1991).

1.2 Modèle de l'apprentissage et ses problèmes

Le second modèle regroupe les systèmes qui sont conçus pour évaluer le processus d'apprentissage des apprenants. Les principales fonctions de ces systèmes sont : la surveillance de l'apprenant au cours de la résolution de problèmes et l'intervention au moment convenable en délivrant des messages

appropriés (Dillenbourg, 1993 ; Paquette, 1999). Ces systèmes prennent en compte la richesse et la complexité des processus cognitifs impliqués dans l'enseignement. Ils se basent sur une vision constructiviste de l'apprentissage, selon laquelle l'apprenant construit ses connaissances en interagissant avec le système. Ainsi, ils aident l'apprenant à comprendre pourquoi le résultat qu'il a fourni n'est pas le résultat escompté (situation de diagnostic).

Il convient cependant de préciser certaines limites de ces systèmes. Comme première limite, on trouve le mode d'évaluation des apprentissages utilisé par ces systèmes. Ce mode est souvent lié aux produits, qui sont les erreurs de l'apprenant, les systèmes opèrent donc à partir d'un produit et non pas à partir d'un processus. Ensuite, l'apprenant ne peut pas apprendre à surmonter ses difficultés par lui-même. Et enfin, on constate que la recherche des sources des erreurs et de leur mode de génération en liaison avec les connaissances des sujets, est encore très embryonnaire dans ces systèmes.

2 Cadre du travail

Notre travail se situe dans le cadre des recherches sur les environnements interactifs d'apprentissage avec ordinateur (EIAO) et s'intéresse plus particulièrement au diagnostic des erreurs de l'apprenant. Comme nous l'avons déjà indiqué en haut, le but général du paradigme EIAO est de concevoir des systèmes qui favorisent l'apprentissage chez leur utilisateur. Pour favoriser cet apprentissage, il est apparu indispensable de disposer d'informations sur les aptitudes, les connaissances et les lacunes propres à chaque apprenant, ce qui est l'objet des recherches qui s'intéressent à la modélisation de l'apprenant. Cette modélisation, par un système tutoriel intelligent, est le processus qui analyse non seulement des informations sur l'apprenant, mais aussi ses produits pour estimer son niveau de connaissance du domaine et adapter les contenus et les interactions en fonction de cette estimation.

3 Approche adoptée

Comme nous avons le souci de construire un modèle de l'apprenant pour obtenir une interaction adaptative entre l'environnement d'apprentissage et l'apprenant, nous proposons une nouvelle approche pour réaliser cet objectif. En effet, notre approche se base sur une méthodologie du Model Tracing d'Anderson (ou traçage de modèle d'Anderson), a un aspect cognitif, elle nous aide à savoir ce qui se passe dans la tête de l'apprenant et à extraire des informations précises et utiles pour le diagnostic de ses erreurs en se fondant sur l'observation de ses actions. Cette observation est interprétée comme l'arrivée de nouvelles informations. De plus, ce modèle peut déterminer, non seulement, le but recherché par l'apprenant mais aussi, il peut détecter les obstacles qui peuvent entraver la réalisation de ce but.

4 Nécessité d'une théorie cognitive

Pour pallier aux faiblesses des systèmes cités plus haut, le besoin de s'appuyer sur des modèles théoriques devient une préoccupation commune à la plupart des chercheurs dans le domaine de l'EIAO. Rappelons que l'enseignement assisté par ordinateur est né dans les années 60 comme extension de l'enseignement programmé hérité des théories de Skinner (1954). Ces théories ont été largement contestées depuis et remplacées par d'autres théories comme, par exemple, la théorie constructiviste ou, plus récemment, différentes théories d'inspiration cognitiviste (voir chapitre2). La référence à une théorie, qu'elle soit implicite ou explicite, est devenue un souci partagé par plusieurs chercheurs.

Cependant, différentes opinions s'opposent en ce qui concerne les référents théoriques sur lesquels il faut s'appuyer pour établir un apprentissage à l'aide d'un système tutoriel intelligent. Certains considèrent qu'il est possible de modéliser l'acquisition des compétences indépendamment de la richesse du domaine, ce qu'on appelle des modèles globaux (Depover et al., 2000). Grâce à ces modèles, on a créé des environnements d'apprentissage aussi variés que des tuteurs intelligents, des hypertextes.

Au contraire, d'autres avancent que les modèles sont spécifiques à une discipline particulière, ce qu'on appelle des modèles locaux. Ces modèles sont beaucoup plus précis que les précédents, mais ne touchent que peu de domaines et dans des tâches très limitées.

Dans notre travail de recherche, nous avons adopté le modèle global ACT*[1] d'Anderson (1996), puisqu'il présente, à notre avis, plusieurs avantages. Tout d'abord, ce modèle a joué un rôle très important dans la compréhension des mécanismes d'apprentissage chez un apprenant. De plus, il a été à la base de la conception et du développement de systèmes d'apprentissage d'Anderson. Ce modèle est exploité par certains pédagogues pour comprendre et analyser les difficultés qui peuvent apparaître en cours d'apprentissage. Ensuite, un des avantages de ce modèle qu'il définit très clairement la construction des activités complexes sur la base d'une interaction entre des connaissances procédurales et déclaratives. Enfin, Anderson a validé son modèle dans une situation réelle (Anderson, 1996). Cette validation lui permet de comprendre les conceptions erronées de l'apprenant, d'analyser ses erreurs de façon plus fine et de savoir leurs origines (faire le diagnostic).

5 Domaine d'application

Malgré que notre méthode de diagnostic se veut générique, quelques choix particuliers s'imposent pour valider cette approche. Nous présentons dans la suite les justifications du domaine d'application que nous avons choisi.

- **Arithmétique cognitive**[2]

Parmi les quatre opérations arithmétiques élémentaires, deux ont donné lieu à de très nombreuses recherches : l'addition et la soustraction. Cela tient au fait que l'étude de la résolution des problèmes liés à ces deux opérations permet d'évoquer une question théorique fondamentale, celle des procédures utilisées par les élèves

---

[1] Nous nous référerons dans notre travail à sa version la plus récente, l'ACT-R (Anderson, 1993, 1996).

[2] L'arithmétique cognitive est un modèle de résolution des problèmes arithmétiques par modèles mentaux. (Lebiere &Anderson, 1998 ; Geary, 1999 ; Ashcraft, 2001)

en fonction de leur niveau d'apprentissage et du système numérique qu'ils utilisent (Fayol et al., 2000).

En outre, les recherches sur les types d'erreurs en arithmétique ainsi que leur classification ont bénéficié de plus d'attention de la part des nombreux chercheurs que celles effectuées dans d'autres domaines de mathématiques (Rojas & al., 2002 ; Beal al., 2000). La recherche bibliographique que nous avons effectuée, nous a donné un nombre considérable de documents traitant des erreurs arithmétiques, notamment en soustraction, parmi les plus récentes publications (Bundy, 2001, 2002 ; Dimitrova al., 2000 ; Hirashima & al., 2000 ; Nakano & al., 2000 ; SiteWeb, 2002 ; Arroyo, 2000 ; Arroyo & al., 2000, Anandeep, 1995).

Plusieurs raisons expliquent cet intérêt pour l'arithmétique, parmi lesquelles, nous pouvons retenir les suivantes :

- le domaine de l'arithmétique se prête un peu plus difficilement à l'étude des principes généraux de l'apprentissage et de l'enseignement et fournit une fenêtre sur les transformations cognitives chez l'enfant (De Corte & al., 1991) ;
- parmi les principaux objectifs de l'enseignement primaire demeure de nos jours celui de former des individus capables d'effectuer correctement les opérations arithmétiques de base. D'ailleurs, aux États-Unis d'Amérique, beaucoup d'instituteurs ou institutrices se comportent comme s'ils croyaient avoir pour seule mission, en enseignant les mathématiques, de développer l'aptitude au calcul arithmétique (Geary & al., 1997, 1999);
- l'addition et la soustraction occupent une place primordiale dans les curriculums mathématiques. Plusieurs chercheurs ont confirmé que le champ conceptuel des structures additives, pour lequel l'addition et la soustraction des nombres naturels sont les exemples les plus élémentaires, est à la base d'une large portion de mathématiques et se développe sur une longue période de temps (Vergnaud, 1982 ; Bundy, 2001, 2002 ; Hirashima & al., 2000) ;
- l'investigation des erreurs des élèves en arithmétique y est relativement compliquée par rapport à d'autres domaines (Chiu & al., 1997).

D'ailleurs, on sait depuis longtemps que la résolution des problèmes de la soustraction écrite soulève de nombreuses difficultés, et pour les apprenants et pour les enseignants (Cox, 1974, 1975 ; Engelhandt, 1979, 1982 ; Resnik, 1982, Ellerton, 1986). En revanche, l'intérêt porté à ce genre de difficultés se révèle beaucoup plus récent au chapitre des domaines techniques et constitue même une des « retombées » des recherches en intelligence artificielle. D'autre part, le secteur dans lequel les obstacles et les difficultés d'apprentissage sont les plus ardus et les plus difficiles à éliminer est celui relatif à la compréhension du concept de la soustraction (Foss, 1987a ; Brown & VanLehn, 1980, 1982; VanLehn, 1990, Hennessy, 1990).

6 Contexte et nature de la recherche

6.1 Objectif de recherche

En regard à ce que nous venons de voir dans la première section de cette introduction, notamment les problèmes des systèmes tutoriels existants (§1.1 et §1.2), notre objectif est de concevoir, d'une part, une méthode générique originale pour la modélisation des apprenants quel que soit le domaine utilisé et d'autre part, d'implémenter cette méthode sur un véritable système tutoriel intelligent, efficace, utilisable, utile pour l'enseignement et qui a pour but de diagnostiquer de façon plus précise les erreurs des apprenants et d'en trouver des causes. Nous voulons offrir un outil d'évaluation qui pourrait faciliter la tâche de l'enseignant qui voudrait l'utiliser et lui permettrait de mieux appuyer ses apprenants dans leur démarche d'apprentissage. Deux questions sont associées à la conception et à l'élaboration de notre système (ces questions seront traitées au chapitre 3):

- **à quelles conditions le modèle ACT* peut-il s'intégrer à un système d'apprentissage en tant que système tutoriel intelligent ?**
- **à quelles conditions ce modèle peut-il s'intégrer à un système d'apprentissage dont le domaine d'apprentissage est l'arithmétique ?**

D'autres sous-questions de recherche sont associées à la mise à l'essai du système d'apprentissage de l'arithmétique ainsi conçu (ces sous-questions seront traitées au chapitre 5) :

- L'utilisation du système d'apprentissage de l'arithmétique par un élève donne-t-elle effectivement lieu à un apprentissage ?
- Cet apprentissage est-il conforme à celui qui était prévu en se basant sur le modèle d'Anderson?
- Le système d'apprentissage de l'arithmétique assure-t-il une grande efficacité de diagnostic à l'aide du « traçage de modèle » d'Anderson ?
- L'utilisation d'une rétroaction à deux niveaux (analyse, diagnostic) permet-elle de concilier les avantages d'une rétroaction immédiate et ceux d'une plus grande latitude laissée à l'élève ?

6.2 Type de recherche

Cette recherche est de type développement. Ce type de recherche porte sur la manière dont le produit élaboré apporte une solution (ou des solutions) aux problèmes définis au départ.

Selon Contandriopoulos et al. (1990), la recherche de type développement vise notamment à mettre au point ou à améliorer considérablement une intervention. Elle ne peut donc pas servir à soumettre à l'épreuve de faits une ou des hypothèses de recherche. Cette recherche doit plutôt permettre de définir explicitement dans quelles conditions une intervention devrait donner les résultats attendus.

6.3 Originalité du projet

Cette démarche est originale à plusieurs égards :

- L'originalité du projet provient, selon nous, dans l'utilisation d'une approche qui traduit l'ambition explicite, d'une part, de décrire plausiblement des processus cognitifs observables chez des apprenants, d'autre part de la modélisation des apprenants selon cette approche afin de réaliser un système d'apprentissage.

- La conception de notre système tutoriel s'appuie sur un modèle de connaissances capable de générer des problèmes et d'analyser et de diagnostiquer les solutions de problèmes que lui soumettent des apprenants
- La stratégie tutorielle est centrée sur le volet diagnostic plutôt que sur l'analyse[3] comme c'est le cas de plusieurs systèmes actuels. Cet accent sur le diagnostic impose une nouvelle approche de diagnostic qui se distingue des approches existantes.
- Une nouvelle stratégie d'intervention est proposée : elle combine les avantages de la rétroaction immédiate d'Anderson et ceux de la rétroaction après une réponse complète de l'apprenant.
- Un autre aspect d'originalité, c'est que notre système sera capable de générer des tests pour savoir exactement les causes des erreurs commises par un apprenant et de choisir à quel moment l'intervention est efficace.

6.4 Limite de recherche

Le fait de choisir une recherche de type de développement impose évidemment des limites quant aux conclusions à tirer de la mise à l'essai du système. En effet, si la mise à l'essai du système élaboré constitue un outil précieux pour observer le comportement d'élèves en situation d'apprentissage, elle ne peut conduire qu'à valider ou infirmer certaines hypothèses concernant différents aspects de ce système. Une telle mise à l'essai ne permet donc pas de tirer une conclusion quelconque concernant le véritable processus cognitif chez ces élèves. Autrement dit, elle ne cherche pas à valider ou invalider la théorie ACT-R.

Il va de soi que le présent travail n'ambitionne pas la résolution d'une problématique aussi vaste mais il se propose néanmoins d'y apporter des éléments de réponse originaux et pertinents. Plus précisément, l'ambition du présent travail se restreint à la modélisation d'un apprenant dans une situation d'apprentissage.

---

[3] L'analyse des erreurs précède le processus de diagnostic. L'analyse consiste en effet à identifier l'erreur, en décrire la nature, définir ses caractéristiques et décrire le contexte où elle se produit. Le diagnostic doit ensuite permettre d'en repérer la source, d'en identifier la cause de façon à préparer une intervention adéquate.

7 Plan de la thèse

Dans le premier chapitre nous présentons les grandes lignes de l'état de l'art en intelligence artificielle sur les problèmes de formalisation des connaissances spécifiques des EIAO. Nous donnons une vision globale sur les systèmes tutoriels intelligents. Nous discutons certaines notions de base, ensuite nous décrivons les différentes techniques de représentation de connaissances, en indiquant pour chacune les mécanismes de raisonnement associés. Dans ce chapitre, nous précisons certains choix que nous faisons au sujet de la conception de notre projet.

Le deuxième chapitre est consacré au cadre conceptuel dans lequel s'inscrit notre recherche. Nous commençons par la présentation d'un certain nombre de théories d'apprentissage et de la cognition. Ensuite, l'accent sera mis sur la théorie cognitive ACT* d'Anderson, théorie que nous avons adoptée dans notre travail. Nous verrons pourquoi nous nous intéressons particulièrement à cette approche.

Le troisième chapitre traite les deux questions de la recherche et présente la méthodologie de conception et l'élaboration de notre système de diagnostic. L'un des objectifs de cette recherche est d'appliquer la théorie cognitive explicite d'Anderson lors du développement de notre système. Il s'agit dans ce chapitre de déterminer à quelles conditions ce modèle (d'Anderson) peut s'intégrer à un système d'apprentissage (1) en tant que système tutoriel intelligent et (2) dont le domaine d'apprentissage est l'arithmétique.

Le quatrième chapitre traite des méthodes qui ont été utilisées pour vérifier que le système développé donne les résultats attendus, notamment en rapport avec le modèle d'Anderson.

Dans le cinquième chapitre, les sous-questions de la recherche sont traitées et les données recueillies au cours de la mise à l'essai du système développé sont présentées et analysées. Ces données comprennent des informations enregistrées dans le fichier LOG au cours des séances d'apprentissage (voir Annexe C), des données recueillies à l'aide du questionnaire sur les perceptions des participants aux séances d'apprentissage.

Le sixième chapitre est consacré à l'extension bayésienne possible du système et aux améliorations possibles que l'on qualifierait d'explorées mais non encore complètement intégrées dans l'état actuel du système.

Pour conclure, nous proposons un bilan de notre travail en mettant en évidence ses résultats, ses limites, ainsi que les perspectives que nous pouvons dès maintenant proposer. Nous terminons par une discussion sur les apports de notre travail.

CHAPITRE I

LES ENVIRONNEMENTS D'APPRENTISSAGE INTELLIGENTS

Dans le présent chapitre, nous présentons les grandes lignes de l'état de l'art en intelligence artificielle sur les problèmes de formalisation des connaissances spécifiques des EIAO ; puis nous traitons le sujet principal du chapitre, à savoir, les systèmes tutoriels intelligents. Dans ce chapitre, l'accent sera mis sur la modélisation de l'apprenant. Nous présentons quelques approches existantes pour la modélisation, ainsi que leurs avantages et leurs limites. Nous discutons les notions de connaissances, de représentation de connaissances et de raisonnement. Nous décrivons en détail les différentes techniques de représentation de connaissances, en indiquant pour chacune les mécanismes de raisonnement associés, les avantages et inconvénients, les hypothèses sous-jacentes et les applications pour lesquelles cette représentation est la plus adaptée. Au cours de cette présentation, nous explicitons aussi les différents choix que nous faisons au sujet de la conception de notre système.

1.1 Introduction

Nous allons dans cette section présenter les grandes lignes de l'état de l'art en intelligence artificielle sur les problèmes de formalisation des connaissances spécifiques des EIAO, c'est-à-dire prenant en compte l'apprenant.

Il est plus facile de comprendre cet état de l'art en adoptant une perspective historique qui montre l'évolution des problématiques. Cependant, il faut noter que cette perspective peut s'écrire de différentes façons selon l'attention portée par les

chercheurs à l'une ou l'autre des composantes des environnements d'EIAO (Mendelsohn, 1995).

Au début des années 70, le besoin de disposer d'une représentation des connaissances d'un apprenant s'est imposé pour l'élaboration des systèmes capables d'adapter leurs stratégies d'enseignement (Self, 1974). Le modèle de l'apprenant est construit dynamiquement à partir de son comportement observable. Ensuite, l'apprenant a été considéré comme un novice qui en saurait moins que l'expert. Il s'agissait de constater qu'il savait ou ignorait telle ou telle règle, tel ou tel fait. Sa connaissance était modélisée comme un sous-ensemble de la connaissance de référence, on a parlé à ce propos de modèle de type « overlay » (Goldstein et Carr, 1977) (ou overlay model selon Van Lehn, 1988). Une telle approche ne permettait pas de rendre compte de la façon dont la connaissance évolue par des processus comme l'analogie ou la généralisation (voir 2.4.5).

La période suivante est caractérisée par la prise en compte de l'erreur au cours de laquelle le problème de la modélisation de l'apprenant repose sur deux modèles : le modèle relevant de l'apprentissage symbolique, le modèle relevant de la simulation de l'apprenant (Py, 1998). Le premier modèle est illustré par le projet PIXIE de Sleemann (1982) en algèbre élémentaire dont le but est d'inférer des règles erronées (mal-rules) à partir de l'observation des activités de l'apprenant. Le second modèle est illustré par l'implémentation de la « théorie de réparation » (Repair Theory) par Brown et VanLehn (Brown et VanLehn 1980, VanLehn 1990), à la suite du projet BUGGY (Brown et Burton, 1978).

Une autre période est dominée par ce que Kayser (1997) appelle la « représentation des connaissances dans un modèle ». Il s'agit d'utiliser le potentiel des modélisations logiques pour rendre compte des connaissances de l'apprenant.

Les travaux de Self (1992) sont un bon exemple de ce courant de recherche. Il s'agit de déterminer les croyances d'un apprenant à partir de l'observation de ses actions à l'interface du système. Ces croyances attribuées à l'apprenant par le système ne cherchent pas à rendre compte d'une réalité mentale mais à produire des données pertinentes pour permettre au tuteur de décider de ses interactions. La

modélisation proposée par Ohlsson (1992) est aussi une autre illustration de cette approche. L'auteur souligne qu'il s'agit de s'appuyer sur le traitement d'un problème par un apprenant et sur sa mise en œuvre d'actions qui le transforment avec le but de le résoudre. Ces actions sont contraintes par des conditions, le modèle consiste en l'ensemble des contraintes que l'apprenant n'a pas respectées (voir 3.9.4).

Dans la section suivante, nous présentons l'architecture classique des systèmes tutoriels intelligents. Nous décrirons ensuite les disciplines impliquées en EIAO ainsi que leurs rôles respectifs dans l'élaboration des différentes composantes d'un tutoriel intelligent.

1.2 Les systèmes tutoriels intelligents

1.2.1 Introduction

Les Systèmes Tutoriels Intelligents (STI) sont censés afficher un comportement proche de celui d'un tuteur humain. Ils sont capables de gérer l'interaction avec leurs utilisateurs. De plus, leurs stratégies tutorielles permettent de déterminer quand et comment organiser les interventions dans le processus d'apprentissage afin de réorienter la démarche de l'utilisateur.

1.2.2 Architecture d'un STI

Comme indiqué précédemment, le problème délicat pour tout STI est celui de déterminer la façon appropriée d'intervenir (le faire efficacement, mais sans s'imposer au cheminement cognitif, lui-même délicat, de l'étudiant). Pour cela, un STI doit comprendre et garder à jour plusieurs modèles sophistiqués : un modèle du domaine (de l'expertise à enseigner), un modèle de l'étudiant (en évolution constante), un modèle de l'acte tutoriel (sous forme de stratégies pédagogiques précises), et l'interface utilisateur (moyen de communication). Ces composantes essentielles constituent l'architecture d'un système tutoriel intelligent (Sleeman et Brown, 1982 ; Ross, 1987 ; Yazdani, 1987 ; Wenger, 1987).

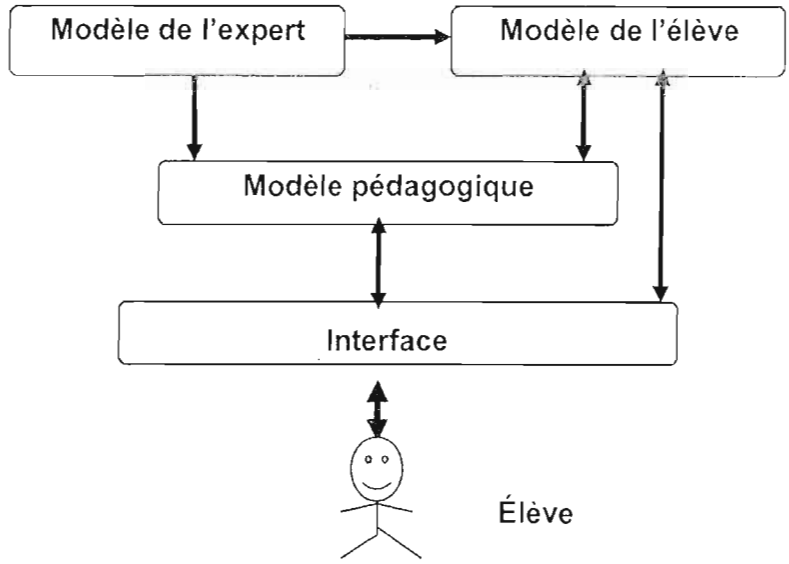


Figure 1.1 Architecture classique d'un STI

Dans la suite, nous décrivons très brièvement les différentes parties de cette architecture, puis nous présentons les interférences qui peuvent exister entre ces modules.

1.2.2.1 Modèle pédagogique

Le modèle pédagogique sert à gérer les interactions du système avec l'apprenant (Mendelsohn & Dillenbourg, 1991). Ces interactions peuvent être plus ou moins directes en proposant des problèmes à l'élève, qu'il doit résoudre, en le conduisant vers la solution, en lui apportant une aide lorsqu'il en a besoin et en approuvant ou critiquant ses performances. Pour ce faire, le modèle pédagogique s'appuie sur des stratégies d'apprentissage qui sont utilisées par un tuteur humain, telles que l'apprentissage par l'exemple, l'apprentissage par erreur, l'apprentissage par analogie.

1.2.2.2 Modèle de l'expert

Une caractéristique principale d'un système tutoriel intelligent est d'être compétent dans le domaine qu'il enseigne, c'est-à-dire de se comporter comme un expert. Cette compétence permet au tutoriel de présenter son raisonnement et de

justifier sa décision. Pour être en mesure de cela, le système doit contenir les connaissances spécifiques au domaine ainsi que les méthodes de raisonnement appropriées intervenant lors de la résolution de problèmes (Swartz, 1992).

1.2.2.3 Modèle de l'élève

Le modèle de l'élève constitue un ensemble d'informations relatives à l'état des connaissances, exactes ou erronées, de l'élève. Sa construction s'établit principalement par comparaisons avec les connaissances de l'expert. Le modèle de l'élève sert ainsi à construire un diagnostic qui pourra servir au système tutoriel pour prendre une décision de nature didactique (questionnement, explications, remédiation...) (Self, 1992). Dans les systèmes tutoriels intelligents, l'état des connaissances de l'élève est généralement représenté comme un sous-ensemble des connaissances du module expert. Dès lors, le modèle de l'élève est construit en comparant la performance de l'apprenant avec celle que l'expert aurait produite dans les mêmes circonstances (Goldstein, 1982).

1.2.2.4 Interface

Ce n'est que tardivement que les chercheurs réalisent que l'interface est une composante de tout premier ordre dans les processus d'apprentissage. Les interactions entre l'élève et le système tutoriel intelligent sont devenues dynamiques grâce à cette interface (Baker, 1997). Son rôle principal est de faciliter la communication des informations entre le système et l'élève. Elle doit offrir à l'élève des activités au travers desquelles il pourra acquérir ou consolider des concepts et des démarches.

1.2.3 Domaines de l'EIAO

La conception d'un système de type EIAO capable de s'adapter à son utilisateur relève de plusieurs disciplines de recherche telles que, la psychologie cognitive, la didactique, les sciences de l'éducation et l'intelligence artificielle (Mendelsohn, 1995). Le rôle de la psychologie cognitive dans ce domaine est de permettre la compréhension de l'état cognitif de l'apprenant au moment de son apprentissage.

Quant à la didactique, elle analyse la pertinence des savoirs enseignés et le savoir-faire de l'apprenant. En ce qui concerne les sciences de l'éducation, elles sont au cœur de la conception des outils pédagogiques. Leur rôle est de fournir les théories qui visent à étudier les méthodes et les moyens à mettre en œuvre pour favoriser l'apprentissage et l'enseignement (Gagné et al.,, 1992). L'intelligence artificielle a pour rôle de proposer des formalismes informatiques qui facilitent la représentation des connaissances ainsi que la conception des systèmes interactifs.

La coopération de ces diverses disciplines s'est en effet avérée nécessaire pour progresser dans la conception des divers modules, pour élaborer et implanter des modèles pertinents qui s'adaptent aux situations d'apprentissage plus complexes (Baron, 1994).

1.2.4 Les tâches principales de l'EIAO

La conception et l'élaboration d'un système tutoriel intelligent feront toujours appel à des tâches principales qui guident cette opération, à savoir la modélisation de l'apprenant, la modélisation des connaissances et l'élaboration du modèle de l'apprenant.

1.2.4.1 Modélisation de l'apprenant

Les recherches sur les premiers tuteurs intelligents se sont focalisées sur la modélisation de l'expertise du domaine. Puis, on a ressenti la nécessité de disposer de connaissances sur l'élève au sein des systèmes tuteurs. En effet, pour que les processus de guidage, d'évaluation et d'interaction soient efficaces et dynamiques, il est apparu indispensable que le système dispose d'informations sur les aptitudes, les connaissances et les lacunes propres à chaque apprenant, c'est ce qu'on appelle « modélisation de l'apprenant ».

La modélisation, par un système tutoriel intelligent, est donc le processus qui analyse non seulement des informations sur l'apprenant, mais aussi ses produits pour estimer son niveau de connaissance du domaine. Mais, cette modélisation ne peut être réalisée sans problèmes. En effet, et selon plusieurs chercheurs, (Kass, 1987 ; Nicaud & al., 1988 ; Holt & al., 1994 ; Danna, 1997), les problèmes liés à la

modélisation de l'apprenant reviennent tout d'abord, à déterminer les informations précises sur l'apprenant qu'on veut modéliser, ensuite, vient le choix d'un formalisme pertinent de représentation de ces informations et, en dernier lieu, l'élaboration des processus qui construisent le modèle de l'apprenant.

Les systèmes de modélisation se distinguent selon les moyens qu'ils utilisent pour stocker les informations sur l'apprenant, les méthodes de construction qu'ils appliquent pour l'élaboration du modèle de ce dernier, ainsi qu'à l'utilisation qu'ils font de cet ensemble d'informations (Danna, 1997). Dans la section suivante, nous présentons certains aspects qui caractérisent le modèle de l'apprenant.

1.2.4.2 Aspects du modèle de l'apprenant

Le rôle de la modélisation de l'apprenant était initialement clairement différencié : l'apprenant était l'objet modélisé. Mais pour que ce rôle soit bien rempli, le modèle de l'apprenant peut être caractérisé par un certain nombre de critères relatifs aux méthodes de construction de ce modèle (Holt & al., 1994 ; Kass, 1987 ; Dillenbourg & Self, 1991). Donc, il peut être :

1. **implicite** lorsque les informations décrivant le comportement de l'apprenant et influençant le déroulement de l'interaction avec le système sont incorporés dans ce dernier ;
2. **explicite** lorsque les informations sur l'apprenant sont intégrées et codées dans le système de manière explicite dans le but de gérer l'interaction avec l'apprenant ;
3. **statique** lorsque les connaissances de l'apprenant sont déterminées avant toute utilisation et ne peuvent être l'objet d'une modification en cours de session ;
4. **dynamique** lorsqu'on peut ajouter ou modifier les données en cours de session ;
5. **spécifique** lorsqu'il peut être adapté à une catégorie d'apprenants ;

6. **de surface** lorsqu'il contient des informations limitées qui ne peuvent expliquer l'état cognitif de l'apprenant ;
7. **profond** dans le cas où il contient des informations plus représentatives de l'état cognitif de l'apprenant.

Nous avons intégré dans notre système certains de ces critères, notamment les critères 5 et 7. En effet, d'une part, notre système est adapté aux élèves du primaire, puisque l'apprentissage de l'arithmétique est essentiel dans cette période, d'autre part, notre système cherche à savoir les causes réelles des erreurs effectuées par l'élève, dans ce cas un diagnostic précis est nécessaire.

Dans la suite, nous présentons, en premier lieu, les informations qui doivent être contenues dans le modèle de l'élève. En second lieu, nous exposons les différentes techniques de représentation de ces informations qui sont utilisées par certains systèmes tutoriels. Dans la dernière section, nous décrirons les processus d'élaboration de ces ensembles d'informations ainsi que les méthodes qui ont été appliquées dans cette optique.

1.2.4.3 Contenu du modèle de l'apprenant

Le contenu du modèle de l'apprenant est indispensable pour la réalisation des systèmes tuteurs intelligents suffisamment adaptatifs. Cette nécessité a conduit certains chercheurs à introduire deux approches complémentaires pour déterminer le contenu du modèle de l'apprenant, à savoir, *l'approche fonctionnelle* et *l'approche extensionnelle* (Self, 1987 ; VanLehn, 1987). La première approche vise à décrire le contenu du modèle sous forme de fonctions que le système doit assurer. La seconde approche, en se basant sur les résultats obtenus par la première approche, essaie de déterminer les informations qui doivent apparaître dans le modèle de l'apprenant (Bruillard, 1997).

Cependant, la mise en œuvre d'un modèle de l'apprenant fondé sur ces approches a, au fil des années, révélé un certain nombre de problèmes difficiles. Ces difficultés ont poussé les chercheurs à explorer de nombreuses alternatives qui vont faciliter la tâche de modélisation en intégrant plusieurs fonctions dans le

modèle à constituer. Un travail de clarification sur les différentes fonctions des modèles de l'apprenant a ainsi été exposé par deux chercheurs VanLehn (1987) et Self (1987).

- **Classification de VanLehn**

VanLehn (1987) a proposé une classification qui contient quatre fonctions. Selon lui, le modèle de l'élève doit permettre :

1. d'augmenter la connaissance de l'élève en passant au thème suivant lorsqu'on constate que ce dernier maîtrise bien le thème en cours d'apprentissage ;
2. d'intervenir au moment où l'élève commet des erreurs et d'offrir des conseils non sollicités;
3. de générer les problèmes de façon dynamique plutôt que seulement présenter les problèmes prédéfinis ;
4. d'individualiser les explications suivant le niveau de la connaissance de l'élève.

Comme on peut le constater, les fonctions énumérées concernent seulement le modèle de l'élève, mis à part le modèle de diagnostic. Ces fonctions donnent des indications importantes quant aux informations qui doivent être contenues dans le modèle de l'apprenant.

- **Classification de Self**

Self (1987) a identifié vingt utilisations différentes d'un modèle de l'apprenant dans des systèmes tutoriels intelligents existants. Mais, il a considéré seulement six fonctions principales, qui dépendent des caractéristiques du modèle de l'apprenant et de l'existence d'un modèle du processus d'apprentissage (Bruillard, 1997, Danna, 1997) :

1. **corrective** : le modèle de l'apprenant doit pouvoir aider à éliminer les connaissances erronées de l'élève ;
2. **élaborative** : le modèle de l'élève doit servir à compléter les connaissances de l'élève, de choisir le prochain sujet à aborder. Ce choix peut être : basé sur le

curriculum, fait par comparaison des réponses expert-apprenant, fait par une analyse interne de la connaissance de l'apprenant, ou laissé à l'élève ;

3. **stratégique** : le modèle de l'élève est utilisé pour contrôler l'interaction et la stratégie d'enseignement suivie par le tuteur, comme par exemple, le plan de déroulement de session ou le style d'interaction ;
4. **diagnostique** : le modèle de l'élève doit servir à la construction d'un diagnostic plus fiable et plus précis surtout dans le cas où plusieurs interprétations sont a priori possibles ;
5. **prédictive** : le modèle de l'élève peut servir à la prédiction du comportement de l'élève face à un problème afin de limiter l'espace de recherche. Cette prédiction peut porter sur la performance de l'élève ou sur les effets des actions didactiques envisagées ;
6. **évaluative** : le modèle de l'élève peut être utilisé d'une part, pour mesurer l'efficacité potentielle d'un système en comparant différentes stratégies didactiques avec un apprenant simulé. D'autre part, le contenu du modèle peut servir à évaluer les performances de l'élève.

Les fonctions du modèle de l'apprenant énumérées ci-dessus tiennent compte surtout de l'aspect évaluatif (stratégique → évaluation → prédiction → diagnostic). Dans notre modèle, nous avons tenu compte de certaines fonctions que nous avons jugées importantes, notamment les quatre dernières. En effet, nous avons utilisé le traçage de modèle (Théorie ACT* d'Anderson) pour savoir les stratégies de résolution de l'élève, évaluer ses connaissances, prédire ses actions face à un problème, évaluer aussi sa performance et faire un diagnostic plus fiable. Les étapes successives de son processus de résolution sont comparées une à une avec celles apparaissant dans la trace du processus de résolution du système.

Comme le contenu du modèle de l'apprenant dépend des stratégies pédagogiques que les concepteurs d'un système tutoriel intelligent veulent implanter dans leur système, nous exposons dans la suite des exemples de stratégies implémentées dans certains systèmes tutoriels.

1.2.5 Les stratégies pédagogiques

Les travaux sur les stratégies d'apprentissage ont soulevé de nombreuses questions en matière d'intervention pédagogique. Un certain nombre de ces stratégies ont été élaborées. Cette diversité reflète les différents points de vue sur l'enseignement. Dans la suite nous présentons très brièvement les modèles liés à l'interaction de l'apprenant avec les systèmes d'apprentissage.

1.2.5.1 L'interaction

Lorsqu'un apprenant s'engage dans des activités d'apprentissage « c'est pour acquérir des connaissances, des attitudes et des compétences qui lui permettent d'interagir adéquatement avec l'environnement dans lequel il évolue » (Brien, Bourdeau et Rocheleau, 1999). Cette interactivité, n'est autre qu'une interaction (Charlier, 1999 ; Baker, 2003), soutient l'engagement de l'apprenant dans une activité d'apprentissage afin de reconnaître ses connaissances antérieures (Brien, Bourdeau et Rocheleau, 1999).

D'ailleurs, l'interaction tient une place centrale dans la conception des systèmes d'apprentissage et son rôle dans ces systèmes est fondamental. En effet, c'est la nature de l'interaction entre l'apprenant et le système qui conditionne l'apprentissage (Py, 2001 ; Core et al., 2003 ; Rosé et Torrey, 2005). Baker (2003) de sa part a souligné que la majorité des théories d'apprentissage accordent un rôle essentiel à l'interaction. Cette interaction peut être considérée comme une séquence d'actions qui se construisent les unes à partir des autres, selon des contraintes données, des connaissances à utiliser et des stratégies à appliquer de manière à atteindre un but. Dans ce contexte, l'interaction ne serait alors qu'une forme de dialogue permettant à un apprenant d'utiliser un système d'apprentissage comme participant à son propre plan (Vernant, 1992 ; Caelen et Villasenor, 1997 ; Baker, 2003).

1.2.5.2 Le dialogue comme moyen d'évaluation

Le dialogue représente un mode d'évaluation qui nous permet d'obtenir un aperçu des connaissances conceptuelles et de la manière de raisonner un apprenant au

regard de certains concepts. Il est difficile de savoir si une réponse incorrecte est le résultat d'une erreur d'inattention ou plutôt d'un manque de connaissances conceptuelles indispensables. De même, il arrive aussi qu'un apprenant trouve la réponse correcte à partir d'un raisonnement erroné. Dans ce contexte le dialogue joue un rôle important. En effet, d'une part il permet à l'apprenant de communiquer ses connaissances et d'autre part, il nous donne la possibilité de sonder la réflexion et le raisonnement de l'apprenant de manière plus approfondie pour mieux déterminer son niveau de compréhension et ainsi diagnostiquer ses lacunes. De plus, il permet aussi à l'apprenant d'augmenter des gains assez considérables au niveau son apprentissage ; selon Graesser et al., (Graesser et al., 2001), le module de dialogue Atlas, ajouté au système tutoriel Andes (VanLehn et al., 2005) a permis de doubler les gains d'apprentissage de 0.9 à 1.8 dérivation standard. Un certain nombre de projets Why-AutoTutor (Graesser et al., 2003), Why2-Atlas (VanLehn et al., 2002), ITSpoke (Litman et Silliman., 2004), ScoT (Pon-Barry et al., 2004) et des travaux (ITS2004 Workshop) récents témoignent d'ailleurs d'un intérêt croissant d'utiliser le dialogue, notamment en langage naturelle, pour encourager l'apprenant à construire lui-même ses connaissances et à augmenter les gains de son apprentissage.

Cependant, pour engager un dialogue avec l'apprenant sur un domaine, le système doit disposer d'une certaine compréhension de ce domaine, c'est-à-dire d'un formalisme pour représenter les connaissances de ce domaine et de mécanismes spécifiques pour les manipuler (Bruillard, 1997 ; Grasser et al., 1995). De plus, et pour favoriser au maximum ce dialogue, il faut doter le système d'apprentissage d'un modèle dynamique de dialogue capable de traiter en temps réel les stratégies d'échange d'information avec l'apprenant (Caelen, 1995a). Cela conduit notamment à utiliser un modèle pédagogique adéquat capable d'élaborer ce véritable dialogue. Ce modèle peut être plus ou moins complexe suivant le degré de l'interactivité « plus l'interactivité est grande, plus le système d'apprentissage sera complexe » (Brien, Bourdeau et Rocheleau, 1999). L'objectif étant de faire acquérir certaines connaissances à l'apprenant, le système d'apprentissage doit suivre des stratégies tutorielles particulières. Mais, le problème est de déterminer

quelles stratégies de dialogue peuvent être utilisées pour faciliter l'apprentissage incrémental et permettre effectivement au système de s'adapter aux besoins de l'apprenant.

Ce problème a conduit Rosé et al. (2001) à distinguer deux modèles d'interaction utilisés dans le domaine des systèmes tutoriels intelligents, à savoir l'interaction de type socratique et l'interaction de type didactique. Dans leur article intitulé « A Comparative Evaluation of Socratic versus Didactic Tutoring », les auteurs ont présenté en détail les résultats de leur étude comparative entre ces deux modèles en fonction de leur mode d'interaction avec l'apprenant.

Dans cette étude, tout d'abord un cours de l'électricité et l'électronique est présenté aux apprenants via le Web en utilisant le système BE&E (Rosé et al., 1999). Ensuite, ces apprenants complètent leur apprentissage en utilisant un laboratoire (ou deux) simulé par le système afin d'évaluer et renforcer les concepts introduits dans le cours. Puis, les apprenants sont invités à résoudre un problème simulé de l'électronique et de l'électricité. Leur tâche consiste à trouver trois endroits, dans un circuit de courant continu, où ils pourraient recevoir une lecture de voltage non nul. Les erreurs commises par les apprenants déclenchent, suivant les situations, les deux modes de dialogue et le tuteur commence à leur poser des questions pour les encourager à bien réfléchir au raisonnement derrière leurs actions incorrectes. Cependant, dans le dialogue didactique, le tuteur explique d'abord tout ce que l'apprenant a besoin de savoir avant de lui poser des questions. Par contre, dans le cas du dialogue socratique, le tuteur commence par poser des questions ouvertes à l'apprenant et lui donne moins d'explication afin de lui laisser la possibilité de juger lui-même son comportement.

Dans la suite nous décrivons très brièvement les deux stratégies de dialogue et nous discutons les résultats de cette évaluation formelle.

1.2.5.3 Dialogue socratique

Par l'interaction socratique, le système tutoriel intelligent engage l'apprenant, lorsqu'il commet des erreurs, dans un dialogue qui le conduit à découvrir ses

erreurs et ses contradictions (Nkambou, 1996 ; Bruillard, 1997 ; Rosé et al., 2001). Ce style est caractérisé par un raisonnement dirigé par lequel le tuteur essaie autant que possible d'éviter de donner des informations à l'apprenant (Rosé et al., 2001). Ce type d'enseignement consiste à poser des questions à l'apprenant de manière à le faire évaluer ses propres hypothèses afin d'y découvrir éventuellement des ambiguïtés, des imprécisions et des contradictions pour finalement tirer les déductions correctes à partir des faits qu'il connaît (Bruillard, 1997). Cela suppose, pour le tuteur, une compréhension des hypothèses de l'apprenant et que le dialogue orienté vers la recherche de contradictions soit maîtrisé (Bruillard, 1997). Cette méthode a été utilisée dans SCHOLAR (Carbonell, 1970) et WHY (Stevens, 1977), premiers systèmes tutoriels intelligents, avec un certain succès (voir section 2.12). Ces systèmes sont orientés vers l'acquisition de connaissances plus factuelles que procédurales, de plus l'apprenant peut reprendre le contrôle de l'interaction (dialogue à initiative mixte).

1.2.5.4 Dialogue didactique

Dans le style d'interaction didactique, le système tutoriel présente à l'apprenant une explication complète de la matière et ce dernier doit l'apprendre. Au cours d'une séance d'apprentissage, le tuteur conduit l'apprenant par un raisonnement dirigé semblable à celui utilisé dans l'interaction socratique, sauf que les questions, posées à l'apprenant, doivent attirer son attention aux informations que le tuteur lui a déjà expliquées (Rosé et al., 2001).

Ce mode d'interaction exige de l'apprenant qu'il assimile bien la matière qui lui est présentée, qu'il doit retourner les informations qui lui sont fournies et qu'il respecte les contraintes qui lui sont imposées. Le dialogue didactique assure un guidage et un contrôle précis du processus d'apprentissage basé sur le traitement des informations fournies par l'apprenant dans son interaction avec le système (Core et al., 2003 ; Bruillard, 1997).

Le système tutoriel basé sur ce mode d'interaction laisse très peu d'initiative à l'apprenant. Le système contrôle ainsi que l'apprenant dispose des connaissances

de référence mais ne laisse pas l'apprenant exprimer sa propre compréhension du problème car l'apprenant doit utiliser les informations fournies par le système (Rosé et al., 2001). Dans ce contexte d'apprentissage, le système évalue les réponses de l'apprenant selon ces informations et non par rapport à la stratégie utilisée par ce dernier. SOPHIE (Brown et al., 1975) est un exemple de système tutoriel intelligent qui utilise ce mode d'interaction (voir section 2.12).

1.2.5.5 Discussion

Les résultats de la comparaison formelle présentés dans l'article de Rosé et al. (2001) démontrent une tendance en faveur du style socratique sur le style didactique « The results of our rule gain analysis demonstrate that students in the Socratic condition learned more effectively than students in the Didactic condition » (Rosé et al., 2001). Ces résultats confirment d'ailleurs ceux trouvés par Collins et Stevens (1982) en utilisant le système Why. Ces derniers chercheurs ont rapporté que les meilleurs enseignants ont tendance à utiliser le style tutoriel socratique dans leur enseignement. Cependant, Graesser et al., (1995) ont rapporté de leur part que les tuteurs humains utilisent rarement le mode d'interaction socratique et que leur enseignement est très efficace.

Toutefois, des faiblesses importantes peuvent apparaître dans ces deux modèles de dialogue. Pour le dialogue de type socratique, malgré la liberté apparente accordée à l'apprenant d'explorer ses propres connaissances et malgré aussi le fait que cette méthode a été utilisée avec un certain succès dans certains systèmes tutoriels, notamment Scholar (l'initiative mixte permet d'obtenir une certaine souplesse dans la conversation), ce modèle souffre au moins de deux limites. La première, la plus courante, est un risque de déstructuration du dialogue si elle est utilisée de manière trop intempestive (Mendelsohn et Dillenbourg, 1991). La deuxième limite, est que, si les stratégies implantées dans certains systèmes sont indépendantes du contenu du modèle de l'apprenant et peuvent être utilisées dans d'autres cadres, elles semblent plus délicates à appliquer à des domaines ne concernant plus simplement des faits mais aussi des procédures.

Pour le dialogue de type didactique, le suivi serré du tuteur ne garantit pas l'efficacité de l'apprentissage. En effet, tout d'abord l'apprenant n'a pas l'occasion d'exprimer explicitement sa propre compréhension, ni d'utiliser ses propres connaissances ; le tuteur est plus concentré sur les connaissances de référence que sur les connaissances dont l'apprenant pourrait disposer (Balacheff, 1994). Ensuite, l'apprentissage dans de tels environnements ne peut conduire l'apprenant à construire de nouvelles connaissances ni d'améliorer celles dont il dispose.

1.2.5.6 Dialogue et approches ouvertes

La tâche principale d'un dialogue est de fournir à l'apprenant la possibilité de juger lui-même son comportement. Il lui permet de porter un regard réfléchi sur ses propres actes, de s'interroger sur le sens de ses acquis, autrement dit, de se dialoguer avec lui même. Ce dialogue avec soi contribue, selon Allal (2001), à alimenter la capacité de l'apprenant à enter dans de nouvelles situations d'interaction. Il apparaît, dès lors, que le dialogue peut conduire l'apprenant à une réflexion plus explicite (Tchetagni et al., 2007) et mieux articulée sur son propre processus cognitif et en cela affecter favorablement son développement cognitif. En d'autres termes, l'apprenant doit être impliqué dans son propre processus d'apprentissage et dans son processus d'évaluation (s'auto-évaluer, Mitrovic et Martin, 2007 ; Allal, 1999) afin de connaître son propre fonctionnement cognitif.

Les systèmes tutoriels intelligents reconnaissent de plus en plus l'importance de mettre l'apprenant en contrôle de son apprentissage et de favoriser le développement de son autonomie (Dimitrova et al., 2000, 2003 ; Tchetagni, 2005). Nous voyons apparaître des approches de modélisations plus ouvertes, comme *Open Learner Modeling*, qui conduisent l'apprenant à réguler ses apprentissages de manière mieux réfléchie en lui facilitant l'accès au contenu de son modèle afin de lui permettre de mieux conceptualiser sa manière d'apprendre et de le rendre responsable de son évaluation ; ce qui revient à mettre en œuvre le processus de la *pensée réflexive* (Dewey, 1933). Dans la suite, nous allons discuter très brièvement cette approche.

1.2.5.7 Le modèle Open Learner Modeling (OLM)

L'idée que l'apprenant explore le contenu de son propre modèle est rendue opérationnelle grâce à l'approche OLM. En effet, cette approche peut mettre à la disposition de l'apprenant le contenu de son modèle qui était jusque là conservé dans les systèmes (Morales et al., 1999); bien plus, elle peut aussi confronter la représentation du système et celle de l'apprenant à travers le dialogue, la négociation, le questionnement et la justification des choix afin que l'apprenant raffine sa compréhension du contenu de son modèle et, par conséquent, raffine la représentation du modèle que le système a généré (Giardina et Laurier, 1999). Cette approche semble efficace puisqu'elle provoque chez l'apprenant une réflexion plus approfondie et lui fait prendre conscience de l'état de ses connaissances, de ses forces et de ses faiblesses, lui facilite sa réflexion sur son apprentissage (Bull et al., 2005 ; Tchetagni, 2005). D'ailleurs, certains travaux permettent à l'apprenant d'agir sur ses propres connaissances en dialoguant et en négociant son modèle avec le système (Dimitrova, 2003). Ces recherches, basées sur l'approche OLM, s'inscrivent dans une démarche visant à soutenir la métacognition chez l'apprenant (Noël, 1997). Dans ces travaux, les modèles présentés aux apprenants sont soit visualisables, soit modifiables (Zapata-Rivera et Greer, 2003), soit négociables avec le système (Dimitrova, 2001, 2003 ; Tchetagni et al., 2005a). Ces techniques de modélisation semblent avoir un impact bénéfique sur la motivation d'apprentissage de l'apprenant (Bull et al., 2005 ; Dimitrova, 2003). En effet, elles ont montré l'intérêt pour l'apprenant de mobiliser son processus métacognitif pour mieux gérer et contrôler ses propres démarches cognitives, d'être plus autonome, plus motivé et davantage responsable de son apprentissage.

Selon les notions développées par Flavell (1979) la métacognition peut être vue comme la représentation que l'apprenant a de ses connaissances et le contrôle qu'il a sur lui-même et sur ses activités cognitives. En d'autres termes, il y a métacognition lorsque l'apprenant engage sa pensée à réfléchir sur lui-même. Cette pensée réflexive (Dewey, 1933) demande de porter un regard critique sur

ses propres connaissances à travers une prise de conscience. Pour cela, il faut qu'il y ait un transfert de contrôle d'apprentissage à l'apprenant. Et pour que ce transfert s'effectue, il est important d'amener l'apprenant à une certaine réflexion cognitive explicite sur les situations proposées et son comportement particulier. Le fait de rendre explicite la pensée réflexive de l'apprenant dans les situations d'apprentissage (Tchetagni et al., 2007) a montré des résultats encourageants en motivant l'apprenant à utiliser les environnements d'apprentissage qui considèrent les activités métacognitives.

Certains travaux axés sur la métacognition s'intéressent davantage à la modélisation de l'apprenant comme un moyen de réflexion sur son propre apprentissage (Baker, 1991 ; Bull et Pain, 1995 ; Kay 1995). Des systèmes tels que Auto-Tutor (Person et al., 2001), Atlas-Andes (Rosé et al., 2001), Why-Andes (Makatchev et al., 2004), Why-AutoTutor (Graesser et al., 2003) et d'autres, font ressortir l'importance d'utiliser des approches de modélisation capables d'amener l'apprenant à réfléchir sur ses propres activités cognitives, sur ses propres actions, sur sa conscience de l'état de ses connaissances. L'idée d'un système, comme Style-OLM (Dimitrova, 2001) par exemple, où l'on mise sur un dialogue collaboratif de l'apprenant avec le système afin de déduire un modèle de l'état cognitif de cet apprenant. Le système, utilisant une méthode diagnostique interactive, est visualisable et modifiable sous la forme de graphes conceptuels. Selon Dimitrova, cette forme de diagnostic mise en place dans les modèles ouverts de l'apprenant (OLM), donne des résultats plus fiables que les approches classiques. Le système Prolog-Tutor (Tchetagni et al., 2005a) procède de la même logique. Ce système est basé sur un dialogue tutoriel avec l'apprenant à travers l'évaluation de sa performance. En effet, un problème est exposé à l'apprenant et pour le résoudre, ce dernier devra répondre à une série de questions liées à ce problème ; le système déclenche un dialogue avec l'apprenant afin de le rendre conscient de ses décisions. En même temps, l'apprenant peut poser des questions prédéfinies au cours de son interaction avec le système.

On peut cependant s'interroger sur la manière dont se comporte l'apprenant dans un domaine d'apprentissage plus complexe. En d'autres termes, quelles sont les activités qu'un système tutoriel doit proposer à l'apprenant, dans cette situation, afin de l'aider à acquérir des connaissances métacognitives à partir de son modèle. D'ailleurs, Rich (1983) souligne que l'apprenant qui ne peut envisager de façon objective ses propres connaissances ou qui maîtrise mal le domaine d'apprentissage, peut perturber son modèle. Dans ce contexte, il n'est donc pas souhaitable de laisser l'apprenant modifier son modèle. Dans le même ordre d'idées, Laveault (2000) confirme qu'il y a un véritable problème qui se pose au niveau du transfert de contrôle d'apprentissage, à savoir comment passer d'une situation d'apprentissage où le système contrôle une grande partie de l'activité de cet apprentissage, à une situation où elle sera de plus en plus assumée par l'apprenant. Il ajoute aussi « qu'il est tout aussi absurde de penser aider l'apprenant en contrôlant tous les aspects de son apprentissage que de croire qu'il apprendra de façon autonome en l'abandonnant à lui-même ». Il est aussi illusoire de croire que tous les apprenants, surtout dans le cas d'activités complexes, peuvent contrôler de façon raisonnable leurs démarches d'apprentissage (Grangeat, 1999). Au contraire, certains apprenants, notamment ceux en difficulté d'apprentissage, « se limitent souvent à un contrôle de leur action en fonction de ses effets immédiats, sans anticipation, ni réelle évaluation » (Grangeat, 1999).

Dans la section suivante, nous allons passer en revue les différents types de modélisation de l'état cognitif de l'apprenant puis les techniques de diagnostic associées.

1.2.6 Différents types de modélisation de l'apprenant

Un modèle de l'apprenant est une structure de données qui caractérise l'état des connaissances de cet apprenant (Self, 1994). Il va se définir par l'écart entre les connaissances propres de l'apprenant et les connaissances cibles de l'expert telles qu'elles sont représentées dans le système. Pour concevoir cet écart, deux grandes approches peuvent être utilisées : modèle d'expertise partielle ou par recouvrement dans lequel la connaissance de l'élève est un sous-ensemble de la connaissance de

l'expert ; modèle différentiel ou par perturbations qui incorpore des connaissances fausses correspondant à des préconceptions erronées (Bruillard, 1997).

1.2.6.1 Le modèle de recouvrement (overlay model) (VanLehn, 1988)

La modélisation par recouvrement consiste à représenter, dans le système, les connaissances de l'élève comme un sous-ensemble de la connaissance experte (figure 1.2). Dans cette approche, le modèle de l'élève est construit en comparant la performance de l'apprenant avec celle de l'expert. Le but du tutoriel est alors de compléter les connaissances de l'élève pour qu'il s'approche de celui de l'expert.

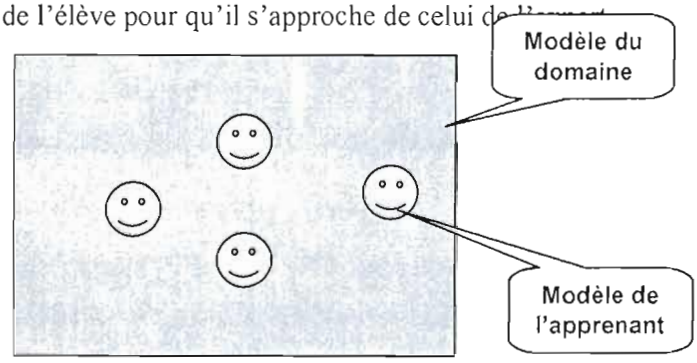


Figure 1. 2 : Modélisation par recouvrement

- **Avantages de ce modèle**

Cette approche présente certains avantages. D'une part, elle se déduit facilement de l'analyse d'un domaine, de sa décomposition en connaissances et compétences élémentaires (Nicaud et al., 1988). D'autre part, l'approche est simple, puisque les seules informations importantes sur l'état cognitif de l'apprenant sont celles décrivant ses acquis et ses lacunes, sans nécessiter d'analyse plus profonde des conceptions de l'apprenant.

- **Problèmes de ce modèle**

Malgré les avantages de cette approche, elle souffre au moins de deux difficultés majeures (Nicaud et al., 1988; Bruillard, 1997). Tout d'abord, ce modèle ne peut donner aucune indication permettant d'expliquer pourquoi un apprenant n'a pas effectué le meilleur choix possible. D'autre part, il ne permet pas de prendre en compte des méthodes ou résultats incorrects que l'élève peut avoir acquis. Or, les études sur les erreurs ont montré que de nombreuses erreurs ne sont pas dues à un

manque de connaissances de l'apprenant, mais plutôt à l'application correcte de procédures fausses.

1.2.6.2 Le modèle par perturbations (buggy model)

Les travaux précédents modélisent la connaissance de l'élève comme un sous-ensemble de la connaissance correcte. Ils ne permettent pas de prendre en compte des méthodes ou résultats incorrects que l'élève peut avoir acquis. Or, des études (comme celles de Cox, 1975 ; ou de Brousseau, 2001) ont montré que de nombreuses erreurs ne sont pas dues à un comportement irrégulier des apprenants, mais à l'application correcte de procédures erronées. Il faut prendre en compte ces erreurs de type systématique, que les chercheurs vont désigner par le terme «bug».

La méthode qui tient compte de ce type d'erreurs, appelée modélisation par perturbations (buggy model) est une extension de la précédente (Brown et Burton, 1978 ; VanLehn 1990). Ce modèle consiste donc à représenter, en plus des règles qu'il possède, les « mal-règles » qui sous-tendent les erreurs systématiques de l'apprenant. Ces erreurs sont considérées comme des perturbations de la connaissance experte (Danna, 1997).

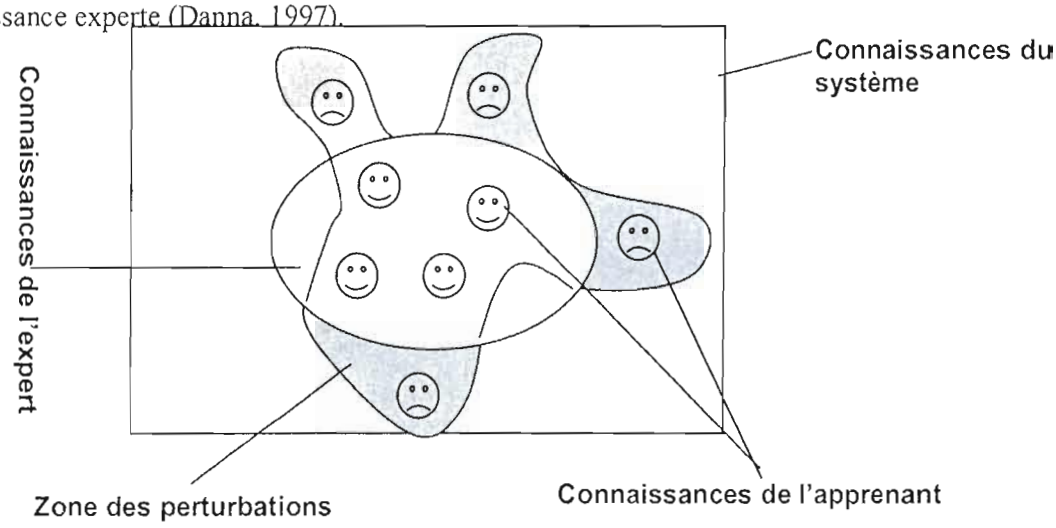


Figure 1.3 : Modélisation par perturbations

- **Avantages de ce modèle**

Cette approche de modélisation a l'avantage de prendre en compte les erreurs de l'apprenant autrement que par des lacunes, contrairement à l'approche précédente. D'autre part, ce modèle a induit des stratégies basées sur la remédiation (Wenger, 1987).

- **Problèmes de ce modèle**

Si une description des erreurs est certainement importante pour caractériser l'apprenant, il n'est pas évident quelle soit suffisante pour déterminer les types d'intervention à effectuer. Étiqueter un comportement ne donne pas les informations nécessaires pour choisir une remédiation adaptée (Bruillard, 1997).

Le modèle par perturbation a été utilisé par Brown et Burton (1978) dans la conception de leur système Buggy. Sur la base d'un catalogue d'erreurs déjà observées, le système essaie de comprendre les origines des « bugs » commis par des élèves (en les comparant à ce catalogue). Il utilise pour cela les règles correctes et les « mal-règles » qui figurent dans son catalogue d'erreurs.

Après avoir étudié les informations qui doivent être contenus dans le modèle de l'apprenant, nous nous intéressons maintenant aux techniques pour représenter ces informations.

1.2.7 Importance des connaissances pour le STI

L'objectif que doit atteindre un système tutoriel intelligent est qu'il accomplisse la tâche qui lui a été demandée de façon efficace, le plus rapidement possible et en fournissant des explications adéquates sur ses décisions. Pour cela, il est donc nécessaire qu'il possède les *connaissances* adéquates de son environnement. Cela amène quelques problèmes de la représentation des connaissances, auxquels nous essaierons de donner un début de solution du point de vue de l'intelligence artificielle conjointement à celui de la psychologie cognitive.

1.2.7.1 Notion de la connaissance

Les connaissances sont vues comme des entités symboliques manipulables par l'ordinateur, correspondant aux significations associées à des descriptions du monde, ou « concepts », qui peuvent être exprimées par le langage naturel. Selon Newell (1981), la connaissance peut donc être définie comme la *perception* et la *compréhension* qu'un agent intelligent a d'un monde externe ; cette connaissance va lui permettre d'avoir un comportement rationnel orienté vers l'obtention d'un but.

1.2.7.2 Notion de la représentation de la connaissance

La représentation de la connaissance est la modélisation du monde réel et la détermination de procédures d'interprétation faisant le lien entre le modèle et le monde, tant au moment de l'acquisition de connaissances qu'au moment de l'élaboration de ce modèle (Haton & al., 1991). La connaissance représentée peut être un concept, une méthode, un fait, un modèle. Elle peut avoir différentes modalités : statique ou évolutive, certaine ou incertaine, explicite ou implicite (Marino, 1993).

1.2.7.3 Deux points de vue sur la représentation des connaissances

Deux points de vue sur la représentation des connaissances peuvent être considérés : celui de l'intelligence artificielle et celui de la psychologie cognitive.

En intelligence artificielle, la représentation des connaissances est décrite comme « une combinaison de structures de données et de procédures interprétatives qui, si elles sont bien utilisées dans le programme, conduira à une expérience de *connaisseur* » (Barr et Feigenbaum, 1986).

Le point de vue de la psychologie cognitive est proche de celui de l'intelligence artificielle sur ce sujet. Pour cette discipline, toute activité mentale humaine se fonde sur une représentation interne du monde extérieur, constituée à la fois de connaissances sur les objets, sur les situations et sur les événements (Shute, 1996).

Cependant, les deux approches doivent résoudre un problème important : celui de la modélisation des différentes formes de connaissances. Pour ce même problème, ces approches vont donner des modélisations de natures différentes et pourtant complémentaires : le développement de modèles dérivés de l'étude expérimentale qui s'attache à une modélisation du fonctionnement cognitif humain (ce qui est la base de la psychologie cognitive) et celui de modèles formels qui s'attache à une modélisation formelle des mécanismes de la connaissance en utilisant la machine comme support (ce qui est la base de l'intelligence artificielle) (Chaudet & Pellegrin, 1998).

Par exemple, elles font toute deux une distinction fondamentale entre les connaissances dites déclaratives (celles qui « déclarent » comment sont les choses du monde, qui en décrivent les caractéristiques et les composantes) et les connaissances dites procédurales (celles qui portent sur les différentes manières de procéder dans une tâche, savoir-faire) (Anderson, 1983).

Cependant, en psychologie cognitive, on considérera une dimension supplémentaire, selon laquelle les connaissances sont catégorisées en deux types d'informations différentes qui sont nommées « mémoire sémantique » et « mémoire épisodique » (Mayers, 1997). Les connaissances générales et permanentes sur les objets du monde de la mémoire sémantique se construisent à partir des informations stockées en mémoire épisodique, par la généralisation d'événements observés de façon régulière. Cette distinction n'est pas prise en compte par les modèles de représentation des connaissances en IA, qui ne concernent que des connaissances générales portant sur un domaine particulier, excepté dans le cadre des réseaux de neurones artificiels aptes à mémoriser les situations épisodiques.

1.2.7.4 Utilisation des connaissances

Les connaissances sont utilisées pour des activités qu'un système tutoriel intelligent est censé pouvoir exécuter (Chaudet & Pellegrin, 1998) :

- *l'acquisition de nouvelles connaissances sur l'apprenant*, classiquement nommée *apprentissage*. Ce processus d'acquisition permet au système non seulement de mémoriser de nouveaux faits, mais aussi d'ajouter les nouvelles connaissances aux anciennes. Il permet aussi au système tutoriel de prendre en charge la mise à jour des connaissances. Le tuteur peut poser des questions à l'élève pour savoir d'autres informations supplémentaires sur l'apprenant ;

- *la récupération de connaissances du domaine déjà acquises*, ce que l'on nomme le *rappel*. Cette récupération de connaissances doit être adaptée au problème posé au système. De plus, il faut préciser la connaissance qui s'applique à un problème donné lorsque le système est censé connaître de nombreuses connaissances différentes ;

- *le raisonnement avec les connaissances dont dispose le système* : le système doit être capable de raisonner sur les connaissances qui lui ont été explicitement fournies pour déduire et vérifier de nouveaux faits. Ce raisonnement est en général dirigé par un but que le système doit atteindre pour résoudre un problème. Il existe plusieurs modes de raisonnement qui peuvent être intégrés dans le système, comme par exemple, le raisonnement formel, le raisonnement par analogie ;

- *le raisonnement avec des connaissances incomplètes et imprécises*, dans la plupart de situations d'apprentissage, le système doit manipuler des connaissances incomplètes et imprécises. Le raisonnement avec des connaissances incomplètes exige l'utilisation des hypothèses, c'est à dire des affirmations provisoires susceptibles de devenir fausses. Tandis que, le raisonnement avec des connaissances imprécises utilise des méthodes d'approximation, comme par exemple, les probabilités et les ensembles flous (Marino, 1993).

Dans les sous-sections suivantes, nous présentons quelques techniques de représentation de connaissances, en indiquant pour chacune, ses éléments de base, le mécanisme de raisonnement qu'elle utilise et ses avantages et inconvénients.

1.2.8 Techniques de représentation

Une technique de représentation des connaissances est un formalisme qui facilite la représentation de la connaissance dans un ordinateur pour qu'elle soit utilisée pour inférer des nouvelles connaissances (Marino, 1993). Dans les trois dernières décennies, l'intelligence artificielle a vu apparaître différentes techniques de représentation de connaissances tels que les systèmes de production, les réseaux sémantiques, les prédicats logiques, les graphes conceptuels.

Nous nous attarderons ici sur les formalismes de représentation logique, sur les réseaux sémantiques et sur les systèmes de production. Ils sont utilisés pour représenter des connaissances déclaratives et procédurales.

1.2.8.1 La logique

La logique a un fondement rationnel solide. Puisque, d'une part, elle possède une sémantique claire (Besnard, 1989), et d'autre part, elle permet de décrire des connaissances d'une façon proche du langage naturel en utilisant un mécanisme inférentiel qui lui est inhérent (Haton et al., 1991). D'ailleurs, les premiers systèmes de représentation des connaissances créés en intelligence artificielle ont utilisé la logique mathématique pour formaliser les connaissances.

Pour l'IA, le rôle de la logique en représentation des connaissances est fondamentale, car tout autre mode de représentation (comme par exemple, les réseaux sémantiques (§2.7.2)) pourra être traduit en logique mathématique.

Il existe plusieurs familles de logiques. Chaque famille hérite des avantages et inconvénients inhérents aux formalismes logiques utilisées. Outre ces caractéristiques, chacune possède des propriétés qui la rendent unique. Le choix entre ces formalismes est effectué au moment de la conception du système. Il dépend des propriétés dont le concepteur veut doter son système final. La divergence sur ces propriétés explique que les systèmes tutoriels intelligents basés sur la logique n'utilisent pas tous le même système logique.

Nous présentons les différents systèmes logiques utilisés dans les tuteurs actuels afin de représenter explicitement la connaissance de l'élève.

- **Logique propositionnelle (ou calcul des propositions)**

La logique propositionnelle a pour objectif de valider la vraisemblance de proposition ou d'énoncés, c'est-à-dire de déclarations portant sur des faits, des événements se déroulant dans le monde (Chaudet & Pellegrin, 1998 ; Thayse et al., 1988). Ces énoncés, formulés dans une langue donnée, peuvent être traduits en logique propositionnelle en attribuant une *valeur de vérité* (vraie ou fausse).

La puissance d'un tel système logique est justement que les déductions qui seront faites seront toujours vraies, tant que les faits desquels elles sont issues le seront aussi. Cependant, le système ne peut servir à simuler le raisonnement humain. Ainsi, un exemple classique montre les limites du raisonnement propositionnel (Thayse et al., 1988) :

> « Tout homme est mortel ; Socrate est un homme ; donc, Socrate est mortel ».

Cette logique ne permet pas de dire si ce raisonnement est correct (ou valide). Sa syntaxe n'est pas assez fine, puisqu'elle prend chaque proposition dans son intégralité (dans l'exemple précédent, elle ne fait pas la différence entre un homme et Socrate). Elle n'est pas en mesure de déduire des connaissances sous-jacentes à ces phrases, comme l'implication suivante : « si un être est un homme, alors il est mortel ». De plus, elle n'est pas en mesure de tenir compte d'un raisonnement qui peut évoluer dans le temps. C'est pour cela que la logique des prédicats est venue compléter le calcul des propositions. Mais, ce formalisme a néanmoins été utilisé dans certains systèmes pour représenter la connaissance de l'apprenant (comme par exemple le système Scent, c'est un système conseiller en programmation Lisp (Brecht et al., 1989)).

- **Logique des prédicats**

La logique des prédicats est très utilisée en intelligence artificielle pour la représentation des connaissances. Elle propose des possibilités supplémentaires au modèle précédent et permet de formaliser un ensemble plus vaste de

connaissances et de raisonnements (Chaudet & Pellegrin, 1998). Cette logique est convenable pour définir les relations entre les différents éléments d'une représentation donnée. L'un des grands apports de la logique à l'IA est le langage Prolog (Colmerauer et al.,, 1983) fondé sur la logique du premier ordre et utilisé comme langage de base de plusieurs systèmes experts développés par la suite. Le système «The logic Theorist » (Newell et Simon, 1963) est l'un des premiers tuteurs intelligents basé sur cette logique pour décrire les connaissances de son utilisateur.

- **Logique probabiliste**

Dans les domaines où interviennent des connaissances approximatives, mal définies ou vagues, les systèmes logiques précédents ne peuvent être utilisés pour représenter ces connaissances. Cependant, d'autres formalismes de représentation de ce type de connaissances résultent des travaux effectués en logique mathématique, à savoir, entre autres, la logique probabiliste (Haton et al., 1991).

La logique probabiliste est utilisée principalement pour la gestion de l'information incertaine à l'aide de mesures de probabilité sur des propositions décrites dans un langage donné (Fabiani, 1996). Les valeurs de vérité de ce formalisme appartiennent à l'ensemble des réels compris entre 0 et 1.

Le système tutoriel intelligent FBM (Feature Based Modeling, soit « modélisation orientée traits ») (Kuzmycz et al., 1992 ; Webb, 1993) est un exemple des systèmes tutoriels basés sur la logique probabiliste. Les connaissances de l'élève y sont représentées sous la forme d'un ensemble de règles de production qui symbolisent les *traits* caractéristiques du comportement de l'apprenant (Danna, 1997). Les techniques d'apprentissage que le système met en œuvre lui permettent de représenter tous les comportements de l'apprenant. De plus, son mécanisme de raisonnement tient compte des éventuelles informations bruitées et permet de suivre les changements dans le comportement de l'apprenant (Weeb, 1993).

Un inconvénient majeur de la logique probabiliste est que ce formalisme exige que la somme des valeurs de vérité associées à des propositions complémentaires

soit égale à 1 (e.g. une proposition *soleil_demain* et son contraire ¬*soleil_demain*). Cet inconvénient fait qu'il est généralement difficile de faire évoluer la base de connaissance (Haton et al., 1991).

- **Logique floue**

Depuis que la logique floue a été introduite par Zadeh en 1965, de nombreuses applications de ce « nouveau » concept d'ensemble ont été développées dans plusieurs domaines. En fait, toute application qui s'intéresse à la manipulation de connaissances vagues ou incertaines peut faire appel à la théorie des ensembles flous (Cayrol et al., 1982).

D'une façon générale, tous les problèmes concrets liés aux traitements des connaissances approximatives sont confrontés aux notions d'incertitude et d'imprécision. L'incertitude peut être traitée de manière probabiliste, c'est essentiellement l'observation statistique qui induit dans la pratique la mesure probabiliste des incertitudes.

Quant à l'imprécision, elle peut être traitée à l'aide de la logique floue. En effet, la logique floue est née de la constatation que la plupart des connaissances ne peuvent pas être représentées à l'aide de valeurs de vérité qui ne peuvent prendre que deux valeurs (vrai ou faux) (Cayrol et al., 1982). En introduisant une infinité de valeurs entre vrai et faux, la logique floue comble donc les lacunes des autres systèmes logiques.

Dans notre vie courante, nous utilisons essentiellement des concepts liés entre eux par des règles logiques. Ces concepts, qui possèdent un fort contenu sémantique, sont matérialisés par des mots, plus ou moins vagues. Cependant, la logique floue se propose de formaliser l'usage de ces termes vagues, dans le but de les rendre manipulables par la machine.

D'ailleurs la logique floue, qui a beaucoup de points en commun avec les réseaux de neurones (Kasabov, 1995), est utilisée pour des applications où l'on dispose d'une expertise humaine pour résoudre un problème (contrairement aux réseaux de neurones qui nécessitent essentiellement des données). En effet, la construction

d'un modèle en logique floue passe toujours par la transcription d'une expertise humaine sous la forme de règles floues. Celles-ci sont ensuite utilisées par le modèle expert dans un système intelligent. Cependant, la manipulation de règles floues (c'est-à-dire non précises) peut générer un nombre d'erreurs non négligeable. La mise en place d'un système basé sur la logique floue nécessite donc une attention particulière pour que le système ne comprenne pas des anomalies.

Le système Sherlock II (Katz et al., 1994) est un exemple de système tutoriel qui utilise ce formalisme pour représenter la connaissance de l'apprenant. Ce système s'intéresse au diagnostic de circuits électriques. En effet, le modèle de l'élève est constitué d'un ensemble de variables floues, évaluant la compétence de l'élève à utiliser des voltmètres pour réaliser des tests sur des circuits électriques, sa capacité d'interpréter les résultats obtenus à ces tests, et son habileté à lire des schémas électriques, etc.

1.2.8.2 Les réseaux sémantiques

Les réseaux sémantiques correspondent à d'autres solutions proposées par l'intelligence artificielle pour la représentation des connaissances. Développé par Collins et Quillian (1969), le réseau sémantique est vu comme un modèle psychologique explicite de la mémoire associative humaine : la mémoire est vue comme un réseau d'unités d'information ; ces unités sont activées par un mécanisme, la « procédure d'activation », qui propage des signaux à travers le réseau (Chaudet & Pellegrin, 1998).

Un réseau sémantique comprend à la fois des relations entre différents concepts et la structure hiérarchique des concepts. Trois éléments importants sont nécessaires dans ce mode de représentation : des concepts, des classes et des relations étiquetées entre les concepts.

Pour le concept, il peut être représenté sous deux aspects : il est défini par un ensemble de propriétés et par ses relations avec d'autres concepts, dont sa classe d'appartenance. En ce qui concerne la *classe*, elle contient tous les concepts ayant

les mêmes caractéristiques, ou propriétés. Quant à la *relation* (ou un lien), elle relie deux concepts entre eux, selon un sens donné, et possède une étiquette qui explicite le type de lien existant entre les concepts de deux classes différentes (Chaudet & Pellegrin, 1998).

Par exemple, la phrase « Ali parle avec Nadia » va être représentée par deux concepts [Ali], [Nadia] et par un lien dont l'étiquette est (parle) (figure 1.4).

Deux relations importantes, qui offrent la possibilité de structurer la connaissance en une hiérarchie de concepts, sont la relation « est-un » qui relie un concept à sa classe et la relation « sorte-de » qui relie la classe à ses sous-classes (Marino, 1993). De plus, un concept qui spécialise un autre (sa classe par exemple), possède toutes les relations de celui-ci avec d'autres concepts, il hérite de son information.

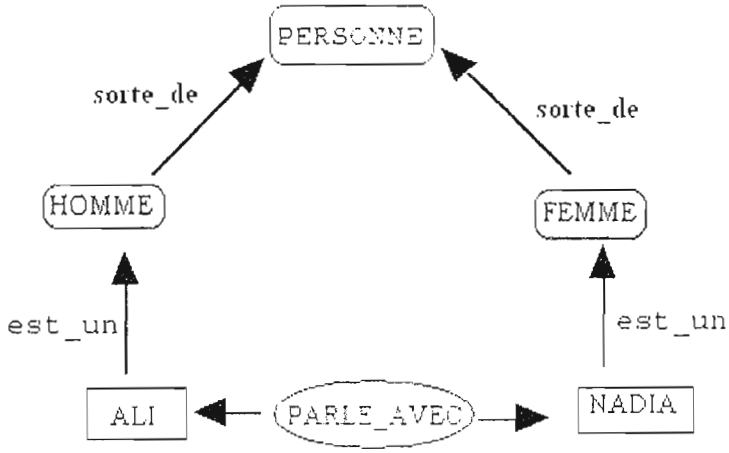


Figure 1.4 : Exemple d'un réseau sémantique

Cette technique de représentation de connaissances a été utilisée par le système tutoriel intelligent SCHOLAR. Ce système écrit par Carbonell (1970) est conçu pour enseigner des connaissances factuelles sur la géographie de l'Amérique du Sud.

SCHOLAR utilise un réseau sémantique pour représenter la connaissance. À chaque nœud de ce réseau, qui correspond à un objet ou un concept géographique, est associé un ensemble de propriétés et à chacune de ces propriétés est affecté un

indice d'importance, utilisé pour mesurer le degré de pertinence d'une propriété selon le contexte du dialogue.

- Avantages de cette technique

Le principal avantage des réseaux sémantiques est qu'ils sont bien adaptés aux domaines où les concepts sont simples et fortement liés entre eux, comme les phrases en langage naturel. La représentation sous forme d'un graphe étiqueté rend visibles les diverses relations existantes entre les concepts (Marino, 1993). De plus, grâce au mécanisme d'héritage des propriétés entre les concepts, le mode de raisonnement devient économique (Haton et al., 1991) .

- **Problèmes de cette technique**

Un premier inconvénient des réseaux sémantiques est qu'il est difficile, au moment de la conception du réseau, de déterminer le niveau de détail et les structures nécessaires à la bonne expression d'une proposition (Marino, 1993). De plus, le seul attribut attaché à un nœud est son nom, précisé par une étiquette. Cette simplification pose quelques problèmes. Tout d'abord, il est difficile de traduire des propositions ayant des quantificateurs (universels, existentiels et numériques), en traitement de langage naturel. Ensuite, la représentation statique du monde offerte par les réseaux sémantiques rend difficile la modélisation de l'évolution de l'information. D'ailleurs, ce problème est traité par Sowa (1991) à l'aide des graphes conceptuels. Sowa a utilisé le lambda calcul pour exprimer les quantificateurs logique.

En ce qui concerne les graphes conceptuels (Sowa, 1984), ils organisent les différents types d'éléments qui les constituent dans des structures différentes, contrairement aux réseaux sémantiques qui représentent dans un même réseau des propositions, des définitions de concepts, des relations et des liens entre les concepts. En effet, la structure hiérarchique de graphes conceptuels facilite la généralisation des inférences obtenues sur un graphe conceptuel particulier (Marino, 1993).

1.2.8.3 Les systèmes de production

Le système de production (appelé aussi système à base de règles) gère un ensemble de connaissances exprimées sous forme de règles de production (Kass, 1987).

Une règle de production a la forme : **Si** *<condition>* **Alors** *<action>* (il peut y avoir plus d'une condition et plus d'une action). La première partie de la règle (condition) est exprimée par un prédicat logique qui doit être vrai pour que l'action soit déclenchée ; la deuxième partie (action), est une partie exécutable de la règle qui indique des modifications à ajouter dans la base de connaissances.

Les systèmes de production comportent trois éléments : (1) une base de règles qui contient un ensemble de règles de production du système, (1) une base de faits qui réunit tous les fais des connaissances générales sur le domaine d'application et (3) un moteur d'inférence qui contrôle toute l'activité du système en utilisant un module qui explique son raisonnement.

Les systèmes de production ont été utilisés dans un grand nombre de tuteurs intelligents. L'adoption du système de production comme modèle pour élaborer un système d'apprentissage détermine bien sûr un ensemble d'hypothèses concernant l'architecture cognitive d'un humain. Newell et Simon (1972) ont identifié un certain nombre de caractéristiques des systèmes de production, caractéristiques qui les rendent particulièrement adaptés à la modélisation du comportement humain : l'homogénéité, l'indépendance, le fonctionnement en parallèle et en série, la démarche de gestion par objectifs et la modélisation de la mémoire.

À l'intérieur des systèmes de production, il est possible d'appliquer différentes théories cognitives, notamment en ce qui concerne la structure et le fonctionnement de la mémoire de travail et de la mémoire à long terme. Klahr et Wallace (1976) ainsi que Young (1976) ont utilisé des systèmes de production pour représenter la théorie de Piaget sur le développement cognitif. D'autres chercheurs dont Brown et Van Lehn (1980) ont également utilisé ces systèmes

pour appliquer leur théorie de réparation (**Repair theory**) dans la compréhension des causes et origines des erreurs d'un apprenant (voir chapitre 3).

Anderson (1983) a largement contribué à l'utilisation des systèmes de production pour simuler le comportement humain, notamment en appliquant sa théorie ACT pour construire des modèles de la cognition humaine portant sur des domaines comme la géométrie ou la programmation.

- **Avantages de cette technique**

Les systèmes de production permettent notamment d'exprimer facilement des processus de résolution de problèmes en termes de buts et de sous-buts qui sont atteints en réalisant des successions d'actions, ainsi ils permettent en général de bien résoudre certains problèmes de causalité ou de diagnostic. De plus, ils sont plus aptes à modéliser les connaissances procédurales, ce qui permet de voir clairement les conditions dans lesquelles une règle est applicable (Marino, 1993 ; Barr et Feigenbaum, 1986).

- **Problèmes de cette technique**

Dans les systèmes à base de règles, les connaissances de la base ne sont pas structurées, et elles sont dispersées dans plusieurs règles. De plus, les règles expriment souvent une connaissance apparente, superficielle qui peut occulter le raisonnement profond de l'expert humain.

Une conclusion que l'on peut tirer de cette étude sur les techniques de représentation de la connaissance d'un apprenant est que le type de représentation de la connaissance qui est choisi par les auteurs du STI dépend généralement de la nature des informations considérées. Par exemple, pour l'enseignement d'un domaine tel que la soustraction, qui est typiquement considéré comme de la connaissance procédurale, la technique employée est très souvent celle des règles de production. Certains chercheurs, dont Ohlsson (1992), ont toutefois adopté pour ce même domaine un point de vue plus proche des connaissances conceptuelles.

Dans la suite, nous nous intéressons à la détermination des processus qui élaborent le modèle de l'élève.

1.2.9 Élaboration du modèle de l'apprenant

La mise en place d'un modèle de représentation de connaissances devient indispensable pour la réalisation de tuteurs intelligents capables d'adapter leurs stratégies d'enseignement (Self, 1974). Cette nécessité a conduit les chercheurs à tenter d'induire un modèle de l'apprenant construit dynamiquement en s'appuyant sur son comportement observable (Bruillard, 1997). Pour ce faire, différents problèmes sont à résoudre : tout d'abord, celui de la détermination des processus d'acquisition d'information sur l'apprenant, qui constitue son modèle de connaissances ; ensuite, la mise en place de mécanismes pour mettre à jour ce modèle tout au long de l'interaction avec le système et, enfin l'implantation d'un processus d'interprétation de ce modèle afin d'orienter les stratégies d'enseignement. Ces problèmes sont très fortement liés et la solution adoptée pour l'un conditionne de manière très importante la façon de traiter les autres.

Nous allons présenter dans la suite les résultats qui ont été élaborés en ce qui concerne tout d'abord les processus d'acquisition, puis les techniques de diagnostic qui sont associées.

1.2.9.1 Acquisition des connaissances

L'élément essentiel dans l'élaboration du modèle d'un apprenant est la détermination des processus d'acquisition d'informations sur celui-ci. Ce processus d'acquisition est généralement appelé apprentissage, par analogie avec les systèmes d'acquisition d'informations sur les systèmes mécaniques ou sur les systèmes médicaux (Wenger, 1987).

Les méthodes d'acquisition de connaissances utilisées dans les systèmes tutoriels intelligents sont nombreuses et diffèrent selon le domaine d'application (Paiva & al., 1994b). Nous pouvons les classifier selon les critères suivants (Danna, 1997):

- l'acquisition basée sur l'observation : c'est-à-dire, les sources d'informations peuvent être implicites, les données traitées proviennent de différentes sources disponibles et en infèrent des informations sur l'élève. Le diagnostic est réalisé en s'appuyant seulement sur des observations ;
- l'acquisition basée sur l'apprenant : les sources d'informations sont explicites ; l'apprenant est considéré comme une source d'information possible. Mais, cette façon de faire a l'inconvénient que l'élève peut ne pas être en mesure de juger ses propres connaissances, et donc fournir de mauvaises informations ;
- l'acquisition basée sur le comportement de l'apprenant : c'est-à-dire, que le système affine son modèle de l'élève au fur et à mesure des interactions, en analysant le comportement de l'élève. Il peut faire en sorte de poser à l'élève des questions dont les réponses apportent des informations supplémentaires sur l'apprenant ;
- le moment de l'acquisition peut être direct : c'est-à-dire que le diagnostic est réalisé au moment où l'élève utilise le système. Comme il peut être indirect, dans un premier temps, l'élève fournit alors ses réponses qui sont enregistrées par le système, ensuite le diagnostiqueur analyse ces réponses et en infère des informations.

Dans les sections suivantes, nous présentons dans un premier lieu les principaux problèmes qui se posent généralement lors de l'acquisition de connaissances, ensuite nous décrivons quelques méthodes de diagnostic du comportement d'un élève qui sont utilisées dans des systèmes tutoriels intelligents.

1.2.9.2 Problèmes d'acquisition

De façon générale, trois principaux problèmes se posent lors de la détermination des processus d'acquisition d'informations basés sur le comportement de l'apprenant, à savoir (Haton & al., 1991, Wenger, 1987) :

- l'identification des connaissances qui expliquent le comportement de l'élève. Le problème d'inférence des informations sur l'état cognitif de l'élève est dû à

la diversité des comportements de l'apprenant, ce qui entrave d'élaborer un système de diagnostic basé sur des connaissances antérieures de l'élève ;

- l'identification des erreurs qui peuvent être commises par l'élève. Le problème d'acquisition des informations fiables à partir de réponses incorrectes de l'élève empêche de déterminer un système de diagnostic profond en plus de celui de surface. Notamment, lorsque ces informations doivent parvenir d'un grand nombre d'erreurs ;
- l'explosion combinatoire qui peut résulter de la procédure de diagnostic. Le problème survient notamment lorsque le système utilise un catalogue d'erreurs et lorsque le diagnostiqueur essaie de faire une combinaison d'un certain nombre d'erreurs qui puisse expliquer le comportement erroné de l'élève.

1.2.10 Méthodes de diagnostic

1.2.10.1 Sources d'erreurs

La recherche des sources d'erreurs et de leur mode de génération en liaison avec les connaissances des apprenants est encore très embryonnaire. On constate cependant que les erreurs proviennent essentiellement de dérivations sur des propriétés de surface et de préconceptions erronées. Il apparaît que les erreurs sont les résultats des tentatives raisonnables bien qu'infructueuses d'adapter des connaissances préalablement acquises à de nouvelles situations. Toutefois, les types d'adaptation susceptibles d'être choisis semblent dépendre fortement des caractéristiques des divers domaines de connaissances. L'analyse de ces caractéristiques est un champ d'études privilégié pour l'épistémologie et la didactique. Des théories explicatives des erreurs devraient faciliter les explications et les suggestions de remédiation.

Mais, avant de procéder à la recherche des sources d'erreurs, il y a une étape importante à ne pas négliger, celle de l'analyse des erreurs. C'est une tâche qui n'est pas évidente, mais c'est indispensable. Cette analyse ne se substitue pas aux méthodes de diagnostic, mais par contre, elle devrait permettre d'expliquer les phénomènes observés chez l'apprenant, de fournir des informations sur sa

compréhension et sur ses habiletés, et de prédire, dans une certaine mesure, les comportements ultérieurs de cet apprenant dans des conditions données.

L'analyse des erreurs n'est pas une opération simple : on est souvent amené à proposer plusieurs hypothèses pour une même erreur alors que dans d'autres cas, il est difficile d'avancer une hypothèse. La difficulté est d'autant plus grande que, pour une même activité, l'élève peut ne pas avoir un comportement stable.

1.2.10.2 Le diagnostic

Disposant d'analyses précises sur les erreurs, l'objet du diagnostic va consister à déceler les actions dans le comportement de l'apprenant pour éventuellement déterminer ses causes. Ce comportement n'étant accessible qu'au travers de certains observables qui reposent sur les possibilités offertes par l'interface, il faut l'inférer à partir des observables disponibles. Cette inférence va s'appuyer sur des techniques d'intelligence artificielle.

Notons que ces différentes techniques aboutissent à ce que l'on peut appeler un diagnostic comportemental. L'analyse porte avant tout sur la détermination de *comment* fait l'apprenant et non sur *pourquoi* il le fait. Elle reste au niveau des erreurs de surface, sans être à même d'en élucider les causes, c'est-à-dire les erreurs profondes.

Le critère qui définit une méthodologie de diagnostic est le type de représentation de la connaissance qui est utilisé dans le tuteur intelligent, procédural vs déclaratif.

VanLehn (1988) a proposé une classification des différentes techniques de diagnostic. Nous en présentons ici les grandes lignes de certaines techniques. Dans la suite, nous exposons les différentes méthodologies de diagnostic, en fonction du type de connaissance qu'elles considèrent (procédural ou déclaratif) ainsi que les techniques de diagnostic utilisées par chaque type de connaissance.

1.2.10.3 Le cas des connaissances procédurales

Plusieurs techniques de diagnostic sont utilisées lorsque les connaissances de l'apprenant sont considérées comme étant de nature procédurale, nous citons entre autres, le diagnostic par induction (Van Lehn, 1988) et le diagnostic par traçage de modèle (Anderson, 1986). Nous présentons seulement la dernière technique que nous adoptons dans notre travail. Nous justifierons notre choix dans le chapitre 2.

- **Diagnostic par traçage de modèle (Model Tracing) (Anderson, 1986)**

La méthode de diagnostic dite par *traçage de modèle* (Model Tracing) a été mise au point par Anderson et son équipe lors de l'élaboration des systèmes basés sur la théorie ACT*. L'idée de base de ce modèle consiste à utiliser le *modèle de performance* pour suivre l'état de la solution d'un élève à l'intérieur d'une tâche et à utiliser le *modèle d'apprentissage* pour déterminer l'état des connaissances d'un élève au fur et à mesure qu'il accomplit les différentes tâches (Anderson, 1986).

Dans le traçage de modèle d'Anderson, on distingue les connaissances que l'élève manifeste extérieurement pendant qu'il résout un problème, le modèle de performance, de celles qu'il possède véritablement, le modèle d'apprentissage, en se référant aux différentes règles de production représentant les connaissances du domaine. Dans le modèle de performance, on observe le comportement de l'élève en situation de résolution de problèmes et on tente d'identifier un ensemble de règles ou mal-règles du système qui pourrait expliquer un tel comportement ; cette identification permet alors de suivre les états cognitifs de l'élève en temps réel. Il faut surveiller et contrôler l'élève à chaque production en supposant qu'il y a une correspondance étroite entre les unités du modèle interne du système, les comportements observables de l'élève. D'ailleurs, Anderson a considéré que, le fait que la correction des erreurs soit effectuée dès qu'une erreur est commise par l'élève est un avantage de cette technique. Cependant, cela sous-entend que l'élève ne peut pas explorer d'autres chemins que celui fixé auparavant par le

modèle. De plus, il ne se rend compte pas par lui-même de ses erreurs lorsqu'il rencontre une impasse.

Un autre avantage de cette méthode de diagnostic est qu'elle est indépendante du domaine enseigné. Les tuteurs LISP, GEOMETRY, ALGEBRA sont des exemples des tuteurs intelligents basés sur cette technique de diagnostic.

Comme nous l'avons indiqué en haut, nous avons adopté cette méthode dans notre modèle de diagnostic.

Nous reviendrons plus en détail sur ce modèle dans les chapitres 2 et 3.

1.2.10.4 Le cas des connaissances conceptuelles

En ce qui concerne les connaissances conceptuelles qui sont représentées le plus souvent sous forme déclarative, il existe aussi plusieurs techniques de diagnostic (VanLehn, 1988 ; Burton & Brown, 1982). Nous présentons par la suite une seule technique de diagnostic dite orientée contraintes.

- **Technique de diagnostic dite orientée contraintes**

Cette technique de diagnostic, proposée par Ohlsson (1992), consiste à mémoriser un ensemble d'événements qui indiquent le comportement de l'élève face à un problème. Selon Ohlsson, ces événements doivent être énoncés sous la forme de contraintes. Ces contraintes constituent alors un sous-ensemble des concepts du domaine enseigné. Le diagnostic représente l'ensemble de ces exigences qui sont respectées et non respectées par la réponse de l'élève.

Ohlsson (1992) a appliqué cette technique de diagnostic dans le domaine de la soustraction écrite à multi-colonnes. Il a posé certaines contraintes qui concernent la gestion de l'emprunt. Comme par exemple, une décrémentation de 1 du chiffre du haut de la colonne située immédiatement à gauche de la colonne active doit être équilibrée par un ajout de 10 au chiffre du haut de cette colonne active. Ainsi, lorsque l'élève affiche les différentes étapes de sa solution, le diagnostic constitue alors l'ensemble du respect et non-respect de cette contrainte par la réponse de l'élève.

Cette technique de diagnostic présente certains avantages. D'une part, elle est indépendante du domaine enseigné, Ohlsson lui-même l'a appliquée dans le domaine de la chimie. D'autre part, la technique n'entraîne pas une explosion combinatoire lors de son application puisque à chaque étape de solution on teste si une contrainte est respectée ou non.

Cependant, ce modèle souffre au moins d'une limite principale. En effet, bien qu'il ait été appliqué dans différents domaines, ces derniers sont bien structurés (mathématique, chimie, etc.). Cependant, la détermination de l'ensemble des contraintes pour des domaines qui sont de nature moins formelle (la linguistique par exemple) n'est pas évidente, ou lorsque les experts du domaine peuvent juger que certaines réponses différentes sont exactes (ce qui n'est pas le cas pour les domaines bien formalisés tels que la soustraction, par exemple).

Pour conclure cette section, il faut souligner que le type de connaissance considéré dans le domaine enseigné constitue un critère permettant de faire un choix entre les techniques de diagnostic, comme nous l'avons dit précédemment. En effet, certaines techniques sont visiblement plus adaptées à des types de connaissance que d'autres. Par exemple, les méthodes dites par traçage de modèle ne sont utilisables que dans le cas de connaissances de nature procédurale, tandis que la technique de diagnostic dite orientée contraintes est particulièrement bien adaptée pour les deux types de connaissances : procédurales et conceptuelles. Dans la suite nous allons passer en revue les premiers systèmes tutoriels intelligents.

1.2.11 Systèmes auteurs de tuteurs intelligents

Dans cette section nous retraçons très brièvement les caractéristiques principales des systèmes auteurs telles qu'elles sont décrites par Murray dans un article de synthèse assez complet sur les systèmes auteurs de tuteurs intelligents (Murray, 1999 ; Murray & al., 2003b). Nous précisons aussi leurs avantages et leurs limites afin de mettre en valeur notre choix et notre idée de concevoir et de développer notre propre système d'apprentissage sans utiliser l'environnement

auteur.

1.2.11.1 Environnement auteur

La réalisation d'un système tutoriel intelligent demeure une tâche difficile à entreprendre ; elle requiert un plus gros effort puisqu'il s'agit de concevoir et de développer un environnement d'apprentissage complet. Le désir de simplifier cette tâche conduit plusieurs travaux de recherche, depuis les années 80, à proposer un environnement de développement dédié à la production des systèmes pédagogiques. L'idée consiste à fournir aux concepteurs pédagogiques expérimentés et non expérimentés un ensemble intégré d'outils, appelé systèmes auteurs, permettant de concevoir et de développer des systèmes tutoriels intelligents (Nkambou, 1996 ; Grandbastien, 1998).

Un système auteur propose donc, à un concepteur humain, un environnement réel de réaliser des systèmes d'apprentissage sans devoir recourir à des activités de programmation. De plus, il lui permet d'organiser sa tâche de conception et de l'assister dans la structuration des connaissances du domaine, dans ses stratégies pédagogiques, dans la spécification de ses objectifs et dans ses méthodes d'évaluation d'apprentissage.

1.2.11.2 Exemples de quelques systèmes auteurs

Dans les dernières décennies, plusieurs travaux de recherche ont été consacrés à l'élaboration de systèmes auteurs. Murray dans (Murray, 1999 ; Murray & al., 2003b) a présenté une analyse de l'état de l'art de développement et de recherche sur les systèmes auteurs de systèmes tutoriels intelligents. Il a fait une classification en sept catégories en fonction des types de STI qu'ils produisent. Ensuite, il les a regroupé en deux grandes familles : il y a ceux qui s'orientent vers la pédagogie et ceux qui s'orientent vers la performance. Nous proposons ci-dessous une liste des principaux systèmes de ces deux familles. Il faut noter que cette liste n'est pas limitative, puisque les systèmes auteurs se sont multipliés ces dernières années (Ferrero & al., 2005 ; Psyché, 2007) de sorte qu'un concepteur d'enseignement qui cherche un outil nécessaire à cet effet, est de plus en plus

confronté au problème du choix d'un système. Donc, il est difficile d'en donner une liste exhaustive, la liste suivante ne reprend, à titre d'exemples, que les systèmes qui ont été les plus cités dans la littérature (pour plus de détails voir Murray, 1999, 2003b).

1.2.11.3 Systèmes auteurs orientés vers la pédagogie

Les systèmes orientés vers la pédagogie supportent la modélisation du contenu de l'enseignement, les systèmes qui vont dans ce sens sont :

- les systèmes d'enchaînement et planification de curriculum : Cream-tools (Nkambou & al, 1996, 2003), ISD Expert (Merrill, 1998, 2003) ;
- les systèmes à stratégies tutorielles : EON (Murray, 1998, 2003a), REDEEM (Ainsworth & al, 2003) ;
- les systèmes de types de connaissances multiples : Cream-tools (Nkambou & al, 1996, 2003), XAIDA (Hsieh & al., 1999) ;
- les systèmes hypermédia intelligents /adaptatifs : CALAT (Kiyama & al., 1997), GETMAS (Wong & Chan, 1997).

1.2.11.4 Systèmes auteurs orientés vers la performance

Les systèmes orientés vers la performance supportent la réalisation d'un environnement d'apprentissage interactif, les systèmes qui vont dans ce sens sont :

- les systèmes de simulation et d'entraînement : SIMQUEST (Van Joolingen & De Jong. 2003), XAIDA (Hsieh & al., 1999) ;
- les systèmes experts en domaine : Demonstr8 (Blessing, 1997), Training Express (Clancey & Joerger, 1988) ;
- les systèmes à buts spécifiques : IDLE-Tool (Bell, 1998), LAT (Sparks & al., 1999).

Malgré que ces environnements auteurs aient des objectifs différents, ils ne sont

évidemment pas concurrents. En effet, bien que certains systèmes auteurs soient des systèmes de simulation et d'entraînement (performance), le cas par exemple du système XAIDA, ils peuvent aussi être vus comme étant des systèmes de types de connaissances multiples (pédagogie).

1.2.11.5 Critères de sélection d'un système auteur

Les concepteurs pédagogiques non expérimentés qui souhaitent réaliser des outils d'apprentissage font généralement appel à un système auteur. Cependant, ils se confrontent au problème de choix entre les systèmes auteurs qui pourront répondre parfaitement à leurs besoins. Pour faciliter la tâche à ces concepteurs, Murray (1999, 2003b) a proposé un certain nombre de critères d'analyse qui peuvent être servis comme guide dans le choix d'un système auteur à des besoins de développement d'un système pédagogique ou la conception d'un nouveau système auteur. Parmi les principaux critères, il y a tout d'abord la fidélité des outils développés, viennent ensuite la puissance et l'utilisabilité de l'environnement auteur et enfin le coût de production de ceux-ci. Cependant, Murray (2003b) a souligné qu'aucun système auteur actuel (ou à venir) ne serait en mesure de remplir tous ces critères puisqu'ils s'opposent les uns les autres.

1.2.11.6 Discussion

- **Avantages des systèmes auteurs**

Un premier avantage des systèmes auteurs, pour un concepteur pédagogique, est qu'il n'est pas nécessaire qu'il soit un expert ou un programmeur pour pouvoir les utiliser, puisqu'un système auteur est sensé être exploitable par tout type d'utilisateur. Mais cette simplicité d'utilisation constitue en fait, selon Ibrahim (Ibrahim & al, 1996), un inconvénient, puisqu'elle « se fait généralement au détriment de la richesse d'expression ». Un autre avantage, c'est qu'avec un système auteur le temps de conception et de développement d'un outil d'apprentissage est beaucoup moins inférieur au temps requis en utilisant les méthodes classiques de programmation (Wu Yao Kuang, 2000).

- **Inconvénients des systèmes auteurs**

Les limites des systèmes auteurs sont nombreuses. Mizoguchi et Bourdeau, dans (Mizoguchi et Bourdeau, 2000), ont énuméré plusieurs désavantages liés à cette approche. Parmi ces limites, on trouve que les systèmes auteurs existants ne répondent pas suffisamment aux attentes pédagogiques des concepteurs, mais ils leur demandent des efforts considérables dans la réalisation de leur projet. Ce qui montre que : « There is a deep conceptual gap between authoring systems and authors » (Mizoguchi R. et Bourdeau J., 2000). Il y a aussi le manque de l'aspect « intelligent », selon plusieurs chercheurs (Nkambou & al., 2003 ; Bourdeau J. et Mizoguchi R., 2002 ; Murray, 1999, 2003) les fonctionnalités qui déterminent que l'environnement auteur est intelligent sont quasiment absentes dans ces systèmes. En effet, par exemple, et comme le soulignent Mizoguchi et Bourdeau, la source d'intelligence dans un système est la représentation déclarative de ce qu'il connaît, notamment en ce qui concerne le modèle de l'apprenant, mais : « What about authoring systems? Do they have a model or declarative representation of what they know? Unfortunately, the answer is No. This is one of the major reasons why authoring systems are not intelligent », « Authoring tools are neither intelligent nor particularly user-friendly » (Mizoguchi R. et Bourdeau J., 2000). Ajoutons aussi le problème de la représentation du design pédagogique dans les systèmes auteurs (Psyché, 2007). En effet, le modèle pédagogique dans ces systèmes ne propose pas des stratégies pédagogiques adaptées à toutes les situations d'apprentissage, soit parce qu'il ne tient pas compte de certaines théories d'apprentissage ou soit parce qu'il est fermé et ne peut être explicite. En conséquence, ces systèmes ne sont pas en mesure de fournir de l'assistance à un concepteur pédagogique dans sa tâche de réalisation d'un outil pédagogique. C'est le concepteur qui doit s'adapter aux contraintes des systèmes et non le contraire (Psyché, 2007).

- **Notre point de vue**

Il ressort des différents problèmes présentés précédemment un certain nombre de constats. Tout d'abord, aucun système auteur existant ne nous aurait permis de

construire notre outil d'apprentissage. En effet, l'utilisation de l'environnement auteur pour la conception et le développement de l'outil demande des efforts importants, notamment en terme de temps. Il faut investir un temps personnel considérable : le temps préalable de formation à cet environnement, le temps de la maîtrise de tous ses composants, le temps accordé à la conception et au développement d'un prototype adapté. Dans ces conditions, cet investissement ne donne aucune garantie de satisfaction sur les résultats à obtenir. Ensuite, et puisque dans notre tâche il faut utiliser deux environnements distincts (environnement cognitif, la théorie ACT* d'Anderson, et environnement intillegent, les STI) pour créer un unique environnement homogène et présentant des interfaces homogènes, mais à travers l'étude des différents systèmes auteurs existants, nous avons pu constater le fait qu'il n'y a pas de système qui nous permet de faciliter cette tâche compliquée.

Ces constats nous amènent à choisir de développer notre propre système sans utiliser aucun système auteur ; d'ailleurs, nos compétences en informatique nous ont permis de dépasser ces limites en utilisant les environnements classiques de programmation. En effet, les langages de programmation nous donnent une liberté plus étendue, nous permettent d'adopter une structure convenable et nous offrent un environnement de développement unique garant d'homogénéité. Toutefois, il faut le souligner, la réalisation de notre système n'est assurément pas une tâche facile ; elle a demandé des efforts considérables surtout au niveau de développement.

Par ailleurs, nous avons développé notre système en utilisant l'outil Amine Plate-forme (Kabbaj et al., 2006) qui est une multicouche java (Open Source). Cet outil est un environnement ouvert qui facilite le développement de plusieurs types de systèmes : les systèmes intelligents, les systèmes à base de connaissances, les systèmes à base d'ontologie, les applications basées sur les graphes conceptuels, et les agents intelligents (nous reviendrons plus en détail sur cette Plate-forme dans le chapitre 4). Ce choix a bien répondu à nos besoins : nous avons pu développer l'outil souhaité et mener les expérimentations avec les publics cibles.

1.2.12 Les premiers systèmes tutoriels intelligents

1.2.12.1 Introduction

Les premiers systèmes d'intelligence artificielle voient dans la capacité de stockage des ordinateurs et la vitesse de traitement de l'information une possibilité de reproduire dans un système tous les processus cognitifs de l'homme à l'aide d'algorithmes généraux. Cependant, ces systèmes généraux s'avèrent irréalisables à cause de l'énorme quantité d'information qu'ils nécessiteraient et à l'inefficacité des algorithmes généraux pour résoudre des tâches complexes particulières.

Les systèmes suivants sont plus évolués que les premiers. Ils traitent des problèmes des domaines de connaissances bien délimités (médecine, mathématique,...) et spécifient les techniques de raisonnement selon le type de problème à résoudre (dépistage, diagnostic, etc.). La plupart de ces systèmes sont des systèmes déclaratifs, c'est-à-dire des systèmes qui séparent la représentation de la connaissance du traitement de cette connaissance.

Dans cette section, nous présentons d'abord un tableau récapitulatif des premiers systèmes, inspiré d'une étude exhaustive des premiers STI faite par Fletcher (1984) (cité dans Paquette, 1999), ensuite nous exposons très brièvement leurs caractéristiques principales, les stratégies tutorielles et leur domaine d'application.

1.2.12.2 Tableau récapitulatif des premiers systèmes

Comme nous l'avons signalé précédemment, Fletcher (1984) a réalisé une étude complète des premiers systèmes tutoriels intelligents. Il a mis en évidence les principaux éléments qui caractérisent les STI, à savoir, le modèle du domaine de connaissances, les stratégies tutorielles et les techniques de représentation de connaissances utilisées par chaque système. Le tableau suivant, inspiré de cette étude, présente ces premiers systèmes avec leurs méthodes de modélisation.

Système	Domaine	Base de connaissance	Modèle de l'étudiant	Modèle tutoriel
SCHOLAR	Géographie	Réseaux sémantiques	Par recouvrement	Dialogue socratique
WHY	Météorologie	Scripts	Par perturbation	Dialogue socratique
INTEGRATE	Mathématiques	Base de règles	Par recouvrement	Conseiller
SOPHIE	Électronique	Réseaux sémantiques	Par recouvrement	Interactions guidées
WEST	Expressions arithmétiques	Base de règles	Par recouvrement	Entraîneur
BUGGY	Arithmétiques	Réseau procédural	Par perturbation	Conseiller réactif
WUSOR	Relations	Réseaux sémantiques	Par recouvrements	Environnement réactif
LISP-TUTOR	Programmation Lisp	Base de règles	Traçage de modèle	Directif
GEOMETRY-TUTOR	Géométrie	Base de règles	Traçage de modèle	Directif

Tableau 1.1 Les premiers STI (Fletcher, 1984 ; Paquette, 1999)

Dans la suite nous présentons très brièvement chaque système.

1.2.12.3 Scholar (Carbonell, 1970)

Le système Scholar, mis au point par Carbonell (1970), est généralement considéré comme le premier tuteur intelligent. Conçu pour l'enseignement de connaissances factuelles sur la géographie de l'Amérique du Sud, son originalité réside dans le type de dialogue (Scholar utilise la méthode du dialogue socratique) qu'il peut instaurer avec l'élève. Dans ce mode d'interaction, qualifiée d'*initiative mixte* (Mendelsohn et Dillenbourg, 1991), l'apprenant et le système peuvent, à tour de rôle, prendre l'initiative et poser des questions. Utilisant un réseau sémantique pour représenter la connaissance et des mécanismes de parcours de ce

réseau et des règles d'inférences, Scholar peut répondre aux questions des apprenants.

Les stratégies d'inférence de Scholar pour répondre aux questions de l'élève sont indépendantes du contenu du réseau sémantique et peuvent être utilisées dans différents domaines (Paquette, 1999).

Les inconvénients de la méthode socratique utilisée dans Scholar sont déjà discutés dans la section 2.5.

1.2.12.4 Why (Stevens et Collins, 1982)

Afin d'étendre les capacités de Scholar dans la conduite de dialogues tutoriels, Collins (1977) a étudié des protocoles de dialogue de ce type avec des enseignants. Il en a tiré des règles permettant de conduire des dialogues de type socratique qu'il a utilisé dans le système nommé Why (Stevens *et al.*, 1982), qui traite de l'étude des causes des chutes de pluie. Le dialogue de type socratique a pour objectif de conduire l'apprenant, en lui proposant des questions, à formuler des principes généraux à partir d'exemples, à évaluer ses propres hypothèses afin d'y découvrir éventuellement des contradictions pour finalement tirer les déductions correctes à partir des faits qu'il connaît (Bruillard, 1997). Cela suppose, pour le système tuteur, une compréhension des hypothèses de l'apprenant et que le dialogue orienté vers la recherche de contradictions est maîtrisé.

Le fonctionnement de Why est conforme aux principes précédents. Le système tuteur demande à l'élève de donner toutes les causes possibles pouvant expliquer les chutes de pluie, ensuite de tirer les causes principales et de déduire une règle générale. Si la règle déduite est fausse ou incomplète, le système présente un contre-exemple à l'élève pour lui montrer que ses connaissances sont erronées et qu'il lui faut donc les remettre en cause. Comme par exemple, si l'élève croit que le riz peut pousser dans n'importe quelle région même chaude (sans ajouter les conditions nécessaires telles que l'irrigation par exemple), le système présente la région d'Arizona comme contre-exemple à l'élève et lui demande ce qu'il pense

de ce cas. L'élève est censé savoir qu'il fait chaud en Arizona et qu'on n'y trouve pas de rizières, le tuteur statue que, à partir de l'instant où il a posé ce contre-exemple, l'élève s'est rendu compte des raisons de son erreur et qu'il a affiné sa connaissance en conséquence (Bruillard, 1997 ; Stevens et al., 1982).

Toutefois, deux faiblesses importantes du système apparaissent : l'absence d'une stratégie tutorielle globale et l'insuffisance de la représentation des connaissances à base de scripts[4], à la fois pour expliquer le processus de chute des pluies et pour diagnostiquer et corriger les conceptions erronées des apprenants.

1.2.12.5 Sophie (Brown et al., 1973)

Brown, Burton et Bell ont développé en 1973 le système Sophie (*SOPHisticated Instructional Environment*), qui est une tentative de création d'un « environnement d'enseignement réactif» dans lequel les étudiants acquièrent des habiletés de résolution de problèmes en expérimentant selon leurs propres idées plutôt qu'en se laissant diriger par le système. Sophie incorpore un modèle du domaine de connaissances ainsi que des stratégies heuristiques pour répondre aux questions des étudiants, pour leur fournir des critiques de leur propre cheminement d'apprentissage et pour générer des voies différentes (Mendelsohn et Dillenbourg, 1991).

L'objectif du système Sophie est ainsi d'entraîner les étudiants au diagnostic de pannes dans le domaine des circuits électroniques, en utilisant un laboratoire simulé sur ordinateur. Un défaut ayant été introduit dans un des composants de circuits électroniques, et la tâche de l'étudiant est de trouver ce composant défectueux. L'environnement de travail est dit réactif puisque le système ne fait que répondre aux actions et sollicitations de l'étudiant et ne prend jamais l'initiative. Cependant, pour utiliser correctement le système, l'apprenant doit avoir des connaissances suffisantes de l'électronique (Bruillard, 1997, Mendelsohn et Dillenbourg, 1991; Wenger, 1987).

---

[4] Why utilise une représentation des connaissances sous forme de scripts, c'est-à-dire de suites ordonnées d'événements correspondant aux différentes étapes temporelles ou causales des processus agissant sur la chute des pluies (Bruillard, 1997).

La recherche autour de Sophie s'est étalée sur plusieurs années et a conduit à l'élaboration de trois versions de Sophie : SophieI, SophieII et SophieIII. Le passage de SophieI à SophieII illustre bien l'alternative dans la conception des systèmes : laisser le maximum de liberté d'exploration à l'apprenant en minimisant les interventions de l'ordinateur ou limiter les possibilités d'exploration offertes pour permettre d'assurer un guidage plus précis de l'apprentissage (Wenger, 1987). Concilier ces deux exigences va être un des objectifs de SophieIII.

Parmi les faiblesses du système Sophie (toutes les versions) est l'incapacité du tuteur à exploiter les erreurs commises par les apprenants (Bruillard, 1997). Cela dû, d'une part à son environnement réactif qui ne peut prendre l'initiative d'explorer les incompréhensions de l'apprenant ou de suggérer d'autres approches et d'autre part, à l'expertise implantée dans le système qui ne permet pas de rendre compte des raisonnements causaux par les experts humains.

Les auteurs décident dans le cadre du projet Sophie, de se concentrer sur la conception du module d'expertise simulant les méthodes utilisées par les experts humains. Pour explorer les problèmes de modélisation de l'élève et de guidage direct, ils choisissent de traiter des domaines considérés comme étant beaucoup plus simples, respectivement les mathématiques élémentaires (Buggy) et les jeux (West) (Wenger, 1987).

1.2.12.6 Buggy (Brown et Burton, 1978)

Buggy est un système tutoriel conçu par Brown et Burton (1978). Leur objectif consiste à utiliser les erreurs les plus fréquemment rencontrées chez les élèves, dans le domaine de l'arithmétique (l'enseignement de la soustraction), comme des variantes possibles du modèle de l'expert.

L'idée à la base de ce système est que la majorité des erreurs commises ne sont pas dues, comme on le croit souvent, à un manque de concentration de la part de l'élève. Au contraire, elles trouvent leur origine dans l'application consciente d'un algorithme partiellement erroné.

Dans le système Buggy, les erreurs de calcul sont expliquées par des perturbations du réseau procédural qui représente les habiletés de calcul (buggy model, §2.4.6). Dans ce réseau, les conceptions erronées simulant les erreurs possibles sont associées à des sous-procédures ; 330 erreurs sont ainsi encodées dans une base de procédures. Ainsi, Buggy peut être vu comme un modèle exécutable d'élève fictif (commet des « bugs »), qui est utilisé pour qu'un enseignant s'entraîne à identifier quelles sont les « procédures erronées » de cet élève fictif. Pour cela, l'enseignant soumet au système des problèmes à résoudre, et Buggy simule les réponses de cet élève fictif aux problèmes soumis. L'enseignant peut alors comparer les « bugs » qu'il a identifiés chez cet élève fictif avec ceux effectivement introduits dans le réseau de procédures (Wenger, 1987).

Malgré que le système Buggy sensibilise les enseignants aux problèmes de « bugs » et les familiarise avec les plus courants, il présente certaines faiblesses. Ainsi, selon Young et O'Shea (1981), les multiples « bugs » répertoriés dans Buggy ne sont pas assez profonds et sont trop limités, ce qui les rend inutilisables pour la remédiation. Pour eux, la chose la plus simple consiste à se concentrer sur les erreurs caractéristiques ou les plus courantes. Ainsi, dans leur étude sur la soustraction, ils soulignent l'existence seulement de 15 erreurs fondamentales (nous reviendrons plus en détail sur ce système dans le chapitre 3).

1.2.12.7 West (Burton et Brown, 1982)

West est un tutoriel conçu pour l'apprentissage de l'utilisation des opérateurs arithmétiques de base (l'addition, la soustraction, la multiplication et la division). L'élève joue contre l'ordinateur. Le jeu ressemble à celui du « jeu de l'oie ». Chaque joueur, à son tour, reçoit trois nombres tirés au hasard, avec lesquels il doit composer une expression arithmétique utilisant l'addition, la soustraction ou la multiplication. La valeur de l'expression obtenue indique le nombre de cases dont le joueur fait avancer son pion. Les connaissances nécessaires à ce jeu sont d'une part des connaissances arithmétiques (bon emploi des opérateurs et des parenthèses), d'autre part des connaissances stratégiques sur le jeu (utilisation des raccourcis, des cases qui permettent de rejouer, etc.). Ces connaissances sont

regroupées dans le modèle de l'expert, qui est ainsi capable de jouer de manière optimale. Après chaque coup joué par l'élève, le système détermine les connaissances effectivement utilisées par l'élève et les compare aux connaissances qu'aurait utilisées l'expert dans la même situation. Le modèle de l'élève mémorise, pour chaque connaissance, le nombre de fois où l'élève l'a utilisée à bon escient, le nombre de fois où l'élève l'a utilisée à mauvais escient, et le nombre de fois où l'élève ne l'a pas utilisée alors que l'expert l'a utilisée. Le modèle de l'élève est ensuite consulté par le tuteur pour individualiser le guidage et les aides.

1.2.12.8 Wusor (Goldstein, 1982)

Wusor est un autre outil de diagnostic basé sur un jeu qui fait manipuler des probabilités à l'élève. Dans ce système, les connaissances de l'élève sont représentées comme un sous-ensemble des connaissances du module expert qui sont de nature opératoire et qui sont organisées sous forme d'un graphe. Chaque nœud du graphe est une procédure. Les arcs représentent les relations entre les procédures (correcte/incorrecte). Les connaissances semblables sont donc représentées par des nœuds voisins dans le graphe, en particulier les procédures visant le même but forment un îlot de nœuds interconnectés (Wenger, 1987).

Quant au modèle de l'élève, il est constitué du sous-graphe représentant les connaissances attribuées à l'élève. Goldstein a souligné que les connaissances nouvellement acquises sont proches des connaissances déjà acquises, et focalise le diagnostic à la frontière du modèle de l'élève. Si l'élève améliore ses performances, le tuteur attribuera de préférence ce gain à l'acquisition de connaissances proches des connaissances déjà acquises. D'autres part, les erreurs commises par l'élève s'expliquent en termes d'absence de connaissances, c'est-à-dire que l'élève ignore les règles en jeu ou le concept qui lui permettra de jouer le meilleur coup possible (Wenger, 1987; Kass, 1988).

Deux faiblesses sont attribuées au système. D'abord le système a un caractère *ad hoc* de représentation de la connaissance qui se retrouve notamment dans la

définition des liens du graphe. En particulier, des liens autres que ceux utilisés peuvent être imaginés (Kass, 1988). De plus, il n'est pas évident qu'un graphe soit concevable pour tout domaine, surtout si ce dernier est complexe et si les connaissances ne sont pas déjà formalisées (Wenger, 1987).

1.2.12.9 Integrate (Kimball, 1972)

Le système Integrate, conçu en Lisp par Kimball en 1982, est une tentative de réaliser un système « connaissant ce qu'il enseigne ». La recherche d'une primitive dans ce système s'effectue en appliquant à une expression mathématique une série de transformations jusqu'à l'obtention d'une expression de forme connue qui conduit à une réponse immédiate (une primitive connue) (Bruillard, 1997).

Le système Integrate utilise une approche probabiliste pour la construction du modèle de l'élève afin de comparer ses performances avec le tuteur ou avec un autre élève, pour le choix des problèmes de remédiation et la description des effets d'apprentissage. L'apprenant peut poser un problème ou le tuteur en sélectionner un, adapté à l'élève en fonction de son modèle.

Lorsque l'élève résout un problème, le système compare la solution de l'élève avec la sienne. Si la solution proposée par l'élève est jugée meilleure que celle du système, le tuteur insère cette solution dans son répertoire. D'après Kimball (cité dans Bruillard, 1997), le système a acquis des approches qui n'avaient jamais été utilisées par l'auteur. Malgré que l'expertise de résolution de problème intégrée dans le système Integrate est jugée faible, l'approche suivie est intéressante. Elle montre en effet ce qu'un système incomplet peut faire.

1.2.12.10 Lisp-Tutor (Anderson et Reiser, 1985)

Le tuteur Lisp est conçu par Anderson et son équipe (1985). Il a pour objectif d'enseigner la programmation en langage Lisp aux débutants. Il propose des problèmes de programmation à l'étudiant et l'assiste dans sa recherche de la solution. Le tuteur Lisp est composé de deux modèles : un modèle idéal de l'élève (ensemble de règles correctes) pour guider l'apprenant à résoudre son problème de

façon correcte et un modèle erroné (ensemble de mal-règles) sert à reconnaître les erreurs commises par l'élève et à les lui signaler (Dion et Lelouche, 1992). L'utilisation combinée de ces deux modèles définit ce qu'on appelle la méthodologie du traçage de modèle. À l'aide de ce modèle, le tuteur trace le chemin emprunté par l'élève. Cette trace représentant l'état de la connaissance de l'élève, c'est-à-dire l'ensemble des règles et mal-règles utilisées par celui-ci.

Le style d'interaction utilisé par le système Lisp est défini par les caractéristiques principales suivantes (Dion et Lelouche, 1992) :

- l'intervention du tuteur est immédiate, l'élève ne peut pas se tromper ni s'éloigner de la solution correcte ;
- l'entrée de code doit se faire successivement étape par étape et « de haut en bas » : l'élève n'a pas la possibilité de choisir une partie du programme sur laquelle il désire travailler ;
- une fois le code entré, il ne peut être changé ni enlevé (même pour en essayer un autre).

Comme on peut le constater, le style du tutoriel de Lisp est très directif. En effet, l'élève reçoit une rétroaction immédiate dès qu'il commet une erreur, ou plus généralement dès qu'il entre une information s'éloignant de la solution correcte connue de tuteur. Cette approche directive, selon Anderson, réduit considérablement la complexité du diagnostic en identifiant les comportements observables et les états mentaux de l'apprenant.

Selon Dion et Lelouche (1992), ce style pédagogique n'est pas approprié pour un tuteur en programmation pour plusieurs raisons. Tout d'abord, dans certaines situations, il est probable qu'une exploration plus libre amènerait parfois l'étudiant à une meilleure compréhension du problème et de ces solutions possibles. Ensuite, il n'est pas préférable d'intervenir au moment où l'étudiant n'est pas encore en mesure de comprendre pourquoi sa solution est incorrecte. Enfin, l'idée que l'étudiant n'ayant pas la possibilité de se tromper, ne permet pas de développer l'habileté primordiale que constitue la programmation.

1.2.12.11 Geometry-Tutor (Anderson et al., 1986)

Le tuteur Geometry (Anderson et al., 1986) est un tuteur intelligent dans le sens où il peut interpréter ce que fait l'élève en comparant les règles qu'il applique avec celles de IBR (Ideal Baggy Rules). Comme le tuteur précédant, le tuteur Geometry ne permet pas à l'élève de s'éloigner d'un chemin de preuve fixé par le modèle idéal. Cela veut dire que d'autres réseaux procéduraux corrects non retenus par le tuteur et que, si l'élève s'écarte trop du modèle idéal, on lui suggère immédiatement la meilleure étape pour s'en rapprocher : si la réaction était retardée, les raisons de l'erreur seraient plus difficiles à trouver, il faudrait analyser le chemin.

Le travail porte essentiellement sur les connaissances procédurales (savoir appliquer un théorème) qui constituent la première étape dans l'acquisition de la démonstration. L'apprentissage est très structuré ; il repose sur l'hypothèse de modularité des connaissances. Chaque chapitre comporte une série d'exercices dont le but est d'appliquer un certain type de théorèmes. Ces exercices ne peuvent donc être résolus par une autre méthode, même si cette dernière fournit une solution plus simple et plus rapide. Il faut noter que l'hypothèse de la modularité des connaissances exprimées sous forme de règles de production (194 règles pour la version initiale, Anderson, 1987) facilite le fonctionnement du tuteur et n'est pas trop gênante dans les toutes premières étapes de l'apprentissage.

Dans ce contexte, les aides sont de deux types : le premier, procédural, concerne l'application d'un théorème (comme par exemple : vous n'avez choisi qu'une prémisse, le théorème choisi en nécessite deux). L'autre, heuristique, comporte la liste des théorèmes applicables qu'un élève peut essayer systématiquement, d'ailleurs sans comprendre du tout ce qu'il fait : il pourra ainsi résoudre tous les exercices s'il sait appliquer correctement les théorèmes proposés.

Comme Anderson le reconnaît lui-même (Anderson, 1987), le tuteur Geometry ne respecte pas le principe 7 (voir chapitre 2, 5.4) de la théorie ACT*, car l'élève expert doit travailler comme le novice. Dans l'utilisation de ce tuteur, Anderson confirme que les élèves novices sont moins gênés par la directivité du système

(principe 6), mais que ceux qui ont déjà une expérience se montrent plus impatients (Anderson, 1987).

Ce tuteur comporte beaucoup d'insuffisances du point de vue mathématique et pédagogique (Wenger, 1987). Il ne permet sans doute pas à un élève de collège de prendre réellement conscience de ce qu'est une démonstration, mais lui permet de voir la structure d'un pas de déduction. Par contre, ce tuteur ne peut guère s'adresser à un élève plus initié car l'atomisation et la lourdeur apparaissent comme des obstacles à la découverte de la solution : une démonstration en géométrie suppose la mise en évidence préalable de plan.

1.2.13 Problèmes des systèmes diagnostiques

Cependant, on peut constater qu'il existe un certain nombre de problèmes récurrents qui se posent à certains systèmes tutoriels. D'abord, on trouve très peu de systèmes qui dépassent le simple stade du prototype. Ensuite, les performances de ces systèmes ne sont pas très satisfaisantes. Ces problèmes sont mis en évidence par plusieurs chercheurs (Sleeman & Brown, 1982 ; Sleeman & al., 1989 ; Self, 1992, 1993 ; Mendelsohn, 1995 ; Paquette, 1999 ; Dimitrova & al., 2000 ; Nakano & al., 2002). Nous en relevons six qui traduisent la difficulté du défi à relever au moment de leur élaboration.

1. Le diagnostic dans ces systèmes porte avant tout sur la détermination de *comment* fait l'apprenant et non sur *pourquoi* il le fait. Il reste au niveau des erreurs de surface. Ce mode de diagnostic invite à des interventions du type compléter et corriger, alors qu'un diagnostic plus profond des causes des erreurs pourrait conduire à d'autres modes d'interventions ;
2. Le niveau d'intervention dans ces systèmes, suite à une erreur commise par l'apprenant est souvent situé à un mauvais niveau de détail, le système supposant trop ou pas assez de connaissance de l'apprenant ;
3. Les conceptualisations des apprenants ne sont représentées efficacement que par trop peu de ces systèmes, ce qui leur permettraient de diagnostiquer efficacement leurs «bugs» sur cette base;

4. La plupart de ces systèmes d'évaluation ont été réalisés à partir de l'intuition du concepteur et non pas à partir d'une théorie cognitive explicite d'enseignement. En effet, sans modèle cognitif, nous ne pouvons avancer aucune hypothèse pour expliquer pourquoi un apprenant ne parvient pas à réaliser correctement une tâche d'apprentissage ;
5. L'interactivité usager-système est encore trop restrictive, limitant l'expressivité de l'apprenant et, par contrecoup, limitant les capacités de diagnostic du système ;
6. Les systèmes tutoriels de diagnostic ne répondent pas encore aux attentes. Parmi les raisons qui expliquent cela, on peut retenir la difficulté d'adapter le raisonnement-expert au raisonnement des novices et aussi la difficulté du transfert, c'est-à-dire la généralisation à d'autres matières d'un processus de diagnostic élaboré pour une matière donnée.

Ces limites pourraient être encore attribuées à la plupart des systèmes actuels bien que sur certains points des progrès notables aient été accomplis. Cela tient au fait qu'ils sont liés à des limitations aux différentes conceptions utilisées dans le développement des tutoriels intelligents. Une manière de résoudre ces problèmes serait de concevoir un système tutoriel intelligent et de fonder plus intimement son architecture et sa conception sur des méthodes appropriées qui peuvent apporter des contributions efficaces à la construction du modèle de l'apprenant. D'ailleurs, nous avons conçu notre système dans cet esprit pour pouvoir résoudre tous ces problèmes. En effet, l'approche adoptée (voir les chapitres 3 et 4) est une approche originale, elle a d'ailleurs servi à la conception et au développement de notre système. Ce système est capable d'analyser et de diagnostiquer non seulement les erreurs systématiques compliquées, mais il est aussi capable de simuler diverses combinaisons des erreurs des apprenants lorsqu'ils ne maîtrisent pas l'habileté de calcul (voir chapitre 4 pour des exemples). De plus, le système utilise des explications pertinentes des procédures erronées pour pousser plus loin le diagnostic des erreurs liées à ces procédures et ainsi ouvrir la voie à la remédiation.

1.2.14 Conclusion

Nous venons de présenter les résultats obtenus en modélisation de l'apprenant. Ces résultats montrent que la grande majorité des systèmes existants sont construits pour un domaine particulier. En conséquence, ils ne sont pas utilisables pour d'autres domaines d'enseignement que ceux pour lesquels ils ont été construits. Certains systèmes ont par contre été conçus de façon à être réutilisables (cf. par exemple les systèmes basés sur le « model tracing » ou le modèle « par contraintes »).

Lors de cet exposé, nous avons présenté une décomposition du problème de l'élaboration d'un STI en distinguant trois points complémentaires : celui de la détermination du contenu du modèle de l'apprenant, du choix d'une technique de représentation de la connaissance et de la conception des processus d'élaboration dynamique du modèle de l'apprenant.

Lors de la conception de notre modèle, nous tenons compte des aspects précédents. Notre modèle de l'apprenant ne décrit que les informations sur l'état cognitif de l'apprenant. Ces renseignements doivent dépeindre aussi bien les connaissances de l'élève que ses lacunes et ses erreurs, c'est-à-dire les mal-règles qu'il utilise. Ces connaissances doivent être décrites quel que soit le type de connaissance considéré dans le tutoriel intégrant notre système : procédural ou conceptuel. Aussi ces deux types de connaissance doivent-ils être contenus dans le modèle de l'apprenant. La technique de représentation de la connaissance utilisée dans notre système est celle à base de règles. L'explosion combinatoire des erreurs est un des problèmes que nous devons considérer puisque nous cherchons à élaborer un diagnostiqueur dynamique, donc la technique de diagnostic la plus pertinente de notre point de vue est celle dite, par traçage de modèle. Elle aide à savoir ce qui se passe dans la tête de l'apprenant et à extraire des informations précises et utiles pour le module pédagogique. De plus, elle permet de prendre en compte un grand nombre d'erreurs sans craindre l'explosion combinatoire. C'est une technique originale et est une des rares utilisables en ligne pour des domaines

relativement complexes. Cette technique sera l'objet de la partie suivante de ce travail.

CHAPITRE II

THÉORIES D'APPRENTISSAGE ET DE LA COGNITION ET LES SYSTÈMES TUTORIELS INTELLIGENTS

Il s'agit, dans ce chapitre, de présenter plus en détail le cadre cognitif de notre travail. Pour commencer, nous allons tout d'abord discuter les apports de la psychologie cognitive au traitement de l'information. Ensuite, nous présenterons un certain nombre de théories d'apprentissage et de la cognition. Nous montrerons notamment le rôle joué par ces théories dans l'évolution des systèmes d'apprentissage. Dans ce chapitre aussi, l'accent sera mis sur la théorie cognitive **ACT*** d'Anderson, théorie que nous avons adoptée dans notre travail. Nous présenterons en détail une description des fonctionnalités de cette théorie que nous souhaitons voir figurer dans notre système tutoriel intelligent. Nous verrons pourquoi nous nous intéressons particulièrement à cette approche. Enfin, nous terminerons ce chapitre en étudiant les apports de la théorie d'Anderson aux systèmes tutoriels intelligents. Il va sans dire que devant l'ampleur du projet, nous n'avons pas l'ambition d'analyser l'ensemble des rapports complexes entre l'intelligence artificielle et les théories de l'apprentissage et de la cognition. Nous nous limiterons volontairement aux principaux travaux qui permettent de comprendre les apports de ces théories dans la conception des systèmes d'apprentissage.

2.1 Introduction

Qui s'intéresse à l'apprentissage des élèves est concerné a priori par les théories de l'apprentissage et donc par les différentes théories cognitives.

D'ailleurs, l'élaboration d'une stratégie tutorielle devant servir à la conception d'un tuteur intelligent doit s'appuyer sur une théorie d'apprentissage. D'une certaine façon, cette stratégie tutorielle est une mise en application d'une théorie d'apprentissage. Plusieurs théories cognitives de l'apprentissage ont été développées durant les dernières décennies. Ces théories peuvent servir en particulier de guide durant l'étape de planification d'un système d'apprentissage, elles visent aussi à étudier les méthodes et les moyens à mettre en œuvre pour favoriser l'apprentissage et l'enseignement (Gagné et al., 1992). Nkambou (1996) souligne que l'intégration ou la considération de ces théories dans les systèmes d'apprentissage au moyen de l'IA, ne peut que contribuer à augmenter leur efficacité.

Il faut souligner aussi que les théories relatives au processus d'enseignement et d'apprentissage ont été développées dans le domaine de l'éducation (Gagné, 1985b). De plus, ces théories influencent et continueront d'influencer non seulement les systèmes d'apprentissage, mais aussi, dans une large mesure, les pratiques éducatives et la vision des processus d'apprentissage et d'enseignement.

Dans le domaine des STI par exemple, et depuis l'apparition des premiers systèmes tuteurs intelligents, plusieurs systèmes ont été développés dans le but d'expérimenter l'intégration de théories reliées aux domaines (IA, sciences de l'éducation, psychologie cognitive) dans un ordinateur (Nkambou, 1996). Ainsi, les systèmes SCHOLAR, SOPHIE, BUGGY, WEST, WUSOR, Geometry-Tutor, LISP-Tutor et autres (voir tableau 1.1, chapitre 1), sont généralement influencés par ces théories. En effet, ces systèmes, en se préoccupant de plus en plus de l'apprenant, ont permis de bien comprendre la complexité reliée à la conception et au développement d'un STI ; notamment les problèmes liés à la modélisation de l'apprenant, à la prise en considération de l'expertise du domaine, à l'intégration des stratégies d'enseignement, à l'interaction avec l'apprenant, etc. Les solutions qui ont été apportées à certains de ces problèmes; elles se sont, pour la plupart, basées sur les théories d'apprentissage et d'enseignement (Nkambou, 1996).

Toutefois, il faut en effet se référer aux théories qui guident à la compréhension du comportement de l'apprenant et qui permettent d'analyser son raisonnement dans un processus d'apprentissage, afin de produire un enseignement efficace, en utilisant les méthodes d'enseignement les plus susceptibles de favoriser cet apprentissage (Gagné, 1985b ; Tardif, 1992). D'ailleurs c'est l'un des apports de la psychologie cognitive.

2.2 Psychologie cognitive et ses apports

La psychologie cognitive postule que l'on peut inférer des compétences cognitives à partir de l'étude du comportement. Elle concerne, d'une part les processus d'élaboration et de représentation des connaissances chez l'être humain, c'est-à-dire, la nature, le format et l'architecture de ces connaissances ainsi que leur fonctionnement cognitif (Grégoire, 1999). D'autre part, elle cherche à comprendre les modalités de traitement de l'information qui se déroulent dans le cerveau humain entre le stimulus et la réponse (Grégoire, 1999 ; Niederhauser et al., 1999).

Les psychologues contemporains ont mis au point des méthodes ingénieuses permettant de tester de manière rigoureuse divers modèles du traitement de l'information. Ils ont ainsi permis un progrès considérable de nos connaissances des mécanismes de la pensée. Les retombées de ces connaissances nouvelles semblent évidentes au niveau de la conception des apprentissages scolaires. En effet, si nous cherchons à trouver des informations utiles pour comprendre et surmonter des difficultés d'apprentissage chez les apprenants, les connaissances issues des recherches en psychologie cognitive se révèlent indispensables (Roblyer et al., 1997).

2.3 Les différents courants de la psychologie

Comme nous l'avons souligné auparavant, nous n'avons pas l'ambition de présenter tout l'ensemble des théories de la cognition et de l'apprentissage, mais plutôt rappeler certains principaux courants de la psychologie qui vont nous permettre de situer l'approche que nous avons adoptée dans notre travail. Nous

allons présenter en premier lieu l'approche behavioriste, puis les théories constructivistes (constructivisme et socio-culturelle), ensuite les théories cognitivistes (cognition située, cognition distribuée et cognition socialement partagée), la théorie sociale de Wenger (l'apprentissage social et l'apprentissage coopératif). Enfin, nous discutons en détail la question centrale de cette thèse, à savoir la théorie cognitive d'Anderson et ses implications pratiques.

2.3.1 Le behaviorisme

Le béhaviorisme ou la psychologie comportementale est une approche qui s'intéresse d'une part à l'étude des comportements observables de l'apprenant et d'autre part à l'environnement qui définit ces comportements (Tavris et Wade, 1999).

La théorie béhavioriste est la première théorie d'apprentissage qui a influencé les recherches dans les domaines de l'éducation et de l'enseignement. Pour cette théorie, il ne faut pas chercher ce qui se passe dans la tête de l'apprenant, c'est une *boite noire* et sa connaissance n'est pas essentielle pour comprendre les éléments qui déterminent l'apprentissage. Ce qui est important, pour ce courant, c'est l'environnement dans lequel s'effectue cet apprentissage. Cet environnement est essentiel pour la détermination et l'explication des comportements observables. Le schéma de l'apprentissage dans le modèle béhavioriste est le suivant :

Environnements d'apprentissage (Stimuli) → (Boite noire) → Comportements observables (Réponses).

2.3.1.1 Le behaviorisme et l'apprentissage

La théorie behavioriste, fondée par Watson, s'intéresse essentiellement au conditionnement d'un comportement (stimulus-réponse). Selon Watson, l'acquisition de nouveaux comportements par les individus peut être réalisée grâce à un mécanisme simple du conditionnement, et que ces comportements se manifestent aussi longtemps que les conditions (stimuli) sont présentées. Tout comportement d'un individu peut être interprété en termes stimuli-réponses.

Le processus d'apprentissage dans le courant behavioriste est expliqué par le conditionnement. Cet apprentissage peut être encouragé en associant des récompenses à une réponse correcte « renforcements positifs » ou, dans le cas contraire, des punitions « renforcements négatifs ». L'entraînement répétitif de l'individu fait disparaître certaines réactions qui lui provoquent des renforcements négatifs et en même temps il adopte un comportement adéquat qui favorise des renforcements positifs.

2.3.1.2 Le behaviorisme et l'enseignement programmé

Skinner, considéré comme l'un des pionniers du behaviorisme, soutient qu'on peut apprendre à l'élève les notions mêmes élémentaires d'un savoir à l'aide d'une méthode pédagogique appelée enseignement programmé (Bruillard, 1997). Pour arriver à cette fin, Skinner a utilisé des machines à enseigner. Dans ce contexte, la matière est divisée en petites unités présentant des contenus assez simples et avec des niveaux de difficultés progressifs. La correction des réponses de l'élève est automatisée, c'est-à-dire qu'un programme informatique intégré dans la machine évalue ces réponses. Suivant que la réponse fournie par l'élève est bonne ou non, la machine renvoie des messages de renforcements positifs (félicitation, continuez,...) ou des messages de renforcements négatifs (refaire l'exercice, réponse incorrecte, ...). L'élève ne peut passer à une étape suivante de son programme d'apprentissage que si l'étape en cours est complètement maîtrisée.

Il y a quatre principes d'apprentissage optimal pour Skinner, principes qui sont réunis dans l'enseignement programmé, (Bruillard, 1997) : le découpage de la matière en fragments élémentaires et l'attention de l'élève doit être focalisée sur un fragment très ciblé ; la stratégie d'enseignement doit être adaptée au rythme de l'élève ; l'élève doit obligatoirement fournir une réponse pour chaque fragment ; connaissance immédiate de la réponse.

Sur la base de ces principes généraux, de nombreux programmes sont conçus. Ils sont de différents types, relevant de cette théorie d'apprentissage, notamment

quant au rôle des erreurs. En effet, selon Skinner, l'apprentissage, pour qu'il soit efficace, il doit être réalisé sans erreur, c'est-à-dire que l'apprenant, s'il maîtrise bien son cours, il doit trouver de bonnes réponses aux questions qui lui sont posées (Bruillard, 1997). L'erreur est ainsi utilisée pour contrôler le cheminement de l'élève.

Pour conclure, il faut souligner que grâce à l'enseignement programmé, l'EAO a vu le jour. En effet, puisque l'individualisation repose sur la possibilité d'assurer un contrôle précis du processus d'apprentissage basé sur le traitement des indices fournis par l'apprenant dans son interaction avec le système enseignant, alors, et pour assurer et supporter cette individualisation, la voie de l'informatique est indispensable. D'où l'apparition de l'Enseignement Assisté par Ordinateur (EAO) comme un véritable mariage de l'enseignement programmé et de l'informatique (Mendelsohn, 1995 ; Bruillard, 1997).

2.3.1.3 Commentaires sur le behaviorisme

Le behaviorisme ignore le contenu de la « boîte noire » et met à part tout le système cognitif de l'individu, c'est-à-dire il ignore toutes les caractéristiques de l'individu apprenant, avec les conséquences qui en découlent et particulièrement l'échec scolaire (Hoover, 1997). En effet, les difficultés d'apprentissage de l'élève ne sont pas traitées dans leurs dimensions cognitives, c'est-à-dire comme des facteurs importants jouant sur les activités de traitement, mais seulement comme des facteurs contextuels de la situation d'apprentissage (Crinon et al., 2000). Cette conception, qui domine encore de nombreuses pratiques enseignantes, ne peut contribuer de façon efficace au développement cognitif de l'élève. Malgré que certains développements plus récents du paradigme behavioriste ont permis de renouveler et d'enrichir les modèles pour les adapter aux nouvelles technologies de l'apprentissage, ces modèles continuent à proposer des programmes de renforcement qui, dans de nombreux cas, reposent essentiellement sur les mêmes principes de base du behaviorisme (Goupil & Lusignan, 1993).

2.3.2 Le constructivisme

Le paradigme « constructivisme », qui apparaît presque dans tous les domaines, notamment dans les milieux pédagogiques, recouvre une multiplicité de significations qui renvoient à une multiplicité de points de vue (Philips, 1995). Selon Duffy et Cunningham (1996), deux idées importantes sont communes à tous ces points de vue. La première idée, c'est que l'apprentissage doit favoriser la construction des connaissances et non pas l'acquisition des connaissances. La deuxième idée, c'est que les activités d'enseignement doivent être des activités d'aide à la construction des nouvelles connaissances et non pas des activités de transmission des connaissances.

Cependant, ce paradigme a donné lieu à deux types d'approches bien distinctes, mais souvent confondues dans les modèles qui sont à la base des environnements d'apprentissage (Hannafin et al., 1997). L'approche constructiviste individuelle dérivée de la théorie piagétienne et l'approche socio-culturelle inspirée des travaux de Vygotski.

Dans les sous-sections suivantes nous présentons très brièvement les fondements de ces deux approches ainsi que les modèles d'apprentissage qui s'inspirent explicitement ou implicitement de ces paradigmes.

2.3.2.1 L'approche constructiviste individuelle

L'approche constructiviste individuelle a pris son extension en réaction de la théorie behavioriste qui centre l'apprentissage à l'association stimulus-réponse. Cette approche met en avant l'activité de l'apprenant et conçoit l'apprentissage comme un processus de construction des connaissances nouvelles (Crinon & al., 2000). Les connaissances nouvelles qui en résultent constituent les nouveaux objets de pensée sur lesquels les sujets s'appuient pour agir (Piaget, 1977).

- **Les fondements du constructivisme**

Le constructivisme est une théorie d'apprentissage développée par Piaget. Il place l'élève au centre du processus d'apprentissage. Il postule que chaque apprenant construit sa propre connaissance et que l'apprentissage passe à travers des

activités mentales et des connaissances de chacun. Et sans ces activités, aucune connaissance nouvelle ne peut être intégrée. De plus, la théorie constructiviste insiste sur le rôle essentiel des interactions incessantes pour que ces activités de construction aient lieu.

Dans ses travaux de recherches, Piaget met l'accent sur le rôle de trois processus essentiels : les processus d'équilibration, d'assimilation et d'accommodation (Piaget, 1977) : « le premier permettant la construction d'une représentation du monde extérieur afin de faciliter le passage d'un stade de déséquilibre à un stade d'équilibre, le deuxième permettant d'assimiler les nouvelles connaissances à celles déjà en place dans les structures cognitives et le troisième permettant une transformation des activités cognitives afin de s'adapter aux nouvelles situations ».

- **Le constructivisme et l'apprentissage**

Contrairement à ce qu'on trouve dans la théorie behavioriste en ce qui concerne l'apprentissage (conditionnement classique ou conditionnement opérant), la théorie constructiviste, dont la notion de l'interaction est importante, précise que l'apprentissage doit être construit grâce aux facteurs cognitifs qui peuvent influencer le comportement de l'apprenant. Les interactions entre l'élève et son environnement jouent un rôle essentiel dans la construction de cet apprentissage (Crinon & al., 2000). Ces interactions dépendent de la perception qu'a l'élève des différentes composantes.

Pour apprendre des nouvelles connaissances, l'élève doit établir des liens entre ce qu'il sait déjà, c'est-à-dire ses connaissances antérieures, et ce qu'il est en train d'apprendre. La réalisation de ce nouvel apprentissage exige que l'élève mette de côté certaines connaissances qu'il juge hors de propos. Mais, ce processus demande de la part de l'apprenant une réflexion à la fois active et constructive. Parmi les moyens qui encouragent l'élève à comprendre les connaissances qui entravent son apprentissage est de le faire participer à des activités qui utilisent la réflexion pour la construction de l'apprentissage. Le rôle de l'enseignant à ce niveau devient très important. En effet, s'il désire favoriser chez l'élève

l'apprentissage constructif, il doit offrir un environnement propice, un climat souple qui encourage l'activité, la motivation et le désir d'apprendre. Il doit aussi proposer des stratégies d'apprentissage qui répondent aux besoins de l'élève.

L'approche pédagogique constructiviste favorise la confiance des élèves en leur propre potentiel et leur propose l'occasion de prendre la responsabilité de leur apprentissage afin de construire leurs propres connaissances. Selon Piaget (Piaget, 1977), l'élève qui construit son savoir à travers sa propre initiative et son effort spontané sera capable de retenir ce savoir et aura acquis une méthodologie qui lui servira toute sa vie. Toutefois, Piaget souligne aussi que, le savoir n'est pas régit par une maturation interne ou un enseignement externe. C'est une construction active dans laquelle l'individu construit progressivement des structures cognitives de plus en plus complexes à travers ses propres activités.

- **L'erreur dans l'approche constructivisme**

Dans la perspective constructiviste, où l'erreur est considérée comme une étape normale, même si elle n'est pas nécessaire. Étape normale car les connaissances n'apparaissent pas subitement, complétées et achevées dans la tête du sujet qui apprend. Elles résultent plutôt d'un cheminement de la pensée, cheminement au cours duquel elles se voient mises à l'épreuve, confrontées à d'autres connaissances ; et les conflits, qui en résultent, contribuent à les parfaire (Dionne, 1986). Les erreurs apparaissent comme des manifestations de ces conflits : produit logique et cohérent des pensées du sujet, ces erreurs le forcent à adapter ses savoirs, le poussent à des progrès (Perkins & Simmons, 1988). Et elles constituent en même temps des sources précieuses de renseignement pour le maître qui veut aider ses élèves, le renseignant sur la pensée de ceux-ci, sur leurs acquis et sur ce qui reste à faire.

- **Le constructivisme et l'EIAO**

EIAO est souvent vu par les psychologues comme un milieu propice pour étudier et expérimenter les processus d'apprentissage. Ils peuvent même tester les modèles théoriques qui les jugent plus aptes à une implémentation (Mendelsohn,

1995). Un exemple de système d'EIAO est celui proposé par Papert (1980). C'est un prototype issu du courant constructiviste du développement de Piaget dans le domaine des micro-mondes, qu'il l'a appelé LOGO. L'utilisation de LOGO s'est présentée d'abord comme alternative à l'enseignement programmé qui pouvait constituer un autre mode d'utilisation de l'ordinateur (Bruillard et Péreira, 1987). Comme le souligne Rouchier (1992), avec LOGO, l'ordinateur n'est plus une machine à enseigner, mais un générateur de micro-mondes, dont la prise de connaissance à travers l'écriture de programmes et la conduite de projets va contribuer au développement cognitif aussi bien qu'à la construction acquisition de savoirs spécifiques.

Le constructivisme piagétien a inspiré d'autres démarches d'intégration de langages informatiques dans l'enseignement des mathématiques. Un exemple est la création par Dubinski (1991, 1992) du langage ISETL destiné à l'enseignement des notions de théorie des ensembles. L'aspect intéressant de ce travail est d'offrir un cadre cognitif pour concevoir l'activité de l'étudiant confronté à des concepts d'algèbre et d'analyse « avancés ». La théorisation associée met l'accent sur l'abstraction réfléchissante[5] (Piaget, 1977) que Dubinski présente comme la principale activité qui doit être prise en considération pour faire, d'une part, un lien entre la théorie de Piaget et la pratique éducative. Et, d'autre part, sur l'exploitation de similarités structurelles entre les processus de construction cognitive piagétiens et les activités mentales impliquées dans l'écriture de programmes informatiques.

- **Commentaires sur le constructivisme**

Durant plusieurs décennies, la théorie piagétienne a offert aux enseignants un cadre conceptuel et un modèle du fonctionnement cognitif qui permet d'expliquer

---

[5] Abstraction réfléchissante est une réflexion sur le développement de la connaissance. L'acte réfléchissant est un acte mental qui permet de faire le passage d'un vécu en acte à un vécu représenté. Il est basé sur un retour réflexif sur un vécu passé, de manière à en opérer le réfléchissement (Piaget, 1977).

le développement de la pensée et l'acquisition des connaissances. Les problèmes d'apprentissage étaient alors compris en termes de troubles opératoires. Les actions d'aide étaient, par conséquent, centrées sur le développement opératoire. Certes, ce type d'intervention a connu un certain succès dans le domaine des mathématiques. Mais, malheureusement, les limites du modèle piagétien sont progressivement apparues. Toutes les difficultés d'apprentissage ne peuvent en effet s'expliquer par des problèmes de développement logique (Bideaud, 1991). En outre, les espoirs mis dans la théorie piagétienne étaient toutefois démesurés et il est apparu que cette théorie ne permettait pas d'expliquer toutes les difficultés d'apprentissage, comme par exemple celle du calcul arithmétique, rencontrées en clinique de l'enfant.

D'ailleurs, Vygotski, de sa part, a critiqué dans son ouvrage intitulé « pensée et langage » la position de Piaget en ce qui concerne le rôle du langage égocentrique chez l'enfant. Il souligne que « les règles découvertes par Piaget ne sont pas des lois éternelles de la nature mais des lois historiques et sociales ... Piaget ne tient pas assez compte de l'importance de la situation sociale » (Vygotski, 1997 ; p : 132).

2.3.2.2 L'approche socio-culturelle (Vygotski)

L'approche socio-culturelle inspirée des travaux de Vygotski, Leontiev et Bakhtine (Wertsch, 1985) et qui met l'accent sur le contexte socio-culturel de la cognition conçoit la connaissance comme le résultat d'une co-construction (Crinon & al., 2000). Cette approche, qui s'inspire aussi des théories cognitivistes, a cependant ouvert la voie à la constitution de nombreux cadres théoriques qui ont pour point commun de concevoir l'apprentissage comme une activité située, distribuée ou partagée socialement (Brown et al., 1989 ; Pea, 1993 ; Lave & Wenger, 1991 ; McClellan, 1996 ; voir §3.4).

- **Les fondements de l'approche socio-culturelle**

Les travaux de Vygotski connaissent un succès marqué depuis de nombreuses années. Ces travaux consistent à reformuler les bases de la psychologie et mettre

l'accent sur la notion de la coopération sociale. Cette coopération permet à l'enfant de développer plusieurs activités mentales supérieures qui ont une origine sociale, ce que Vygotski nomme les fonctions psychiques supérieures, comme par exemple l'attention volontaire, la mémoire logique, l'abstraction et l'habileté à comparer et à différencier (Vygotski, 1997).

Son travail sur les activités mentales supérieures s'organise sur trois notions théoriques (Vygotski, 1997). La première est que ces activités mentales sont le résultat de toute une évolution historique et culturelle. En effet, selon Vygotski, chaque société développe sa propre culture au cours de l'histoire et un individu à l'intérieur de cette société va être amené à développer lui aussi ses fonctions psychologiques en fonction de la culture développée par cette société.

La seconde notion porte sur l'étude du développement des fonctions psychologiques à un ancrage social, selon Vygotski, la transmission culturelle se fait via le groupe culturel auquel on appartient. La médiation sociale s'inscrit dans tout ce qui est de l'ordre de l'interaction avec autrui, caractérisée dans le développement par le fait de l'existence d'une asymétrie au niveau des interactions.

Le dernier axe, est que pour que tout puisse fonctionner, l'individu doit disposer d'instrument psychologique, notamment le langage qui est l'instrument privilégié. L'idée est que ces instruments psychologiques sont tous des produits culturels issus de la société et partagent tous un caractère social.

- **L'approche socio-culturelle et apprentissage**

Vygotski s'intéressait surtout au développement des compétences cognitives. Pour Vygotski, ces compétences sont les résultats d'un « apprentissage de la pensée », c'est un ensemble d'interactions entre le novice et les membres expérimentés de la société qui lui servent de tuteurs ou de guides (Maurice, 2002). Le but de ces interactions est d'une part, de soutenir nécessairement l'acquisition des connaissances, et d'autre part de fournir en outre le contexte propice à la maîtrise des outils d'apprentissage. À l'encontre de Piaget, Vygotski croyait que le langage

constitue le principal outil d'apprentissage : l'individu devient capable, en interagissant avec son partenaire social, à exprimer ses idées et à absorber celles des autres. De plus, il affirme que le langage ne peut être égocentrique, comme il le soutient Piaget, mais qu'il a un caractère social.

- **La zone proximale de développement**

Pour soutenir ses propositions, Vygotski suppose qu'il existe une zone sensible nommée *Zone Proximale de Développement* (ZPD) qui détermine la distance entre ce que l'enfant est capable de réaliser seul et ce qu'il serait en mesure de réaliser avec l'aide de l'adulte ou dans une activité collective.

Selon Vygotski, s'il n'y a pas des échanges entre l'enfant et son entourage, on ne peut parler de développement de ce dernier. Par contre, si l'enfant se réagit dans un contexte d'interaction social (école, paires,...) alors on peut parler de développement possible. Ce développement ne peut être associé à un apprentissage. En effet, comme le souligne toujours Vygotski, le développement n'est pas réductible à l'apprentissage ni aux conditions nécessaires et suffisantes permettant l'apprentissage. Selon lui, c'est l'apprentissage qui est à l'origine de développement « l'apprentissage donne donc naissance, réveille et anime chez l'enfant toute une série de développements internes qui, à un moment donné, ne lui sont accessibles que dans le cadre de la communication avec l'adulte et la collaboration avec les camarades, mais qui une fois intériorisés, deviendront une conquête propre de l'enfant » (Vygotski, 1997).

- **L'approche socio-culturelle et l'EIAO**

Comme nous l'avons souligné plus haut, l'interaction sociale joue un rôle fondamental dans le développement de la cognition. En outre, certaines catégories d'interactions entre apprenants favorisent l'apprentissage (Dillenbourg et Jermann, 1996). Plusieurs chercheurs ont tenté d'appliquer cette théorie dans le domaine de l'EIAO pour observer les interactions entre élèves face à la machine. Ainsi, Staub et ses collègues (1994) ont conçu un système appelé HERON qui présente une utilisation originale de ce modèle. Ces auteurs ont remarqué que

lorsque deux apprenants travaillent avec le système HERON, le nombre d'interactions entre ses apprenants était trois fois plus élevé que pour les apprenants résolvant la même tâche sur du papier. Puisque, chaque apprenant est engagé dans deux types d'interactions, celles avec le partenaire et celles avec le système. De plus, les divergences entre apprenants les obligent à se justifier et à rendre explicites certains aspects du problème dont ils n'avaient pas nécessairement conscience (Dillenbourg et Jermann, 1996).

2.3.3 L'approche cognitivisme

En psychologie, le cognitivisme désigne un paradigme scientifique qui a vu s'unifier plusieurs domaines de recherches, notamment la psychologie, l'intelligence artificielle, la linguistique, etc. Il s'intéresse essentiellement au traitement en mémoire, à la perception, à l'étude du fonctionnement de l'intelligence, au traitement du langage et ce, en regard du fonctionnement du cerveau.

Gaonac'h (1991) souligne que dans une perspective cognitiviste, « les connaissances qu'un individu possède déjà sont le principal déterminant de ce que cet individu peut apprendre ». Selon Weil-Barais (1993), il existe deux types de cognitivisme :

- le cognitivisme structural qui peut être illustré, d'une part, par la théorie de la forme et d'autre part, par le structuralisme piagétien ;
- le cognitivisme computationnel, qui s'intéresse à la représentation du flux informationnel dans le système cognitif et sur le traitement de ce système.

Dans une telle perspective, le système cognitif humain est modélisé sous la forme d'un système de traitement de l'information (Rézeau, 2001).

- **Le cognitivisme et l'apprentissage**

L'apprentissage dans la perspective cognitiviste est considéré comme un processus constructif qui peut être réalisé lorsque l'apprenant modifie sa structure cognitive en traitant de façon active les informations nouvelles (Shell, 1988, cité

dans Lise, 1991). Selon les cognitivistes, l'interaction entre connaissances antérieures et informations nouvelles est au cœur du processus d'apprentissage. De plus, si la relation entre ces informations nouvelles et les connaissances en place est bien établie et intensive, alors le nouvel apprentissage pourra aussi être bien approfondi. Toutefois, l'interaction joue un rôle essentiel dans l'apprentissage. En effet, d'une part, elle facilite la compréhension et la maîtrise des contenus qui font même l'objet de cet apprentissage et d'autre part, elle développe chez l'élève certaines attitudes de pensée qui lui permettront d'être plus actif et responsable de son apprentissage.

2.3.4 Extensions de la théorie cognitiviste

Dans cette section, nous présentons trois courants de recherche qui proviennent de la théorie cognitive et nous les considérons comme des extensions de cette théorie, à savoir cognition située, cognition distribuée et cognition socialement partagée (Salembier, 1996). Ces trois approches ont certains points communs. Tout d'abord, elles s'intéressent au groupe d'individus et le considère comme un seul système cognitif. Ensuite, elles accordent une grande importance à l'environnement social et physique dans l'explication et l'analyse de l'apprentissage. De plus, ces approches étudient de façon plus profonde certains concepts comme la mémoire, le traitement de l'information,...etc. afin de les appliquer à un groupe d'individus, c'est-à-dire un système cognitif, en utilisant un outil informatique (Jermann, 1996).

2.3.4.1 Cognition et Action Situées

La cognition située est un modèle récent, issu de la psychologie cognitive, qui propose une nouvelle conception de connaissances et de représentations. En effet, les connaissances sont redéfinies comme la capacité humaine de comprendre le sens d'une situation donnée en indiquant en même temps les actions à accomplir ou les représentations à construire (Brown et al. 1989 ; Lave & Wagner, 1991).

Deux notions essentielles qui caractérisent ce courant (Suchman, 1987 ; Winograd & Flores, 1986 ; Brown et al., 1989) : le contexte d'apprentissage et l'activité de

l'apprenant. Selon ce paradigme, les connaissances sont liées à leur contexte et aux activités dans lesquelles elles sont situées. En outre, et pour que l'apprentissage soit efficace, il faut que l'apprenant se place dans un contexte et une activité authentiques (Brown et al. 1989).

Trois hypothèses sont défendues par les tenants de la cognition située :

- la première hypothèse considère la cognition située (ou action située) comme alternative à la thèse cognitiviste qui considère l'homme comme système symbolique de traitement de l'information (Suchman, 1987 ; Theureau, 1999) ;
- la deuxième hypothèse insiste sur le contexte socio-culturel dans lequel s'inscrit toute activité cognitive (Theureau, 1999) ;
- la troisième et la dernière hypothèse soutenue par les tenants de la cognition située remet en question le rôle fonctionnel des plans sur lesquels se base l'IA symbolique et hérités de la théorie cognitiviste (Salembier, 1996 ; Theureau, 1999 ; Miller et al., 1964, cité dans Suchman, 1986).

- **Problèmes de l'approche située**

La notion de cognition symbolique n'est pas acceptée par tous les tenants de l'approche située. Certains même la considèrent comme un cas particulier de l'activité cognitive (Greeno & Moore, 1993) ; d'autres encore repoussent la nature symbolique des représentations internes, mais acceptent leur existence (Salembier, 1996).

La remise en question de la notion de représentation interne par le courant de l'action située peut poser certains types de problèmes. Tout d'abord, il est impossible de faire une description complète de l'environnement, et les règles d'action exigées par l'action située sont aussi incomplètes et amènent à se questionner sur la manière dont on peut représenter le comportement d'un système intelligent. Ensuite, un autre problème survient au moment de l'analyse d'un système cognitif, c'est-à-dire lorsqu'on analyse l'activité cognitive d'un individu ou d'un groupe d'individus afin d'expliciter les étapes d'un processus de

raisonnement. Ce qui provoque une réduction de certaines activités implicites qui ne peuvent être réductibles à une symbolisation mentale selon l'approche située (Salembier, 1996).

2.3.4.2 Apprentissage situé

Généralement les connaissances acquises à l'école ne sont pas toujours exploitables dans une situation réelle et restent souvent mal intégrées et ne peuvent être utilisées dans la résolution de problèmes complexes. De ces constatations, s'est développé un courant qui considère que l'activité cognitive est totalement située et doit avoir un sens dans une situation d'apprentissage semblable à une situation réelle où les connaissances acquises peuvent être appliquées.

Cette conception de l'activité cognitive a été reprise et développée dans certains domaines, notamment celui de l'éducation et a donné naissance à un courant appelé *apprentissage situé* (Resnick, 1989). Ce courant souligne le caractère important du contexte social de tout apprentissage, de plus il insiste sur le fait que l'élève devrait être impliqué dans une communauté de pratique (Brown, 1989).

Cependant, l'apprentissage ne peut être transféré d'une situation à une autre ni d'une matière à l'autre (Mendelsohn, 1994, 1996). Mais, il faut qu'il s'inscrive dans un contexte pour que les apprenants puissent lui donner un sens. Ces apprenants ne sont pas sans connaissances, ni des simples récepteurs de connaissances. Ils viennent en contexte avec des connaissances antérieures qu'il faut tenir en compte.

Les études comparant les activités de calcul arithmétique dans un contexte de tâches standardisées (de type scolaire), et dans celui de la vie quotidienne (activités d'achat, calculs au cours du travail), montrent que le contexte fait partie intégrante de la situation et influe considérablement sur les structures de la cognition et de l'action. Ces travaux montrent, par exemple, une profonde transformation du fonctionnement de la cognition, dès lors que l'on passe d'une

situation de tests standardisés, à une situation quotidienne (Lave et al., 1984 ; Scribner, 1986).

Plusieurs exemples de ce phénomène sont décrits dans la littérature. Ainsi, Lave (1988) rapporte une observation faite sur des clientes de grands magasins. Celles-ci réussissaient 98% des opérations complexes relatives aux produits à acheter. Par contre, ces mêmes personnes ne réussissaient que 70% des mêmes opérations présentées de manière scolaire. Des exemples similaires sont rapportés par Nunes et al., (1993) qui ont étudié eux aussi les calculs pratiqués par les enfants vendeurs des rues au Brésil. Selon cette étude, les enfants calculent généralement bien et sans erreur dans le contexte de vente. Mais, par contre, lorsque des calculs similaires leur sont présentés en contexte scolaire, sous forme d'opérations, leurs performances de calcul ont chuté.

Comment peut-on expliquer de telles différences de performance en fonction du contexte ? Pourquoi le sujet n'applique pas systématiquement les procédures qu'il connaît dans toutes les situations où celles-ci sont demandées? Simplement parce qu'il n'existe pas de procédure adaptée *a priori* à tous les contextes (Grégoire, 2001). En effet, selon Anderson (1993), une procédure est une règle de production qui lie les conditions aux actions. Les conditions définissent les circonstances dans lesquelles les actions peuvent être produites. Plus la situation rencontrée par le sujet recouvre les conditions d'une règle de production, plus il y a de chance que les actions qui lui sont liées soient déclenchées (Anderson, 1996). Si une procédure n'est pas activée, c'est qu'il n'y a pas d'adéquation suffisante entre ses conditions de déclenchement et les éléments du contexte. Ce phénomène apparaît clairement dans les observations de Nunes et de ses collègues.

L'apprentissage en situation se présente donc comme le résultat d'un processus de confrontation de la personne à un contexte à la fois matériel, culturel et social objectif, dans lequel cette personne redéfinit subjectivement ce contexte en fonction du sens donné à son activité. Ces contextes subjectifs, matériels, culturels et sociaux constituent autant de mondes d'action différents.

2.3.4.3 Cognition distribuée

Comme nous l'avons souligné un peu plus tôt, la psychologie cognitive met l'accent sur les processus mentaux d'un apprenant, en utilisant certaines notions particulières telles que le traitement de l'information et la représentation de la connaissance. De nombreuses fonctions cognitives sont étudiées à cette fin, notamment, la mémoire, la perception, l'attention et le langage (Reuchlin, 1991).

Cependant, les récentes recherches en sciences cognitives précisent que la cognition et l'apprentissage sont des processus qui peuvent être réalisés dans un contexte collectif et non une affaire individuelle. La cognition est *distribuée* (Resnick, 1991), elle se construit dans l'interaction et l'activité collective.

Cette nouvelle tendance de sciences cognitives a donné naissance, depuis plus d'une décennie, à un nouveau courant nommé *cognition distribuée* (Resnick, 1991), qui s'intéresse au groupe, plutôt qu'à l'individu. Ce courant se distingue des modèles traditionnels issus des sciences cognitives. Il soutient que l'objet d'étude n'est plus considéré uniquement dans la tête d'un individu mais comprend les processus de coopération et de collaboration entre les individus (Villon, 2003).

Selon Hutchins (1992), un *système fonctionnel* est constitué d'individus et d'artefacts avec lesquels ils interagissent. La cognition distribuée propose donc d'étudier le fonctionnement de ce système de façon cognitive par rapport aux approches classiques des sciences cognitives et de décrire les propriétés cognitives en termes de structure et de traitement interne du système (Rogers, 1993).

Trois points de vue sont issus du courant de la cognition distribuée. Le premier, concerne le rôle des ressources environnementales (physique et social) qui participent à la cognition distribuée (Hutchins, 1991 ; Perkins, 1995). Le deuxième s'intéresse à la distribution de l'intelligence dans l'environnement (Pea, 1993 ; Halverson & Rogers, 1995), puisque les objets présents dans l'environnement constituent une forme de représentation externe qui interagit avec les représentations internes dans une tâche cognitive distribuée (Scaife & Rogers, 1995). Le troisième et le dernier point évoquera les possibilités d'auto-

organisation du système fonctionnel. L'organisation des interactions entre différents médias (représentations internes et externes) peut être variée en fonction des acteurs (Dillenbourg & Traum 1999).

2.3.4.4 Cognition socialement partagée

Le courant de la cognition socialement partagée (Resnick, 1991) réunit plusieurs aspects évoqués dans les sections précédentes, notamment l'aspect cognitif social, mais qui fait intervenir de nouveaux concepts (cognition socialement partagée, apprentissage collaboratif,...).

- **La dimension sociale de la cognition**

Pour le courant de la cognition socialement partagée, le rôle du fonctionnement social dans l'étude de la cognition humaine est déterminant (Resnick, 1991). En effet, pour ce courant, le social ne constitue pas seulement une composante implicite ou un élément du contexte à prendre en compte dans l'étude des mécanismes cognitifs, mais « il détermine largement la nature des processus cognitifs mis en œuvre et la performance d'un individu rapportée à un contexte social de référence » (Salembier, 1996).

Le social exerce son influence sur la cognition à travers deux aspects. D'une part, à travers l'environnement dans lequel se déroule une activité et d'autre part, à travers les outils qu'on peut utiliser pour penser. En effet, comme le souligne Resnick, la cognition utilise souvent des outils qui comprennent des histoires, des traditions intellectuelles qui ne sont pas neutres socialement. Selon Lave (1988, cité dans Salembier, 1996), cette influence du social sur la cognition peut être, dans certains cas, retraduite en terme d'influence du culturel sur le cognitif.

- **Cognition partagée**

La théorie sociale cognitive est basée sur la notion d'interaction. En effet, les interactions entre individus constituent un champ privilégié d'application et de mise en évidence du rôle joué par la cognition socialement partagée. La cohérence et la réussite de ces interactions, qui sont produites conjointement par les individus, dépendent en grande partie d'une compréhension partagée de

l'historique de l'interaction à un moment donné et des différentes alternatives possibles à venir (Schegloff, 1991).

La notion de cognition partagée a été envisagée de différents points de vue (connaissance mutuelle chez Clark & Marshall, 1981 ; environnement cognitif partagé chez Sperber & Wilson, 1986) selon notamment que l'on s'intéresse plus particulièrement à un corps de connaissances stabilisées supposées partagées ou à la co-construction par les individus d'un contexte nécessaire à la compréhension mutuelle.

En parlant de la cognition socialement partagée, on met souvent l'accent sur le « savoir commun ». Cet aspect est souligné par plusieurs chercheurs, notamment Schegloff (1991) et Cole (1991). Ce dernier par exemple, met en évidence la possibilité de penser la cognition socialement partagée en termes de « savoir commun ». À ce moment, la connaissance n'est plus simplement disponible, déposée dans quelques mémoires individuelles, mais toutefois elle demande à être partagée dans quelques mémoires collectives (notion avancée par Smith, 1994). Ainsi, en effet, « si les connaissances privées sont du ressort de l'individu alors que les connaissances partagées sont communes à tous les acteurs du système. Le fait de partager une connaissance privée avec les autres participants lui confère le statut de connaissance partagée » (Jermann, 1996).

2.3.4.5 Cognition socialement partagée et théorie de la collaboration

L'article de Roschelle et Teasley (1995), qui traite de la résolution collaborative de problèmes, illustre bien le courant de la cognition socialement partagée. L'accent est mis sur le processus d'établissement d'une conception partagée du problème. En effet, selon ces auteurs, la notion de la collaboration est une activité qui permet aux individus d'établir et de maintenir une conception partagée du problème. La collaboration engage les intervenants à apprendre une action commune sur le même objet.

Depuis peu de temps, on croyait que l'apprentissage individuel, notamment par ordinateur, était plus efficace. Or une étude conduite par Pea (1993, cité dans

Mendelson et Jermann, 1997, p : 39) révèle que l'apprentissage en collaboration serait le plus efficace. Selon lui, en effet, les apprenants qui collaborent dans leur apprentissage obtiendraient des bons résultats par rapport à ceux qui apprennent seuls.

Lorsque deux apprenants collaborent, ils *partagent* leur charge cognitive (Dillenbourg & al. 1996). En effet, pour résoudre une tâche sur un ordinateur ou sur un papier, les partenaires peuvent proposer ensemble une démarche commune qui leur facilite la résolution de cette tâche. De plus, chaque étape du raisonnement sera débattue, contestée, précisée, acceptée ou refusée. Un apprenant peut mettre en cause son raisonnement si son collaborateur lui en fournit un meilleur. Dans ce mode d'apprentissage collaboratif, les apprenants se surveillent mutuellement (L'haire, 2000). D'une part, et grâce à l'interaction avec leurs partenaires, ils intègrent mieux les explications sur une étape de résolution. D'autre part, ils doivent trouver une solution à leurs différends afin d'arriver à une résolution acceptée par tous les partenaires.

Cependant, certains problèmes peuvent nuire à l'efficacité de l'apprentissage collaboratif. On peut citer entre autres, la non homogénéité des groupes (Dillenbourg & Schneider 1995), la différence entre les niveaux des apprenants, c'est-à-dire si un apprenant a un haut niveau de connaissances, les autres membres du groupe comptent sur son savoir. A l'inverse, si les membres de groupe ont tous un niveau faible, la collaboration ne pourra être efficace, car ces participants ne pourront pas apporter des connaissances à leurs camarades. Ajoutons aussi que, lorsqu'un apprenant (ou des apprenants) laisse les autres participants accomplir la tâche sans qu'il ne fasse un effort, l'apprentissage collaboratif, dans ce cas, dérape de son but (L'haire, 2000).

2.3.5 Théorie de l'apprentissage social (Wenger)

Dans le domaine de l'apprentissage social, on voit apparaître certains concepts (comme par exemples : connaissances partagées, engagement mutuel, apprentissage collectif,...) qui amènent à la bonne gestion de connaissances dans

un contexte de pratique entre des personnes. Lave et Wenger (1991) l'appellent *communauté de pratique*.

Selon Wenger (1998), une communauté de pratique est un groupe de personnes qui mettent en commun des connaissances. Ce groupe « travaille et apprend ensemble, interagit, contribue à la création et au partage des connaissances, construit des relations, et à travers cela développe un sentiment d'appartenance et d'engagement mutuel » (Wenger et al., 2002 p.34). Être membre de la communauté de pratique permet de partager des connaissances, d'apprendre des autres membres de la communauté et de chercher essentiellement à développer ses compétences dans la pratique considérée.

L'approche de Wenger met l'accent sur la nature sociale de l'apprentissage intégré dans une communauté de pratique. Cet apprentissage, selon Wenger, est essentiellement réalisé dans un contexte coopératif et collaboratif.

Sept principes fondamentaux sont à l'origine de la théorie sociale de l'apprentissage de Wenger (Wenger et al., 2002 ; Benoît, 2000).

1. La nature sociale et culturelle de l'apprentissage est favorisée par la communauté de pratique.
2. Pour que l'apprentissage soit efficace, il faut qu'il y ait un dialogue entre les participants afin de tirer parti des expériences des autres membres des communautés de pratiques.
3. L'apprentissage développe les compétences d'un individu et les capacités d'interaction au sein de la communauté.
4. L'engagement mutuel des membres de la communauté dans une activité pratique a une conséquence importante sur l'apprentissage, puisque l'utilisation des connaissances et des compétences de chacun reproduit et innove de nouvelles connaissances.
5. L'apprentissage favorise différents niveaux de participation et encourage l'interaction entre les membres de la communauté.

6. L'apprentissage se développe dans une communauté de pratique parce qu'elle est source d'enrichissement pour les membres de cette communauté. La négociation, l'échange et les divers points de vue favorisent le développement de cet apprentissage.

7. L'apprentissage, dans une communauté de pratique, évolue dans le temps à travers les actions et les interactions entre les membres de la communauté.

Ces principes sont à l'origine de travaux plus exhaustifs de Wenger publiés dans son ouvrage intitulé « Communities of Practice : Learning, Meaning, and Identity » (1998). L'auteur a développé une théorie sociale d'apprentissage appelée théorie des communautés de pratique. Trois aspects fondamentaux caractérisent cette théorie de l'apprentissage social de Wenger (Wenger, 1998 ; Benoît, 2000; Henri, 2002) : (1) *l'engagement mutuel* ou la contribution à la construction de la communauté, ainsi au processus d'apprentissage. Dans cet aspects, les individus s'engagent mutuellement à partager leurs connaissances avec celles des autres membres de la communauté ; (2) *l'entreprise commune* est considérée comme le résultat d'un processus collectif où les négociations, les interactions et les actions communes établissent des relations de responsabilité mutuelle entre les membres de la communauté ; (3) *le répertoire partagé* est un ensemble de ressources qui peuvent être utilisées individuellement ou collectif par une communauté de pratique afin de faciliter les négociations de significations et de sens .

La théorie de communauté de pratique remet en cause l'influence technologique sur les efforts de gestion de savoirs qui a conduit à des stockages d'informations, des accumulations de bases de données (Beauchamp, 2002). En revanche, cette théorie fait des connaïssances quelque chose appartenant à une communauté capable de maintenir, de partager ces connaissances entre ses membres. Elle contribue à établir une nouvelle compréhension de l'apprentissage. Cet apprentissage, non seulement a toujours une dimension sociale, mais de plus, il se manifeste principalement dans les interactions sociales de personnes engagées dans une pratique commune.

2.4 Notre point de vue

Après ce passage en revue des différentes approches de l'apprentissage et de la cognition, nous pouvons constater que la psychologie cognitive n'est pas un corpus théorique unifié (Gardner, 1985 ; Bideaud & Houdé, 1991 ; Grégoire, 1999). Il n'y a pas une psychologie cognitive, mais plutôt un ensemble de modèles du fonctionnement du cognitif qui ne partagent parfois que quelques postulats essentiels (Grégoire, 1999). Cependant, tous ces modèles se rejoignent par leur volonté d'étudier les processus mentaux de manière scientifique. Ils cherchent tous à comprendre les modalités de traitement de l'information qui se déroulent dans le cerveau humain.

Mais à quels modèles théoriques pouvons-nous se référer pour réaliser notre système tutoriel de diagnostic qui s'appuie sur la psychologie cognitive ? La réponse à cette question est loin d'être simple.

Nous avons pu constater que le behaviorisme ne s'intéresse pas à l'explication interne, ni à la conscience d'un apprenant, il se centre sur les comportements d'un élève en terme de stimuli- réponses ; le constructivisme piagétien, quant à il, s'occupe aux processus de changements internes de l'élève. Mais les deux approches « ignorent les conditions réelles de l'apprentissage scolaire qui mettent en présence l'enseignant, les élèves, le savoir et les contraintes de mise en œuvre » (Amigues, 2001). De plus, de nombreux systèmes d'apprentissage développés par les chercheurs sur la base de ces deux approches sont trop limités et peu de ces systèmes sont réellement utilisables dans le milieu scolaire (Larochelle et Bednarz, 1994). En somme, ces modèles théoriques classiques sont de plus en plus contestés et considérés comme inadéquats pour répondre aux attentes actuelles des chercheurs (dans différents domaines) et des praticiens ainsi qu'aux demandes des institutions scolaires.

D'autre part, l'approche socio-culturelle de Vygotski se centre sur la nature sociale de l'apprentissage, et se distingue de l'approche behavioriste et de l'approche constructiviste piagétienne. Mais, cette thèse connaît aussi ses propres limites pratiques et théoriques. Notamment, l'hypothèse défendue par Vygotski,

que la représentation est une relation de type symbolique, c'est-à-dire une relation de signification entre la représentation et ce qu'elle représente. Lorsque nous entendons un mot ou que nous le lisons, par exemple, nous ne sommes pas seulement sensibles à sa sonorité ou à sa forme, nous sommes directement renvoyés à l'objet qu'il désigne. Or dans quelle mesure pouvons-nous parler de représentation symbolique pour le genre d'organismes artificiels que nous avons réalisé ? La question est loin d'être résolue et la difficulté provient ici d'un manque de détermination conceptuelle plutôt que d'un défaut d'appuis empiriques. Selon Proust (1995), le simple fait d'avoir des contenus représentationnels n'est pas une condition suffisante pour affirmer que ceux-ci sont des symboles de quelque chose. Il reste à prouver que ce contenu est effectivement utilisé par l'organisme comme un moyen de faire référence à quelque chose.

Concernant les autres approches de la cognition (située, distribuée, partagée), il n'est pas toujours facile d'établir une distinction entre ces paradigmes. La théorie de la cognition distribuée insiste sur la structure des connaissances et leur représentation. La théorie de la cognition située insiste davantage sur l'aspect contingent de l'activité humaine qui laisse une place relativement importante à l'environnement d'apprentissage du sujet. Pour le courant de la cognition socialement partagée, le rôle du fonctionnement social dans l'étude de la cognition humaine est déterminant. Il existe néanmoins des points communs importants entre ces paradigmes ; ils partagent en effet un certain nombre de positions théoriques ou méthodologiques qui contribuent à les rapprocher. D'où la difficulté parfois à décider à quel courant spécifique rattacher une recherche donnée. De plus, ces modèles ont également la particularité d'être souvent très locaux et partiels. Ils ne concernent alors qu'un type précis d'activité cognitive, et lui seul.

Comme nous l'avons souligné auparavant, certains psychologues contemporains ont mis au point des méthodes intéressantes permettant de tester de manière rigoureuse divers modèles du traitement de l'information. Ils ont ainsi permis un

progrès considérable de nos connaissances des mécanismes de la pensée. Les retombées de ces connaissances nouvelles semblent évidentes au niveau de la conception des systèmes d'apprentissage (Grégoire, 1999). Mais ces modèles présentent cependant plusieurs limites pour l'évaluation diagnostique des erreurs de l'élève. Leur principale limite résulte de leur complexité. Ainsi, pour pouvoir utiliser de tels modèles, il est nécessaire d'en maîtriser toutes les subtilités conceptuelles et de disposer des systèmes d'évaluation très sophistiqués. Ces contraintes conduisent à réserver ces modèles à des spécialistes formés pour réaliser un diagnostic très précis. Outre leur complexité, ces modèles ont également l'inconvénient d'être focalisés sur des activités cognitives très spécifiques. Si l'on souhaite appréhender une large gamme d'apprentissage, il est dès lors nécessaire de disposer de multiples modèles locaux ; à supposer que tous ces modèles existent ou soient suffisamment élaborés pour être utilisés de manière opérationnelle. Enfin, la majorité des modèles locaux ont été élaborés à partir d'études de sujets adultes (Seron, 1991). Ils nous donnent une représentation d'un système cognitif arrivé à un point de perfection.

Puisque nous désirons réaliser une évaluation diagnostique à l'aide d'un tuteur intelligent, nous devons pouvoir disposer d'un cadre théorique plus général qui nous permette de comprendre la construction progressive des apprentissages scolaires et les obstacles que les élèves peuvent rencontrer en cours de route. Ce cadre théorique doit être aisé à opérationnaliser dans des évaluations diagnostiques réalisables dans le contexte de l'EIAO.

Nous devons reconnaître que les modèles théoriques qui satisfont ces contraintes ne sont pas nombreux. Selon nous, la théorie ACT d'Anderson est une des seules théories cognitives qui soient applicables pour l'évaluation diagnostique en milieu scolaire. C'est cette approche que nous avons adoptée dans notre travail.

Dans la section suivante, nous présentons de façon générale cette théorie cognitive sous laquelle nous avons implémenté notre système tutoriel (chapitre 3). Ensuite, nous discutons de ses implications pratiques dans le domaine d'EIAO.

2.5 La théorie ACT d'Anderson

2.5.1 Introduction

L'objectif principal d'Anderson est d'explorer son modèle général d'apprentissage, en s'appuyant sur sa théorie ACT, dans l'implémentation de ses systèmes tutoriels intelligents. Son modèle constitue des réponses plausibles à des questions aussi importantes que : comment un élève acquiert-il des habiletés intellectuelles complexes pour accomplir certaines tâches telles qu'écrire un programme d'ordinateur ? Est-il utile qu'un élève soit guidé durant un tel apprentissage et si oui, à quel moment, de quelle façon et avec quelle fréquence un tuteur humain doit-il intervenir? Comment doit-on effectuer la conception d'un système d'EIAO qui joue le rôle d'un tel tuteur?

À travers son modèle d'apprentissage, Anderson propose des réponses à ces questions. D'abord en adoptant une façon particulière de modéliser la représentation et l'apprentissage de connaissances procédurales (le système de production) dans chacun de ses tuteurs intelligents, il reflète largement sa façon de concevoir l'architecture cognitive d'un humain. Ensuite, la théorie ACT (au début, puis la théorie ACT* (Anderson, 1983) et maintenant la théorie ACT-R (Anderson, 1993)) a déjà sa lettre de noblesse comme théorie d'apprentissage appliquée à plusieurs domaines différents. Enfin, la méthodologie d'implantation « traçage de modèle » (Model Tracing) proposée par Anderson peut servir de guide pour effectuer le design pédagogique des différents modules du système : le modèle de l'élève, le modèle tutoriel et l'interface de communication.

2.5.2 Histoire de la théorie ACT

ACT-R est une architecture cognitive développée par Anderson (1993), simulant les principes de base du système cognitif humain. ACT-R est la dernière version d'une longue série de travaux commençant avec le système HAM (pour Human Associative Memory) (Anderson & Bower, 1973), se poursuivant par ACTE (Anderson, 1976) et ACT. Cette théorie a d'abord évolué en ACT* (Anderson 1983), puis a été reprise et étendue par le système PUPS (pour

PenUltimate Production System) (Anderson & Thompson, 1989). Considérant que la cognition est rationnelle (Anderson, 1990) (i.e. qu'elle maximise la réalisation des buts en minimisant les coûts de traitements), Anderson révise sa théorie et produit finalement ACT-R (Anderson, 1993). La théorie ACT (pour Adaptative Control of Thought) est basée sur les systèmes de production. Son architecture prend en compte deux types différents de mémoire à long terme, une mémoire déclarative et une mémoire procédurale. Ces deux différents types de mémoire interagissent entre eux au travers d'une mémoire de travail, comme l'illustre la figure ci-dessous (Figure 2.2).

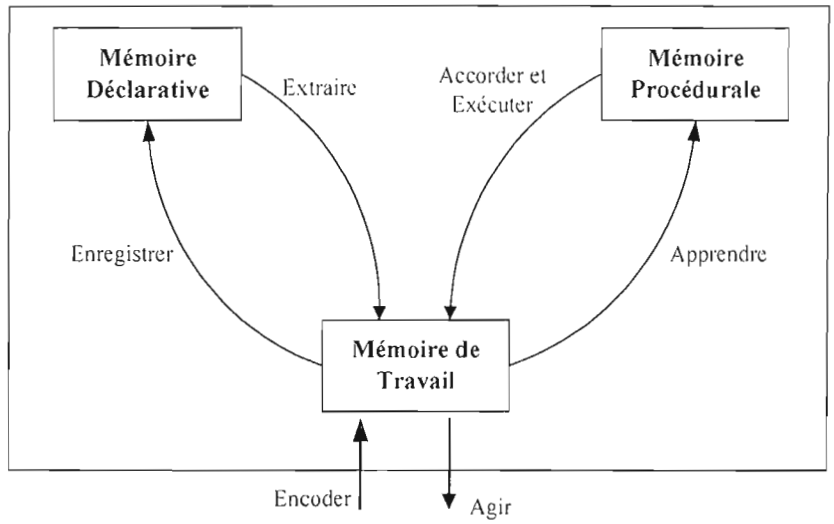


Figure 2.2 : Architecture de la théorie ACT* d'après Anderson (1989)

La théorie ACT a encore aujourd'hui un grand succès en tant que modélisation cognitive, car elle permet d'exprimer une large gamme de phénomènes (comme l'acquisition et le transfert de connaissances déclaratives et procédurales (Johnson, 1998). De plus, cette théorie a reçu plusieurs validations expérimentales, dans de très nombreux types de tâches, y compris dans le traitement du langage. Pour ne mentionner que quelques travaux récents : modélisation des mouvements oculaires en lecture (Salvucci, 2001), de l'analyse syntaxique (Emond, 1997), du traitement des métaphores (Budiu et Anderson, 2000), de l'apprentissage (Taatgen & Anderson, 2002), de la mémoire des phrases (Anderson, Budiu & Reder, 2001), etc. D'autres travaux ont récemment montré

qu'ACT est largement compatible avec les acquis de la neurophysiologie de la cognition humaine (Anderson & Lebiere, 1998b).

Nous nous référerons ici à sa version ACT-R (Anderson, 1993, 1996).

2.5.3 Principes essentiels de la théorie ACT

2.5.3.1 Distinction entre connaissance déclarative et procédurale

ACT est une théorie cognitive qui s'intéresse à la représentation des connaissances dans la mémoire d'un être humain et à la manière dont ces connaissances lui permettent de produire des comportements adaptés à la structure de son environnement. Deux types de connaissances sont à distinguer dans cette théorie : les connaissances déclaratives qui concernent le savoir (faits, théories, ...) et les connaissances procédurales qui concernent le savoir-faire (savoir calculer, ...) (Grégoire, 1999).

Les processus d'apprentissage de ces deux types de connaissances sont très différents. En effet, dans le cas des connaissances déclaratives, un fait peut être mis en mémoire après quelques secondes d'étude. Mais par contre, une connaissance procédurale ne s'acquiert qu'en l'appliquant, c'est un apprentissage par action.

La représentation de ce ces types de connaissances en mémoire est aussi différente. Les connaissances déclaratives sont représentées sous la forme de blocs d'informations « chunks » (Anderson & Lebiere, 1998a). Chaque chunk est organisé comme un réseau de liens, qui peut s'enrichir en fonction des expériences (figure 2.3). L'activation qui se produit au niveau d'un chunk s'étend à l'ensemble des chunks qui y sont liés et qui, dès lors, sont mis en mémoire déclarative.

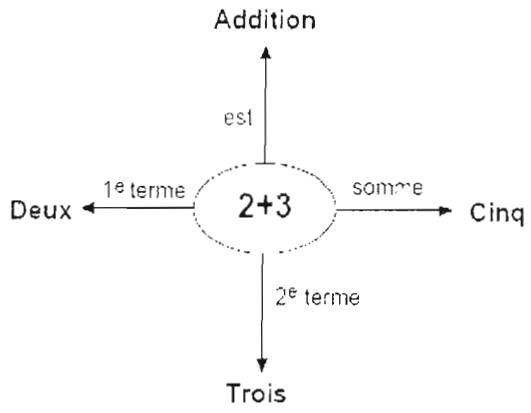


Figure 2.3. Exemple de chunk organisé en réseau (Anderson, 1983)

Quant aux connaissances procédurales, elles sont représentées sous forme de règles de production. Une règle de production est définie comme une association entre une condition (ou plusieurs conditions) et une action (ou plusieurs actions). Les conditions définissent les circonstances dans lesquelles les actions peuvent être produites. Les actions peuvent être motrices ou intellectuelles. Chaque règle de production représente une étape bien définie d'une activité cognitive. Par exemple si un sujet a le but d'effectuer une addition, toutes les règles dont la partie condition spécifie que le but est de faire une addition reçoivent de l'activation (Figure 2.4). Face à un problème, plusieurs règles de production peuvent être activées en parallèle. Parmi les règles examinées, celle exécutée est celle qui s'apparie le mieux à la structure de l'environnement perçue par le sujet.

SI
J'identifie une addition
Mon but est de trouver le résultat de cette addition

ALORS

1. Je repère le plus grand des deux termes
2. Partant de ce terme, je compte le nombre d'éléments qu'il y a dans le second terme
3. Je prends le dernier nombre compté comme le résultat de l'addition

Figure 2.4. Exemple de règle de production (Anderson, 1987)

Cependant, les connaissances déclaratives et procédurales sont intimement imbriquées. En effet, les conditions d'une règle de production sont définies sous

forme d'une structure déclarative. La capacité de l'élève à identifier des informations pertinentes dans son environnement et les connaissances utiles qu'il a pu accumuler sur cet environnement sont des éléments essentiels dans la sélection des règles de productions les plus adaptées (Grégoire, 1999).

2.5.3.2 Compilation de connaissances

Dans la théorie ACT comme dans toutes les autres versions suivantes d'ACT, une connaissance procédurale est acquise en trois étapes (Mayers, 1997) : d'abord, l'acquisition de cette connaissance sous forme déclarative, ensuite son interprétation pour produire le comportement et enfin la transformation de cette interprétation sous forme compilée.

Le mécanisme de compilation permet de transformer une connaissance déclarative en une connaissance procédurale (Anderson 1989). ACT propose deux manières de compiler les éléments mémorisés, par *composition* et par *procéduralisation*. La composition permet de comprimer plusieurs règles procédurales en une seule en court-circuitant les étapes intermédiaires. La procéduralisation transforme des règles procédurales générales en des règles spécifiques au problème à résoudre en éliminant les références aux éléments déclaratifs actuellement mémorisés.

2.5.4 Les huit principes d'ACT

ACT est une théorie de l'architecture cognitive (Anderson, 1983) au sens où elle propose un ensemble relativement complet de principes permettant de rendre compte de la construction et de l'organisation du système cognitif. Des systèmes tutoriels ont été développés autour de ce modèle cognitif. La construction de ces tuteurs a été guidée par un ensemble de huit principes basés sur la théorie d'ACT. Voici très brièvement ces principes (Anderson et al., 1983, 1995) :

- *Principe* 1 : Représenter le modèle (ou comportement) de l'élève comme un ensemble de règles de production < Si condition Alors action > ;
- *Principe* 2 : Mettre en évidence la structure hiérarchique de buts sous-jacente à la résolution de problèmes ;

- *Principe* 3 : Enseigner dans un contexte de résolution de problèmes (les compétences sont acquises par l'action) ;

- *Principe* 4 : Favoriser un niveau abstrait de représentation de connaissances pour la résolution de problèmes. En effet, de nombreux problèmes d'utilisation des connaissances proviennent d'un défaut dans leurs modalités de représentation. Les règles de productions doivent être abstraites et peuvent, par conséquent, s'appliquer à des situations parfois très différentes de la situation d'origine.

- *Principe* 5 : Réduire au minimum la charge de mémoire de travail (afin d'éviter les erreurs dues à une perte de celle-ci) ;

- *Principe* 6 : Réagir immédiatement aux erreurs ;

- *Principe* 7 : Adapter l'enseignement à l'expertise de l'élève ;

- *Principe* 8 : Faciliter la résolution par approximations successives.

2.5.5 Les tuteurs intelligents développés selon la théorie ACT*

Contrairement aux chercheurs en psychologie cognitive qui manifestent assez rarement de l'intérêt pour l'application de leur théorie dans le domaine de l'éducation, Anderson manifeste le souci constant de mettre sa théorie en pratique. Il a créé à cet effet, avec certaines personnes de l'université Carnegie Melon, le Pittsburgh Advance Cognitive Tutor Center (PACT) dont le but est de développer des tutoriels intelligents destinés à l'apprentissage (Anderson et al., 1990). Trois domaines ont conduit à des réalisations importantes : l'apprentissage du langage LISP avec LISP-TUTOR (Anderson & Reiser, 1985), la géométrie élémentaire avec GEOMETRY TUTOR (Anderson & al., 1986) et les manipulations algébriques avec ALGEBRA (Lewis & al., 1987 ; Anderson, 1992).

Tous les tutoriels développés par le PACT sont construits selon les huit principes de la théorie ACT. Chaque système est conçu à partir d'un ensemble de règles de productions nécessaires pour maîtriser une compétence complexe. De plus, un certain nombre de mal-règles est inclus dans le système qui sous-tendent les erreurs les plus fréquentes. Le tuteur peut inférer, à partir des erreurs commises par un élève en cours d'apprentissage, les mal-règles qu'il met en œuvre

Les expériences montrent que les tuteurs peuvent répondre à la majorité des problèmes que rencontrent les élèves en cours d'apprentissage. On constate également que les élèves qui utilisent ces tuteurs parviennent à maîtriser les nouvelles compétences deux fois plus vite que par des méthodes traditionnelles. Ceci n'est pas surprenant puisque les tuteurs favorisent un enseignement plus individualisé.

La théorie cognitive ACT d'Anderson permet une modélisation efficace des apprenants, facilite le processus d'apprentissage scolaire et résout certaines difficultés qui peuvent surgir en cours de route. En particulier, cette théorie permet aussi une évaluation diagnostique des difficultés d'apprentissage réellement informative et utile pour la remédiation.

2.5.6 Formes de rétroaction

Un principe important d'ACT* est la réaction immédiate aux erreurs de l'élève (principe 6). Il n'est pas possible de s'éloigner trop d'un chemin de preuve, même si les règles appliquées sont correctes. Anderson justifie cette intervention immédiate par le faite qu'il est plus facile pour l'élève d'analyser le cheminement mental qui l'a conduit à une erreur et d'effectuer la correction appropriée si cette erreur est toute fraîche au moment de la rétroaction. La rétroaction immédiate rend donc, selon Anderson, le tutorat plus facile, parce qu'on peut toujours retracer où en est exactement l'élève. Elle facilite aussi l'analyse des données, du fait qu'on peut contrôler les actions de l'élève, action par action et les règles de production impliquées dans chaque action. La rétroaction immédiate rend également l'apprentissage plus efficient, parce qu'elle empêche l'élève de perdre trop de temps à errer dans de fausses pistes.

Pourtant, plusieurs recherches (Rankin & Trepper, 1978 ; Sturges, 1978 ; Gaynor, 1981) démontrent que ce type de rétroaction immédiate n'est pas toujours celle qui favorise le plus l'apprentissage. Par exemple, l'étude de Gaynor (1981) propose qu'une rétroaction différée est plus convenable dans les cas d'apprentissage d'habiletés d'ordre supérieur.

Toutefois, Anderson lui-même reconnaît aussi qu'un certain nombre de problèmes sont causés par l'utilisation de ce type de rétroaction. Il y a d'abord le risque que cette rétroaction fournisse une réponse toute faite à l'élève au lieu d'inciter ce dernier à repenser ou à rechercher la bonne réponse (Anderson & al., 1987). Il semble en effet important, sur le plan de l'apprentissage, que l'élève passe à travers un processus cognitif qui génère la bonne réponse, et non qu'il la copie à partir de la rétroaction fournie. Il y a aussi le problème des erreurs qu'un élève pourrait lui-même corriger, si en lui en laissant seulement le temps. Une telle autocorrection est évidemment préférable quand elle survient spontanément plutôt qu'à la suite d'une rétroaction.

Cependant, Anderson (1987) a proposé plusieurs façons de remédier à ces problèmes. Parmi les solutions envisagées, il y a celle d'une rétroaction seulement après une réponse complète d'un élève, permettant ainsi à ce dernier de s'auto-corriger. Une autre solution, proposée cette fois par Corbett et al. (1999), consiste à laisser le contrôle de la rétroaction à l'élève.

La nature des messages de rétroaction émis par le tuteur doit en effet permettre à l'élève d'identifier précisément son erreur. Chaque message du tuteur doit spécifier quel élément critique du contexte est en cause dans la solution erronée formulée par l'élève. C'est à partir de cette information de type déclaratif que l'élève peut réussir sa compilation, c'est-à-dire établir le bon lien entre le but poursuivi dans un certain contexte et la production spécifique associée.

2.5.7 Traçage de Modèle

Pour faciliter l'implantation de sa théorie ACT*, Anderson a proposé une méthodologie appelée *traçage de modèle* (Model Tracing, Anderson, 1986). Ce modèle permet à un système d'apprentissage de simuler le processus dynamique d'un élève en situation de résolution de problèmes et utilise cette simulation pour interpréter le comportement de l'élève. Cette interprétation se fait par comparaison entre le modèle idéal et le modèle erroné. Le traçage de modèle peut tracer le chemin solution de l'élève en identifiant la règle qu'il applique à une des règles du modèle idéal ou erroné. Cela simplifie grandement la tâche de

diagnostic des erreurs, et l'on peut donner ainsi une aide adaptée (nous reviendrons plus en détail sur ce modèle de traçage dans le chapitre 3).

Dans son traçage de modèle, Anderson utilise deux modèles de contrôles : un *modèle de performance* pour suivre la performance d'un élève en cours d'exécution d'une tâche et un *modèle d'apprentissage* pour suivre son apprentissage (Anderson et al., 1990). Le premier modèle fournit des informations en temps réel sur l'état cognitif de l'élève pendant que ce dernier résout un problème. Le tuteur est donc capable d'intervenir durant le processus de résolution parce qu'il connaît les différents états à travers lesquels passe l'élève durant ce proccssus. Quant au deuxième modèle, il est utilisé pour déduire l'état des connaissances de l'élève à partir de sa performance à résoudre des problèmes. Cet état des connaissances, global par opposition à l'état dans l'espace problème en cours, peut être utilisé pour mieux interpréter le comportement de l'élève et surtout pour choisir les problèmes les plus susceptibles d'optimiser son apprentissage.

2.6 Conclusion

La psychologie cognitive n'apporte pas de solution à tous les problèmes posés par l'apprentissage. Son champ d'application privilégié est essentiellement limité à étudier chez le sujet humain, la nature et le format des connaissances, et l'architecture qui les organise. Elle apporte des modèles du fonctionnement cognitif qui permettent d'expliquer la façon dont l'apprentissage se construit dans la mémoire de l'apprenant et de comprendre les difficultés qui peuvent survenir au cours de cette construction (Grégoire, 1999).

Cependant, le choix d'un modèle cognitif n'est pas chose facile. Comme nous l'avons souligné dans ce chapitre, de nombreux modèles développés par les chercheurs sont à la fois trop limités dans leur étendue et trop détaillés dans les mécanismes qu'ils prennent en compte. Et peu de ces modèles sont réellement utilisables dans le cadre de l'EIAO.

La théorie d'Anderson est l'une des rares théories cognitives qui a fourni un cadre théorique général permettant d'éclairer la construction de l'ensemble des apprentissages. Elle est aussi l'une des seules théories qui a été fait l'objet d'applications rigoureuses dans le contexte scolaire. Cette théorie s'est révélée particulièrement efficace lorsqu'elle a été mise à l'épreuve de la réalité scolaire au travers de tutoriels d'apprentissage de l'algèbre, de la géométrie et de la programmation. Cette théorie se montre également très prometteuse au sujet de l'élaboration d'outils d'évaluation diagnostique.

Se référer à la théorie d'Anderson pour la conception d'un système de diagnostic a plusieurs implications. Tout d'abord, l'accent est mis sur les modes de représentations des connaissances en mémoire à long terme. Ensuite, il vient la nécessité d'analyser les compétences complexes en leurs composantes élémentaires que sont les règles de production. L'évaluation de ces règles de production doit porter non seulement sur la qualité des procédures mises en œuvre mais aussi sur la capacité à activer ces procédures dans des contextes déterminés.

CHAPITRE III

MÉTHODOLOGIE DE DÉVELOPPEMENT

Ce chapitre traite de la méthodologie de conception et d'élaboration du système TIDES (Tutoriel Intelligent pour le Diagnostic des Erreurs en Soustraction). Cette méthodologie met en relation les prescriptions et contraintes du modèle d'Anderson, celles des systèmes tutoriels intelligents, et celles de l'arithmétique comme domaine d'apprentissage en mettant l'accent sur ce qui doit orienter et guider la conception et l'élaboration du système TIDES.

Comme nous l'avons souligné à l'introduction de cette thèse, l'un des objectifs de cette recherche est appliqué la théorie cognitive explicite d'Anderson lors du développement du système TIDES et est abordé en premier lieu dans ce chapitre. Il s'agit ensuite de déterminer à quelles conditions ce modèle (d'Anderson) peut s'intégrer à un système d'apprentissage, en tant que système tutoriel intelligent et dont le domaine d'apprentissage est l'arithmétique.

3.1 Introduction

La problématique principale de notre recherche concerne la modélisation des apprenants à l'aide d'une nouvelle approche. Cette problématique nous amène à examiner une situation particulière : un diagnostic précis des compétences et connaissances arithmétiques des élèves au niveau primaire. Ces compétences et connaissances, clairement identifiées un peu plus loin dans ce chapitre et dans les deux chapitres suivants, sont essentielles à maîtriser au terme de l'enseignement primaire. C'est pour cette raison que nous avons choisi de travailler, lors de la

mise à l'essai du système TIDES, avec des élèves ayant terminé la troisième année primaire. Nous reviendrons plus en détail sur les critères de sélections de ces élèves dans le chapitre 5. Toutefois, les compétences et connaissances arithmétiques concernent notamment : les compétences logiques de raisonnement, la connaissance du système de numération, les procédures de comptage, les connaissances algorithmiques, les connaisances conceptuelles et les connaissances procédurales. Il faut souligner que ces connaissances sont intégrées dans le modèle de l'élève afin de faciliter l'évaluation diagnostique. Nous les présenterons plus en détail dans la section 4 de ce chapitre.

Cette problématique de modélisation de l'apprenant à l'aide d'un système tutoriel intelligent nous oblige de traiter la compétence arithmétique de façon très rigoureuse. Nous ne pouvons pas nous contenter de proposer une solution basée sur des aspects partiels des connaissances de l'apprenant, ni en donnant une description quantitative en termes de réussite ou d'échec, ni en répertoriant les erreurs de l'élève, ni même en fournissant l'ensemble des mal-règles qu'il utilise. Un diagnostic de surface ne suffit donc pas, constater simplement qu'un élève ne parvient pas à effectuer correctement une opération de soustraction se révèle, en général, insuffisant pour pouvoir agir de manière efficace. Une analyse profonde des connaissances et compétences des élèves est indispensable, et qui nécessite une modélisation cognitive qui prend en compte les multiples aspects des connaissances de ces apprenants.

Pratiquement, l'élaboration d'un tel système tutoriel intelligent nécessite d'abord la détermination des tâches susceptibles d'être accomplies par l'élève et qui se manifestent par différents comportements en situation d'apprentissage et ensuite la traduction de ces tâches en règles de production qui représentent bien les connaissances (correctes et incorrectes) utilisables lors de l'accomplissement de ces tâches. L'élaboration d'un tuteur intelligent nécessite également l'intégration de quatre modules qui assurent les différentes fonctions du système : le module de l'expert, le module de l'élève, le module pédagogique et l'interface.

Dans le cas présent, et compte tenu des objectifs de la recherche, cette démarche doit satisfaire trois soucis majeurs :

- appliquer le modèle d'apprentissage d'Anderson ;
- utiliser ce modèle pour la modélisation de l'apprenant ;
- adapter le modèle au domaine particulier de l'arithmétique et y incorporer un système de rétroaction.

Dans la section 2, la théorie cognitive d'Anderson est réexaminée, notamment au chapitre d'acquisition de connaissances, afin d'en dégager les éléments caractéristiques, ainsi que les contraintes ou prescriptions que ceux-ci imposent, d'une part à l'organisation et à la structuration des connaissances, d'autre part au modèle pédagogique d'un système d'apprentissage. Ces éléments constituent une partie de cadre d'analyse de chacune de deux questions traitées par la suite, soit l'applicabilité du modèle d'Anderson à un système d'apprentissage en tant que système tutoriel (section 3) et dont le domaine d'apprentissage est l'arithmétique cognitive (sections 4 et 5).

3.2 Acquisition d'habiletés et théorie ACT-R

Comme nous l'avons souligné dans le chapitre précédent, le modèle d'Anderson est un modèle cognitif fondé sur des hypothèses quant à l'organisation et à l'acquisition d'habiletés cognitives complexes. Il est décrit dans la théorie d'apprentissage ACT-R (Adaptative Control of Thought, et R signifie *Rational*) (Anderson, 1993) et dans la méthodologie d'implantation du traçage de modèle (Model Tracing) (Anderson, 1995).

Ce modèle d'apprentissage a été retenu pour guider l'élaboration du système TIDES. D'abord, la théorie ACT-R s'impose par le fait qu'elle tente d'expliquer l'ensemble du processus d'acquisition d'habiletés intellectuelles complexes dans un système d'EIAO. En outre, cette architecture a reçu de très nombreuses validations expérimentales et a encore aujourd'hui un grand succès en tant que modélisation cognitive (Anderson et Lebiere, 1998, Taatgen et al., 2004). Enfin,

une bonne description de la théorie est disponible à travers différents volumes et articles de revue.

Le modèle d'Anderson est défini ici, comme pour d'autres auteurs (Wenger, 1987 ; Orey et Burton, 1990 ; Mayers, 1997), à partir des quatre éléments distinctifs suivants :

1. les règles de productions peuvent être utilisées pour représenter les connaissances à acquérir ;
2. l'acquisition de connaissances par l'apprenant d'abord sous une forme déclarative, puis la conversion de ces connaissances en procédures par l'exécution des tâches ;
3. les modalités d'intervention en cas d'erreur ;
4. l'utilisation de la mémoire de travail de l'élève doit être minimisée.

Dans les sous-sections suivantes (sous-sections 2.1 à 2.4), ces éléments seront explicités afin d'identifier les contraintes et prescriptions qu'ils imposent (sous-sections 2.5) à la conception et au développement d'un système tutoriel intelligent à base de système de productions et dont le domaine d'apprentissage est l'arithmétique, et enfin d'élaborer un premier jet de modèle pédagogique du système (sous-sections 2.6).

3.2.1 Modélisation des connaissances et système de production

La modélisation des connaissances est un sujet délicat à aborder car il n'existe pas un unique formalisme de modélisation accepté par tous.

Les connaissances sur un domaine peuvent être empiriques ou théoriques. Les connaissances empiriques représentent les résultats d'expériences ou les exemples de cas pratiques ; elles n'ont pas encore subi de transformations en vue d'obtenir une théorie plus générale sur le domaine. Quant aux connaissances théoriques, elles modélisent les connaissances sur un sujet à l'aide d'une théorie correspondant au problème posé (Osório, 1998).

Les connaissances empiriques et théoriques doivent être transformées dans une représentation compatible avec les outils informatiques de raisonnement automatique afin de pouvoir être exploitées. Parmi les différentes approches de modélisation de connaissances, on trouve les systèmes de productions.

3.2.1.1 Systèmes de productions

Les systèmes de productions constituent la base de la majorité des modélisations cognitives existantes en résolution de problème. Ces systèmes se sont montrés bien adaptés pour simuler les stratégies de résolution de problème (Klahr & Robinson, 1981). Ils gèrent un ensemble de connaissances exprimées sous forme de règles de production grâce à un moteur d'inférences extérieur aux connaissances. Le contrôle du processus de résolution passe d'une production à une autre quand les actions produites par le déclenchement d'une production créent les conditions nécessaires à l'activation d'une autre production. L'automaticité est une caractéristique importante de ce type de système. En effet, selon Gagné (1985a), le caractère d'automaticité des systèmes de productions a comme conséquence que, dans un système humain de traitement de l'information, les connaissances procédurales (représentées par les règles de production) occupent peu d'espace dans la mémoire de travail, et par conséquent une autre activité peut être accomplie consciemment en même temps.

3.2.1.2 Règles de production dans la théorie ACT-R

Dans la théorie ACT-R d'Anderson, les fonctions cognitives sont représentées par un ensemble de règles de production. Ces règles constituent, ce que Anderson appelle, « le modèle idéal » de la façon dont un élève doit résoudre divers problèmes. Si on y ajoute des règles représentant les façons correctes ou incorrectes pour cet élève d'arriver à des solutions de ces problèmes différentes de celle du modèle idéal, on obtient le *modèle de performance* (Anderson, 1983). Dans les systèmes tutoriels intelligents adoptant le formalisme des règles de production, comme le cas du système TIDES, cela se traduit par la modélisation des stratégies de solutions correctes et incorrectes en règles correspondantes :

- des *règles de références* : celles utilisées par le système pour obtenir la solution correcte, c'est-à-dire utilisées par le modèle idéal ;
- des *règles équivalentes* : celles menant à une solution équivalente à celle du système;
- des *règles généralisantes* : celles menant à une solution correcte mais plus générale que ce qui est demandé à l'élève ;
- des *mal-règles* : celles utilisées pour interpréter le comportement d'un élève qui commet des erreurs en tâchant de résoudre le problème.

Une règle de production, selon ACT-R, est une association entre une ou plusieurs conditions et une ou plusieurs actions. Les conditions définissent les circonstances dans lesquelles les actions peuvent être produites.

Les règles de production dans le système TIDES, d'ailleurs comme dans les systèmes d'Anderson, sont utilisées dans le cadre d'une *structure de buts explicite*. De fait, chaque règle est un module qui représente une étape bien définie d'une activité cognitive et se réfère explicitement à un but dans ces systèmes. Cela permet au tuteur de situer ses messages en fonction d'un but clair et précis. Cela facilite également la décomposition d'un problème en une hiérarchie explicite de tâches et de sous-tâches. Cela permet enfin de regrouper toutes les règles partageant un même but en une *unité de connaissances* qui définit en fait toutes les façons identifiées d'atteindre, correctement ou incorrectement, le but en question en utilisant la stratégie du tuteur ou toute autre stratégie (Anderson, 1983).

Face à un problème, plusieurs règles de production peuvent être activées en parallèle. Les règles finalement mises en œuvre sont celles qui s'apparient le mieux à la structure de l'environnement perçue par le sujet.

3.2.1.3 Traçage de modèle

Comme nous l'avons souligné au chapitre 2, la théorie ACT-R d'Anderson s'appuie sur la méthodologie dite du traçage de modèle (Model Tracing), discutée

également en 4.2 sous l'angle de l'arithmétique, qui permet de guider le design pédagogique des différents modules du système TIDES, et notamment le modèle de l'expert et le modèle de élève. Le traçage de modèle consiste à simuler dynamiquement, à l'aide d'un système de production, la résolution effectuée par l'élève et à utiliser cette simulation pour interpréter son comportement. Ceci implique de modéliser convenablement l'élève à deux niveaux. D'une part, en utilisant un *modèle de performance* pour suivre l'état de la solution de l'élève à l'intérieur d'une tâche et d'autre part, en utilisant un modèle d'*apprentissage* pour déterminer l'état des connaissances de l'élève au fur et à mesure qu'il accomplit les différentes tâches. Le mécanisme permettant le passage du modèle de performance au modèle d'apprentissage est détaillé à la section 4.

La méthodologie du traçage de modèle a montré la possibilité d'interagir avec l'apprenant en pas à pas au cours de la résolution de problème et d'interpréter son comportement en termes de règles cognitives. Pour Anderson (1992), le succès de cette méthode provient d'abord de la création d'un modèle cognitif adéquat et de le communiquer à l'apprenant, ensuite, de la minimisation du temps d'apprentissage et enfin, du contrôle et la vérification de l'acquisition des connaissances (règles individuelles) (Bruillard, 1997).

3.2.2 Acquisition de connaissances procédurales

3.2.2.1 Apprentissage en contexte de résolution de problème

Dans la théorie ACT-R, l'acquisition des connaissances déclaratives précède toujours celle des connaissances procédurales. En effet, pour mettre en œuvre des connaissances procédurales, il faut d'abord connaître les conditions d'exécution de la procédure qui sont fournies sous la forme de connaissances déclaratives. Ces connaissances déclaratives trouvent en principe leur origine dans l'encodage des stimuli issus de l'environnement qui sont reconnus au niveau de la mémoire de travail puis transférés en mémoire à long terme. Quant aux connaissances procédurales, elles résultent d'un mécanisme de transformation des connaissances déclaratives dans lequel l'analogie et l'imitation jouent un rôle essentiel.

D'ailleurs, le passage d'un état déclaratif à un état procédural d'une connaissance sous l'effet d'un apprentissage en contexte de résolution de problèmes est appelé par Anderson (1989) *compilation*. Cet apprentissage est efficace quand il facilite à l'élève d'encoder à la fois l'applicabilité de la connaissance ainsi que sa pertinence en rapport avec le but poursuivi.

Pour présenter la connaissance en contexte de résolution de problèmes, le tuteur doit être capable de communiquer avec l'élève en utilisant le vocabulaire des différentes tâches requises pour atteindre le but poursuivi. Ces tâches constituent l'espace-problème[6] (Newell et Simon, 1972) à l'intérieur duquel des règles de production doivent être sélectionnées pour suivre et guider l'élève.

Le mécanisme d'apprentissage en contexte de résolution de problèmes peut prendre deux formes : *procéduralisation* et *composition*. Dans sa forme procéduralisation, une connaissance déclarative est transformée en une procédure spécifique s'appliquant dans une certaine classe de cas. Dans sa forme composition, plusieurs procédures utilisées sont réunies, pour atteindre un but donné, en une procédure unique qui combine les effets des procédures de départ. On peut concevoir le mécanisme de composition comme la combinaison de plusieurs règles existantes qui a comme résultat la réduction du nombre d'étapes utilisées par l'élève pour résoudre un problème.

3.2.2.2 Traçage de connaissances

Le *traçage de connaissances* (Corbett et Anderson, 1995) constitue l'aboutissement du *traçage de modèle* et dans lequel on utilise le *modèle de performance* pour tracer *le modèle d'apprentissage*. Il consiste à mettre en œuvre un ensemble d'hypothèses sur la façon dont l'état de connaissances d'un élève change en fonction de son comportement pendant qu'il résout un problème. Pour ce faire, on assigne d'abord à chaque règle une probabilité initiale que l'élève a appliqué correctement cette règle ; cette probabilité initiale est fondée sur

[6] L'espace-problème (Newell & Simon, 1972) est une construction mentale composée d'un ensemble d'états et d'opérateurs permettant de passer d'un état à un autre (ce sont les règles de production). Un problème est défini par la donnée d'un état initial et d'un état final. Une solution est alors un chemin dans cet espace entre les deux états déterminé par une suite d'opérateurs.

l'expérience et rajustée au besoin, à mesure que de nouvelles données statistiques deviennent disponibles (corbett et Trask, 2000, Beck et Sison, 2004). Ensuite, cette probabilité évolue de manière différente pour chaque élève en fonction de la performance manifestée au cours de son apprentissage.

3.2.3 Modes d'intervention en cas d'erreur

Pour que le mécanisme de compilation soit efficace, il doit permettre à l'élève d'appliquer correctement les règles de production spécifiques pour atteindre un but donné. Cette façon de faire n'est pas compatible avec le fait de laisser l'élève explorer des chemins de solution erronée. C'est pourquoi la théorie d'Anderson, du moins dans sa formulation originale, suggère qu'une rétroaction doit être rapide, sinon immédiate (Dion et Lelouche, 1992). Par conséquent, le tuteur peut formuler un message de rétroaction qui identifie l'élément critique en cause dans l'erreur détectée de l'élève sans avoir à se référer à l'ensemble de la solution courante. Si par contre, la rétroaction était retardée, il serait beaucoup plus difficile au tuteur d'identifier le message de rétroaction le plus approprié et de n'afficher que celui-ci, puisqu'il faudrait procéder à une analyse de toute la solution en cours formulée par l'élève.

Les messages de rétroaction émis par le tuteur doivent en effet permettre à l'élève d'identifier clairement ses erreurs. Chaque message du tuteur doit spécifier, en effet, quel élément critique (ou éléments critiques) du contexte est en cause dans la solution erronée formulée par l'élève. C'est à partir de cette information de type déclaratif que l'élève peut réussir sa compilation, c'est-à-dire établir le bon lien entre le but poursuivi dans un certain contexte et la production spécifique associée.

3.2.4 Gestion de la mémoire de travail de l'élève

La mémoire de travail de l'élève, utilisée pour le stockage temporaire d'informations, a une double limitation, de capacité et de durée. Pour Anderson, le caractère limité de la mémoire de travail de l'élève constitue une sérieuse contrainte à sa capacité d'apprendre. En conséquence, un système tutoriel doit

tenir compte de cette contrainte et réduire au minimum la charge imposée à la mémoire de travail de l'élève.

Étant donné les capacités cognitives limitées de l'élève, il peut y avoir incompatibilité entre les activités de résolution de problèmes et celles d'apprentissage si ses capacités intellectuelles sont surchargées par la mécanique complexe de résolution de problèmes. Cette tendance peut ainsi interférer sur l'apprentissage et retarder l'acquisition des connaissances (Sweller et Chandler, 1991)

Afin de permettre à l'élève de se concentrer sur les éléments critiques du problème et d'accorder une part de son attention à l'analyse et à la solution du problème et ainsi d'apprendre rapidement, Anderson suggère qu'il faut limiter au maximum tous les éléments qui peuvent surcharger la mémoire de travail de l'élève. Cette considération est particulièrement importante en ce qui a trait à la conception de l'interface du système. En effet, le choix de la conception doit être effectué selon les objectifs de la tâche et doit permettre à l'élève de se concentrer sur l'apprentissage de cette tâche et d'améliorer sa compréhension.

3.2.5 Contraintes d'application du modèle d'Anderson

À partir de la discussion précédente sur les caractéristiques du modèle d'Anderson, il est maintenant possible de dégager un certain nombre de prescriptions ou contraintes touchant la construction de la base de connaissances et le design pédagogique d'un système tutoriel intelligent qui vise à constituer une application de ce modèle.

La base de connaissances de tel système doit être constituée d'un ensemble de règles effectivement organisées, structurées et exploitées de la façon suivante :

- Chaque règle de production représente une connaissance à acquérir.
- Ces règles sont structurées de façon hiérarchique en fonction du but explicite de chacune.

- Ces règles constituent les différentes étapes du processus de résolution de problèmes.
- Ces règles permettent de suivre le raisonnement d'un élève pendant le processus de résolution.
- Chaque règle peut être accompagnée d'un certain nombre de mal-règles qui sous-tendent les erreurs d'un élève.
- Ces règles facilitent l'intervention du tuteur auprès de l'élève afin de l'aider dans son raisonnement et le guider en cas d'erreur.

Le modèle d'Anderson (Anderson, 1987) représente aussi un guide intéressant pour élaborer un système tutoriel intelligent. À ce contexte, il suggère un ensemble de démarches qui peuvent être appliquées pour déterminer le design pédagogique du système.

- Au début, le système doit présenter déclarativement à l'élève les connaissances à acquérir.
- Le système doit fournir à l'élève un environnement d'apprentissage en contexte de résolution de problèmes.
- Le tuteur doit être capable d'interagir avec l'élève en utilisant un vocabulaire clair des différentes sous-tâches requises pour atteindre le but poursuivi.
- Le système doit être capable de sélectionner chaque nouvelle tâche en fonction des connaissances courantes de l'élève et des connaissances requises pour accomplir chaque tâche possible.
- L'intervention du tuteur doit être rapide et immédiate lorsque le système détecte une erreur dans la solution formulée par l'élève.
- Cette intervention du tuteur prend la forme de messages explicatifs qui spécifient les éléments critiques en cause dans la solution erronée de l'élève et permettent une interprétation correcte du problème à résoudre.

- Le système doit assurer une bonne gestion de la mémoire de travail de l'élève pendant le processus d'apprentissage.
- L'élève doit pouvoir formuler facilement sa solution en se libérant le plus tôt possible des préoccupations d'écriture.
- Pendant le processus d'apprentissage, l'élève doit continuellement disposer à l'écran des informations essentielles qui l'aideront à accomplir sa tâche.
- L'élève doit également avoir accès en tout temps, aisément et sur sa demande, à un ensemble d'autres informations utiles.

D'autres contraintes sont décrites concernant l'interface d'un système tutoriel intelligent :

- Être aussi facile que possible à utiliser, c'est-à-dire minimiser le nombre d'actions nécessaires pour communiquer avec le tuteur.
- Comporter une structure ou représentation aussi congruente que possible avec la structure sous-jacente des problèmes à résoudre.
- Être hautement interactive et fournir le plus d'informations possibles sur les étapes intermédiaires de résolution.
- Pouvoir faire varier la charge de la mémoire de travail en donnant accès aux informations reliées au problème.

3.2.6 Design pédagogique du système TIDES

Avant de discuter les aspects méthodologiques liés à l'application du modèle d'Anderson à notre système et au domaine de l'arithmétique, un premier design du système TIDES peut être effectué. Même si ce premier jet est exprimé en termes généraux, il fournit quand même un cadre adéquat pour aborder notre problématique.

3.2.6.1 Éléments à introduire dans le système

La conception des différentes fonctions du système TIDES est élaborée à partir des prescriptions et contraintes du modèle d'Anderson (§2.5). Chacune de ces fonctions constitue un élément essentiel du cadre d'apprentissage.

- **Sélection des tâches**

La sélection des tâches est une fonction importante qui doit être introduite dans le système TIDES. Cette introduction doit se faire en fonction du niveau de difficulté cognitif de la tâche à réaliser compte tenu de l'état des connaissances de l'élève.

- **Environnement d'apprentissage**

L'apprentissage en situation de résolution de problèmes doit se dérouler dans un environnement de travail adapté aux besoins de l'élève. L'interface de résolution est une zone de travail (ou bloc de résolution) bien organisée et qui permet à l'élève de formuler sa solution de façon claire comme s'il travaille avec l'outil papier-crayon. Un tel environnement de travail permet de libérer la mémoire de travail de l'élève qui peut davantage se concentrer sur la résolution du problème courant.

- **Aide à la formulation des solutions**

Afin d'axer les efforts de l'élève sur les stratégies de résolution plutôt que sur l'écriture et l'édition des solutions, plusieurs informations utiles sont accessibles à l'élève dans un format et selon des modalités d'accès qui évitent d'encombrer inutilement sa mémoire de travail.

- **Double rétroaction**

Comme nous l'avons vu dans le chapitre précédent, le modèle d'Anderson prévoit une intervention rapide et immédiate après la détection d'une erreur dans la solution formulée par l'élève. Cette intervention prend la forme de commentaires explicatifs qui identifient l'élément critique en cause dans la solution erronée. À cet égard, le système développé inclut deux formes de rétroaction.

Dans la première rétroaction, et à la demande de l'élève, le système émet un message d'analyse explicite, dans le cas où la solution de l'élève est erronée. L'élève peut en tout temps demander l'exécution de l'analyse de sa solution partielle ou complète. Ensuite vient la deuxième rétroaction, dans laquelle le système émet cette fois-ci un message de diagnostic, toujours à la demande de l'élève, pour que l'élève sache les causes réelles de ses erreurs, ce qui constitue une double rétroaction.

Comme on a pu le constater, cette double rétroaction présente plusieurs points communs avec la stratégie adoptée dans certains systèmes d'Anderson, notamment la version 2 de Lisp-tutor (1989). En effet, dans les deux cas, la rétroaction s'exécute en deux temps et incorpore un message plus explicite. Dans le système d'Anderson, le premier message n'apparaît qu'à la suite d'une erreur de l'élève et prend la forme d'un message d'erreur affiché en caractère gras. Dans le système TIDES, le premier message apparaît dès que l'élève juge sa solution terminée et se traduit par un message simple qui affiche le comportement de l'élève.

3.2.6.2 Déroulement d'une séance d'apprentissage

- **Phase instruction**

Dans le modèle d'Anderson, et notamment dans le Lisp-Tutor, la phase instruction correspond à la présentation de problèmes résolus et commentés dans un cahier avant la séance d'apprentissage proprement dite. En effet, les travaux sur l'analogie (Gick et Holyoak 1980, 1983 ; Anderson, 1993) montrent qu'il faut présenter à l'élève plusieurs exemples pour que celui-ci puisse construire une sorte de structure abstraite de connaissances, appelée *schéma*. Les élèves qui tirent profit de l'analyse de ces exemples, selon Renkl (1997), sont ceux qui peuvent rendre explicite les principes du domaine utilisés ainsi les sous-buts à atteindre, et ceux aussi qui anticipent les étapes du raisonnement.

Toutefois, pendant l'analyse de ces exemples (corrigés-types), certains élèves cherchent à anticiper les étapes de la résolution et se manifestent ainsi des attentes

déçues (Hammond, 1990) lorsque les corrigés ne correspondent pas à leurs prévisions. D'autres élèves découvrent comment résoudre certaines étapes spécifiques d'un problème et construisent ainsi des *cas*, en mémorisant cette résolution spécifique, ils peuvent résoudre certains problèmes proches de ces exemples, mais généralement n'arrivent pas à résoudre des problèmes dont l'aspect contextuel est différent (Guin, 1997).

D'ailleurs, la phase instruction doit insister sur les points critiques du problème et de sa résolution. Il ne s'agit pas ici de présenter seulement un ensemble d'exemples divers à partir desquels l'élève pourrait induire des règles de généralisation et discrimination. À cet égard, la théorie ACT-R d'Anderson privilégie plutôt d'indiquer clairement à l'élève les points critiques dans la résolution d'un problème.

L'approche qui a été retenue dans le système développé est essentiellement celle d'Anderson. En effet, suivant l'architecture du système TIDES, dans la phase instruction, qui doit précéder l'étape de mise à l'essai, un bref exposé est présenté oralement aux élèves rappelant la structure du système, son mode de fonctionnement, les méthodes de contrôle, son interface et ses objectifs. Ensuite, une démonstration suit dans laquelle deux problèmes sont proposés et résolus avec eux selon le mode de travail du système (voir chapitre 4). Une fois la phase instruction terminée, les élèves sont invités à procéder de la même manière que lors de la démonstration.

- **Étapes de déroulement d'une séance d'apprentissage**

Après la phase d'instruction, un scénario de six étapes est adopté pour une séance d'apprentissage.

Étape 1. Assignation d'une tâche

En fonction de l'état des connaissances de l'élève et en fonction du niveau de difficulté associé aux différentes tâches, le tuteur sélectionne une tâche (figure 3.1) à accomplir et présente cette tâche à l'élève avec certaines contraintes à respecter (notez que l'élève peut proposer sa propre tâche).

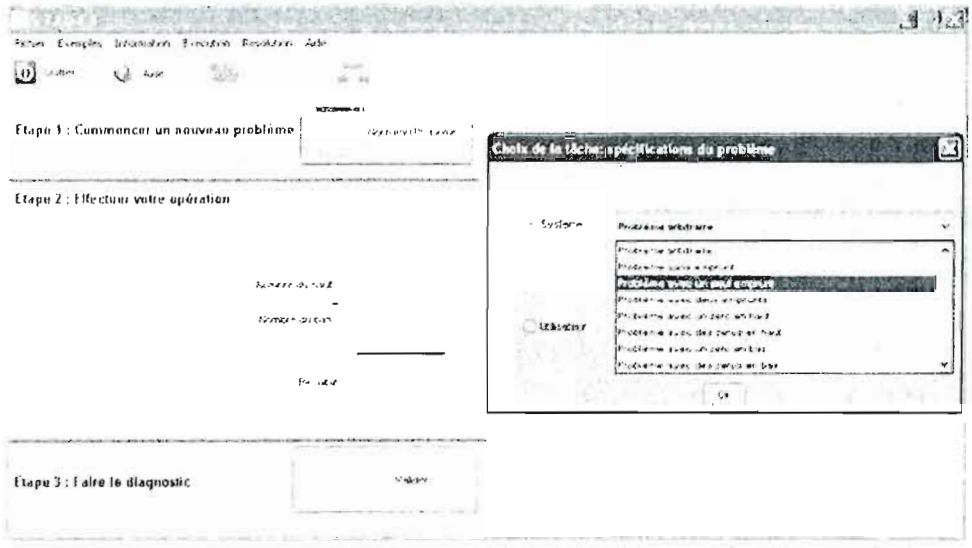


Figure 3.1 Assignation d'une tâche

Étape 2. Résolution du problème

L'élève propose une solution en travaillant au niveau de l'interface (figure 3.2). Il peut formuler sa solution de façon libre et sans aucune contrainte. De plus, en vue de simplifier le mode de résolution, c'est en pointant sur une case destinée à la solution du problème que l'élève arrive à donner sa solution. La suite de ses sélections est consignée à l'écran au fur et à mesure. En cours de résolution, l'élève peut à tout moment procéder au changement de sa décision.

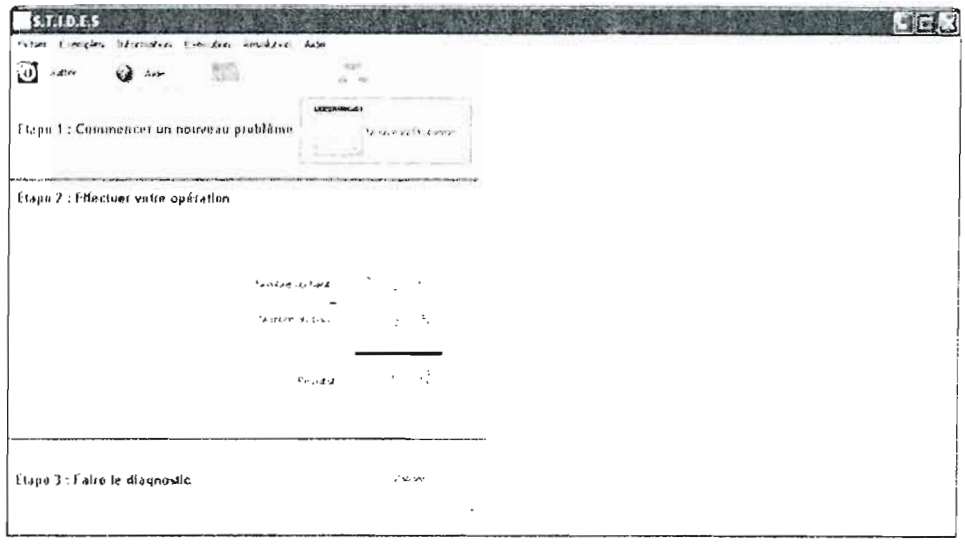


Figure 3.2 Formulation de la solution

Étape 3. Analyse de la solution

Lorsque l'élève termine sa solution, il le signale au système. Le tuteur analyse la solution proposée par l'élève et réagit par un message initial selon que la solution est correcte ou erronée. À la demande de l'élève, le système émet un message d'analyse explicite (figure 3.3). Cette intervention du tuteur correspond en fait à la première phase de la rétroaction.

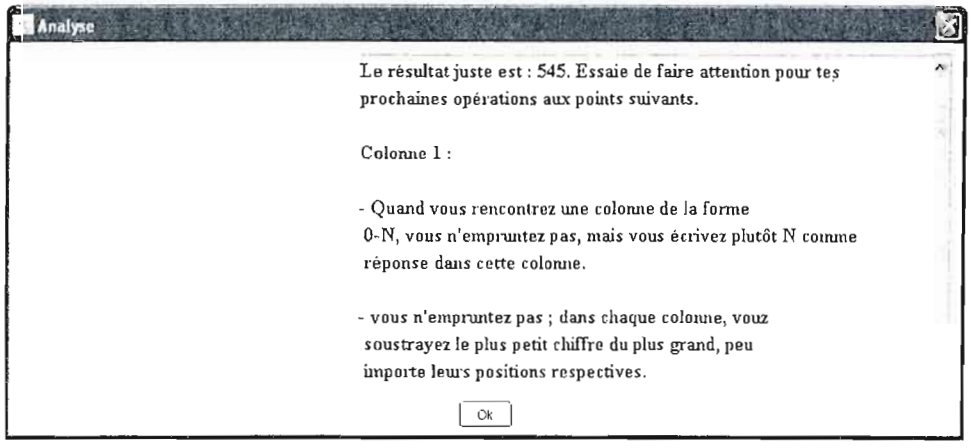


Figure 3.3 Message d'analyse lié aux erreurs précédentes

Si l'élève a commis plus qu'une erreur (comme dans l'exemple précédent), le système l'invite à résoudre certains problèmes tests qui sont liés à sa conception

erronée (figure 3.4). Devant ce message, l'élève peut alors choisir de demander un diagnostic plus explicite pour savoir exactement les causes de ses erreurs. Ce qui nous ramène à l'étape suivante.

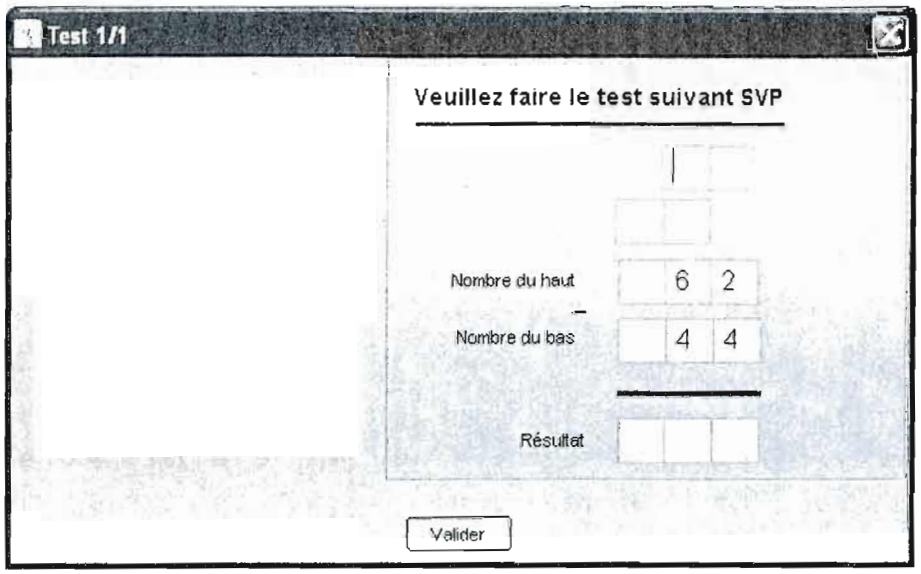


Figure 3.4 Problème test

Étape 4. Diagnostic de la solution

Lorsque l'élève demande un diagnostic plus profond de ses erreurs, ce qui amène alors le tuteur à s'exécuter (figure 3.5) ; cette intervention correspond à la deuxième phase de la rétroaction : le message affiché explicite l'évaluation de la solution proposée, indiquant pourquoi elle est incorrecte ou non-diagnosticable.

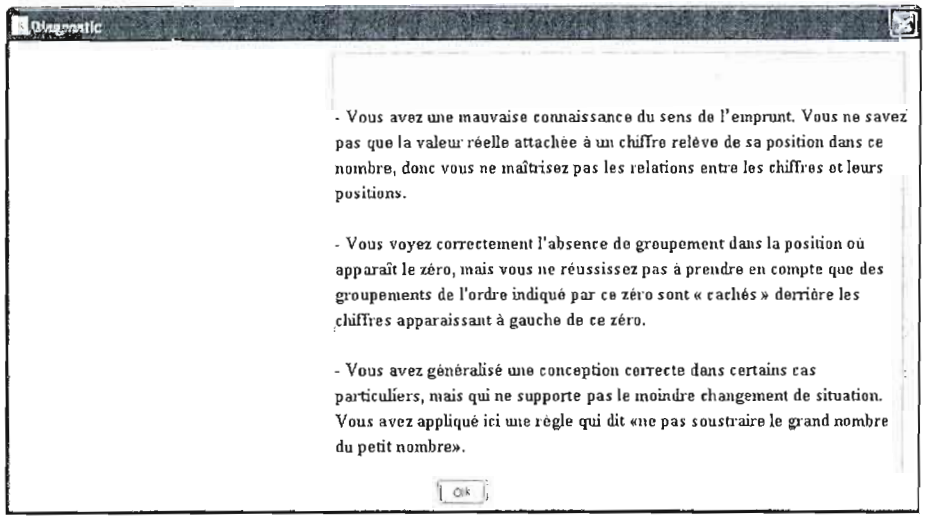


Figure 3.5 Message de diagnostic indique les causes des erreurs

De plus, l'élève peut utiliser « l'exécution graphique (graphe de solution) » pour comprendre mieux les causes de ses erreurs (voir chapitre 4 pour plus de détails).

Étape 5. Fin de la tâche séance

Lorsque la solution de l'élève a été analysée et diagnostiquée, l'élève peut déclarer la session terminée ; on revient alors à l'assignation d'une nouvelle tâche s'il le désire.

Étape 6. Fin de la séance

C'est la fin de la séance d'apprentissage avec l'affichage des messages globaux appropriés.

3.3 Modèle d'Anderson et système tutoriel intelligent

Nous avons abordé dans la section précédente la question de l'applicabilité du modèle d'Anderson à un système tutoriel intelligent. Pour compléter l'analyse dans ce cadre, il faut souligner certaines caractéristiques des systèmes intelligents. Ces informations, combinées à celles présentées dans la première étape sur le modèle d'Anderson (section 2), constituent le cadre d'analyse pour étudier spécifiquement la conception et le développement du système TIDES en tant que système tutoriel intelligent incorporant le modèle d'apprentissage d'Anderson.

3.3.1 Système tutoriel intelligent

Les systèmes tutoriels intelligents se répartissent en catégories distinctes selon le but qu'ils permettent d'atteindre et l'importance de l'aide a apporté (Paquette et al., 1994) :

- les systèmes qui se contentent de donner une simple rétroaction à l'élève. C'est le cas, par exemple, des systèmes experts ; aucune connaissance de l'élève ou connaissance tutorielle n'est requise, et les connaissances du domaine n'incluent que celles de l'expert. Le système ne contient pas de véritable module d'explication, il affiche seulement quelques conséquences des actions de l'élève d'une façon qui devrait l'aider à cheminer vers une solution ;

- les systèmes qui agissent, sur demande de l'élève, pour fournir une explication sur le rôle d'une composante de l'interface qu'il utilise pour réaliser sa la tâche Un exemple d'une telle approche se retrouve dans le module d'explication d'un système expert ;
- les systèmes qui se veulent des conseillers actifs. Ils accompagnent l'élève dans ses activités pour lui fournir, non seulement de l'aide demandée, mais des conseils lorsqu'il leur semble que celui-ci a des difficultés sérieuses avec la tâche en cours ;
- enfin, dans certains systèmes tutoriels intelligents, l'initiative est entièrement laissée à l'élève et le système n'exerce pas un guidage serré de ce dernier. Son intervention est minimale. Elle devient nécessaire dans le cas où l'élève s'éloigne trop du chemin de la solution ou si l'élève demande lui-même cette intervention pour corriger ses erreurs.

3.3.2 Architecture d'un système d'EIAO

Les recherches en EIAO ont pour objectif général de concevoir des systèmes intelligents dont l'utilisation favorise des apprentissages chez les apprenants. Une manière de favoriser ces apprentissages est de prendre en compte les caractères spécifiques de chaque apprenant (Jean, 2000). L'identification de ces caractères spécifiques est une tâche importante qui peut être réalisée par l'obtention d'informations sur les connaissances ou compétences de l'apprenant, d'ailleurs c'est l'objet des courants de recherche qui s'intéressent à la modélisation de l'apprenant. Celle-ci se fait à partir des quatre modules distincts qui composent un système d'EIAO, à savoir (Py, 2001)(voir chapitre 1 section 2) :

- *Le module expert du domaine* : les connaissances reliées à la discipline enseignée et au savoir de la résolution des problèmes ;
- *Le module de l'élève* : le profil évolutif de l'élève au fil de l'interaction avec le système ;
- *Le module tutoriel* : l'ensemble des stratégies pédagogiques et d'intervention tutorielle pour gérer les sessions d'enseignement ;

- *Le module de communication* : l'interface élève-système pour faciliter l'apprentissage.

Il faut souligner que les systèmes développés dans la cadre d'EIAO ne sont pas tous dotés de connaissances suffisantes leur permettant de générer de façon efficace les solutions aux problèmes qu'ils posent aux élèves. Par exemple, les connaissances intégrées dans le système Sophie-I ne lui permettent pas de résoudre les problèmes posés à l'élève sur les circuits électroniques (Brown & al., 1975). De plus, même si on trouve un système capable de générer les solutions aux problèmes qu'il pose à un élève, cela n'implique pas automatiquement qu'il va utiliser les mêmes étapes que cet élève pour y arriver.

3.3.3 Base de connaissances du système à développer

Nous avons déjà mentionné dans l'introduction du chapitre 2, que le design du système TIDES s'appuie sur un modèle cognitif d'apprentissage, l'ACT-R d'Anderson. Cette caractéristique entraîne que la forme des connaissances utilisées doit être adaptée au domaine de l'arithmétique en tant que domaine d'apprentissage. Il faut cependant vérifier que le grain[7] utilisé, est nécessairement adapté au système en tant que système pédagogique capable de suivre le raisonnement d'un élève.

D'ailleurs, il faut bien noter que l'organisation d'un système de représentation de connaissances pour un domaine spécifique conduit également à déterminer sinon la structure idéale, du moins une structure apte à faciliter la résolution des problèmes de ce domaine.

Contrairement à un système expert classique, le modèle d'Anderson s'applique à un système d'EIAO, il vise à ce que l'apprenant *acquière* les connaissances du système et non pas seulement à les consulter. Toutefois, la base de connaissances d'un tel système présente plusieurs caractéristiques intéressantes (Paquette, 1999). Tout d'abord, cette base est considérée comme une façon adéquate de combiner le

[7] Le terme est utilisé dans l'EIAO pour désigner la taille à laquelle les connaissances sont considérées (Py, 1998).

savoir accumulé dans un domaine donné, quel que soit le type de savoir. De plus, la modularité de cette base de connaissances rend le système capable de suivre l'évolution de connaissances de l'apprenant ; elle peut aussi rendre plus facile les modifications à apporter à la base de connaissances. Ensuite, un système doté d'un moteur d'inférence, peut manipuler les connaissances de base ou mettre en évidence des contradictions entre celles-ci et certaines connaissances nouvelles qu'on lui apporte. Enfin, un système peut « expliquer » ses démarches, à la demande de l'apprenant, en lui explicitant les règles qu'il a utilisées pour produire par exemple, une solution particulière. L'apprenant, à ce moment peut alors évaluer la pertinence des règles et du raisonnement du système et ainsi décider dans quelle condition il tiendra compte des recommandations qui lui sont proposées.

3.3.3.1 Mode de raisonnement

Pour être capable de suivre le raisonnement d'un élève pendant le processus de résolution de problème, un système tutoriel intelligent doit avoir des connaissances correspondant aux étapes que pourrait emprunter cet élève pour résoudre le même problème. Le raisonnement dans les systèmes conçus par Anderson est dit « articulé », dans le sens que les règles de production dans ces systèmes représentent les connaissances que l'élève doit acquérir au cours des séances d'apprentissage. Les auteurs qualifient ce raisonnement ainsi par opposition aux systèmes experts classiques à raisonnement *opaque*, dans lesquels les règles ne cherchent nullement à représenter les étapes empruntées par un apprenant en train de résoudre le même problème mais simplement à résoudre le problème le plus efficacement possible.

3.3.3.2 Mode de diagnostic

Le mode de diagnostic d'un système intelligent détermine la qualité de ce système et sa pertinence dans la modélisation de l'apprenant. En ce qui concerne le traçage de modèle d'Anderson, il comporte un mode unique pour l'établissement des diagnostics. En effet, selon Wenger (1987), aucun autre système d'apprentissage

n'effectue le diagnostic pendant le processus même de résolution. Certains systèmes tels que PROUST et DEBUGGY tentent de reconstruire les intentions de l'élève une fois que ce dernier a terminé sa solution ; ces systèmes opèrent donc à partir d'un produit et non pas à partir d'un processus. Cette particularité du modèle d'Anderson constitue un élément positif relativement à la conception du système à développer, puisque dans celui-ci, l'accent est justement mis sur l'apprentissage d'un processus plutôt que sur l'analyse d'un produit.

3.3.3.3 Adjonction des nouvelles connaissances

Il est de grande importance d'annoter chaque règle de la base de connaissance pour lui adjoindre des connaissances spécifiques dans un contexte d'apprentissage. Ces connaissances incluent nécessairement :

- une description exacte de la règle en langage naturel ;
- le type de connaissances arithmétiques utilisées et son degré de difficulté ;
- le genre de problème (ou les genres) dans lequel (lesquels) la règle peut être utilisée ;
- le plan auquel elle donne naissance.

Pour fins d'analyse et de diagnostic, il vaut mieux cerner les différents aspects des règles faisant partie de la base de connaissances, et de les rassembler dans une table de règles avec certains commentaires facilitant les justifications et les assertions relatives aux états avant et après l'activation de chaque règle, même si ces éléments ne font pas partie de modèle d'Anderson. Le système pourrait alors générer des messages explicatifs à partir de ces connaissances. Ces explications pourraient notamment inclure des assertions quant à l'état initial et à l'état final de la règle concernée.

3.3.4 Modèle d'Anderson et systèmes à base de connaissances

Le tableau suivant (tableau 3.1) présente des liens entre les qualités que l'on doit retrouver dans la base de connaissances d'un système tutoriel incorporant le modèle d'Anderson et certains systèmes à base de connaissances comme les systèmes experts de type classique, les systèmes d'EIAO et le système TIDES.

Comme on peut le constater dans cette grille, la base de connaissances d'un système expert classique ne se prête pas bien à une intégration du modèle d'Anderson. Les considérations précédentes fournissent certaines explications à cet égard. Elles permettent aussi de comprendre pourquoi il est autrement avec les systèmes d'EIAO.

Qualité de la base de connaissances dans le modèle d'Anderson	**Système expert classique**	**EIAO**	**TIDES**
Les règles sont structurées hiérarchiquement en fonction du but explicite de chacune	**Parfois**	**Parfois**	**Oui**
Les règles correspondent aux étapes de résolution qu'utiliserait un élève pour résoudre un même problème	**Non**	**Parfois**	**Oui**
Des mal-règles représentent les façons de se tromper en résolvant des problèmes	**Non**	**Parfois**	**Oui**
Les règles permettent de suivre le cheminement de l'élève pendant le processus de résolution	**Non**	**Parfois**	**Oui**
Les règles permettent au tuteur d'intervenir auprès de l'élève pour l'aider et le guider en cas d'erreur	**Non**	**Parfois**	**Oui**

Tableau 3.1 – Prescription du modèle d'Anderson et système à base de connaissances

3.4 Modélisation cognitive de l'arithmétique

Comme nous l'avons souligné à plusieurs reprises, le domaine choisi pour élaborer le système TIDES est l'arithmétique cognitive. Il est maintenant pertinent de réexaminer ce domaine afin d'identifier ce qui le caractérise et se donner aussi des balises sur la façon d'aborder la conception et l'élaboration du système TIDES.

3.4.1 Caractéristiques de l'arithmétique en tant que domaine d'apprentissage

L'étude des procédures de résolution des problèmes d'arithmétique, notamment de la soustraction, apparaît indispensable dès que l'on cherche à comprendre l'origine des erreurs commises par les enfants. Elle montre le caractère profondément organisé de certaines erreurs.

Parce que souvent les connaissances que l'enfant construit ne sont pas isomorphes à celles qu'on voudrait bien leur enseigner. Les enfants ne reçoivent pas passivement la connaissance qu'on veut leur transmettre : ils l'interprètent, ils la structurent, ils l'assimilent à la manière de leurs constructions mentales existantes. Beaucoup de recherches (notamment celles de : Brown et Burton 1978 ; Brown et Van Lehn, 1980, 1982 ; VanLehn (1990) ; Bednarz & Janvier (1984) ; Arsenault, & Lemoyne (2000) ; Carpenter, Moser & Romberg (1982) et Resnick (1982)) en enseignement arithmétique ont été amenées à étudier explicitement comment les enfants s'approprient des concepts arithmétiques, comment ils développent des habiletés spécifiques aux calculs arithmétiques, et quelles sont les stratégies utilisées dans une situation de résolution de problèmes arithmétiques.

Les moyens utilisés pour mettre en évidence les modalités de résolution consistent à observer et à analyser les stratégies utilisées par un enfant dans une situation de résolution de problème et à repérer ses erreurs. Ces dernières sont les résultats des tentatives raisonnables d'adapter des connaissances nouvelles sur le concept des nombres, sur la valeur de position des chiffres, sur la numération positionnelle, sur les procédures de calcul, sur les cardinalités entre les nombres, sur les propriétés logiques des nombres, sur les opérations arithmétiques, sur le fonctionnement des algorithmes de calcul (Fayol & al., 2000; Arsenault & Lemoyne, 2000).

Sur la base des ces connaissances, nous présentons plus en détail, dans la suite, six facettes des compétences arithmétiques qui devraient retenir l'attention lors de l'évaluation diagnostique (Grégoire, 2001). Ces facettes sont représentées dans la

figure 3.6 de façon non hiérarchique qui laisse en suspens la question des interactions entre ces facettes.

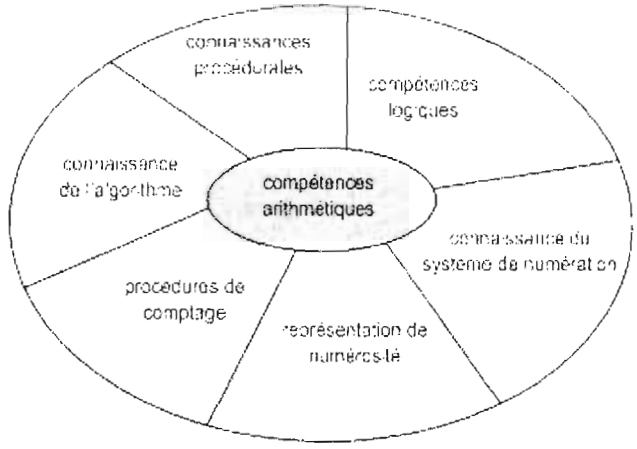


Figure 3.6 Les six compétences arithmétiques

- Les *capacités logiques* constituent une première facette des compétences arithmétiques. Le calcul numérique ne peut se réduire seulement à l'application automatique d'algorithmes ni à la récupération de faits arithmétiques en mémoire. Pour calculer, en particulier mentalement, l'élève doit pouvoir comprendre la notion de nombre. Pour Piaget, la compréhension de cette notion est déterminée par le développement et la coordination de certaines opérations logiques (notamment la sériation logique et la classification logique[8]). La maîtrise opératoire du nombre permet à l'enfant les raisonnements sur le nombre qu'impliquent les calculs arithmétiques. La non-maîtrise des caractéristiques du nombre peut être une source d'erreur. En effet, l'enfant qui, par exemple, voit le nombre comme une simple collection de chiffres collés les uns à côté des autres sans aucune prise en considération des mots centaines, dizaines ni unités, et sans établir de lien entre ces chiffres et leurs positions, peut même ne pas s'étonner d'obtenir, dans une opération de soustraction, une différence de deux nombres plus grande que le premier terme de l'opération. Le sens que l'enfant attribue au nombre est intimement lié au développement de ses compétences logiques.

[8] Selon Piaget (cité dans Bideaud, 1991, p.20), la sériation logique sous-tend la compréhension de la suite ordonnée des nombres naturels. L'opération de sériation consiste à distinguer les objets et à créer ainsi une relation asymétrique entre les objets. La classification logique sous-tend, quant à elle, la compréhension de l'aspect cardinal du nombre.

- Une seconde facette concerne la connaissance du *système de numération*. Dans la numération de position chaque chiffre occupe une place qui correspond à son ordre décimal (unités, dizaines, centaines, ...) ; de plus la mécanisation du calcul n'est possible que grâce à l'utilisation de cette numération (Marguin, 1994). La compréhension de la numération positionnelle est, elle aussi, importante dans l'apprentissage d'un algorithme : en fait, les algorithmes sont, par plusieurs aspects, des applications, des règles et principes de la numération positionnelle aux opérations arithmétiques. Il est bien connu qu'une très large part des erreurs commises par les élèves lorsqu'ils utilisent les divers algorithmes arithmétiques sont des manifestations d'une compréhension déficiente des principes de la numération positionnelle. Elle constitue un obstacle concernant l'apprentissage des techniques opératoires. Ainsi, Dionne (1995) a souligné que la numération positionnelle s'avère toujours l'élément le plus délicat dans la construction de l'arithmétique et que la plupart des difficultés éprouvées par les élèves trouvent là leur source, dans une compréhension encore incomplète, trop partielle, mal assurée de cette convention, base de notre système numérique. D'ailleurs, certains chercheurs (Balka, 1993 ; Fuson et Briars, 1990 ; Hiebert et Wearne, 1992 ; Ross et Kurtz, 1993), comme le rapportent Arsenault et Lemoyne dans (Arsenault & Lemoyne, 2000), suggèrent que la numération de position doit être enseignée conjointement avec les procédés de calcul, pour que les élèves puissent apprendre à faire des opérations à plusieurs chiffres, corrigent en même temps leurs erreurs et assimilent ce concept de numération. Mais cette approche est critiquée par Bednarz et Dufour-Janvier (1986). Ils font remarquer que « les représentations proposées adoptent toujours l'ordre de l'écriture conventionnelle des nombres, rendant difficile la construction de liens entre les procédures de calcul et le travail de groupement effectué sur les objets » (Arsenault & Lemoyne, 2000).

- Une troisième facette désigne la représentation de la *numérosité*. Ce terme désigne la quantité symbolisée par le numéral ou verbal, c'est-à-dire qui peut être représenté par les chiffres ou les phonèmes. L'étude de la numérosité a été

initiée par Piaget et Szeminska en 1941 quant à son apprentissage par les enfants. Ils ont mis l'accent sur l'acquisition de la conservation des quantités continues et discontinues dans la genèse de la notion de nombre afin d'étudier les capacités de l'enfant à construire une représentation mentale indépendamment des objets, de leur propriété perceptive, et aussi de leur diversité. L'enfant doit en effet se libérer des apparences sensibles pour qu'il puisse raisonner correctement sur les nombres. Ses représentations mentales immédiates doivent être corrigées par sa raison. Comme l'expérience de Piaget l'a montré, pour un enfant, les quantités discontinues ne se conservent pas lorsqu'on modifie leur disposition. Si on présente, par exemple, à l'enfant deux rangées de jetons équivalentes (même nombre de jetons) : une rangée (B) équivalente à une rangée modèle (A), et si on écarte les uns des autres les jetons de la rangée (B), et si on demande à l'enfant de comparer à nouveau les deux rangées, l'enfant considère généralement que la rangée ainsi transformée contient plus de jetons car elle est plus longue (Piaget et Szeminska, 1941, p. 104). Pour accéder à la conservation des quantités discontinues, l'enfant doit pouvoir se dégager des changements de surface et revenir mentalement à la situation initiale. Il doit comprendre aussi que la modification de la disposition spatiale des objets ne modifie nullement le cardinal de la rangée. Si rien n'a été retiré ni ajouté, le cardinal reste identique malgré certaines modifications apparentes.

Par ailleurs, la réussite aux tâches d'acquisition de la conservation témoigne de la capacité d'un enfant à faire une distinction entre les transformations qui affectent le nombre (comme l'ajout et le retrait d'objets) et celles qui le laissent inchangé (comme leur déplacement) (Barrouillet et Camos, 2002). Cependant, selon Barrouillet et Camos, malgré que cette capacité de conservation aide à la compréhension de la notion de nombre, on ne peut affirmer qu'elle semble déterminer directement l'acquisition d'autres concepts ou habiletés numériques.

- La quatrième facette représente la capacité de *comptage* et de *dénombrement*. Le dénombrement est le processus de quantification (Barrouillet et Camos, 2002).

Il est souvent considéré comme étant à la base de l'apprentissage arithmétique. En effet, comme le mentionne Grégoire et van Nieuwenhoven (1995), le dénombrement est une façon de prouver et de vérifier empiriquement la validité d'un raisonnement dans la résolution d'une opération arithmétique. Quant au comptage, il désigne selon Nieuwenhoven (1999, 2001) l'activité de récitation de la suite des noms de nombres. Cette activité influence l'acquisition des principes numériques de base de l'arithmétique, telle que la correspondance terme à terme et la cardinalité. Elle influence également la façon dont l'enfant effectue les différentes opérations arithmétiques, telle que la soustraction.

Certains chercheurs, comme Carpenter et Moser (1982), Carpenter (1983), Carpenter, Hiebert et Moser (1981), ont étudié les procédures de comptage utilisées par les enfants pour résoudre certains problèmes d'addition ou de soustraction. Ils ont relevé l'existence d'une pluralité de procédures, pour les problèmes avec addition comme pour les problèmes avec soustraction. Deux stratégies de comptage sont observées, dites "counting up" et "counting down". La stratégie "counting down" consiste à compter à rebours à partir du nombre le plus grand un nombre de pas équivalent au plus petit (exemple : pour effectuer la soustraction 9 - 5, l'enfant décrémentera par 1 et comptera (8, 7, 6, 5, 4), le résultat est 4). La stratégie "counting up" consiste quand à elle à compter en partant du nombre le plus petit jusqu'à atteinte du nombre le plus grand : le nombre de pas constitue le résultat (dans l'exemple précédent, on a : (6, 7, 8, 9) la réponse est 4, c'est le nombre d'éléments ajoutés). Toutefois, la stratégie "counting up" semble la plus fréquemment utilisée (Siegler, 1989) en raison de son plus faible coût cognitif que "counting down". Cette dernière nécessite en effet un comptage à rebours, difficile pour les enfants, et le contrôle simultané du nombre de pas.

Cependant, le principe de cardinalité a été critiqué par certains chercheurs (Frye et al., 1989). Le fait que les jeunes enfants répètent souvent le dernier mot-nombre en dénombrant une collection peut être une simple imitation (Barrouillet et Camos, 2002). En effet, Fuson et Hall (1983) ont montré dans leur expérience

que de jeunes enfants ayant déjà dénombré une collection, ne pouvaient pas répondre par un mot-nombre à la question « combien y a t-il objets ? », mais ils recommençaient à nouveau le dénombrement de cette collection. De plus, et devant une classe d'objets, par exemple, (combien y a-t-il d'arbres dans cette forêt?) leurs réponses étaient différentes qu'à une collection (combien y a-t-il de fleurs?) (Barrouillet et Camos, 2002). Les enfants avaient tendance à recompter lorsqu'ils étaient face à une classe d'objets (Markman, 1979, cité dans Barrouillet et Camos, 2002).

- La cinquième facette concerne la connaissance d'un *algorithme*. Comme l'indique la quatrième facette, la capacité de dénombrer une collection constitue un acquis fondamental. Il arrive souvent que l'on ait à évaluer successivement plusieurs ensembles pour indiquer ensuite la différence qui existe entre eux. Une méthode élémentaire pour résoudre ce type de problème consiste à rassembler les composantes de deux de ces ensembles et à dénombrer les éléments du nouvel ensemble pour en déduire cette différence. Mais, les procédures évoquées ne sont pas très économiques : rares sont les élèves qui vont par exemple réunir une collection de 230 jetons pour ensuite en prélever 148 afin d'effectuer la soustraction 230 - 148. La solution d'un tel problème passe par le recours à des procédures – algorithme - de soustraction que les élèves mettent en œuvre de manière le plus souvent quasi-automatique et inconsciente. Cet algorithme est un ensemble de procédures ou de règles d'action qui doivent être appliquées d'une façon ordonnée, étape par étape, pour arriver à un résultat. L'application d'un algorithme arithmétique suppose la compréhension de plusieurs éléments, notamment du concept du nombre, de la numération positionnelle et du sens de l'opération utilisée. La résolution des opérations arithmétiques, notamment la soustraction, passe habituellement par des algorithmes de calcul reposant sur la numération positionnelle. Selon Barrouillet et Camos (2002), trois facteurs semblent contribuer à la difficulté de ces opérations. D'une part les algorithmes les plus utilisés nécessitent une manipulation mentale des nombres, ce qui entraîne d'ailleurs une charge en mémoire de travail qui peut être source d'erreurs (Widaman et al., 1992).

D'autre part, ces algorithmes nécessitent des connaissances conceptuelles sur la numération positionnelle (Fuson & Briars, 1990). Enfin, la résolution des opérations arithmétiques à l'aide d'algorithmes nécessite une décomposition du problème en tâches élémentaires. Cependant, les stratégies algorithmiques utilisées pour résoudre ces diverses étapes entraînent des durées de résolution très élevées, en particulier pour la soustraction.

- Enfin, la sixième et dernière facette désigne les *connaissances procédurales*. La préoccupation prioritaire ne doit pas être de vérifier simplement si les élèves connaissent les données numériques élémentaires ainsi que l'algorithme approprié, mais qu'il est plus important de s'assurer qu'ils savent aussi appliquer des procédures de façon efficace pour résoudre un problème donné. En effet, au cours de la résolution de problème, l'acte intellectuel est orienté non seulement vers la compréhension de la tâche, c'est-à-dire l'élaboration d'une représentation mentale de la tâche, mais aussi vers l'action à mettre en œuvre.

  Pour acquérir des connaissances procédurales, l'élève doit recourir aux processus de procéduralisation et de composition (Anderson, 1989). Le premier processus transforme des règles procédurales en des règles spécifiques au problème à résoudre. Il peut s'agir donc d'une suite d'actions ou d'opérations que l'élève sait dire mais qu'il ne sait pas encore faire aisément. Quant au deuxième processus, il permet à l'élève d'automatiser une procédure. Il faut souligner que les objectifs de ces deux processus ne peuvent pas, dans la très grande majorité des cas, être atteints autrement que par la pratique des procédures suffisamment longtemps, et ce, chaque fois que si possible.

Les six facettes que nous venons de décrire seront l'objet d'une modélisation possible, par un réseau bayésien, dans le cadre d'amélioration du système mais non intégré dans ce dernier (cf. chapitre 6).

3.4.2 Modélisation cognitive de la soustraction

Avec l'arithmétique comme domaine d'apprentissage, l'accent est mis sur les difficultés de l'opération de soustraction, sur l'application de son algorithme et aussi sur les erreurs commises dans l'utilisation de cet algorithme.

Depuis plus de deux décennies, la résolution des opérations arithmétiques a fait l'objet de nombreuses investigations ; les recherches conduites depuis ont confirmé que les difficultés essentielles des activités de résolutions de problèmes arithmétiques résidaient, d'une part, dans le traitement des opérations, et, d'autre part, dans la compréhension/interprétation des énoncés (Fayol et al., 2000).

Les difficultés rencontrées par les élèves lors de l'apprentissage de l'arithmétique cognitive, notamment en soustraction, ont souvent été soulignées par différents chercheurs. En effet, les études réalisées par Brown, Burton et Van Lehn (Brown et Burton 1978 ; Brown et Van Lehn, 1980, 1982 ; Van Lehn, 1990) dans ce domaine ont permis de franchir une étape importante dans la compréhension des erreurs des élèves (Arsenault & Lemoyne, 2000), notamment en termes de «bugs». Dans la suite, après la précision de la notion de « bug » dans la soustraction, nous présentons deux principaux modèles cognitifs développés par ces chercheurs et reliés à l'étude diagnostique des erreurs commises par les élèves en soustraction. Les critiques que l'on peut adresser à ces modèles d'un point de vue didactique et pédagogique sont aussi présentées.

3.4.2.1 Bug de soustraction

Brown et Burton (1978) ont été les premiers à parler des « bugs » en analysant les erreurs de soustraction écrite. Ils ont souligné que les erreurs ne sont pas le résultat du hasard mais, qu'elles sont liées à une *compréhension incomplète* ou à une *procédure erronée* et utilisée systématiquement. Selon ces auteurs, les bugs correspondent à des variations erronées de l'algorithme de soustraction écrite, comme par exemple le bug « soustraire le plus petit du plus grand » (Substract Smaller from Larger), constitue une modification de la règle qui dit « qu'il faut soustraire le chiffre du dessous du chiffre du dessus dans chaque colonne ».

VanLehn (1982), pour sa part, a souligné que les erreurs de soustraction peuvent être modélisées à l'aide de deux mécanismes: bugs et erreurs d'inattention. Les erreurs d'inattention sont des phénomènes de performance qui sont attendus mais instables, alors que les bugs sont des phénomènes de compétence qui sont aussi attendus mais stables.

Pour notre part, nous considérons que les bugs sont des stratégies d'adaptation qui apparaissent quand des difficultés surgissent dans la réalisation d'une routine d'un algorithme correct.

3.4.2.2 Modèle BUGGY

Selon Brown et Burton (1978), il est souvent difficile de déduire un bug d'un élève à partir de sa réponse. Mais la nécessité de faire comprendre ce problème aux enseignants, de développer chez eux des stratégies d'analyse pour faire face aux erreurs d'un élève a conduit Brown et Burton à construire le système BUGGY. Ce système est destiné à l'entraînement des enseignants dans le diagnostic de « bug », en soustraction. Il se présente comme un jeu où l'utilisateur (en général l'enseignant) est censé être un expert consulté par une commission. L'enseignant découvre alors que les erreurs de calcul sont très souvent associées à un *réseau procédural* bien défini. Cette découverte s'effectue ainsi : l'enseignant propose au système un problème de soustraction, le système produit une réponse erronée et demande à l'enseignant de reconnaître l'erreur commise et d'appliquer la stratégie qu'il juge responsable de l'erreur à d'autres exercices suggérés par le système ; c'est-à-dire qu'au lieu de fournir un diagnostic sous forme verbale, l'enseignant doit montrer qu'il a reconnu l'erreur en produisant le résultat fautif du système sur les exercices proposés. Le système compare alors les réponses induites par la procédure erronée qu'il a choisie à celles fournies par l'enseignant et, en cas de désaccord, donne sa réponse à un des exercices et demande à nouveau à l'enseignant de trouver et d'appliquer la stratégie erronée afin de se forger une idée précise de ces perturbations. Le système est capable de dépister plus d'une centaine de manières erronées d'effectuer des soustractions en calcul écrit. Il possède une base de *procédures erronées* de 330 bugs.

Ce système présente trois avantages. Tout d'abord, il sensibilise les enseignants aux problèmes de « bugs » et les familiarise avec les plus courants. Ensuite, l'efficacité de ce système s'explique par la pédagogie exploitée ; cette pédagogie s'appuie sur la notion de conflit cognitif qui est essentielle dans une perspective de formation des enseignants. Enfin, il assure un diagnostic des erreurs systématiques et des procédures qui les produisent en conjecturant et en trouvant des exemples à poser à l'utilisateur pour infirmer ou confirmer une hypothèse. Cependant, il existe un certain nombre de problèmes d'ordre didactique qui se posent à ce modèle. Tout d'abord, il comprend un trop grand nombre de « bugs » (330 pour la soustraction seulement) dont beaucoup ne sont pas souvent rencontrés chez les élèves. Cette multiplicité a parfois pour résultat qu'on ne sait plus quoi faire pour aider l'enfant, notamment parce que la diversité des « bugs » rend instables les façons d'expliquer les erreurs des élèves. En effet, il est souvent difficile de déduire exactement un seul « bug » à partir de la réponse de l'élève pour donner une explication exacte ou de proposer une hypothèse précise sur le comportement de cet élève. Ensuite, les multiples « bugs » répertoriés dans Buggy ne sont pas assez profonds et sont trop poches, ce qui les rend inutilisables pour la remédiation.

3.4.2.3 Théorie de réparation (Repair theory)

Comme nous l'avons souligné précédemment, les travaux de Brown et VanLehn (1980) ont facilité la compréhension des causes des erreurs en arithmétique. Pour pousser plus loin le raisonnement initié par le système BUGGY, ils ont développé une théorie appelée « *théorie de réparation* (*Repair Theory*) » pour donner une explication concernant l'apparition de ces erreurs (Renaudie, 2005).

Dans cette théorie, l'erreur est considérée comme le produit d'une impasse que l'élève essaie de franchir en proposant sa propre solution (Arsenault, & Lemoyne, 2000). En effet, lorsque l'élève est devant une impasse, due généralement à l'oubli de la procédure qui peut lui aider à dépasser ce blocage, et au lieu d'interrompre sa solution, il devient inventif. Il essaie d'effectuer une *réparation* (repair) et de rafistoler une solution pour surmonter l'obstacle afin de poursuivre

sa tâche (Renaudie, 2005 ; Arsenault, & Lemoyne, 2000). L'exemple suivant (tableau 3.2) donne une idée du déroulement de la séquence de décisions et d'opérations aboutissant à un bug (selon la théorie de réparation).

2 0 7 − 1 6 9	Dans la colonne des unités, je ne peux enlever 9 de 7 donc, je vais devoir «*emprunter*»	**Algorithme enseigné**
2 0 7 − 1 6 9	Pour emprunter, il faut que je décrémente de 1 le chiffre du haut de la colonne suivante. Mais je ne peux toujours pas enlever 1 de 0	**IMPASSE**
2 0 7 − 1 6 9 2	Alors je reviens à la colonne des unités, je ne peux toujours pas soustraire 9 de 7, je vais soustraire 7 de 9	**Réparation avec retour en arrière**
2 0 7 − 1 6 9 2	Je ne peux pas soustraire 6 de 0, donc j'emprunte de la colonne suivante, j'enlève 1 de 2 et je remplace 0 par 10	
2 0 7 − 1 6 9 4 2	6 ôté de 10 il reste 4. Aux centaines, 1 ôté de 1 il reste 0.	

Tableau 3.2 – Exemple de génération d'un bug

L'erreur présentée dans le tableau 3.2 est causée par la règle '*soustraire le petit nombre du plus grand*' (qui devient ici mal-règle, puisque ce n'est pas son contexte d'application). Cette erreur peut être expliquée par l'existence de ce que Vergnaud (1991) appelé un *schème* de soustraction sur les entiers. Ce schème dit : « qu'il ne faut pas soustraire un nombre plus grand d'un plus petit » ce qui conduit l'élève à intervertir la position des chiffres dans la colonne (Arsenault, & Lemoyne, 2000).

La contribution la plus importante de cette théorie est qu'elle a montré le caractère profondément organisé de certaines erreurs. De plus, elle a permis d'expliquer l'apparition des bugs dans les procédures apprises par le sujet. Cependant, l'aspect didactique est absent dans cette théorie de réparation, d'ailleurs VanLehn lui-même a indiqué que la théorie de réparation nécessite un complément didactique. Il suggère que l'étude des enseignements donnés en classe permette de trouver non seulement l'origine de ces montages (stratégies) imparfaits de règles mais encore celle des réparations que le sujet est à même d'envisager. Ajoutons aussi,

que les résultats trouvés par VanLehn ne sont pas décisifs, puisque sa théorie ne permet d'expliquer que 28 des 75 bugs «observés» et génère 26 bugs non observés (VanLehn, 1990).

3.4.2.4 Choix d'un modèle pour cette thèse

Malgré que les deux modèles précédents ont poussé plus loin l'explication de l'apparition de procédures erronées, ils n'ont pu avancer des hypothèses sur les causes réelles derrière ces procédures erronées, puisqu'ils cherchent d'une part, à déterminer quelles sont les conditions idéales d'apprentissage d'une procédure, et d'autre part, à démontrer comment cet apprentissage influe sur la procédure effectivement mémorisée. Un diagnostic plus profond des causes de ces procédures erronées est nécessaire. Pour cela, nous avons choisi d'utiliser le modèle d'Anderson, le traçage de modèle (voir sous-section suivante), pour comprendre les difficultés d'apprentissage qui sont souvent à l'origine des erreurs commises par les élèves. Ces erreurs proviennent essentiellement de dérivations procédurales et de conceptions erronées. Les difficultés manifestées par ces erreurs sont nombreuses, et le système TIDES doit pouvoir traiter certains types de ces difficultés, comme :

- la difficulté d'appliquer un algorithme de soustraction lorsqu'il s'agit d'emprunt ;
- la difficulté de considérer le problème comme un tout, les élèves isolant les colonnes les unes des autres et les considérant comme autant de problèmes particuliers ;
- la difficulté d'appliquer une procédure appropriée à un problème donné ;
- la difficulté d'ordre conceptuel relative à l'opération de la soustraction écrite ;

L'utilisation du traçage de modèle simplifie grandement le diagnostic des erreurs.

3.5 Modèle d'Anderson et arithmétique

Cette dernière étape de la première partie de la méthodologie de conception et d'élaboration vise à montrer l'applicabilité du modèle d'Anderson à un système tutoriel intelligent, dont le domaine d'apprentissage est l'arithmétique. Dans ce cas, il faut d'abord compléter le cadre d'analyse en abordant différents problèmes associés à la conception et au développement du système TIDES en tant que système tutoriel intelligent de l'arithmétique incorporant le modèle d'apprentissage d'Anderson (sous-sections 5.1 à 5.4).

3.5.1 Traçage de modèle et arithmétique

Comme il a été indiqué dans le chapitre 2, le traçage de modèle d'Anderson consiste à utiliser les deux modèles de la théorie : le modèle de performance et le modèle d'apprentissage. Le premier est utilisé pour tracer l'état de la solution d'un élève à l'intérieur d'une tâche et le deuxième est utilisé pour tracer l'état des connaissances d'un élève à mesure qu'il accomplit les différentes tâches. Pendant qu'un élève résout un problème, des messages explicatifs sont générés et permettent une interprétation correcte de la solution. C'est essentiellement grâce à ce scénario d'apprentissage qu'un élève est susceptible de franchir les étapes du processus de compilation du modèle d'Anderson (voir 2-5.3.2).

Dans le traçage de modèle, l'élève interagit avec le système strictement au niveau de la résolution de problèmes. L'élève doit bien sûr utiliser une certaine stratégie pour résoudre son problème. Mais celle-ci ne se manifeste qu'à travers l'organisation de sa solution.

En effectuant le design du système TIDES, et notamment en cherchant à appliquer la méthodologie du traçage de modèle au domaine de l'arithmétique, il faut identifier puis comparer :

- la nature et la structure des connaissances requises pour l'apprentissage des habiletés de résolution des problèmes arithmétiques ;
- la liberté de manœuvre de l'élève et les modalités d'utilisation des messages de rétroaction pour ces types d'habiletés.

3.5.1.1 Nature et structure des connaissances

La création des règles de production nécessaires au traçage est la tâche la plus importante pour implanter le traçage de modèle. Cette tâche implique d'abord l'identification des connaissances que l'apprenant devra acquérir. Une autre tâche complémentaire doit être impliquée aussi est l'élaboration d'un ensemble de mal-règles dans le modèle de l'élève. Ces règles facilitent l'identification des erreurs rencontrées et constituent les connaissances du tuteur. À partir des premières expérimentations avec les trois tuteurs qu'il a réalisés, Anderson rapporte qu'une proportion de 80% d'erreurs identifiées par des mal-règles de production constitue un maximum difficile à dépasser. Les autres erreurs sont trop particulières (ou trop éloignées des connaissances du système) pour être identifiable par le système.

L'identification puis la construction d'un ensemble de mal-règles constituant les connaissances du tuteur peut être élaborée suivant deux approches. Il y a bien sûr comme première approche, la consultation de tuteurs humains ayant une longue expérience avec des élèves en situation d'apprentissage. La deuxième approche est une analyse systématique des règles de production représentant les connaissances de l'élève et l'identification des erreurs théoriques possibles en jouant avec les différents paramètres de ces règles.

Dans le cas du système TIDES, les connaissances relatives aux règles et certaines mal-règles ont été identifiées à partir de différentes sources :

- base de connaissances développée par Van Lehn (1990) ;
- approches développées par Brown & Burton (1978, 1982), Bednarz & Janvier (1984) Arsenault, & Lemoyne (2000), Carpenter, Moser & Romberg (1982) et Resnick (1982).

3.5.1.2 Liberté de manœuvre de l'élève

Dans le domaine impliquant la résolution de problèmes, comme l'arithmétique, le système doit être capable de résoudre le problème présenté à l'élève, étape par étape, en suivant différents chemins. Ces étapes doivent être représentées de façon structurée de telle sorte que l'élève n'ait pas à cheminer en utilisant des étapes

inappropriées. Il ne faut pas non plus empêcher l'élève d'ajouter des étapes intermédiaires en exprimant un cheminement vers une solution. Dans les systèmes d'Anderson, chaque règle de production représentant les connaissances du domaine correspond à une inférence de la solution (Anderson & al., 1985).

La méthodologie à définir doit permettre de vérifier d'abord si les étapes de résolution des problèmes prévues dans les règles du système conviennent également à l'élève qui résout les mêmes problèmes. Pour ce faire, il faudra, pour un certain nombre de tâches, élaborer puis comparer la décomposition en sous-tâches pour le système et pour l'élève.

Les habiletés cognitives à acquérir en arithmétique se réfèrent essentiellement aux stratégies de résolution de problèmes. Dans la plupart des problèmes, de nombreuses stratégies peuvent être utilisées avec succès pour arriver à une solution (voir sous-section suivante 5.2). Le choix d'une de ces stratégies pourra avoir une répercussion directe sur la facilité ou la difficulté de résolution complète du problème.

C'est lors de cette recherche générale dans l'espace-problème que l'élève peut innover dans la solution adoptée. Il est donc impératif qu'un élève dispose d'une grande liberté de manœuvre afin d'explorer et/ou d'expérimenter différentes stratégies ou pistes de solution.

3.5.2 Stratégies de résolution

La soustraction est une opération aux usages multiples : elle permet de comparer deux nombres et de connaître la différence qui existe entre eux, de déterminer de combien le plus grand dépasse l'autre ou ce qu'il faut ajouter à ce plus petit pour obtenir le plus grand. Mais, pour faire avec plus de sûreté et d'exactitude cette opération, il faut suivre les règles et les algorithmes appropriés en utilisant des méthodes courantes de la soustraction.

Au cours des siècles, plusieurs algorithmes ont été développés (Kamii, 1990) : quelques-uns requièrent la manipulation d'abaques, d'autres exploitent des

techniques de calcul mental ou s'appuient sur l'utilisation de papier et crayon dans l'application d'une série de règles.

Un regard sur ce qu'on trouve dans les écoles primaires, permet d'identifier facilement deux algorithmes distincts : l'un utilise la méthode dite « **par emprunt** » où l'on « casse les dizaines » et a été l'objet de l'étude de plusieurs recherches, notamment celles de Cox (1974, 1975), Brown et Burton (1978), Brown et VanLehn (1982), VanLehn (1990), et Resnick (1982). Actuellement, cette méthode est la plus courante dans les écoles nord-américaines. L'autre recourt à la méthode dite par « **Equal Addition** » ou « **addition équivalente** » où l'on ajoute des quantités égales aux deux termes de la soustraction dans l'ordre en utilisant les faits fondamentaux de la soustraction : suivant cette méthode utilisée dans plusieurs pays de la francophonie, pour effectuer 52 - 18, on ajoute dix unités au premier terme (i.e à 2) afin de pouvoir en soustraire huit et on ajoutera une dizaine au second terme (i.e à 1) pour compenser le premier ajout. Ces deux méthodes utilisent le mécanisme de la soustraction verticale dans laquelle les nombres impliqués sont écrits l'un au-dessus de l'autre et les colonnes sont alignées en ordre de droite à gauche. Cependant, elles se distinguent seulement dans l'utilisation de la technique d'emprunt. Comme conséquence de cette différenciation, la remédiation des erreurs doit être différente lorsque le problème exige l'emprunt. Ainsi, Favart (1987, cité dans Fayol, 1990), a comparé les *bugs* relevés dans la littérature anglophone, où l'on traitait de l'algorithme « par emprunt » à ceux observés dans une population d'élèves francophones utilisant l'autre algorithme. L'auteur met en évidence que s'il existe certes des erreurs systématiques communes aux deux algorithmes (notamment 0-N = N ; ..., ou 6-9 = 3), chacun d'eux tend à induire un certain type de *bugs* : par exemple, l'algorithme «par emprunt» entraîne des erreurs nombreuses lorsqu'il s'agit d'emprunter sur un zéro, alors que l'algorithme « Equal Addition » ou « addition équivalente » induit souvent des erreurs consistant à soustraire l'emprunt au lieu de l'ajouter au chiffre correspondant.

3.5.3 Contraintes d'apprentissage de l'arithmétique

Après les discussions que nous avons vu sur l'arithmétique comme domaine d'apprentissage, nous allons maintenant présenter, en résumé, les caractéristiques de ce domaine. Ces caractéristiques qui deviennent des contraintes quant il s'agit d'élaborer le design d'un système tutoriel intelligent en arithmétique.

- **Relativement à la représentation des connaissances et au modèle de l'élève**

  - Pour être exercée, l'arithmétique (notamment la soustraction) doit s'incarner dans une activité réelle (le plus souvent les opérations arithmétiques).

  - Elle implique l'explicitation d'une stratégie de résolution de problèmes par l'élève.

  - Cette stratégie est exprimée sous forme d'actions (simples ou complexes) correspondant à la méthode de résolution de problèmes et à la décomposition de la tâche à résoudre en sous-tâches.

  - Pendant qu'il résout un problème, l'élève doit être en mesure d'explorer librement différentes pistes de solution.

- **Relativement au design de l'interface**

  - Dans la mesure du possible, l'élève doit pouvoir se concentrer sur les aspects stratégiques de sa solution par opposition aux aspects liés à la décomposition de la tâche.

  - L'élève doit s'y retrouver dans l'abondance d'informations relatives à la tâche qu'il est en train de résoudre.

Les contraintes mentionnées précédemment ne sont pas spécifiques au domaine de l'arithmétique. Elles aident à comprendre cependant, comment certaines prescriptions du modèle d'Anderson s'appliquent plus difficilement quand le domaine d'apprentissage est l'arithmétique. Par exemple, une contrainte comme « pendant qu'il résout un problème, l'élève doit être en mesure d'explorer librement différentes pistes de solution » rend difficile l'application d'une prescription du modèle d'Anderson (traçage de modèle) telle « le tuteur doit intervenir rapidement quand le système détecte une erreur dans la solution formulée par l'élève ».

Ces zones de conflit possible entre les prescriptions d'Anderson et les caractéristiques de l'arithmétique comme domaine d'apprentissage constituent en quelque sorte notre cadre d'analyse pour aborder les principaux problèmes rencontrés en essayant d'appliquer la méthodologie du *traçage de modèle* à ce nouveau contexte d'apprentissage.

Les éléments à analyser aux fins de cette démarche (application du traçage de modèle à un système tutoriel favorisant une approche pédagogique plus libre) constituent les sous-sections du chapitre 4 (4-3.1- 4.3.4) portant sur la conception et l'élaboration du système TIDES :

- Identification et représentation des connaissances d'arithmétique à acquérir pour les fins du traçage de modèle.
- Analyse et diagnostic des actions de l'élève.
- Analyse d'un processus plutôt que d'un produit.
- Design de l'interface : messages d'interaction et accès aux informations.

CHAPITRE IV

CONCEPTION ET DÉVELOPPEMENT DU SYSTÈME TIDES

Ce quatrième chapitre traite la conception et le développement du système TIDES (Tutoriel Intelligent pour le Diagnostic des Erreurs en Soustraction). Ce système est élaboré dans le cadre de cette recherche, avec comme objectif de prendre en charge les aspects analyse et diagnostic des erreurs d'un apprenant. Les caractéristiques principales du système TIDES sont qu'il utilise une méthodologie de diagnostic originale dans un contexte inhabituel, soit un environnement où l'élève a le plein contrôle sur le déroulement de la session d'apprentissage.

L'autre objectif de cette recherche, appliqué une théorie cognitive explicite lors du développement du système TIDES, est le sujet du présent chapitre. Il s'agit de réunir les conditions permettant au modèle d'Anderson de s'intégrer à un système d'apprentissage, en tant que système tutoriel intelligent et dont le domaine d'apprentissage est l'arithmétique.

4.1 Introduction

La conception d'un système tutoriel présentant un comportement intelligent implique souvent l'utilisation et la manipulation des connaissances, généralement riches et complexes, de nature et de fiabilité différentes, bruitées, imprécises. Ces connaissances proviennent de sources d'informations diverses et hétérogènes.

Du point de vue de la modélisation d'un apprenant, cette problématique de conception et d'usage du système en contexte réel nous impose de modéliser, non seulement le comportement de l'élève, mais aussi les raisons qui sous-tendent son comportement. Car, la modélisation seulement du comportement aboutit à ce que l'on peut appeler un diagnostic comportemental. L'analyse porte avant tout sur la

détermination de *comment* fait l'élève et non sur *pourquoi* il le fait. Elle reste au niveau des erreurs de surface, sans être à même d'élucider les causes réelles de ces erreurs. Cette distinction rejoint la différenciation entre le diagnostic de surface, qui ne concerne que les produits, et le diagnostic plus profond, qui consiste à comprendre les causes des difficultés qui peuvent surgir à divers moments d'apprentissage. Toutefois, cette modélisation est loin de se limiter à un problème technique seul ; elle s'appuie sur l'articulation et la coopération des différents modèles, qu'ils soient techniques ou cognitifs. Ils coordonnent leurs actions dans un environnement pour l'accomplissement d'un but commun.

Notre approche de la conception et du développement du système TIDES est basée en grande partie sur la modélisation cognitive (traçage de modèle). Cette conception décrit, d'une part, le comportement observé de l'apprenant depuis le début de la réalisation d'une tâche. Et d'autre part, modélise l'état cognitif de l'élève en termes de la maîtrise et de l'utilisation ou non de concepts corrects à bon ou à mauvais escient. Cette séparation entre ces deux façons de modélisation a l'avantage de faire une distinction claire entre les observations du comportement de l'élève et les raisons qui sous-tendent son comportement.

Dans la suite de ce chapitre, nous décrivons les choix que nous avons faits en ce qui concerne l'implémentation du modèle cognitif. La structure de cet exposé est identique au cadre de description que nous avons défini et utilisé lors du chapitre 3. Ainsi, nous donnons les détails de cette implémentation qui est relative au contenu du modèle d'Anderson, qui est utilisé pour le diagnostic des erreurs de l'apprenant. Nous décrivons tout d'abord l'architecture globale du système dans laquelle on trouvera les modules d'analyse et de diagnostic ; le premier identifie les observations recueillies à l'aide de l'interface élève, et le deuxième analyse les observables et produit, d'une part une description très fine du comportement de l'élève, et d'autre part, les causes qui sous-tendent ce comportement. On trouvera aussi dans cette architecture les modules de la base de connaissances et du profil de l'élève. Pour finir, nous donnons des exemples d'évolution du modèle de diagnostic. En parallèle, nous montrons, pour chaque point étudié, les adaptations

éventuellement nécessaires en vue de l'obtention d'une implémentation effective pour le système TIDES. L'implémentation que nous proposons se place dans le cadre de la programmation orientée objet (Java) et la programmation logique (Prolog) qui semble particulièrement adapté pour la conception d'un système tutoriel intelligent.

4.2 Les outils de développement du système TIDES

4.2.1 Outils de développement existants et leur pertinence

Au début de la réalisation du système TIDES, nous avons choisi certains outils spécialement conçus au développement de systèmes à base de connaissances, notamment les systèmes tutoriels intelligents (comme par exemple CLIPS[9]). Il va de soi que ce genre d'outils facilite le développement d'interfaces utilisateurs, formalisme de représentation de la connaissance, etc. Notre choix, quant aux outils, a été guidé par différents éléments, notamment le coût d'acquisition, qui est pratiquement nul, ainsi que leur réputation, et la simplicité qu'on y propose, laquelle offre une base de connaissances suffisante pour construire un système tutoriel efficace.

Mais, malheureusement, après plusieurs essais, nous avons dû reconnaître l'insuffisance de certains outils. D'ailleurs, leurs concepteurs ont averti que ces outils ne peuvent être destinés à faire un produit fini. En effet, même s'ils ne demandent pas un matériel puissant, ni un système d'exploitation très avancé, ces outils ne peuvent être utilisés, dans ses versions actuelles, pour développer un système tutoriel intelligent à cause de la non compatibilité avec des langages orientés objets comme Java. Toutefois, nous avons choisi d'utiliser un outil puissant, *Amine Platform*, dédié au développement de systèmes intelligents. Nous exposons dans la suite cet outil qui nous a facilité grandement le développement de notre système.

4.2.2 Description d'outils de développement

4.2.2.1 Plate-forme Amine

Comme nous l'avons souligné dans le chapitre 1, le développement du système TIDES est basé sur l'outil *Amine Platform*[10] (Kabbaj et al., 2006). Cet outil, multicouche et Open Source, est un environnement ouvert de développement de

[9] CLIPS (C-Language Integrated Production System) repose sur le langage C ainsi que sur un autre langage de programmation (inspiré de Lisp) pour le développement de son moteur d'inférences.
[10] Pour plus de détails : http:// amine-platform.sourceforge.net/

différents types de systèmes (systèmes intelligents, systèmes à base de connaissances, systèmes à base d'ontologie, applications basées sur les graphes conceptuels, et agents intelligents). Les composants intégrés dans cette plate-forme présentent l'avantage d'offrir aux concepteurs pédagogiques un environnement homogène. Ils permettent également aux chercheurs d'explorer librement de nombreuses pistes, puisqu'ils mettent à leur disposition une bibliothèque très riche de classes java qui peut faciliter leur tâche de développement. En revanche, ils leur demandent un plus gros effort puisqu'il s'agit de maîtriser les composants pour développer un environnement complet.

La plate-forme Amine est composée de quatre couches qui forment une hiérarchie (figure 4.1). Chaque couche peut être utilisée avec les autres couches, comme elle peut être utilisée de manière indépendante.

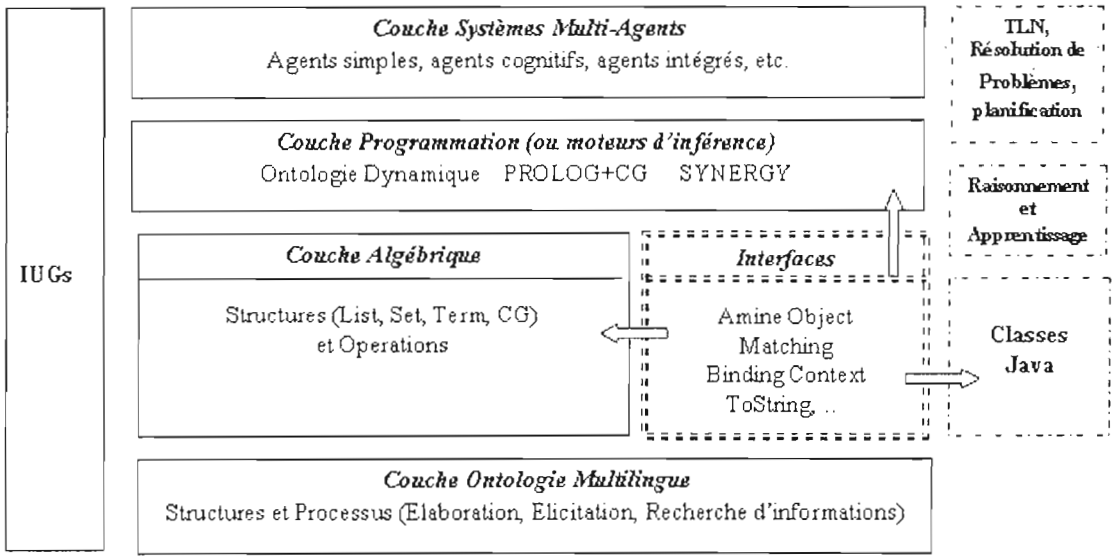


Figure 4.1. Architecture de la plate-forme Amine (version 3)

La figure 4.1 montre l'architecture de la version Amine 3 avec les quatre couches. La première couche constitue le noyau et concerne la création, l'édition, la mise à jour et la manipulation d'ontologies multilingues. La deuxième couche algébrique fournit plusieurs types de données élémentaires, et composées de plusieurs opérations associées. La troisième couche offre des mécanismes d'inférence

incluant des langages de programmation. La quatrième et dernière couche offre la possibilité d'implémenter les concepts d'agents et systèmes multi-agents.

Amine plate-forme fournit également plusieurs interfaces utilisateurs graphiques (IUG). La figure suivante (figure 4.2) présente la fenêtre principale « Amine Suite Panel » qui permet l'accès à toutes les IUG, ainsi que l'accès à quelques exemples d'ontologie et à des tests qui illustrent l'utilisation des structures fournies et leur API (Application Programming Interface).

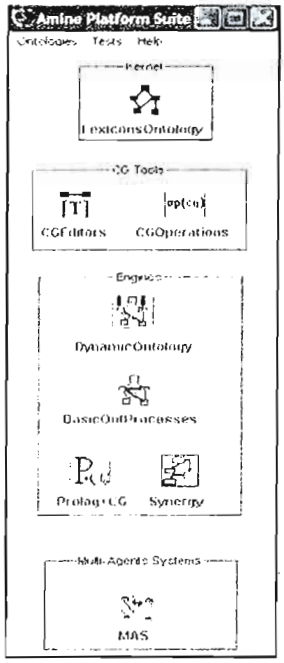


Figure 4.2. Fenêtre principale d'Amine « Amine Suite Panel »

La couche de programmation fournit trois paradigmes de programmation : un paradigme de programmation basée sur la mémoire, un paradigme de programmation à base de règles, incorporé dans le langage PrologPlucCG et un paradigme de programmation visuelle à base d'activation et de propagation, incorporé dans le langage SYNERGY. Cette couche, qui nous a permis de développer notre système, fait l'objet d'une discussion dans la sous-section suivante.

4.2.2.2 Description de l'outil

Avons-nous décidé de construire notre système en langages PrologPlucCG[11] pour construire le moteur d'inférence et Java (JBuilder 2005). Nos premières expériences nous avaient d'ailleurs montré une meilleure adéquation de ces deux langages, pour une telle réalisation, par rapport à certains outils existants. En effet, PrologPlusCG (une extension conceptuelle et orientée objet de Prolog) est un environnement de développement de systèmes à base de connaissances conçu en Java et qui intègre un langage de représentation des connaissances (faits et règles de production), un langage de commande, un moteur d'inférence d'ordre 1 fonctionnant en logique non monotone et une interface utilisateur (sous Unix, Windows ou MacOS). De plus, les avantages industriels du langage Java facilitent le développement d'applications multi-plateformes.

PrologPlusCG comprend un éditeur de texte, un compilateur, un débogueur et un interpréteur. Le compilateur exécute une analyse syntaxique du programme et si l'analyse réussit, il génère un fichier objet contenant une représentation interne du programme en termes de structures Java (Vector, Hashtable, etc.). Ensuite, l'interpréteur opère sur le fichier objet afin de répondre aux requêtes de l'utilisateur.

Un utilisateur de PrologPlusCG interagit avec le système en utilisant deux fenêtres principales. Une première fenêtre permet de créer/ouvrir/éditer un programme alors que la seconde est utilisée comme console pour un langage de commande (figure 4.3).

[11] PrologPlusCG est un langage de **pro**grammation en **log**ique **Plus** Graphe Conceptuel (d'où son nom). PrologPlusCG est open Source et peut être accessible à l'adresse : http ://sourceforge.net/projects/prologpluscg/

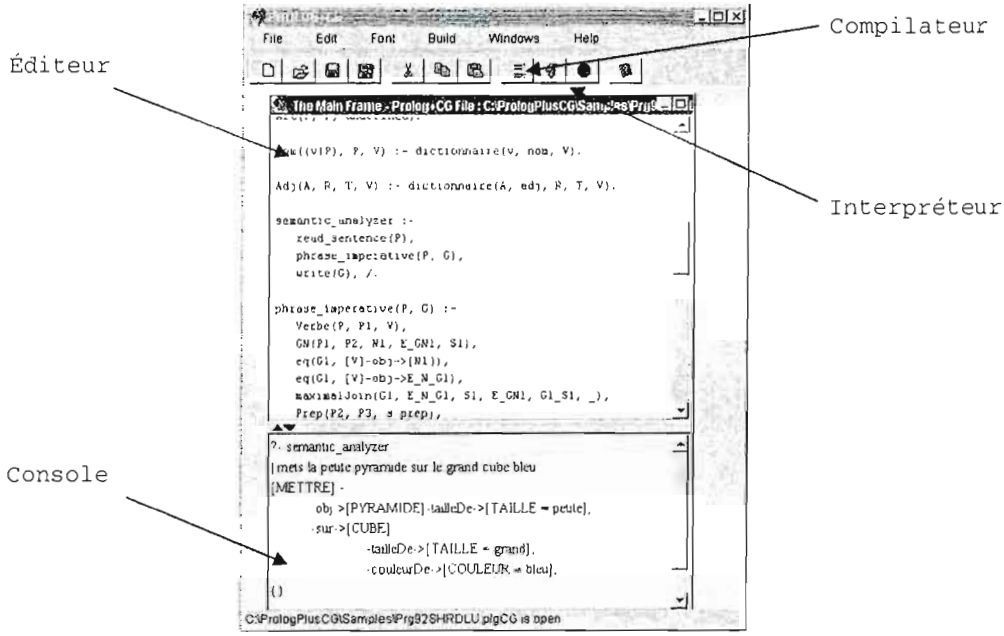


Figure 4.3 Fenêtre d'éditeur et console du PrologPlusCG

PrologPlusCG adopte une notation préfixée pour la formulation d'une expression. Ainsi, l'expression `3+(4-5)` doit être formulée de la manière suivante : `add(3,sub(4,5))`. Par ailleurs, l'opérateur `val(Var,Expr)` évalue l'expression `Expr` et associe sa valeur à la variable `Var`. Par exemple, le code suivant permet d'associer à la variable `x` la valeur de `4+(5*3)` :

```
?- val(x, add(4, mul(5, 3))).
{x = 19}
```

Par ailleurs, afin de permettre aux apprenants d'interagir avec le système de manière facile et ergonomique, nous avons développé le système TIDES en Java. Il comporte environ 4000 lignes de code et une interface appropriée à des utilisateurs de l'âge des enfants (interface avec animation[12]). Cette interface permet principalement à l'élève de saisir son problème de soustraction et d'afficher l'analyse et le diagnostic de ses erreurs détectées. Ceci est rendu possible grâce à l'interfaçage entre Java et PrologPlusCG.

[12] Nous avons utilisé le logo Tweety pour l'expérimentation.

4.3 L'architecture globale du système TIDES

L'architecture du système TIDES (figure 4.4) est modélisée à l'aide du langage de modélisation unifie *UML* (Unified Modeling Language, 2008) et l'outil *Rational Rose* (2003). Elle comporte les modules d'analyse et de diagnostic, la base de connaissances et le profil de l'élève (Danine et al., 2006a).

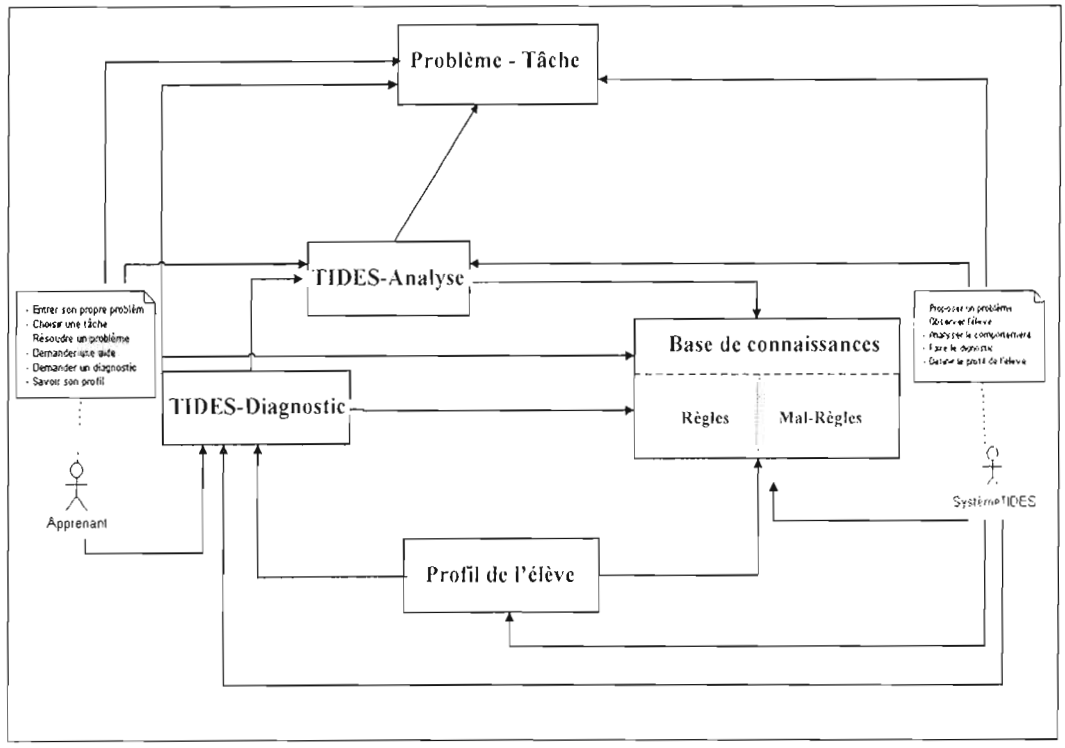


Figure 4.4. L'architecture globale du système TIDES

4.3.1 Description sommaire de chaque module et de son rôle dans l'interaction

Dans cette sous-section nous décrivons brièvement les modules qui constituent l'architecture du système TIDES (figure 4.4), ainsi que les interactions entre ces modules. Cette description sert aussi à expliquer les diagrammes des cas d'utilisation présentés dans la figure 4.7. Nous y reviendrons par la suite pour donner plus de détails sur les fonctionnalités de chaque module. Nous précisons les interactions entre ces modules ; nous expliquons comment les messages

diagnostics sont déterminés et avec quel module ; nous donnons aussi des exemples détaillés qui expliquent comment le système analyse et diagnostique certaines erreurs non systématiques en proposant des problèmes tests et nous montrons, avec des exemples aussi, comment les règles sont utilisées dans tous ces processus.

4.3.1.1 Module Apprenant

Pour commencer sa leçon d'apprentissage, l'apprenant doit choisir une des deux options proposées par le tuteur : option utilisateur où l'apprenant entre lui même son propre problème ou option système où le tuteur propose une tâche à élève (figure 4.5). Dans ce dernier cas, l'apprenant est invité à spécifier lui-même le type de problème qu'il va résoudre selon son

Figure 4.5. Choix du problème

niveau de connaissance (figure 4.5), le tuteur génère ensuite le problème et le présente à l'élève dans l'interface (voir section 4). Dans les deux cas de figure, l'élève doit résoudre le problème et le système recueille ses réponses. Après la résolution de son problème, l'apprenant peut savoir l'analyse de son comportement ainsi que le diagnostic de ses actions. Il peut aussi demander de l'aide (via la base de connaissances) comme il peut savoir son profil (ce qui justifie d'ailleurs les interactions entre les composantes du système). Il faut noter

que, dans ce module, l'élève travaille exactement comme s'il réalise sa tâche avec l'outil papier-crayon, (par exemple, il peut écrire ses emprunts au-dessus d'une colonne comme il le fait habituellement lorsqu'il résout un problème de soustraction avec l'outil papier-crayon).

4.3.1.2 Module Système TIDES

Lorsque l'élève entre son propre problème (section 4), le système TIDES contrôle d'abord les zones de saisie des chiffres (s'ils sont bien des chiffres) et ensuite il vérifie leur position (le grand nombre en haut, le petit nombre en bas et s'ils sont écrits de droite à gauche) (pour un exemple détaillé voir Annexe B).

La résolution d'un problème proposé par le système ou par l'apprenant peut en principe être effectuée par l'élève dans un mode d'*action*. Cette activité cognitive de résolution de problèmes est difficile en elle-même car elle demande à l'élève de pouvoir faire le tri dans ses connaissances pour arriver à élaborer une solution. Pour faciliter un peu cette tâche, nous avons incorporé au système TIDES un générateur de classes de problèmes (figure 4.6) pour chacune desquelles on dispose de connaissances simples menant à la solution. Ces classes sont organisées hiérarchiquement en une classification. Pour résoudre un problème, l'élève doit reconnaître d'abord à quelle classe correspond le problème, puis appliquer les connaissances associées à la classe. La figure suivante (figure 4.6) montre les différentes classes de problèmes générés par le système.

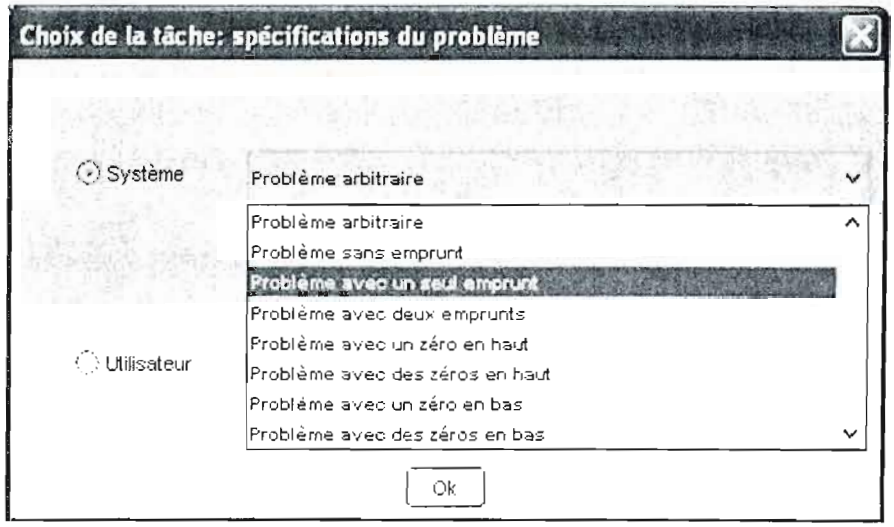


Figure 4.6 : génération de problèmes par le système suivant la classe de problème

Les connaissances liées aux problèmes sont organisées et adaptées aux caractéristiques de l'apprenant. Ces caractéristiques pouvant être la connaissance de l'élève sur un concept spécifique ou sur un ensemble de concepts interdépendants. Les connaissances ne sont pas présentées séquentiellement à l'élève. Ce dernier peut passer à la connaissance suivante même s'il n'a pas bien assimilé la connaissance courante.

Toutefois, cette classification est très utile aussi pour analyser et évaluer une solution fournie par l'élève et donner des explications claires de la démarche de sa résolution. De plus, le système TIDES, en utilisant cette classification, pourra poser un diagnostic précis, ainsi par exemple, il pourra distinguer une erreur conceptuelle d'une erreur d'application d'une procédure ou d'une erreur de raisonnement logique d'une erreur de regroupement en utilisant la base de connaissances et il détermine enfin le profil de l'élève (ce qui justifie d'ailleurs les interactions entre cette composante et les autres composantes de l'architecture du système).

Il est à noter que ces classes de problèmes constituent les éléments de base d'un ensemble de tâches. Ces tâches sont structurées en plusieurs types. Par exemple, les premières tâches à réaliser pour résoudre un problème étant « savoir à quelle classe appartient le problème », puis « déterminer les procédures de

résolution du problème ». On peut toutefois, découper la structure d'une tâche en petites tâches successives ce qui permet à l'élève d'assimiler et de comprendre plus facilement ces connaissances et de travailler sur des parties spécifiques d'un problème et d'améliorer ses compétences.

4.3.1.3 Module Base de connaissances

Ce module joue un rôle essentiel dans cette architecture du système. En effet, il contient toutes les connaissances nécessaires du domaine d'apprentissage qui doivent être mises en œuvre par le système TIDES. Ces connaissances, qui ont été définies et rendues plus modulaires, permettent d'augmenter sensiblement l'efficacité de l'analyseur et du diagnostiqueur, de générer les messages d'analyses et de diagnostics appropriés et de faciliter la construction des profils d'élèves. D'autres connaissances ont été ajoutées pour développer davantage les informations associées à chaque règle et, d'autre part pour représenter les différentes erreurs commises par les élèves. Cet ensemble de connaissances constitue les connaissances du domaine d'apprentissage du système TIDES. Nous exposons dans la suite ce module en détail avec des exemples.

4.3.1.4 Module TIDES-Analyse

Afin d'interpréter les solutions formulées par l'élève, le système TIDES est doté d'un analyseur dont les connaissances se trouvent dans les règles de la base de connaissances. Ces règles ont une structure qui permet une analyse très fine de la production de l'élève et peuvent produire des messages interactifs. Ce module prend en entrée les réponses de l'élève aux différentes tâches, les interprète et les présente à l'élève. Pour ce faire, le système compare la réponse de l'apprenant avec celle du tuteur et essaye de trouver les stratégies utilisées par l'élève dans sa résolution du problème. Cependant, si le système trouve des erreurs ambiguës, il invite l'élève à résoudre certains problèmes tests qui sont liés à sa conception erronée (un exemple détaillé qui explicite cette démarche est donné un peu plus loin, sous-section 6.5).Toutefois, l'analyse porte avant tout sur la détermination de

comment fait l'apprenant. Elle facilite le diagnostic qui permet d'identifier les sources des erreurs.

4.3.1.5 Module TIDES-Diagnostic

Le système TIDES est doté aussi d'un diagnostiqueur qui peut inférer les informations du module de l'élève à partir de ce qui est perçu (par l'analyseur) du comportement de celui-ci, c'est-à-dire d'effectuer un diagnostic et une interprétation du comportement de l'apprenant. Pour cela, il utilise des règles qui comportent des connaissances permettant aussi des messages diagnostiques et des interventions tutorielles mieux adaptés aux fonctions du système TIDES (nous expliquons un peu plus loin comment les messages diagnostics sont générés). De plus, et après cette interprétation du comportement de l'élève, il peut préciser les causes réelles qui ont conduit l'élève à une production erronée. Le diagnostic porte sur la détermination de *pourquoi* il l'a fait. Il établit aussi le profil de l'élève à partir de l'analyse fournie par le module TIDES-Analyse.

4.3.1.6 Module Profil de l'élève

Précisons que le profil de l'élève correspond en fait, à la conception qu'a le système sur l'apprenant. Il s'agit d'établir une description exacte qui donne le modèle de l'élève. Cette description est de plus haut niveau, rendant compte des compétences et des connaissances en arithmétique de l'apprenant.

La construction du profil de l'élève se fait en trois temps. L'apprenant passe tout d'abord une séance d'apprentissage (où il doit résoudre un problème, entré par lui ou proposé par le système). Ensuite, le tuteur analyse les tâches, interprète les productions et diagnostique les actions de l'apprenant en utilisant la base de connaissances. Enfin, les enregistrements constitués au cours de la séance d'apprentissage ont permis de recueillir des données résumant les interactions entre l'élève et le système. De fait, l'utilisation du traçage de modèle d'Anderson a rendu la présentation du profil de l'élève claire et efficace et a justifié le diagnostic que le système a établi. Notons que toutes les actions de l'élève dans le système TIDES sont enregistrées dans un fichier trace (voir chapitre 5). Ce

fichier facilite l'analyse et le diagnostic des comportements de l'élève face à un problème. Il faut noter aussi que le profil de l'élève évolue au fil des interactions avec le système.

4.3.2 Diagrammes détaillés des cas d'utilisations

Les cas d'utilisation représentent les fonctionnalités que le système doit savoir faire. Chaque cas d'utilisation décrit un ensemble d'interactions successives d'une entité en dehors du système (Apprenant) avec le système lui-même pour réaliser une fonctionnalité. Les diagrammes détaillés des cas d'utilisation sont illustrés dans la figure 4.7 suivante.

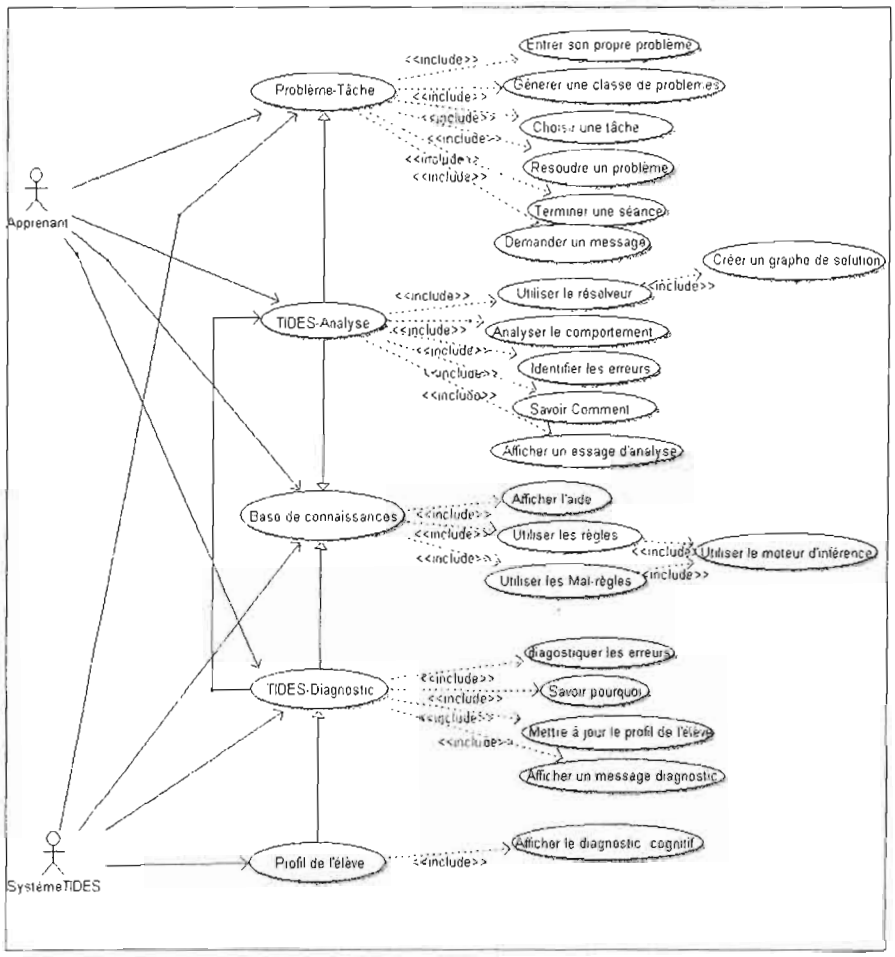


Figure 4.7. Diagrammes détaillés des cas d'utilisation

Dans la suite nous présentons plus en détail les fonctionnalités de chaque module avec des exemples.

4.4 Présentation de l'interface du système TIDES

4.4.1 Interface du système TIDES

L'interface est le lieu de communication entre l'élève et le système. Sa conception doit satisfaire au moins à trois contraintes : le système doit être simple à utiliser et ne doit pas nécessiter beaucoup de temps pour la prise en main, elle doit permettre une manipulation directe de problème équivalente au fonctionnement habituel de l'élève en environnement papier–crayon. Enfin, l'interface doit, par les observables qu'elle fournit, rendre possible l'analyse et le diagnostic.

Pour créer sa propre solution, l'élève doit interagir avec le système et ses interactions sont le reflet de son comportement au cours de son apprentissage. Pour une analyse des comportements fondée sur les interactions entre l'élève et le système, il est nécessaire que l'interaction soit déterminante. En effet, la résolution du problème dépend en grande partie de la capacité de l'élève à interagir correctement avec le système pour trouver une solution. Par conséquent, les interactions traduisent bien le comportement de l'apprenant face à la résolution de son problème.

Plusieurs modalités relatives à la gestion d'interaction et de décomposition des tâches sont abordées et discutées ci-dessous. Ces éléments de l'interface constituent des facteurs critiques eu égard aux prescriptions du modèle d'Anderson touchant la charge imposée à la mémoire de travail. Ils déterminent plusieurs aspects de l'interaction entre le système et l'élève durant l'étape de résolution du problème. Les aspects suivants sont notamment abordés :

1. spécifications et présentations du problème à l'élève ;
2. bloc de résolution comme zone de travail de l'élève ;
3. formulation d'actions par l'élève ;
4. proposition du vocabulaire à l'élève ;
5. accès aux autres informations accessibles.

4.4.2 Spécifications du problème

L'interface de résolution de problèmes doit faciliter à l'élève la formulation de sa solution à un problème posé par le système, mais surtout de profiter au mieux de cette activité pour améliorer ses connaissances et ses capacités de résolution. Pendant tout le processus de résolution, l'élève a sous les yeux les spécifications du problème à résoudre et notamment le but poursuivi, il doit effectuer des opérations de transformation permettant de passer d'un état initial à un état final désiré. Un exemple de présentation d'une tâche à l'élève, en suivant l'enchaînement des fenêtres et sous-fenêtres est fourni dans la fenêtre *spécifications du problème* de la figure 4.8. Évidemment, toutes les sous-fenêtres existantes ne seront pas montrées ici, seulement celles utilisées dans cet exemple type. L'affichage permanent de ces informations diminue la charge imposée à la mémoire de travail de l'élève pendant le processus de résolution.

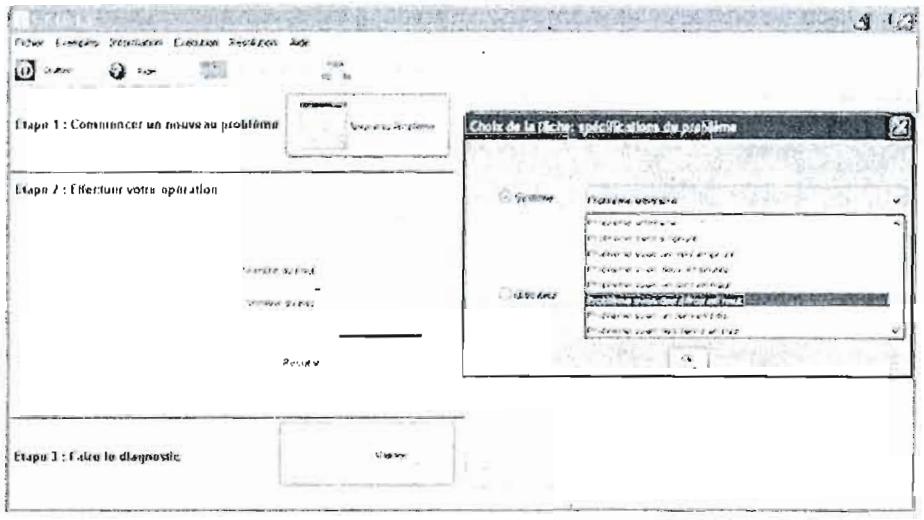


Figure 4.8. Organisation des fenêtres en mode résolution : exemple d'une tâche simple

4.4.3 Bloc de résolution (zone de travail)

4.4.3.1 Affichage de bloc de résolution

L'interface est une fenêtre 960x500, qui gardera la même apparence tout au long de la résolution du problème. Des fenêtres apparaîtront et disparaîtront à

l'intérieur de celle-ci, comme la sous-fenêtre de choix d'une tâche dans la figure 4.8. Grâce à ce mode d'affichage, il est possible d'afficher le bloc de résolution utilisé par l'élève ainsi que les autres informations essentielles. Pour ce faire, l'interface est divisée verticalement en deux parties de dimensions à peu près égales. La partie de droite est principalement constituée d'une fenêtre du traçage de modèle tandis que celle de gauche est constituée d'un bloc de travail pour inscrire la tâche et les actions de l'élève (figure 4.9).

L'élève peut accéder à différents modes pendant sa séance d'apprentissage : *aide*, *exemples*, *informations*, *exécution*, *résolution* et *sortie*. C'est à l'aide d'une barre de menus apparaissant dans la partie supérieure de l'écran que l'élève choisit le mode dans lequel il désire travailler (Voir annexe B pour un exemple détaillé). Chacun des éléments de cette barre (choix de premier niveau) peut faire l'objet de l'ouverture d'une fenêtre de type menu déroulant à partir de laquelle l'élève peut effectuer un choix de second niveau (figure 4.9).

Le mode *Aide* lui présente les procédures et l'algorithme de la résolution d'un problème de soustraction. Le mode *Exemples* lui permet d'examiner quelques exemples de problème avec solution commentée. Le mode *Informations* lui donne accès aux informations sur les consignes de résolution ainsi que sur le mode et le savoir-faire du système TIDES. Avec le mode *Exécution,* il peut obtenir un graphe de solution formulée (dans le cas d'erreurs). Mais d'une façon générale, l'élève travaille en mode *Résolution* en utilisant le bloc de résolution pour formuler sa solution.

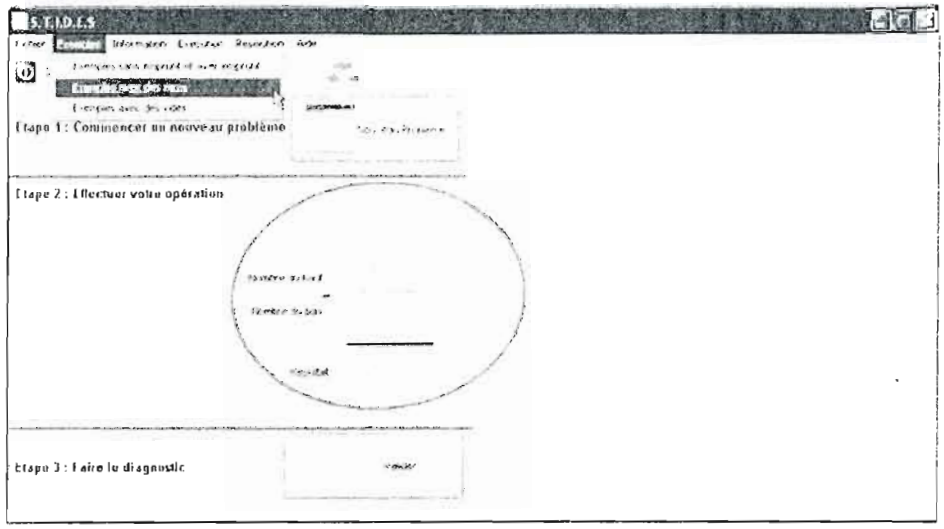


Figure 4.9. Bloc de résolution (zone de travail) et un sous-menu

En ce qui concerne l'interface graphique utilisée pour faciliter la démarche de solution à l'élève pendant le processus d'apprentissage, le bloc a d'abord été identifié comme l'unité de résolution du problème sur lequel travaille cet élève. Ainsi, le bloc définit le problème dans son ensemble en exprimant la tâche générale à accomplir et en fournissant à l'élève un espace de travail pour formuler les sous-tâches requises.

4.4.3.2 Laisser l'élève choisir

Comme nous l'avons souligné dans la section 3.1.1 (module Apprenant), une autre option a également été implantée dans le système TIDES, elle consiste à laisser l'élève choisir son mode de travail, entre l'affichage de la tâche par le système et l'entrée du problème par l'élève lui-même (figure 4.8). Dans les deux cas, l'élève travaille dans un bloc de résolution actif avec curseur et possibilité d'édition.

Il est à noter que dans un tel système, c'est l'élève qui gère son espace-travail avec les avantages qu'une telle autonomie apporte sur le plan pédagogique, mais avec aussi le fardeau additionnel que cette gestion impose sur la mémoire de travail de l'élève.

4.4.4 Formulation des actions

L'élève qui est en train de résoudre un problème opère toujours au niveau d'un bloc de résolution. Il peut choisir de formuler une action en pointant les cases appropriées. À chaque fois que l'élève choisit une case, le curseur apparaît et la case devient active. Le processus continue jusqu'à ce que l'élève ait signifié qu'il a complété la formulation de l'action en cours. De telles formulations libres sont sauvegardées dans un fichier trace pour améliorer les connaissances futures du système d'une part, et d'autre part, pour faciliter l'analyse et le diagnostic du système TIDES.

4.4.5 Vocabulaire proposé à l'élève

Au départ, la nature du vocabulaire proposé à l'élève par le système, peut se montrer incompréhensible pour un élève de primaire. Il a donc fallu l'adapter au niveau de l'élève en utilisant des phrases très simples, susceptibles de faire passer des messages liés à certaines tâches compliquées.

Cependant, nous avons eu quelques difficultés à construire des phrases simples, en quelques mots, pour faire passer un message d'analyse ou de diagnostic, surtout lorsque la tâche peut être décomposée en plusieurs sous-tâches. À ce moment le niveau d'abstraction devient plus élevé. De plus, l'espace alloué aux fenêtres des messages est très limité, il est donc impératif de faire des phrases courtes.

Pour déterminer la nature du vocabulaire proposé, différentes options ont été explorées, notamment : pour limiter le vocabulaire proposé aux règles et mal-règles d'une tâche particulière plutôt qu'à celles de l'ensemble des tâches, et d'ajouter au vocabulaire proposé seulement les formulations des phrases les plus courantes.

Certains ajustements ont été apportés au vocabulaire proposé à la suite d'un test préliminaire du système. Cependant, après l'analyse des actions des élèves à la suite de la mise à l'essai, il sera peut être nécessaire d'ajouter quelques expressions faciles au vocabulaire proposé à l'élève pour que ce dernier

comprenne plus facilement les messages d'analyses et de diagnostics de ses erreurs.

4.4.6 Accès aux autres informations

L'élève peut en tout temps accéder à des informations sur les spécifications du problème en cours, sur un des problèmes types et sa solution commentée, sur les différentes aides disponibles dans toutes les fenêtres, et enfin sur les caractéristiques du système TIDES. Il peut également accéder à des fonctions comme l'explicitation de l'utilisation du système, la présentation des algorithmes de résolution des problèmes et l'exécution pas-à-pas de la solution formulée. Il peut enfin demander un diagnostic de ses erreurs pour en savoir les causes.

4.5 Présentation du module Base de connaissances

4.5.1 Connaissances du système développé

Les systèmes tutoriels intelligents se distinguent des systèmes experts par les connaissances particulières dont ils sont dotés pour remplir des fonctions pédagogiques spécifiques. Ces connaissances rendent un système tutoriel capable d'interventions pédagogiques adaptées au comportement de chaque élève et fondées sur une stratégie tutorielle.

Le développement du système TIDES a permis d'identifier plus spécifiquement les connaissances à utiliser à tous les niveaux, au niveau des connaissances du domaine, de l'analyseur, du module tutoriel et diagnostic et du profil de l'élève, pour en faire un champ d'application du modèle d'Anderson. Ces connaissances incluent les suivantes :

- connaissances du domaine découpées en unités correspondant au processus de résolution de l'élève et représentant la solution du système, et les autres solutions (équivalentes, générales ou erronées) ;
- connaissances permettant l'analyse d'une solution partielle ou complète de l'élève ;

- connaissances associées aux interventions tutorielles et permettant de déterminer la nature de ces interventions et de générer les messages diagnostiques ;
- connaissances permettant la mise à jour du modèle de l'élève en fonction des résultats de l'analyse du comportement de l'élève et du diagnostic.

4.5.2 Connaissances d'arithmétique et traçage de modèle

Dans la présente section, nous identifions les connaissances en arithmétique et nous les caractérisons aussi en tant que domaine d'apprentissage d'un système d'EIAO. C'est en fonction de ces caractéristiques que nous avons ensuite abordé différents problèmes associés à la conception et au développement du système TIDES en tant que système d'EIAO de l'arithmétique incorporant le modèle d'apprentissage d'Anderson.

L'une des prescriptions du modèle d'Anderson implique que *le système d'apprentissage sélectionne les différentes tâches à exécuter en fonction de l'état des connaissances de l'élève et des connaissances requises pour accomplir ces tâches.* Anderson a élaboré la méthodologie du traçage de modèle dans le but de guider cette démarche. En cherchant à appliquer cette méthodologie au système TIDES, plusieurs difficultés surgissent. C'est là l'objet des sous-sections suivantes.

4.5.2.1 Identification des connaissances à acquérir dans TIDES

Plusieurs types de connaissances sont manifestés par l'élève pendant qu'il résout un problème. Il y a d'abord les connaissances liées à l'utilisation des différentes stratégies de résolution. Il s'agit essentiellement de choisir la stratégie qui convient, compte tenu du but poursuivi, puis de manipuler correctement cette stratégie. Pour les fins du modèle de l'élève, des connaissances spécifiques sont associées à chaque stratégie de résolution. Elles correspondent davantage à la capacité d'appliquer la bonne stratégie au bon moment. Les connaissances comptabilisées dans le modèle de l'élève sont donc associées aux différentes stratégies de résolution. Ces connaissances sont ainsi les suivantes :

- Action simple (pas de stratégie).
- Actions multiples (stratégie numérique : numération et sens de l'opération).
- Séquences d'actions pour obtenir une solution (stratégie algorithmique).
- Stratégie conditionnelle (**SI** le chiffre du haut est plus petit que celui du bas **ALORS** emprunter).
- Stratégie itérative (**RÉPÉTER** *deux* **FOIS** l'emprunt, si les deux chiffres du haut sont plus petits que ceux du bas).
- Stratégie alternative (**SI** le chiffre du haut est plus grand que celui du bas **ALORS** faire la soustraction **SINON** emprunter).

D'autres types de connaissances ont trait au niveau de complexité des problèmes et sont associés à la capacité de formuler des stratégies de résolution correctes ainsi qu'à la capacité d'abstraction manifestée par le fait de nommer des actions complexes durant le processus de décomposition d'un problème. Pour appréhender ce type de connaissances, les différentes actions à accomplir pour résoudre les problèmes du système TIDES ont été divisées en trois niveaux de complexité en fonction de stratégies de résolution qu'elles impliquent. Il est ainsi possible de représenter, avec ces trois connaissances additionnelles, les capacités manifestées par les élèves compte tenu des niveaux de difficultés rencontrées. Ces connaissances nouvelles sont les suivantes :

- Niveau de complexité 1 : problème avec un seul emprunt
- Niveau de complexité 2 : problème avec deux emprunts
- Niveau de complexité 3 : problème d'emprunt sur un seul zéro ou deux zéro

Le niveau de complexité d'une tâche donnée est fonction des séquences d'actions (élémentaires ou complexes) requises (à l'aide des stratégies de résolution) pour accomplir cette tâche. La détermination exacte des trois niveaux de complexité sera abordée plus en détail dans le chapitre suivant section 3.1.

4.5.2.2 Représentation des connaissances à acquérir dans TIDES

Les règles de production représentent-elles vraiment les connaissances que le système veut faire acquérir aux élèves? Il est attendu qu'un élève maîtrise l'utilisation des procédures de résolution après avoir complété avec succès une séance d'apprentissage. Cette maîtrise implique notamment l'utilisation de stratégies à plusieurs niveaux. En effet, et pour gérer l'évolution des connaissances de l'élève dans le modèle de l'élève, on pourrait retrouver dans ce modèle une habileté telle que «*l'utilisation d'une stratégie conditionnelle à l'intérieur d'une stratégie itérative*». Comme, par exemple, **SI** le chiffre du haut de la première colonne est plus petit que celui du bas dans la même colonne **ALORS** emprunter (stratégie conditionnelle) et **SI** le chiffre du haut de la deuxième colonne est plus petit que celui du bas dans la même colonne **ALORS** emprunter une deuxième fois (une autre stratégie conditionnelle, mais le tout représente une stratégie itérative). Dans les règles de production, une telle habileté est représentée par un ensemble de règles. Une telle représentation des connaissances est parfaitement adaptée à l'approche par décomposition d'une tâche.

L'approche finalement adoptée est davantage conforme à un élément-clé de la théorie ACT-R d'Anderson, à savoir que *les règles de production représentent véritablement les connaissances que l'élève doit acquérir.* Une telle correspondance entre les règles de production et les connaissances à maîtriser n'est pas évidente, du moins à première vue. En effet, on compte plus de 200 règles dans le système TIDES alors que le nombre de connaissances à maîtriser atteint à peine une douzaine. À cet égard, il faut se rappeler que c'est à une unité de connaissances (l'ensemble des règles partageant un même but) et non pas à chacune des règles individuelles qu'il faut associer une connaissance à maîtriser. En effet, l'ensemble des règles d'une unité de connaissances correspond à tous les comportements, corrects et incorrects, manifestés par un élève et à partir desquels on peut inférer qu'il maîtrise bien, assez bien ou pas du tout l'habileté sous-jacente ainsi que sa compétence par rapport à cette habileté.

Pour les fins du traçage de modèle de l'élève, des connaissances ont été identifiées et incorporées au système sous forme de règles de production comme celles associées à l'utilisation pertinente des différentes stratégies de résolution et celles ayant trait au niveau de complexité des problèmes.

4.5.3 Structure d'une règle dans le système TIDES

Chaque règle du système TIDES est constituée d'une tâche qui peut être décomposée en un ensemble de sous-tâches correspondant à cette décomposition. De plus la structure d'une règle du système est définie de telle façon qu'elle comprend au moins les éléments suivants :

1. un identificateur de la règle ;
2. la tâche assignée à la règle ;
3. la décomposition de la règle en sous-tâches ;
4. le niveau de complexité de la tâche ;
5. la connaissance arithmétique à laquelle elle se réfère ;
6. le message d'analyse ou de diagnostic associé à son déclenchement.

Les trois premiers éléments de la structure d'une règle ont pour but de vérifier, d'une part, la présence de la règle dans la base de connaissances du système TIDES pour la déclencher au besoin. D'autre part, de faciliter la vérification du cheminement possible que pourrait emprunter un apprenant pour résoudre un problème. Il est possible de vérifier aussi dans la base de connaissances du système la présence d'un élément-clé du formalisme de la représentation des connaissances du modèle d'Anderson, un *ensemble de règles structurées hiérarchiquement en fonction du rôle explicite de chacune.*

Cependant, les trois derniers éléments sont essentiels dans système TIDES. L'élément 4 (niveau de complexité de la tâche) sert à limiter l'espace-problème lors des processus d'analyse et de diagnostic, évitant ainsi de rechercher d'une façon aveugle soit une explication, soit une interprétation du comportement d'un élève. Pour apprécier l'importance de cet élément, il faut se rappeler que dans le système TIDES, c'est toujours à un certain niveau de l'espace-problème que se trouve la tâche ou la sous-tâche sur la décomposition de laquelle l'élève travaille.

La possibilité d'identifier ce niveau permet d'augmenter sensiblement l'efficacité de l'analyseur et du diagnostiqueur en restreignant le champ de recherche pour expliquer ou interpréter le comportement de l'élève. Cette question est abordée plus en détail en sections 6 et 7.

L'élément 5 (connaissance arithmétique à laquelle la règle se réfère) permet au système de mettre à jour les connaissances du modèle de l'élève et de choisir la tâche qui lui est proposée en tenant compte des connaissances acquises par l'élève et des connaissances requises pour solutionner les différents problèmes possibles. Cette question est associée de près à l'arithmétique comme domaine d'apprentissage et est déjà discutée dans la sous-section 5.2.

Enfin l'élément 6 (message d'analyse ou de diagnostic associé au déclenchement de la règle) contient les connaissances qui permettent de générer les messages d'analyses et de diagnostics appropriés étant donnés les comportements manifestés par l'élève. La génération de ces messages est détaillée en section 5.

4.5.4 Unité de connaissances dans le système TIDES

Un ensemble de règles ayant un même but constitue une *unité de connaissances* au sens d'Anderson (1983). Ces règles représentent en fait différentes façons, correctes et incorrectes, d'atteindre ce but. Une unité de connaissances peut ainsi comprendre quatre types de règles :

- des *règles de références* : utilisées par le système pour générer correctement les solutions de références ;
- des *règles équivalentes* : qui occasionnent la génération d'une solution équivalente à la solution du système (solution de référence) ;
- des *règles généralisantes* qui conduisent à une solution correcte mais plus générale que celle qui est demandée à l'élève ;
- des *mal-règles* qui provoquent la génération d'une solution erronée et permettent l'identification d'une erreur dans la réalisation d'une tâche par un élève.

Les règles de références correspondent aux premières règles d'une unité de connaissances, celles qui sont déclenchées d'abord et qui contribuent à la solution de références. Les autres règles d'une unité de connaissances représentent les autres comportements observables chez un élève cherchant à accomplir la même tâche. Il peut s'agir de comportements erronés mais il peut aussi s'agir d'une action ou stratégie équivalente ou différente.

À titre d'exemple, voici quelques règles faisant partie d'une même unité de connaissances et exécutant la tâche (voir Annexe D): *Emprunter de la colonne 2.*

Règle de référence (*Rule 22)* :
SI LA TÂCHE EST *emprunter de la colonne2*
ALORS *ajouter 10 au chiffre du haut de la colonne1*

Règle équivalente (*Rule 23)*:
SI LA TÂCHE EST *emprunter de la colonne2*
ALORS *décrémenter la colonne 2 de1*

Règle généralisante (*Rule 24)* :
SI LA TÂCHE EST *emprunter de la colonne2*
ALORS *décrémenter la colonne2 de 1 et ajouter 10 au chiffre du haut de la colonne1*

Mal-règle (*Rule 25)* :
SI LA TÂCHE EST *emprunter de la colonne2*
ALORS *décrémenter la colonne suivante à gauche de la colonne 1*

La présence de ces différents types de règles dans le système TIDES reflète la façon d'intégrer un autre élément du modèle d'Anderson : la représentation à l'aide de règles de production des divers comportements d'un élève qui élabore des solutions correctes, plus générales ou erronées à un problème donné.

4.5.5 Construction de la base de connaissances

La base de connaissances du système TIDES a été conçue à partir de recherches expérimentales faites dans le domaine de la résolution des problèmes de soustraction écrite (Roberts, 1968 ; Cox, 1974, 1975 ; Brown & Burton, 1978 ; Brown & VanLehn, 1980, 1982 ; VanLehn, 1990). Cette base de connaissance du système TIDES est construite en PrologPlusCG et contient plus de 200 règles structurées hiérarchiquement en fonction du but explicite de chacune. Ces règles peuvent résoudre environ 88 problèmes types de la soustraction (voir annexe A et annexe D). Les problèmes couvrent tous les programmes des 3 dernières années de

l'enseignement de soustraction dans les écoles primaires. De plus, chaque règle se réfère explicitement à un but pour faciliter la décomposition d'une étape, nécessaire pour résoudre un problème, en une hiérarchie explicite de buts et de sous-buts.

L'incorporation de règles générales pertinentes et la formulation adéquate de ces règles ont permis au système TIDES de réagir dynamiquement dans une situation d'apprentissage donnée. De plus ces règles sont utilisées pour reconnaître les étapes utilisées par un élève pour résoudre un problème afin d'être à même de le suivre et de le guider (l'élève peut éventuellement regrouper plusieurs règles dans une même seule étape de raisonnement). Les types des règles incorporées dans la base du système sont illustrés dans le tableau suivant (tableau 4.1).

Numéro Règle	**Type de règles**	**Exemples**
Règle1 (*Rule 32)*	les règles qui correspondent aux étapes de résolution qu'utiliserait un élève pour résoudre un problème	***Si*** *le but est de faire un problème de soustraction et que ce problème comprend plusieurs colonnes* ***Alors*** *suivre les étapes suivantes :* - *commencer par la colonne le plus à droite* - *s'il n'y a pas d'emprunt, soustraire le chiffre du bas de celui du haut* - *s'il y a emprunt, passer à la colonne suivante* - *décrémenter cette colonne de 1 et changer son chiffre du haut* - *ajouter 10 au chiffre de la première colonne* - *soustraire le chiffre du bas de celui du haut et écrire le résultat de la colonne 1* - *passer à la colonne suivante, faire la même chose que la première colonne et écrire le résultat de cette colonne.* - *passer à la dernière colonne, soustraire le chiffre du bas de celui du haut et écrire le résultat de cette colonne.*
	les règles qui	***Si*** *le but est de faire un problème de soustraction* *et que l'élève commence à le résoudre* ***Alors*** *prendre les démarches suivantes :* - *vérifier si la première colonne nécessite un emprunt et si l'élève l'a cherché de la colonne suivante*

Règle2 (*Rule 43)*	permettent de suivre le cheminement de l'élève pendant le processus de résolution	- *vérifier si la colonne suivante est décrémentée de 1* - *vérifier si 10 est ajouté au chiffre de la première colonne* - *vérifier le résultat de la première colonne* - *passer à la colonne suivante* - *faire la même chose que la première colonne* - *passer à la dernière colonne et vérifier le résultat de cette colonne* - *vérifier le résultat du problème* - *afficher les règles et les mal-règles utilisées par l'élève*
Règle03 (*Rule 48)*	les règles qui permettent au système d'intervenir auprès de l'élève pour l'aider ou le guider en cas d'erreurs (les règles précédentes déterminent les procédures erronées de l'élève ; ces règles présentent de l'aide à l'élève s'il en a besoin	***Si*** *le but est de faire un problème de soustraction* ***Alors*** *prendre comme sous-buts :* - *comparer le résultat de la première colonne avec celui du système* - *s'il y a une erreur, chercher d'où vient cette erreur* - *présenter de l'aide à l'élève lorsqu'il en a besoin* - *continuer à comparer les autres résultats* - *afficher l'aide si l'élève la demande*
Règle04 (*Rule 37)*	les règles qui définissent les différents types d'erreurs, exemple : erreur de type 0 - N = N	***Si*** *le but est décrémenter une colonne qui comprend un 0* ***Alors*** *prendre comme sous-buts* - *vérifier le résultat de la colonne qui comprend 0* - *si le résultat est égal au chiffre du bas de cette colonne, l'erreur est de type 0--N=N*
Règle05 (*Rule 39)*	les règles qui définissent une unité de connaissances	Règle1 ***Si*** *le but est de faire un problème de soustraction* ***Alors*** *soustraire chaque colonne séparément de droite à gauche* Règle2 ***Si*** *le but est d'emprunter* ***Alors*** *décrémenter la colonne suivante à gauche* Règle3 ***Si*** *le but est de décrémenter une colonne* ***Alors*** *soustraire 1 au chiffre du haut et écrire le résultat au-dessus de ce chiffre* Règle4 ***Si*** *le but est d'emprunter* ***Alors*** *ajouter 10 au chiffre du haut de cette*

		colonne

Tableau 4.1. Illustration des règles de la base de connaissances du système TIDES

On outre, ces règles de productions ont les particularités suivantes :

- *elles sont autonomes* : chacune contient à la fois la connaissance et les conditions dans lesquelles elle est utilisable (conclusions et prémisses) ;
- *elles sont indépendantes les unes des autres* : chacune renferme une connaissance limitée mais complète et ne réfère explicitement à aucune autre ;
- *elles sont modulaires* dans le sens où on peut à tout moment ajouter, supprimer ou modifier une règle sans modifier les autres règles.

Chaque règle de la base a été considérée comme une fraction de connaissance, qui est destinée à être combinée avec les autres sous la forme d'un réseau, en reliant ensemble les règles dont la conclusion de l'une apparaît dans les prémisses de l'autre. C'est l'ensemble de ces réseaux qui constitue le corps de la connaissance experte. Afin d'illustrer ce point, prenons (à titre d'exemple) le fragment de connaissance suivant, portant sur la tâche *faire un problème de soustraction* :

Règle1
Si *le but est de faire un problème de soustraction*
Alors *soustraire chaque colonne séparément de droite à gauche*

Règle2
Si *le but est de soustraire une colonne*
Alors *soustraire le chiffre du bas de celui du haut*

Règle3
Si *le but est de soustraire une colonne, et qu'il n'y a pas d'emprunt et que cette colonne comprend un 0 en bas*
Alors *écrire le chiffre du haut comme réponse à cette colonne*

Règle4
Si *le but est de soustraire une colonne, et qu'il n'y a pas d'emprunt et que le chiffre du bas est un vide*
Alors *écrire le chiffre du haut comme réponse à cette colonne*

Règle 5
Si *le but est de faire un problème de soustraction*
et qu'il faut soustraire une colonne
et que cette colonne comprend un 0

et que ce 0 surmonte un vide
et qu'il n'y a pas d'emprunt
Alors *écrire le 0 comme réponse à cette colonne*

Il est possible de relier ces règles entre-elles suivant le principe énoncé plus haut pour former un réseau.

L'avantage que présente la formulation de la connaissance procédurale sous forme des règles de production est que celle-ci se fait de façon modulaire. Chaque règle est un granule de connaissance. Compte tenu de cette granularité, l'introduction de la connaissance peut se faire en vrac, à charge pour le système de reconstruire le réseau complet de la connaissance suivant les modalités vues plus haut.

4.5.6 Utilisation des règles dans le système TIDES

4.5.6.1 Rôle du mécanisme d'inférence

Le mécanisme d'inférence a un rôle important dans le système TIDES. En effet, il exploite méthodiquement la base de connaissances pour exécuter plusieurs tâches : il peut détecter les connaissances utilisées, sélectionner les règles pertinentes à appliquer (et éventuellement les mal-règles), les enchaîner afin de construire un plan de résolution. Le mode de raisonnement de ce mécanisme implique des contraintes et des qualités spécifiques. Par exemple, dans le contexte de l'observation d'un élève effectuant une tâche, le mécanisme d'inférence est en mesure de travailler avec la base de connaissances au fur et à mesure que les informations sur l'élève se présentent, bien qu'il lui faille réserver son jugement puisqu'il demeure délicat de faire une analyse ou de poser un diagnostic précis sans avoir toutes les informations. Il apporte un aspect dynamique au système.

Cependant, ce qui facilite le travail de ce mécanisme, c'est l'encodage des règles en langage PrologPlusCG. Cet encodage a été fait de façon à ce que le mécanisme d'inférence puisse contrôler chaque colonne de l'opération de soustraction à part, comme le montre la figure suivante (figure 4.10). De plus, chaque « bug » de la soustraction (voir annexe A) est représenté par une règle PrologPlusCG.

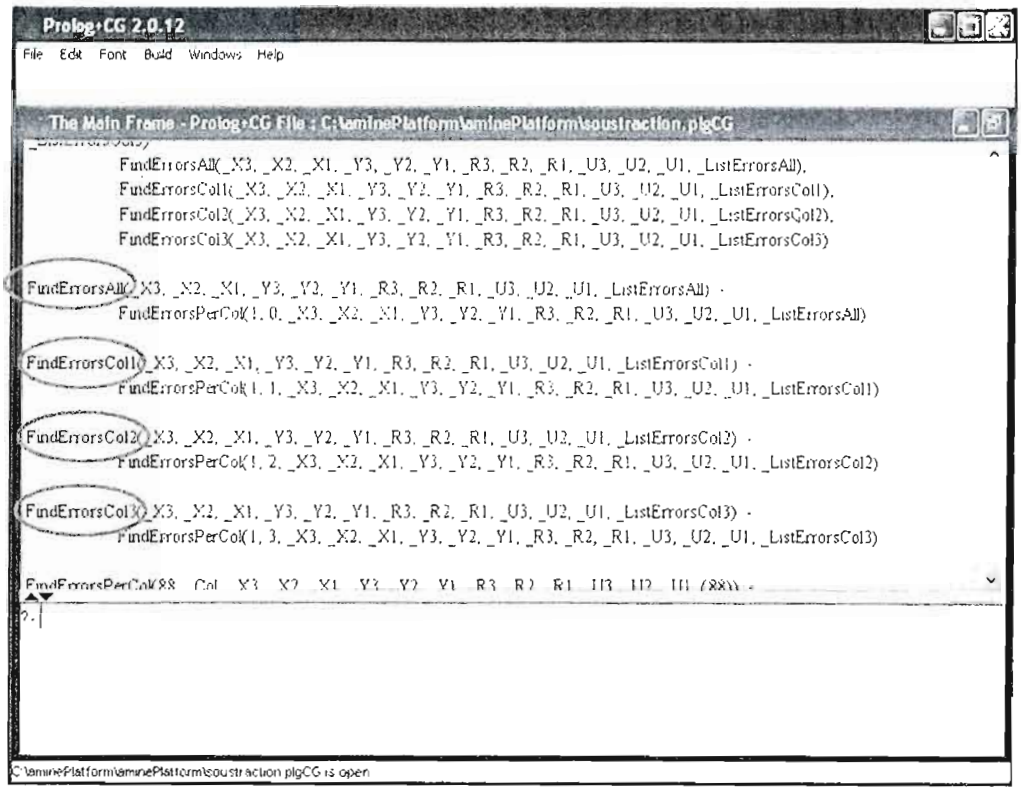


Figure 4.10. Encodage des règles et « bug » par PrologPlusCG. .

4.5.6.2 Exemple d'utilisation des règles

Afin de montrer comment les règles de la base de connaissances du système TIDES sont appelées par le mécanisme d'inférence (soit pour l'analyse, pour le diagnostic ou pour l'aide,...etc.), nous présentons dans la suite un exemple détaillé qui explique ce processus d'utilisation.

Supposons qu'un apprenant, pour résoudre le problème suivant 304-179 (figure 4.11), a procédé de la façon suivante : au lieu d'emprunter pour la première colonne, l'élève écrit zéro comme réponse à cette colonne ; de plus si une colonne dont le chiffre du haut est zéro, il écrit le chiffre du bas comme réponse (0 – N = N) (règle présentée dans le tableau 5.1 précédent).

3 0 4
1 7 9
2 7 0

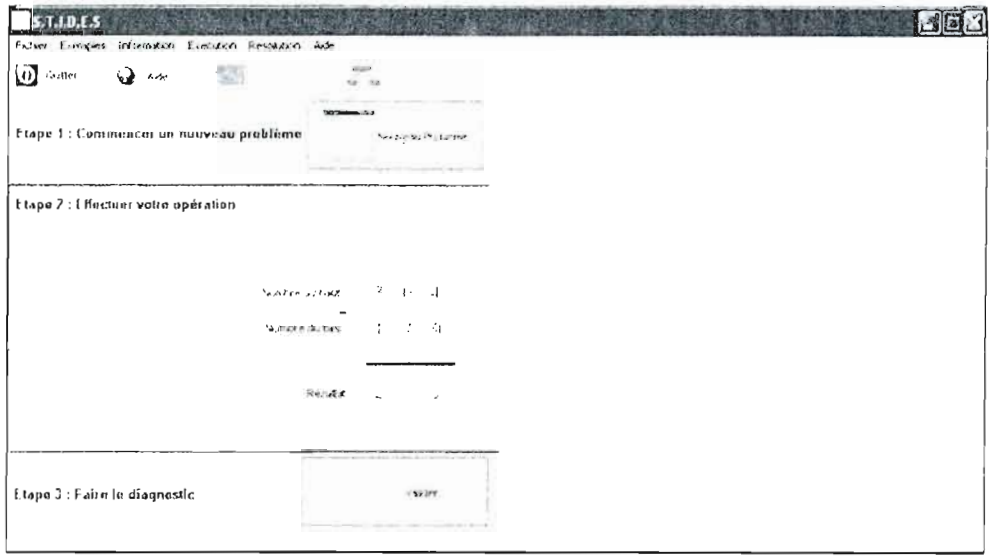


Figure 4.11. Résolution du problème

La compilation de cette procédure de l'apprenant par PrologPlusCG donne le résultat affiché par la figure 4.12 suivante.

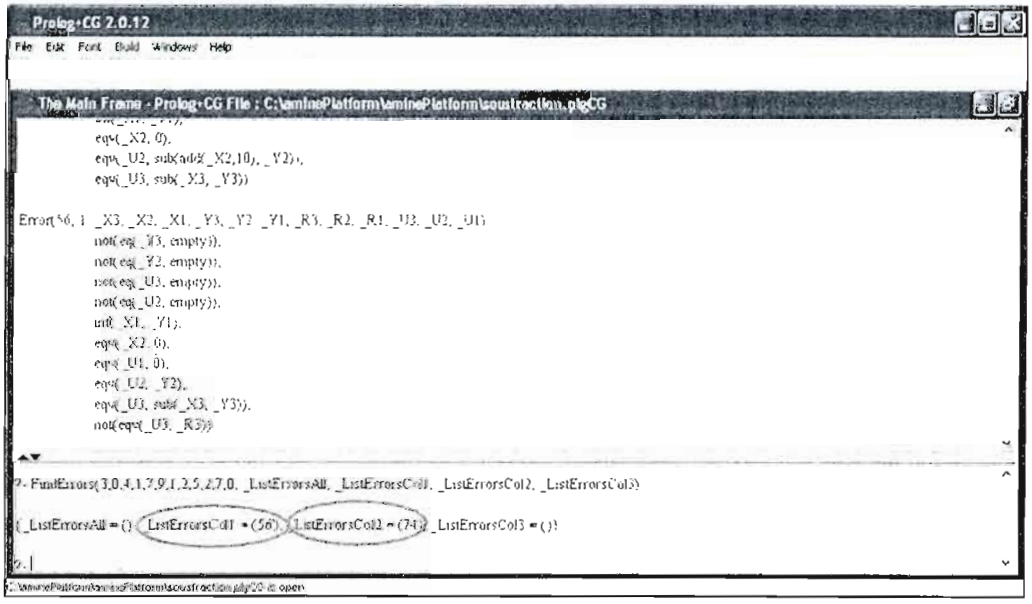


Figure 4.12. Compilation par PrologPlusCG de la procédure de l'apprenant

Ainsi, il nous a été facile de représenter chacun des bugs (voir annexe A et annexe D) de la soustraction par une ou plusieurs règles PrologPlusCG. Notre exemple montre qu'il y a au moins deux règles (mal-règles) qui sont appelées suite au résultat de l'apprenant.

La règle présentée dans la figure 4.12 stipule que si les variables[13] *_Y3*, *_Y2*, *_U3* et *_U2* ne possèdent pas la valeur symbolique « *empty* » et si *_X1* est inférieure à *_Y1* et si *_X2* est égale à 0 et si *_U1* est égale à 0 et si *_U2* est égale à *_Y2* et *_U3* n'est pas égale *_R3*, alors la règle numéro 56 (voir annexe D) est donc satisfaite et sera déclenchée par le mécanisme d'inférence. Ce dernier est entièrement réalisé en PrologPlusCG et nommé *soustraction.plgCG*. Ainsi, à partir de Java, nous chargeons en mémoire le programme PrologPlusCG (Figure 4.13) : *PrologPlusCGFrame.LoadFile("soustraction.plgCG")*

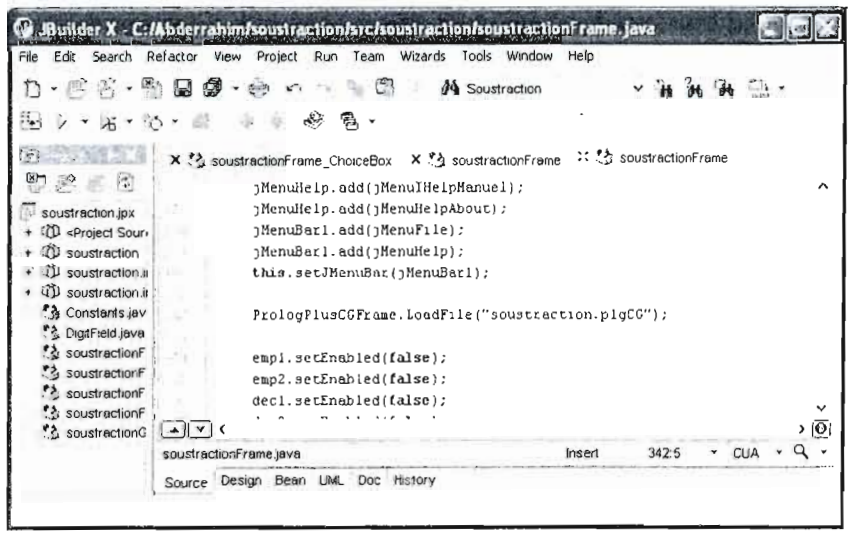


Figure 4.13. Appel du moteur d'inférence par Java

Comme on peut le constater, dès que l'apprenant termine sa résolution de problème, le mécanisme d'inférence commence à chercher dans la base de connaissances les règles et les mal-règles qui expliquent le comportement de l'élève. Si l'apprenant a entré une réponse fausse, un message est affiché pour l'informer que sa réponse est incorrecte, et le mécanisme d'inférence analyse les colonnes une par une pour savoir dans quelle colonne l'erreur s'est produite (nous

[13] Les variables _X1, _X2, et _X3 représentent les chiffres de la première ligne de l'opération, les _Y1, _Y2, et _Y3 représentent la deuxième ligne, les _R1, _R2, et _R3 sont les résultats du systèmes, tandis que les _U1, _U2, et_U3 sont les résultats de l'élève pour chaque colonne.

donnons plus de détails sur ce fonctionnement un peu plus loin, voir module Analyseur). Il commence d'abord par la première colonne ; il vérifie si le résultat de cette colonne est différent de celui du système, d'ailleurs comme le cas ici, il déclenche les règles ou les mal-règles fautives (la mal-règle 56 est déclenchée dans le cas présent, figure 4.12). Ensuite le même processus recommence avec la deuxième colonne ; encore ici, il a détecté une autre mal-règle (mal-règle 74, figure 4.12). Enfin, le mécanisme d'inférence va successivement activer toutes ces mal-règles pour conclure le comportement final de l'apprenant qui a commis ces erreurs en essayant de résoudre son problème (Figure 4.14).

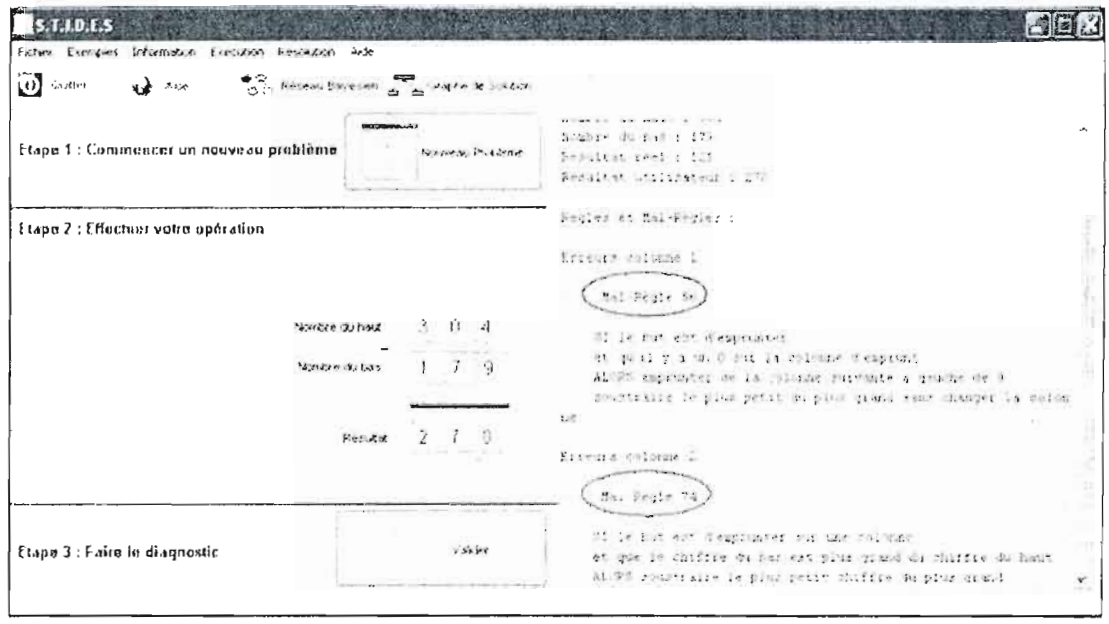


Figure 4.14. Affichage du résultat final par le mécanisme d'inférence

Ainsi, comme le montre la figure 4.15, nous faisons appel à partir du programme Java à la requête principale du mécanisme d'inférence agissant comme si nous utilisions directement PrologPlusCG :

```
     request="FindErrors("+x3String+","+x2String+","+x1String+","
                  +y3String+","+y2String+","+y1String+","
                  +r3Number+","+r2Number+","+r1Number+","
                  +u3String+","+u2String+","+u1String+
 ",_ListErrorsAll,        _ListErrorsCol1,        _ListErrorsCol2,
_ListErrorsCol3)." ;
```

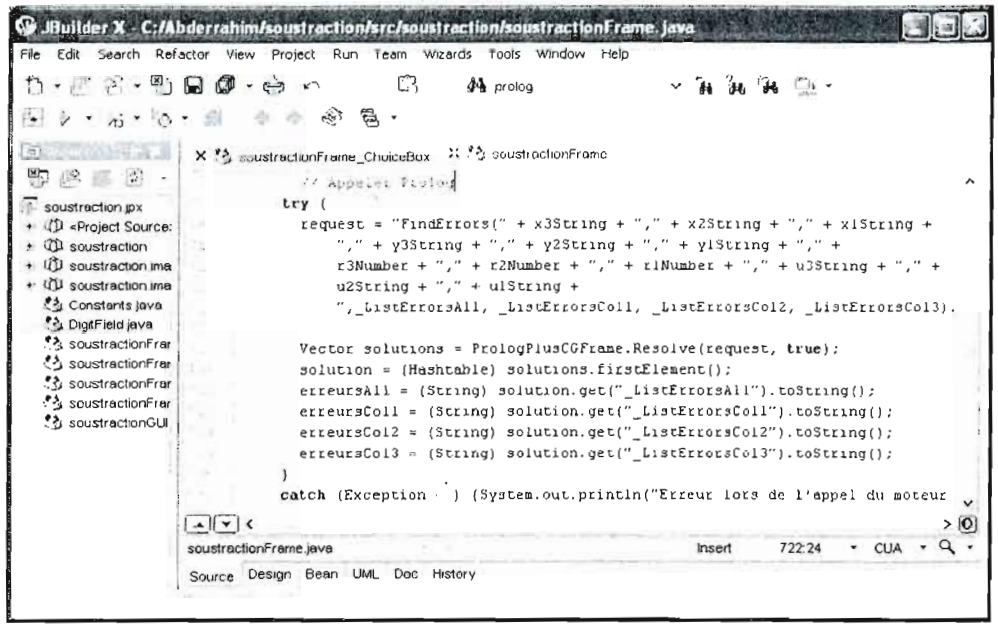


Figure 4.15. Appel d'une requête et récupération de la réponse

Finalement nous récupérons la réponse de l'appel à la requête et nous construisons les listes d'erreurs (ou de règles) :

```
   Vector solutions = PrologPlusCGFrame.Resolve(request, true) ;
   solution = (Hashtable) solutions.firstElement() ;
   erreursAll = (String) solution.get("_ListErrorsAll").toString()
;
   erreursCol1                    =                    (String)
solution.get("_ListErrorsCol1").toString() ;
   erreursCol2                    =                    (String)
solution.get("_ListErrorsCol2").toString() ;
   erreursCol3                    =                    (String)
solution.get("_ListErrorsCol3").toString() ;
```

4.5.7 Génération des messages d'analyse et de diagnostic

Nous avons souligné précédemment que chaque règle de la base de connaissances se réfère à un but clair et précis. De plus, elle contient un message si l'application de cette règle implique une erreur. Ainsi, dès que l'analyseur identifie un ensemble de règles dont l'activation reproduit le comportement de l'élève, il est capable de générer deux messages. Un premier message d'analyse au niveau du premier module (TIDES-Anlyse) qui constitue une première forme de rétroaction et ne concerne que l'identification de la nature de l'erreur seulement (voir figure 4.16 tirée de l'exemple précédent).

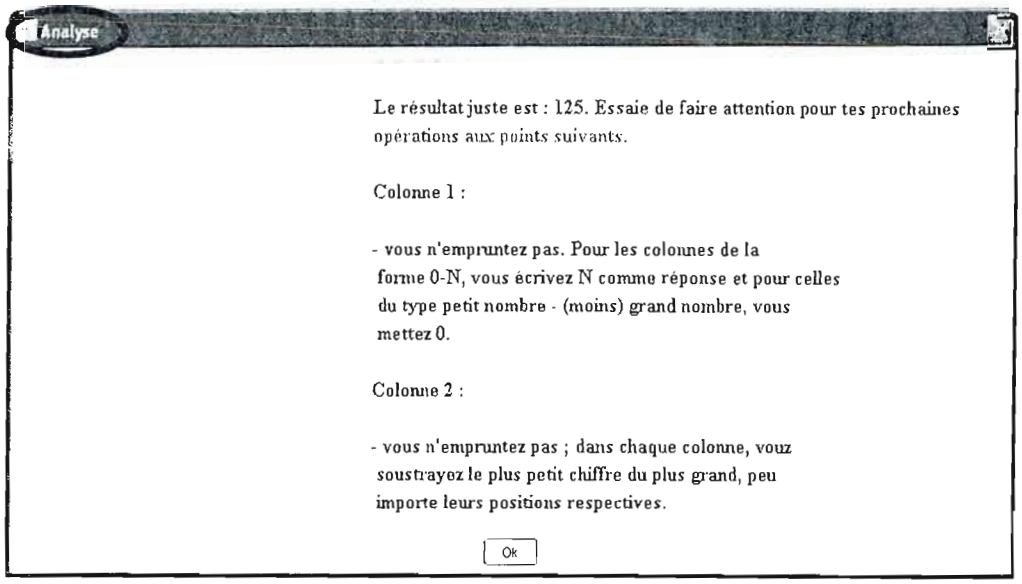


Figure 4.16 - Exemple d'un message d'analyse (exemple précédent)

Ensuite on a un autre message de diagnostic plus explicite au niveau du deuxième module (TIDES-Diagnostic) qui constitue une deuxième forme de rétroaction pour savoir les causes réelles de cette erreur (voir figure 4.17). Ce message permet alors de définir l'état cognitif de l'élève en temps réel, ce qui constitue le diagnostic cognitif.

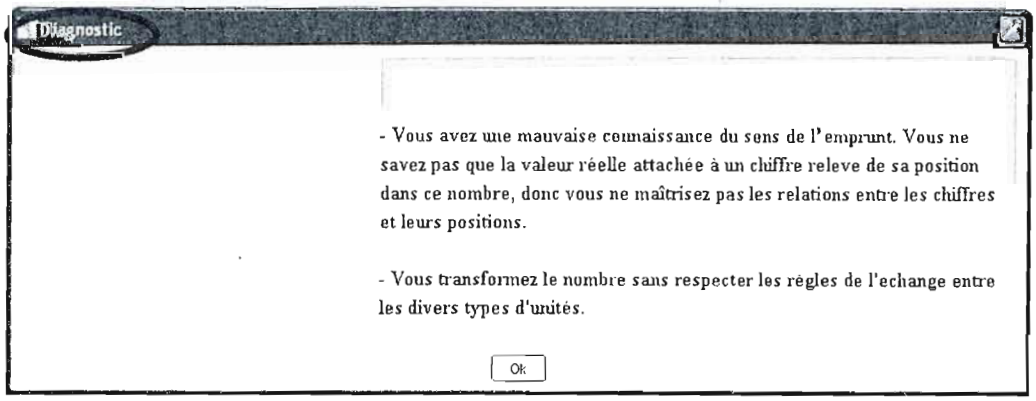


Figure 4.17 - Exemple d'un message de diagnostic (exemple précédent)

Il faut noter que dans les deux cas, le mécanisme d'inférence se charge de déclencher ces messages qui sont intégrés dans les règles de la base de connaissances.

Toutefois, lors d'une séance d'apprentissage, et en particulier lorsqu'il estime avoir terminé sa résolution, l'élève peut demander au système de vérifier sa solution (finale ou partielle). Dans ce cas, un message associé à sa résolution et généré à partir de l'identification de son cheminement, est toujours l'un des *verdicts* suivants :

1. La solution est correcte.
2. La solution est incorrecte (le système fournit la localisation de l'erreur et sa description).
3. La solution est incorrecte et a plusieurs explications (le système présente un test avant de donner une description de l'erreur).
4. La solution est obtenue par des actions plus complexes que nécessaire.
5. Connaissances trop limitées pour poser un diagnostic.

Dans les cas 2 et 4, le message diagnostic comporte aussi des informations plus précises sur les causes de l'erreur ou des erreurs détectées.

Il est à remarquer qu'une fois le problème soumis, l'élève garde le contrôle absolu sur le déroulement de la session d'apprentissage. Il peut développer, vérifier et corriger sa solution sans faire appel au tuteur ni être interrompu par celui-ci. Dans ce contexte, les messages d'analyse et de diagnostic n'apparaissent que si l'élève en fait directement la demande. Cependant le système essaye de discriminer, d'isoler, de connaître le mieux possible le raisonnement qui a mené l'élève à donner une réponse fausse. En effet, c'est à partir des réponses de l'élève que le système pourra élaborer des hypothèses sur sa compréhension et sur la nature des procédures utilisées pour arriver à ce résultat. Pour ce faire, il pourra donner plus d'une explication à une réponse erronée pour bien cerner la conception de l'élève. De ce fait, il est capable d'évaluer d'une façon précise les difficultés de cet élève afin de faire des interventions d'aide efficaces. Cela conduit, bien sûr, à une remédiation. Le système est capable de choisir un mode de remédiation en fournissant des problèmes tests à l'élève (voir sous-section 6.5 pour un exemple détaillé).

4.5.8 Catalogue d'erreurs

Dans le but d'élaborer les mal-règles de la base de connaissances, le catalogage des erreurs susceptibles d'être commises par des élèves a été réalisé en distinguant six types d'erreurs : les erreurs conceptuelles, les erreurs logiques de raisonnement, les erreurs de regroupement, les erreurs d'algorithme, les erreurs dues à la charge cognitive et les erreurs procédurales. Cette distinction repose sur une discrimination des différents éléments du contexte d'une tâche. C'est ainsi que plusieurs problèmes peuvent être générés à partir d'un même type de tâches en spécifiant successivement différents contextes d'exécution.

Par exemple, pour la tâche *emprunter sur un zéro,* le fait que le concept de la numération positionnelle soit connu ou non constitue une information utile puisque le zéro signifie qu'il y a absence de groupement dans la position correspondante, mais dans la soustraction, l'emprunt sur le zéro exige qu'on aille à la valeur suivante pour emprunter le groupement nécessaire. Cette confusion de deux groupements pourra cependant être la source d'erreurs de la part de l'élève, erreurs globalement qualifiées de *groupement.* En fait, l'élève qui commet une telle erreur, la fait parce qu'il a mal compris le concept du nombre. En effet, pour que l'élève arrive à une maîtrise intelligente de la numération positionnelle, il doit d'abord se donner une compréhension suffisante du concept de nombre.

Les travaux concernant le nombre, son acquisition et son utilisation chez les élèves s'articulent autour de deux grandes préoccupations : la première concerne la conception du nombre, en le considérant comme centre d'apprentissage mathématique à travers les six années de l'école primaire (tous les niveaux). La seconde renvoie à un point de vue plus didactique cherchant à analyser les erreurs et tenter d'expliquer les difficultés relatives aux procédures arithmétiques dans le sens que beaucoup d'*erreurs procédurales* sont dues à la représentation du nombre chez l'élève (Fayol, 1990).

D'ailleurs, les opérations elles-mêmes portent sur les nombres. Et il n'est pas surprenant que la compréhension du nombre soit à la base de celle des algorithmes et que les erreurs dans ce domaine relèvent souvent de difficultés avec le concept

de nombre lui-même (*erreurs conceptuelles*). En d'autres termes, un élève qui ne comprend pas le concept du nombre ne pourra pas maîtriser, ni l'algorithme de la soustraction écrite (*erreurs d'algorithme*), ni l'algorithme de toute autre opération arithmétique. C'est le nombre lui-même qui donne sens aux procédures que l'élève doit mettre en place pour réaliser une opération arithmétique.

Toutefois, la non-maîtrise des caractéristiques du nombre peut être une source d'erreurs. L'élève qui, par exemple, voit le nombre comme une simple collection de chiffres collés les uns à côté des autres peut même ne pas s'étonner d'obtenir une différence de deux nombres plus grande que le premier terme de l'opération (*erreurs logiques de raisonnement*). Donc, prenant en considération les difficultés dans l'utilisation du nombre lui-même, l'erreur de l'élève ne peut être considérée comme découlant uniquement d'une procédure erronée.

Une toute autre catégorie d'erreurs n'a rien à voir avec la conception du nombre. Il est possible, par exemple, qu'un élève fasse une erreur pendant le processus même d'élaboration de sa solution. Cette erreur peut provenir du fait que certaines notions font appel à des opérations indisponibles chez l'élève. Les limites de la mémoire de travail, par opposition à la mémoire à long terme, et la charge cognitive réelle de l'activité sont souvent les causes de cette erreur (*erreur due à la charge cognitive*). Il est plus difficile de systématiser l'identification de ce type d'erreurs puisque c'est par l'expérience et au cas par cas qu'on pourra réussir à formuler certaines mal-règles permettant de déceler ces erreurs.

Dans la suite, nous présentons plus en détail le fonctionnement de l'analyseur avec des exemples d'utilisation illustrant ses composantes. De plus, nous donnons un exemple dans lequel on propose des problèmes tests pour montrer la puissance de l'analyseur.

4.6 Présentation du module analyseur

4.6.1 Fonctionnement de l'analyseur

L'analyse de l'action de l'élève est confiée à la base de connaissances grâce au mécanisme d'inférence. En associant différentes règles de cette base, on peut

générer des méthodes (des plans) permettant d'atteindre correctement ou incorrectement le but poursuivi. L'analyseur cherche à interpréter les actions de l'élève en essayant de trouver un ensemble de règles qui peut produire le même résultat que celui de l'élève (utilisations des règles qui correspondent aux étapes de résolution). Dans ce processus d'identification du comportement de l'élève, l'analyseur procède de la façon suivante : dans chaque unité de connaissances (utilisations des règles d'une unité de connaissances), il cherche d'abord à utiliser les règles de références, puis les règles équivalentes ; en cas d'échec, il poursuit avec les règles généralisantes et enfin avec les mal-règles.

On procède donc en décomposant la tâche de départ en sous-tâches et chacune de ces sous-tâches en d'autres sous-tâches jusqu'à l'obtention d'un ensemble de sous-tâches exécutables[14]. Par exemple, la tâche *Faire une soustraction* conduit à la décomposition indiquée à la figure 4.18. Dans cette décomposition, les actions *Soustraire chaque colonne séparément* et *Soustraire le bas du haut* sont des sous-tâches exécutables. Les autres actions cependant (par exemples, *Emprunter sur zéro*, *Emprunter sur un chiffre*) devront être l'objet de décompositions additionnelles puisqu'elles ne sont pas directement exécutables par le système (par exemple, est-ce que ce zéro surmonte un autre zéro ou un vide) (utilisations des règles qui permettent de suivre le cheminement de l'élève).

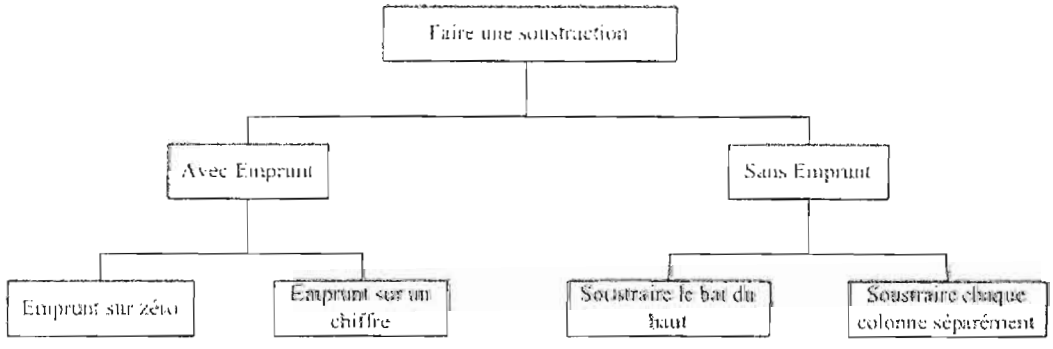


Figure 4.18 - Décomposition partielle de la tâche Faire une soustraction

[14] Une tâche exécutable est une tâche qui ne peut être décomposée en sous-tâches, de plus ne demande pas des connaissances avancées pour qu'elle soit résolue.

Dans le système TIDES, si la décomposition effectuée par l'élève ne correspond pas au problème de référence généré par le système à partir des règles de références, l'analyseur cherche à expliquer les différences (utilisations des règles qui définissent les types d'erreurs). D'abord en vérifiant que la solution proposée par l'élève n'est pas la même que la solution de référence ; ensuite, en cherchant à expliquer la différence par des mal-règles.

Il faut noter que, la génération de plans se fait avec soin pour éviter toutes sortes d'explosion combinatoire des règles. C'est pourquoi il est important de se donner des heuristiques permettant de réduire l'espace-problème dans la procédure d'analyse. Comme il fut mentionné en 5.3, en donnant la structure d'une règle du TIDES, une telle heuristique est la raison d'être de l'élément 4 (niveau de complexité) de chaque règle, élément qui identifie le niveau de complexité de la prémisse (élément 2) de la règle par rapport à l'ensemble des tâches.

4.6.2 Analyse des actions de l'élève

Dans le système TIDES, l'élève est totalement libre dans la formulation de buts essentiels de la connaissance envisagée. S'il n'arrive pas à la formulation désirée, c'est-à-dire, s'il manipule des buts dont la formulation est différente des buts correspondants du système, ce dernier va se charger de l'analyse des actions de cet élève.

D'ailleurs, le principe de l'analyse des actions de l'élève est de comparer les réponses trouvées par l'élève avec les réponses simulées par le système. Cette comparaison génère seulement des hypothèses sur la fraction de connaissances pouvant être fausses ou absentes.

Cependant, un problème d'analyse se pose en tentant de rendre le système capable de poser un diagnostic adapté au niveau de concentration où se situe l'élève dans le processus de résolution. D'une part, il est essentiel de donner à chaque action proposée par l'élève un sens précis si l'on veut que le système soit capable de poser un diagnostic aussi précis. D'autre part, l'élève qui travaille à un niveau de concentration avancé durant les premières étapes de la résolution doit utiliser des

procédures adaptées à ce niveau. Ainsi, par exemple, l'action *emprunter sur un chiffre* (non nul) n'a pas le même niveau de concentration que l'action *emprunter sur zéro*. On pourrait même imaginer que, durant les premières étapes de la résolution, un élève qui inscrit une action comme *emprunter sur un chiffre,* a en tête une action supposant un positionnement préalable par l'action *emprunter sur zéro*. Plusieurs solutions ont été explorées pour faire face à ce problème.

- *Inviter l'élève à préciser son but* : un élève qui formule une action à caractère ambigu, détectable par le système, est invité à préciser son but. Ces précisions viennent réduire les problèmes d'analyse associés aux actions de l'élève. Il est certain que cette façon de procéder facilite l'analyse des erreurs par le système ainsi que la génération de messages diagnostic.

- *Analyser les actions de l'élève par détection d'éléments critiques* : une autre piste de solution consiste à analyser les actions proposées par l'élève à partir de certains critères plus sélectifs (comme l'emprunt, par exemple). Il s'agit tout simplement de vérifier la présence ou l'absence d'actions jugées critiques dans la solution proposée par l'élève et d'exprimer une analyse en conséquence.

- *Adapter les messages diagnostiques au niveau de concentration* : la résolution d'un problème en utilisant un algorithme ou une procédure de résolution, peut amener l'élève à travailler sur plusieurs niveaux de raisonnement et le passage d'un niveau au suivant se fait de façon continue. Le message de diagnostic formulé par le système doit être adapté au niveau de concentration où se situe l'élève dans la démarche de résolution, c'est-à-dire, qu'il doit atteindre un haut niveau de précision quand la formulation de l'élève peut se traduire en actions précises.

Les considérations qui précèdent ont abouti à une approche dans laquelle le système est plus précis au départ dans son diagnostic car il suppose que l'élève qui ne spécifie pas précisément certaines actions moins importantes peut

néanmoins les avoir en tête. Elles sont alors sous-entendues quelque part à l'intérieur des autres actions proposées.

C'est pourquoi quand une erreur est détectée, en utilisant une analyse fine, et que cette erreur implique un mauvais choix de stratégie, le système TIDES émet une première rétroaction en décrivant le comportement de l'élève. Si un message plus explicite est demandé par l'élève, alors une deuxième rétroaction prend la forme d'un diagnostic plus précis. L'utilisation de ces messages permet de distinguer, pour les fins des interventions du tuteur, entre une imprécision de formulation de haut niveau et une erreur en bonne et due forme.

C'est la méthode qui a été retenue pour contourner les problèmes d'analyse à certains niveaux de concentration. Voici, à titre d'exemple, le message de diagnostic affiché à la suite d'une erreur d'un élève consistant à omettre d'emprunter correctement avant de faire la soustraction dans une colonne (figure 4.19):

$$\begin{array}{r} 953 \\ -\,406 \\ \hline 553 \end{array}$$

«*Vous avez généralisé une conception correcte dans certains cas particuliers, mais qui ne supporte pas le moindre changement de situation*[15]».

[15] Dans le cas où l'élève doit faire, par exemple, la soustraction : 3 – 6, il applique ici une règle qui dit : «ne pas soustraire le grand nombre du petit nombre», il change donc le problème en 6 – 3 au lieu d'emprunter sur la colonne suivante.

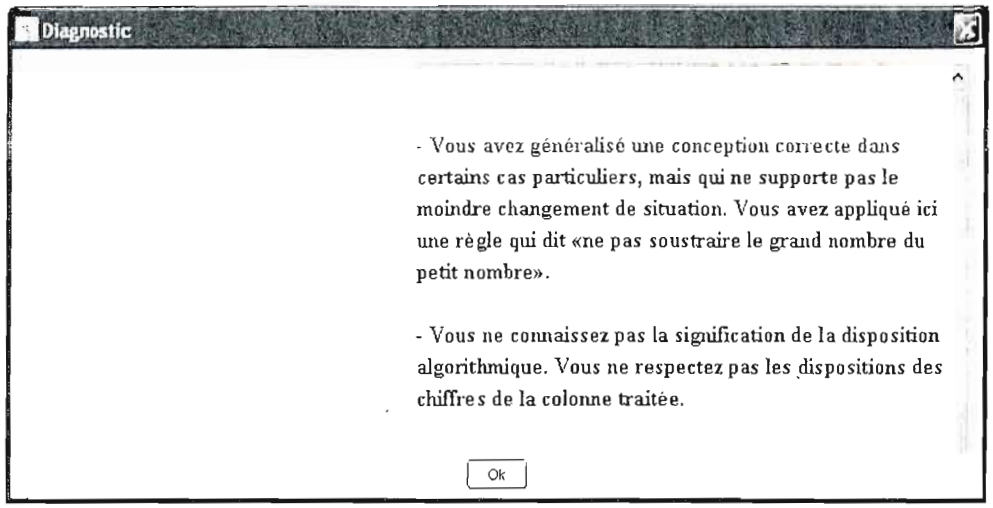


Figure 4.19. Exemple d'un message de diagnostic

4.6.3 Analyse d'un processus plutôt que d'un produit

Comme il a été indiqué dans le chapitre 2, le modèle d'Anderson est indépendant du domaine d'apprentissage. Mais, on peut se poser la question suivante : jusqu'à quel point ce traçage de modèle est-il indépendant du domaine d'apprentissage ? Ce modèle a été appliqué à des domaines aussi variés que la programmation en langage LISP, la démonstration de théorèmes en géométrie et la résolution d'équations algébriques. Dans toutes ces applications cependant, le tuteur intervient immédiatement après l'identification d'une erreur.

Dans le système TIDES, le niveau d'intervention (analyse et diagnostic) est l'interface de résolution (appelée aussi bloc de résolution). Par conséquent, le processus diagnostic est plus complexe, notamment parce que la solution formulée par l'élève peut contenir plusieurs différences (erreurs) avec la solution générée par le système pour atteindre le but poursuivi. Il est également plus complexe parce qu'il est possible (et même probable) que la démarche de résolution formulée par l'élève contienne une ou plusieurs décompositions d'une tâche en sous-tâches.

4.6.4 Précision de l'analyseur

Puisque l'analyse de la démarche de résolution de l'élève peut porter aussi bien sur un premier niveau de décomposition d'une tâche (sur des actions complexes, comme par exemple, emprunt sur deux zéro) que sur une résolution finale (sur des actions élémentaires, comme par exemple, soustraction sans emprunt), sans omettre toutes les situations intermédiaires possibles, l'analyseur du système TIDES doit manifester une précision toute particulière.

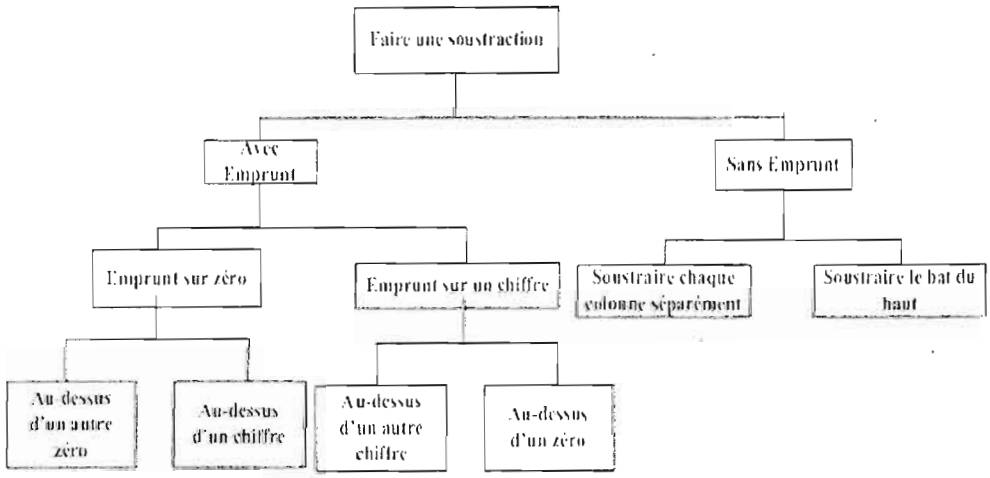


Figure 4.20. Décomposition de la tâche Faire une soustraction

Dans le graphe de la figure 4.20, l'arbre de décomposition de la tâche «*Faire une soustraction*» en sous-tâches, sous-sous-tâches, etc. est présenté afin de montrer la plus grande précision de l'analyseur du système TIDES et les contraintes qui s'imposent à cet analyseur pour que l'analyse porte sur le processus de la résolution plutôt que sur un produit final (réponse finale). On peut constater, par exemple, que dans ce dernier cas, et en supposant pour simplifier l'exposé que la décomposition ne fait intervenir que des décompositions en séquences d'actions, l'objet de l'analyse peut être représenté par un produit final correspondant à la séquence de nœuds *Avec_Emprunt* - *Emprunt_sur_un_chiffre* – *Soustraire_chaque_colonne_séparément* - *Soustraire_le_bat_du_haut*.

En comparaison, l'analyseur du système TIDES doit être capable d'analyser une solution partielle ou complète d'un élève. En conséquence, cette solution doit

pouvoir être analysée en fonction des ensembles d'actions suivantes (complexes ou élémentaires) :

- *Avec_Emprunt-Sans_Emprunt*
- *Avec_Emprunt-Emprunt_sur_zéro-Au_dessus_d'un__autre_zéro*
- *Avec_Emprunt-Emprunt_sur_zéro-Au_dessus_d'un_autre_chiffre*
- *Avec_Emprunt-Emprunt_sur_un_chiffre-Au_dessus_d'un_autre_chiffre*
- *Avec_Emprunt-Emprunt_sur_un_chiffre-Au_dessus_d'un_autre_zéro*
- *Avec_Emprunt-Emprunt_sur_un_chiffre-Soustraire_chaque_colonne_séparément -Soustraire_le_bas_du_haut*

IL s'agit donc d'une problématique d'analyse plus complexe. Et cette complexité croît exponentiellement avec le nombre de niveaux.

4.6.5 Puissance de l'analyseur

Comme le cas dans le LISP-TUTOR d'Anderson, une rétroaction rapide sinon immédiate du tuteur, a deux effets sur l'analyse à effectuer. D'abord, l'analyse n'est faite qu'en fonction d'une seule ou d'un nombre très réduit de règles et ensuite, le nombre d'erreurs peut difficilement dépasser un, puisque le tuteur intervient dès qu'une erreur est détectée.

Dans le cas du système TIDES, la liberté de manœuvre exigée par le domaine d'apprentissage, liberté qui doit permettre à l'élève d'explorer librement différentes pistes de solution sans crainte d'être interrompu par le tuteur, implique que l'analyseur peut retracer simultanément plusieurs erreurs potentielles.

Nous avons annoncé à plusieurs reprises que le système TIDES est capable d'analyser et de diagnostiquer les erreurs ambiguës en invitant l'apprenant à résoudre certains problèmes tests liés à sa conception erronée. Les problèmes proposés ne dépassent pas le nombre d'hypothèses du système (c'est-à-dire le nombre d'explications d'erreurs possibles). Si certaines hypothèses s'accordent avec une réponse erronée, le système peut déduire les causes de l'erreur sans

fournir d'autres problèmes. Selon les réponses de l'apprenant, le système peut prédire quelle conception erronée l'élève manifeste. Dans la suite nous exposons un exemple détaillé de cette capacité du système.

- **Exemple d'analyse et de diagnostic des erreurs ambiguës**

Supposons qu'un apprenant, pour résoudre le problème 730 – 185 =, ait trouvé le résultat 655 présenté dans la figure suivante (figure 4.21). Selon ce résultat, le système ne peut savoir exactement comment l'apprenant arrive à cette solution : pour la première colonne, est ce que l'apprenant a soustrait le plus petit du plus grand (5 - 0 = 5) ou a-t-il emprunté une dizaine de la colonne suivante sans changer cette colonne (10 – 5 = 5) ? D'ailleurs, les deux procédures donnent le même résultat. Pour la deuxième colonne, la même question se pose : est ce que l'apprenant a fait (8 – 3 = 5, le plus grand moins le plus petit) ou a-t-il fait (13 – 8 = 5) en empruntant de la colonne suivante sans la changer ?. Ce type d'erreurs est appelé « erreurs ambiguës » (Attisha et Yazdani, 1983).

$$\begin{array}{r} 7\ 3\ 0 \\ 1\ 8\ 5 \\ \hline 6\ 5\ 5 \end{array}$$

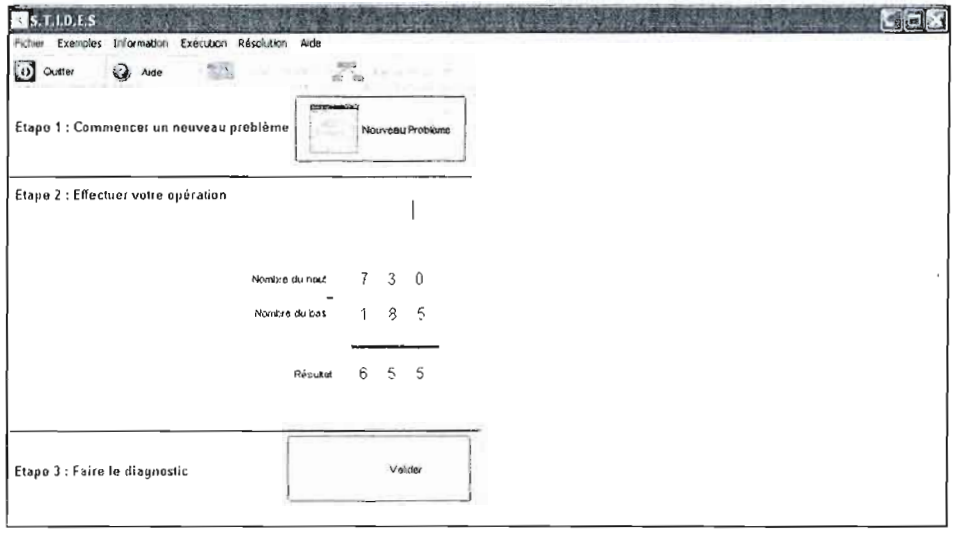


Figure 4.21. Problème de l'élève avec des erreurs ambiguës

Lorsque l'apprenant demande la validation de son résultat, le système TIDES échoue au premier coup pour trouver la procédure utilisée par l'élève ; cependant, il affiche un message avec deux hypothèses différentes (figure 4.22) :

- Hypothèse 1 : *vous n'empruntez pas, dans chaque colonne vous soustrayez le plus petit chiffre du plus grand.*

ou

- Hypothèse 2 : quand *vous empruntez, vous ajoutez 10 correctement mais vous ne changez aucune colonne à gauche.*

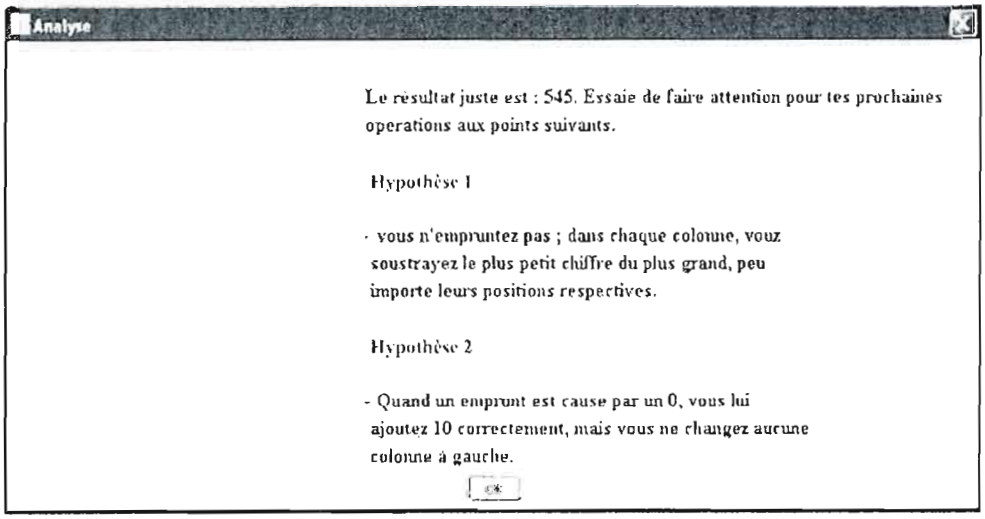


Figure 4.22. Message d'analyse avec deux hypothèses

Ensuite, le système génère un problème test pour savoir exactement quelle procédure est utilisée par l'élève. La figure 4.23 illustre ce problème test.

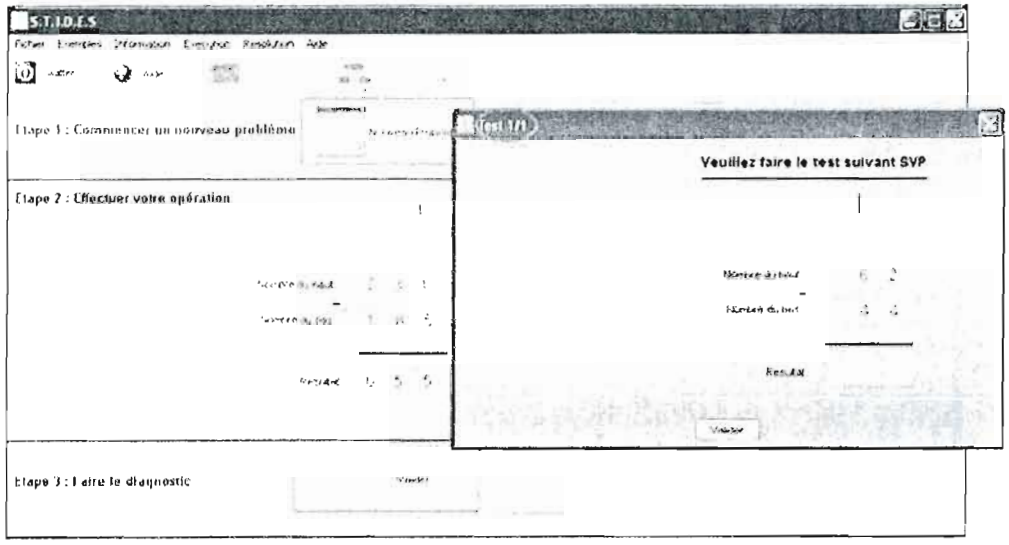


Figure 4.23. Problème test

Deux cas sont possibles : soit la solution de l'élève à ce problème test est correcte soit elle est fausse.

- **Premier cas** : la solution de l'apprenant au problème test est fausse

Si la réponse de l'apprenant au problème test est fausse, comme l'indique la figure suivante (figure 4.24), le système dans ce cas écarte la deuxième hypothèse et garde la première puisque cette dernière s'accorde avec la réponse erronée du problème initial.

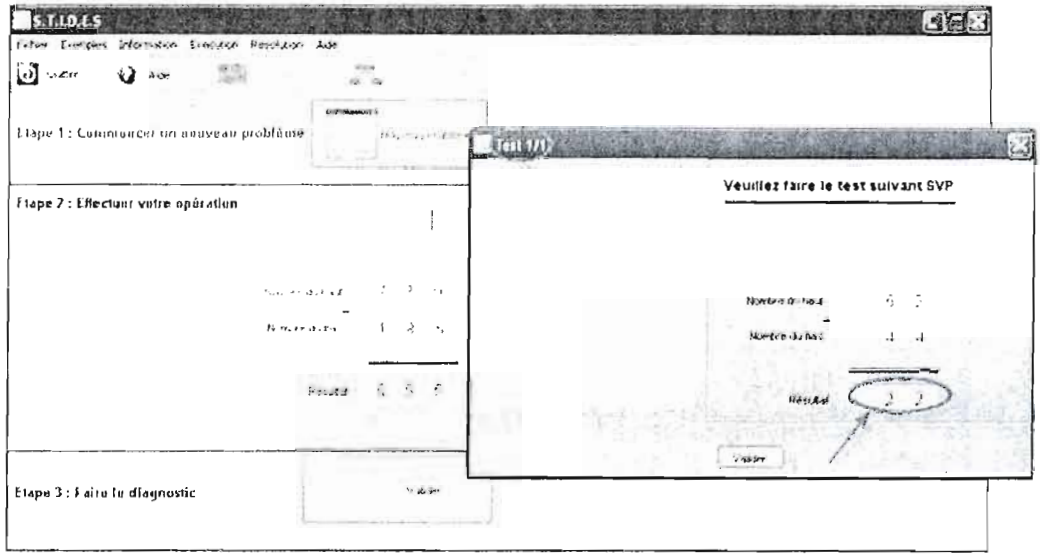


Figure 4.24. Solution fausse du problème test

Le système est maintenant capable d'identifier la nature de l'erreur et par conséquent il peut afficher un premier message d'analyse (figure 4.25).

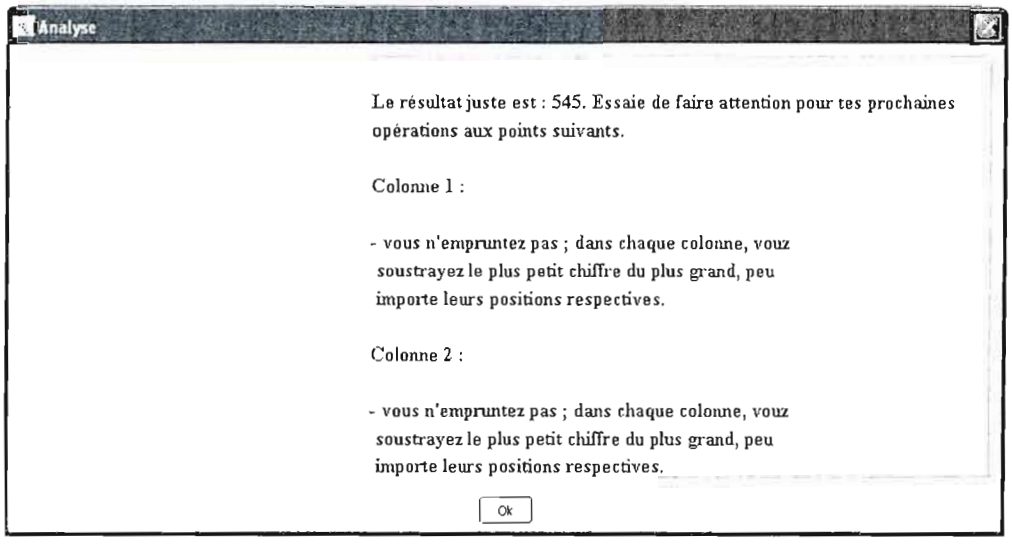


Figure 4.25. Message d'analyse après le problème test

Enfin un message diagnostic (figure 4.26) du deuxième module (TIDES-Diagnostic) très précis est affiché. Ce message détermine les causes réelles de cette procédure erronée.

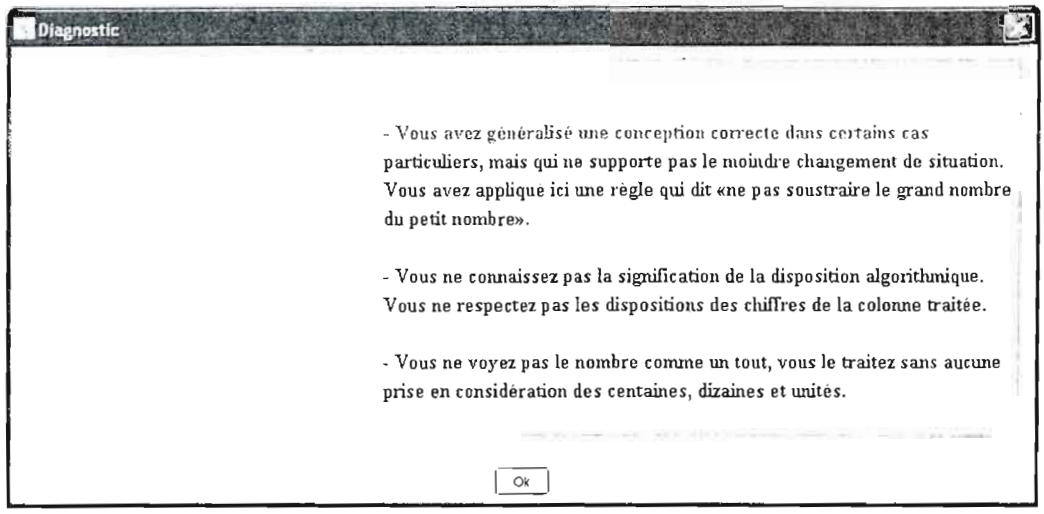


Figure 4.26. Message de diagnostic après le problème test

- **Deuxième cas** : la solution de l'apprenant au problème test est correcte

Si la réponse de l'apprenant au problème test est correcte, comme l'indique la figure suivante (figure 4.27), le système dans ce cas écarte la première hypothèse et garde la deuxième, puisque cette dernière s'accorde avec la réponse erronée du problème initial.

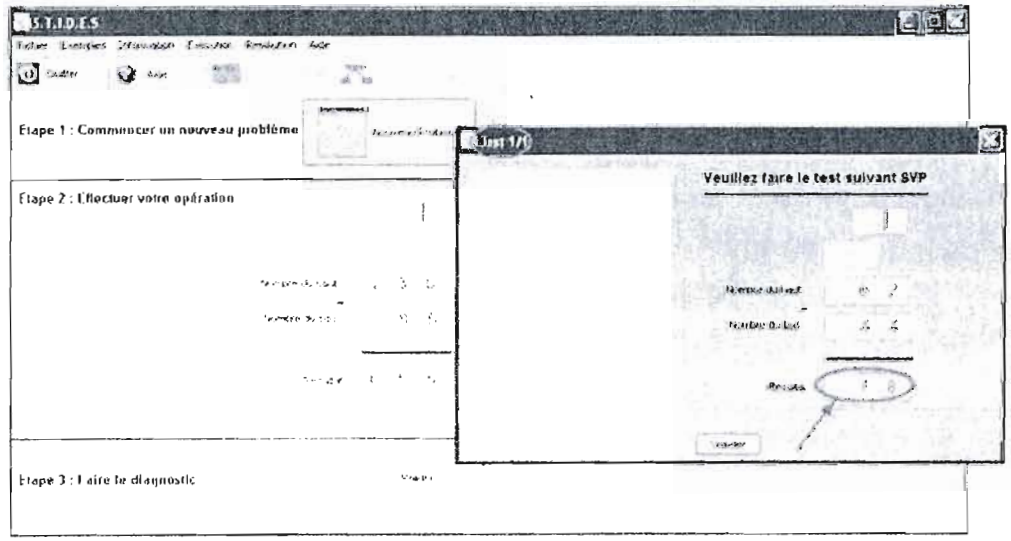


Figure 4.27. Solution correcte du problème test

Le système affiche un autre message d'analyse (figure 4.28) du premier module (TIDES-Analyse) qui interprète le comportement de l'élève de manière exacte.

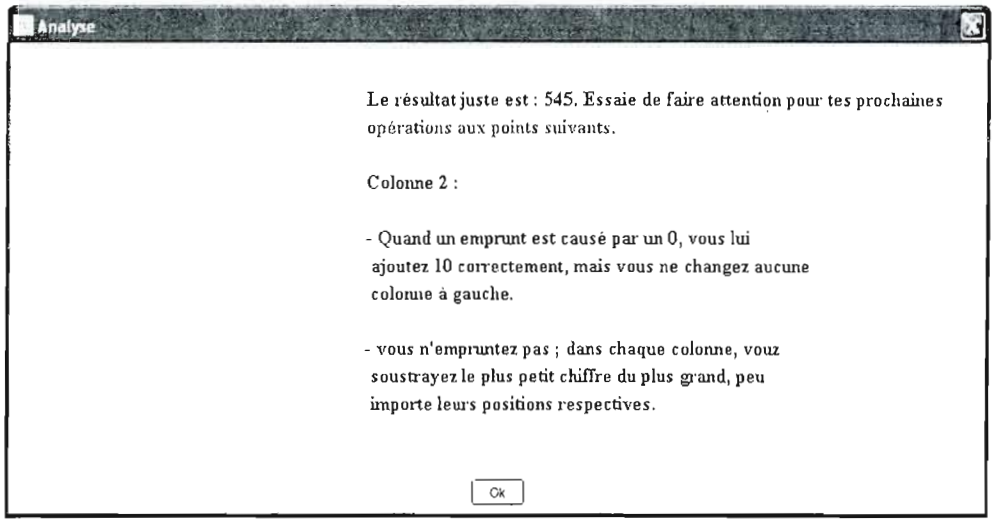


Figure 4.28. Un autre message d'analyse du problème test

Après ce dernier message d'analyse (figure 4.29), le système affiche un autre message de diagnostic qui lui aussi détermine les causes possibles derrières la procédure erronée de l'apprenant.

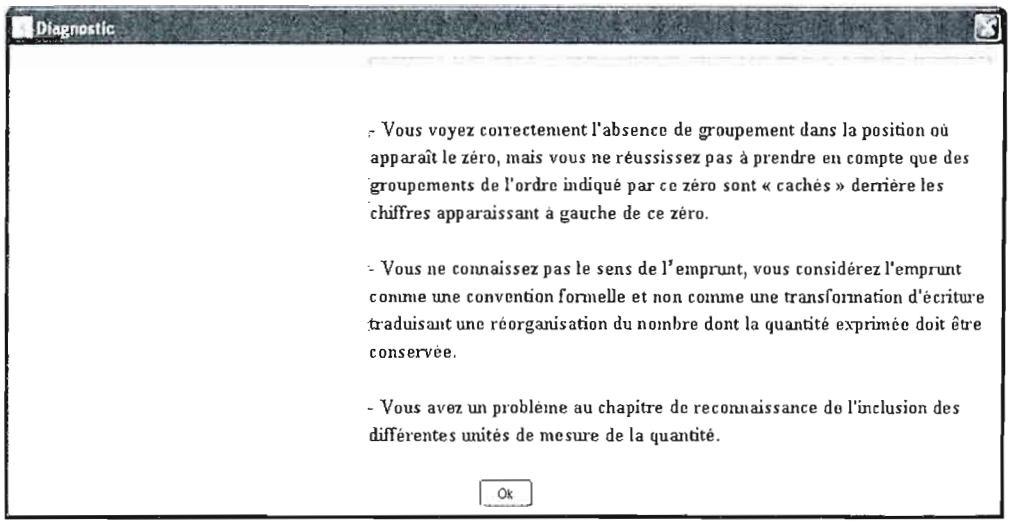


Figure 4.29. Un autre message de diagnostic du problème test

Cet exemple montre de façon très claire la puissance du système TIDES à diagnostiquer les erreurs ambiguës en proposant des problèmes tests pour bien cerner les difficultés qu'éprouve un apprenant lorsqu'il résout un problème de la soustraction écrite.

Par ailleurs, le système peut identifier les lignes de raisonnement qui sous-tendent les action de l'apprenant en utilisant des méthodes très efficaces. Ces méthodes sont présentées dans la sous-section suivante.

4.6.6 Génération dynamique du graphe de solution

Afin de déterminer quelle ligne de raisonnement sous-tend l'action de l'élève, le système doit avoir un ensemble de toutes les lignes de raisonnement qu'un élève peut poursuivre. Cet ensemble représente l'espace de solution du problème. La structure de données que nous avons utilisée pour représenter cet espace de solution est appelée *graphe de solution*. Ce graphe de solution, utilisé par le modèle de l'élève dans le système TIDES, est généré automatiquement par un résolveur de problèmes basé sur les règles de la base de connaissances (Figure 4.30).

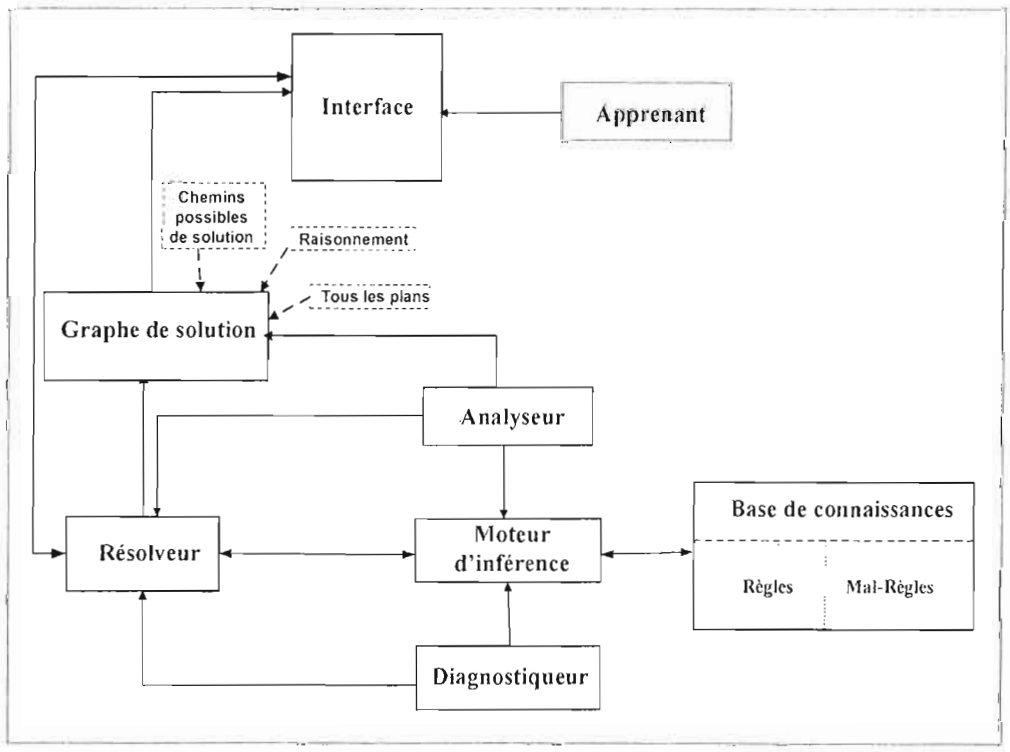


Figure 4.30. Mécanisme de génération dynamique du graphe de solution

Le résolveur, construit sur des caractéristiques de base pédagogique, joue un rôle important dans cette architecture. En effet, d'une part, il facilite l'analyse et le diagnostic des activités de l'élève, d'autres part, il permet de générer un graphe de solution correspondant aux chemins de la solution de l'apprenant. Une des caractéristiques particulières de ce résolveur, est qu'il fonctionne quel que soit le jeu de données introduit par l'apprenant.

4.6.7 Graphe de solution et le module analyseur du système TIDES

Voyons maintenant comment le système TIDES se servit de module analyseur pour appliquer la méthode du traçage de modèle en utilisant le graphe de solution.

Comme il a été indiqué un peu plus tôt dans ce chapitre, que ce soit pour aider l'élève ou pour analyser sa solution, le tuteur fait appel au module *analyseur*. Ce module interprète le graphe de solution pour simuler le traçage de modèle. Il simule en temps réel l'activité cognitive de l'élève, mais entreprend l'analyse de la solution seulement à son instigation. De plus, il utilise un ensemble de règles d'une unité de connaissances correspond à tous les comportements manifestés par l'élève. L'utilité de ces règles réside dans le fait qu'elles permettent l'économie d'un nombre considérable de règles, simplifient la codification de ces règles (Dion & Lelouche, 1994), et réduisent la taille du graphe de solution.

Le module analyseur doit aussi reconnaître la stratégie (ou les stratégies) utilisée par l'élève parmi l'ensemble des stratégies représentées dans le graphe de solution, ce qui montre le rôle de ce graphe dans l'analyse du *comment*. Contrairement à certains tuteurs d'Anderson (notamment le tuteur Lisp) qui doivent établir a priori la stratégie adoptée par l'élève, l'analyseur du TIDES peut la déterminer a posteriori en utilisant le graphe de solution. Puisque, en effet, l'analyseur peut comparer la solution de l'élève à la trace des solutions possibles conservée dans le graphe de solution et ainsi d'identifier le type de solution envisagée par l'élève.

Le processus mis en œuvre par l'analyseur pour donner un indice, aider l'élève ou évaluer une solution est essentiellement le même. Lorsque l'élève fait appel au

tuteur, l'analyseur entreprend de comparer la solution de l'élève à la trace de la solution enregistrée dans le graphe de solution.

4.6.8 Exemples d'un graphe de solution

Le graphe de solution contient trois types d'information (Conati & VanLehn, 1996b) qui représentent (1) tous les plans qui peuvent être dérivés par les règles de la base de connaissances pour résoudre un problème de soustraction ; (2) tous les chemins possibles (calculs numériques) de solution qui développent ces plans ; (3) le raisonnement derrière chaque étape dans un plan. Considérons, par exemple, le problème de soustraction présenté dans la figure 4.31. Il y a deux plans différents pour ce problème : plan qui utilise l'emprunt pour trouver la solution et plan qui cherche la solution sans appliquer l'emprunt. Ces deux plans sont représentés par les deux ensembles de solution liés aux connaissances primitives[16] « Avec Emprunt » et « Sans application d'Emprunt » (figure 4.31).

Par exemple, les connaissances primitives ci-dessous correspondent à l'application des règles de la base de connaissances du système pour l'interaction $84 - 26 = 52$, qui sont aussi considérées dans la figure 4.32 :

- primitive `différence(chiffre1, chiffre2, colonne)` :
  `/*` *chiffre1* est le chiffre situé en haut de la colonne,
  chiffre2 est le chiffre situé en bas de la colonne `*/`

  si [on est en train de traiter la colonne *colonne1*]
  [et qu'il n'y a pas encore de résultat pour la colonne *colonne1*]
  et que *chiffre1* est supérieur ou égal à *chiffre2*
  alors écrire la différence entre *chiffre1* et *chiffre2*
  comme résultat pour la colonne *colonne1*

- primitive `traitement_colonne(colonne)` :

  si on est en train de traiter la colonne *colonne1*
  et que la colonne à gauche de *colonne1* est *colonneG*
  [et qu'il y a un résultat pour la colonne *colonne1*]

  alors déplacer le *focus* de la colonne *colonne1* à la colonne *colonneG*

- primitive `decrement(chiffre, colonne)` :

[16] Selon VanLehn (1988), les connaissances primitives sont des applications directes, par l'élève, des règles de production représentant des connaissances (comme par exemple : *décrément*, *traitement_colonne*, ...)

si on est en train de traiter la colonne *colonne1*
et que la colonne *colonne1* demande un emprunt

alors déplacer le focus de la colonne *colonne1* à la colonne *colonneG*

et soustraire 1 au chiffre1 de la colonne *colonneG*

Résoudre le problème : 84 - 26	
Plan de solution Avec Emprunt	**Plan de solution Sans application d'Emprunt**
• soustraire 6 à 4 • décrémenter 8 • remplacer 8 par 7 • ajouter 10 à 4 • soustraire 6 à 14 • écrire résultat 8 (colonne 1) • soustraire 2 à 7 • écrire résultat 5 (colonne 2)	• soustraire 6 à 4 • faire l'emprunt • décrémenter 8 • remplacer 8 par 7 • soustraire 4 à 6 • écrire résultat 2 (colonne 1) • soustraire 2 à 7 • écrire résultat 5 (colonne 2)

Figure 4.31 : Problème de soustraction avec deux plans de solution (correct et incorrect)

Les connaissances primitives pour un plan peuvent être générées et combinées de différentes manières produisant un ensemble de chemins de solution. Les systèmes tutoriels existants qui interviennent pendant la résolution de problèmes réduisent le nombre de chemins de solution en forçant l'élève à suivre une stratégie particulière de résolution de problèmes (Anderson & al., 1995). Contrairement à ces systèmes, TIDES laisse la liberté à l'élève de choisir sa stratégie de résolution et son graphe de solution fournit une représentation compacte de tous les chemins possibles de solution qui développent un plan donné (figure 4.32) et soutient une interaction tutorielle plus flexible et semblable à celle produite par les tuteurs humains.

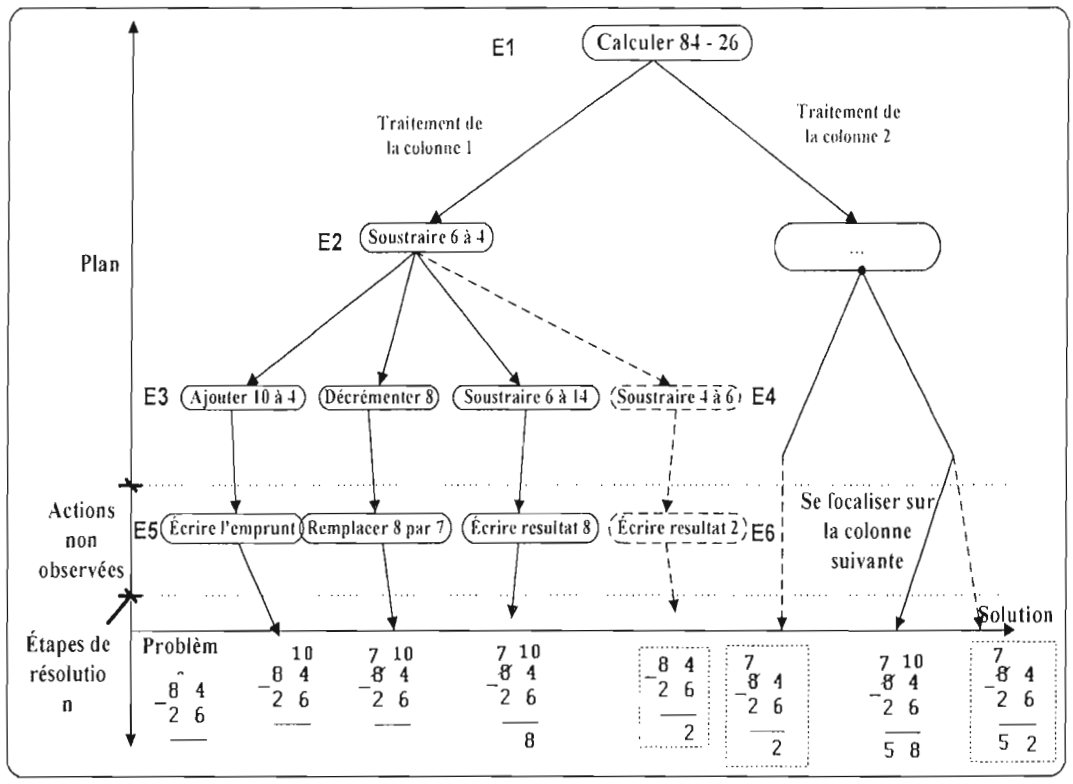


Figure 4.32 Identification du graphe de solution de l'élève

Pour expliciter un peu mieux le principe du graphe de solution, reprenons plus en détail notre exemple présenté dans la figure 4.32. Deux scénarios différents se présentent : dans le premier scénario l'élève a suivi un plan correct pour arriver à une solution correcte (utilisation de l'emprunt) ; dans le deuxième scénario (celui qui nous intéresse le plus) le plan suivi par l'élève n'aboutit pas à une solution correcte, l'élève a commis plusieurs erreurs.

- **Premier scénario : solution correcte**

La démarche que nous décrivons ici consiste à identifier les étapes que l'élève a suivies et qui ont abouti à une solution correcte et à la construction du graphe de solution présenté dans la figure 4.32a. Pour résoudre le problème de soustraction $84-26$, l'élève va commencer d'abord par la colonne la plus à droite (4,6) et la considère comme colonne active. Il doit soustraire 6 à 4, mais comme 6 est plus grand que 4, il doit « emprunter de la colonne suivante à gauche ». Pour cela, il se

focalise sur cette colonne à gauche, il décrémente 8 de 1 et le remplace par 7, il ajoute ensuite 10 à 4, puis il soustrait cette fois 6 à 14 et écrit le résultat dans la première colonne (figure 4.32a).

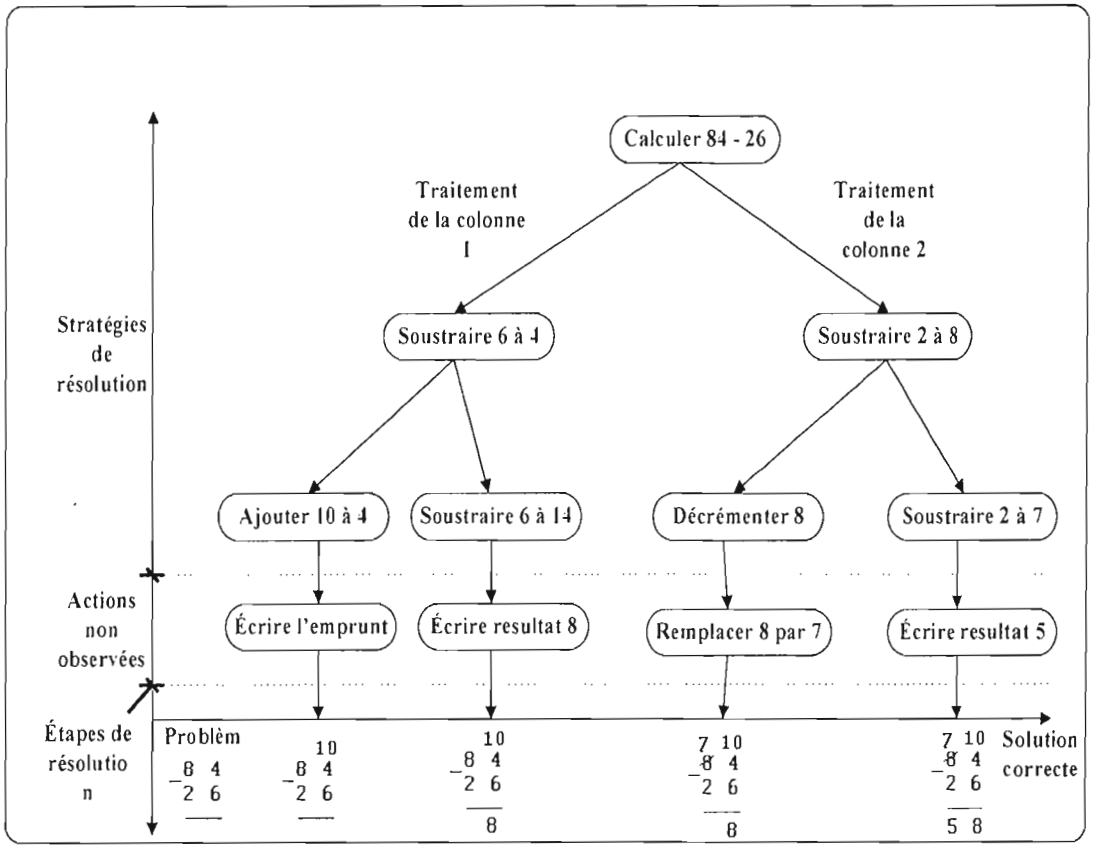


Figure 4.32 a. Graphe de solution menant à la solution correcte du problème 84 – 26

Ensuite, il poursuit sa résolution en soustrayant 2 à 7 et en écrivant 5 comme résultat de cette colonne. Enfin, il arrive à donner une solution correcte à son problème (réponse 58).

Les changements que l'élève a faits sur les deux colonnes : la colonne d'emprunt (première colonne à gauche de la colonne active) en la décrémentant de 1 et la colonne qu'il traite (colonne active) en ajoutant 10 au chiffre du haut de cette colonne, ne sont possibles que grâce à l'application des règles appropriées de la base de connaissances et maîtrisées par l'élève.

Notons que le scénario qui donne une solution correcte n'est pas assez intéressant pour le système TIDES, puisque ce système a pour but d'analyser les erreurs commises par l'élève et d'en trouver les causes. Cependant, le scénario suivant sera examiné avec plus d'attention.

- **Deuxième scénario : solution incorrecte**

Dans ce deuxième scénario, l'élève a commis une erreur dans sa démarche de résolution. Cette erreur n'est pas due, comme on le croit souvent, à un manque de concentration de la part de l'élève. Au contraire, elle trouve son origine dans l'application consciencieuse d'une procédure partiellement erronée. L'erreur produite n'est pas donc aléatoire, elle est déterminée par la stratégie erronée suivie. Quand il traite une colonne dont le chiffre du dessous est plus grand que le chiffre du dessus, l'élève emprunte 1 de la colonne à gauche correctement, mais au lieu d'ajouter cet emprunt au premier chiffre de la colonne courante, il procède d'une autre manière. Il intervertit la position des chiffres de la colonne active afin de soustraire le plus petit chiffre du plus grand. L'élève a commis ce qu'on appelle un bug SSL (Substract Smaller from Larger) « **soustraire le plus petit du plus grand** » (VanLehn, 1990, voir aussi annexe A, bug 26). Le graphe de solution de sa démarche est donné par la figure 4.32b.

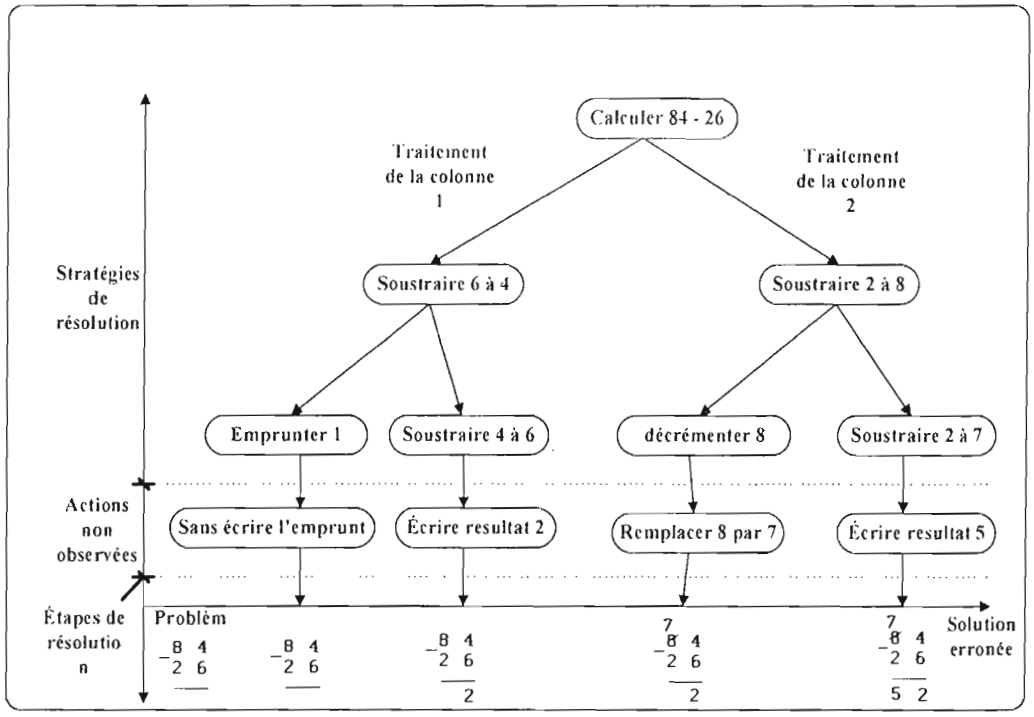


Figure 4.32 b. Graphe de solution menant à la solution incorrecte du problème 84 – 26

Par ailleurs, le graphe de solution permet au tuteur d'accepter les actions de l'élève dans n'importe quel ordre, tant qu'elles appartiennent à un plan connu. De même, il permet aussi de garder la trace des actions que l'élève a effectuées pendant sa résolution de problème. Cependant, il y a des situations où le comportement de l'élève ne détermine pas l'état réel de connaissances de cet élève. En effet, un écart entre les véritables connaissances de l'élève et le comportement observé peut survenir pour deux raisons. Le premier risque est que l'élève manifeste un comportement correct par un coup de chance, alors qu'il ne maîtrise pas réellement la connaissance associée à ce comportement. Le second risque est qu'il manifeste un comportement erroné à la suite d'une erreur d'attention, alors qu'il maîtrise en fait la connaissance sous-jacente.

4.7 Présentation du module diagnostiqueur

4.7.1 Fonctionnement du diagnostiqueur

L'objectif de la modélisation de l'apprenant à l'aide d'un tuteur intelligent est d'obtenir des informations sur l'état de connaissances de l'élève en analysant son comportement. Le processus de diagnostic a pour but d'évaluer les connaissances de l'apprenant. Il s'agit, en fait, de déduire les connaissances « cachées » de l'élève sur la situation-problème à partir de comportements observables. Il nous permet, non seulement de déduire des informations sur les actions de l'apprenant, mais de comprendre aussi l'état cognitif de l'élève. Cependant, l'acquisition des informations pertinentes pour établir ce diagnostic n'est pas toujours facile. Dans le système TIDES, les situations ainsi que l'interface élève ont été élaborées pour faciliter cette tâche. D'ailleurs, le fonctionnement du diagnostic se fait en trois temps. D'abord, le système choisit dans une classe de problèmes le problème à soumettre à l'élève (ou l'élève propose son propre problème). Une fois le problème soumis, l'élève garde le contrôle absolu sur le déroulement de la session d'apprentissage et le tuteur n'intervient qu'à la demande de celui-ci. Dans un deuxième temps, le système analyse les productions de l'élève en utilisant le module d'analyse, décrit dans la sous-section précédente, afin d'identifier les comportements de l'apprenant. Le système procède enfin à un diagnostic plus précis en interprétant ces comportements pour identifier l'état des connaissances de l'élève et construire le profil de ce dernier.

En fait, la deuxième étape (appelée aussi diagnostic comportemental) se limite à la reconstruction des actions qui ont été nécessaires à la résolution de problèmes afin d'en déduire les comportements de l'élève sans chercher à les interpréter. Tandis que la troisième étape (appelée diagnostic cognitif) va plus loin et consiste en une interprétation de ces comportements pour élaborer un modèle de l'état des connaissances attribuées à l'apprenant. Dans ce contexte, il convient alors de prendre en compte à la fois les comportements observables (actions externes

visibles à l'interface du système) et les comportements non-observables (les processus mentaux internes de l'élève).

4.7.2 Analyse diagnostique et traçage de modèle

Pour que les connaissances dans le système TIDES puissent servir à analyser la solution soumise par l'élève, il faut que ces connaissances représentent les étapes de résolution que pourrait adopter un apprenant accomplissant ces tâches. Comme nous l'avons souligné précédemment, ces connaissances sont représentées sous forme de règles de production. Le diagnostiqueur exécute l'ensemble de ces règles en les appliquant au problème posé à l'élève. Lorsque l'élève résout un problème, les étapes successives de son processus de résolution sont comparées une à une avec celles apparaissant dans la trace du processus de résolution du système (figure 4.33) (Danna, 1997).

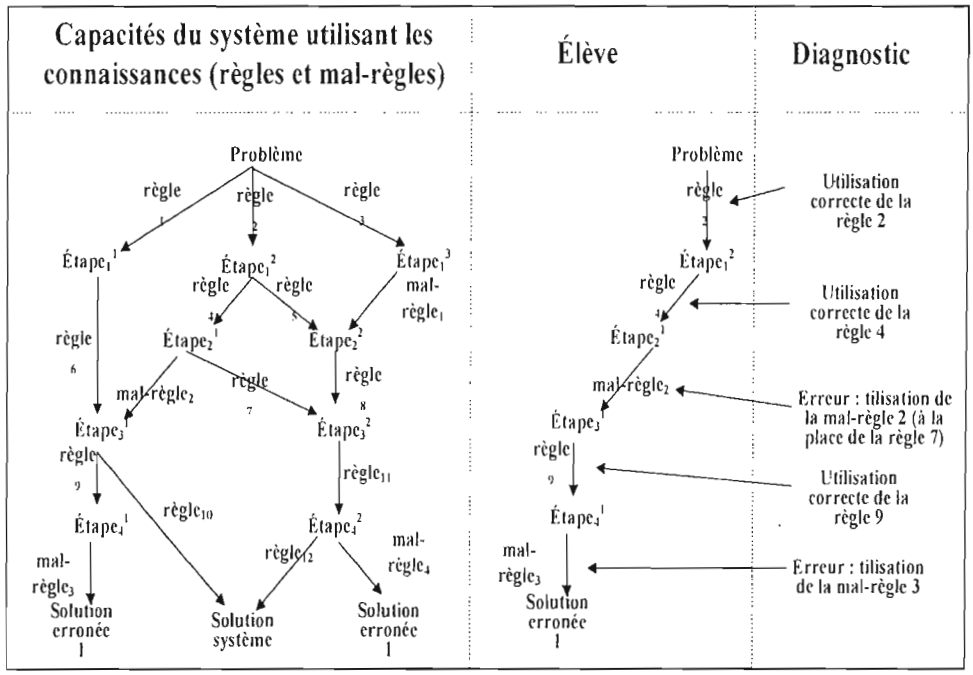


Figure 4.33 Principe du diagnostic par traçage de modèle.

Le système ne vise pas simplement à générer une solution correcte au problème posé, mais plutôt à générer un grand nombre de cheminements possibles vers cette solution, cheminements que pourrait emprunter un élève en train de résoudre le

même problème. Cependant, il reste à savoir dans quelle mesure les cheminements empruntés par un apprenant utilisant le système seront détectés et, par conséquent, pourront faire l'objet d'un diagnostic fiable. Pour terminer, il faut noter que le traçage de modèle a facilité grandement la reconnaissance de plan de l'élève.

4.7.3 Mise à jour du modèle de l'élève

Au moment où l'élève qui résout un problème comme par exemple *emprunter sur un zéro*

$$\begin{array}{r} 904 \\ \underline{486}^{-} \end{array}$$

peut obtenir un message d'analyse (à sa demande) dans les premiers pas de résolution notamment en formulant l'un ou l'autre des ensembles suivants d'actions et de stratégies :

- Emprunter sur zéro \ Décrémenter la colonne.
- Emprunter sur zéro \ Emprunter 2 fois (avancer d'une colonne) \ Décrémenter la colonne.
- Emprunter sur zéro \ Avancer d'une colonne \ Avancer d'une colonne \ Décrémenter la colonne.

Comment interpréter ces différents comportements *corrects* pour mettre à jour le modèle d'apprentissage? Et plus spécifiquement, un élève manifeste-il une connaissance particulière.

- du fait même de décomposer la tâche sans égard à la complexité du problème?
- ou du fait d'utiliser une stratégie de résolution sans égard au nombre d'emprunts?
- ou encore du fait de formuler une solution générale, utilisant ainsi des actions ou stratégies particulières?

La réponse à ces questions est bien sûr qu'il ne faut pas simplement se fier aux règles déclenchées pour les fins de la mise à jour du modèle de l'élève, même si chacune de ces règles manifeste un certain comportement de l'élève. Et c'est pourtant ce qui arrive actuellement dans le système TIDES. Dans la présente version en effet, un élève se trouve parfois *félicité,* en termes de connaissances réputées maîtrisées, par le fait qu'il utilise un plan spécifique pour atteindre un but donné, et cela simplement parce que ce plan occasionne le déclenchement d'un plus grand nombre de règles. Des versions subséquentes du système TIDES devront être dotées d'un processus d'analyse plus raffiné afin de discriminer davantage entre des solutions équivalentes en apparence mais qui, en tenant compte du contexte, s'avèrent différentes sur le plan stratégique.

4.7.4 Acquisition des probabilités et modèle d'Anderson

4.7.4.1 Contexte d'activation d'une règle

Le modèle d'Anderson postule que les règles de production sont sélectionnées selon des critères probabilistes. Une règle a de forte chance d'être activée si elle est souvent utilisée et si ses conditions de production sont appropriées au contexte en question. Si elle n'est pas activée, c'est tout simplement parce qu'il n'y a pas d'adéquation suffisante entre ses conditions de production et les éléments du contexte. Cependant, une question que nous devons dès lors nous poser lorsque nous concevons une évaluation diagnostique est : *dans quel(s) contexte(s) l'élève devrait-il être capable d'activer les règles de production qu'il a apprises* ?

Si un élève se révèle capable d'activer une règle de production dans un contexte donné, il y a de fortes chances qu'il active cette règle de production dans des contextes semblables. Par contre, nous ne pouvons faire aucune prédiction sérieuse quant à la probabilité que l'élève active cette règle de production dans des contextes très différents. Soulignons-le à nouveau : une règle de production a d'autant plus de chance d'être activée qu'elle a été souvent mise en œuvre et que ses conditions de production sont en accord avec les caractéristiques du contexte. Par conséquent, la maîtrise d'une compétence suppose un travail intensif où

chacune des composantes est acquise tour à tour et où ses conditions de production sont progressivement affinées.

4.7.4.2 Estimation des probabilités de départ

À la suite d'expériences avec des élèves en situation de résolution de problèmes, Anderson et son équipe ont pu établir, pour chaque règle de production en cause, des probabilités que l'élève possède ou non la connaissance, et qu'il le manifestera ou non par son comportement à la première occasion. Ces probabilités de départ sont présentées comme des moyennes mais aucune indication sur les écarts entre les règles de production n'est fournie. Quelles estimations doit-on utiliser au départ pour les fins de la mise à l'essai du système? Un même ensemble de probabilités peut-il être utilisé pour toutes les règles de production ou faut-il au contraire appliquer des probabilités différentes en fonction du niveau de difficulté de chaque règle, par exemple ?

Pour réaliser les premières mises à l'essai, le modèle de l'élève est fondé sur les estimations moyennes de probabilité obtenues à partir des travaux d'Anderson et de son équipe (notamment le modèle de traçage de connaissances). Les mêmes probabilités de départ sont assignées à toutes les règles de production et les mal-règles. Ce modèle probabiliste est fondé sur les paramètres suivants :

- une probabilité p(A) qu'une règle est acquise avant la première occasion de son application ;
- une probabilité p(B) que, si cette règle n'est pas acquise à la première tentative, elle le sera probablement à la suite ;
- une probabilité p(C) que l'élève manifestera le bon comportement alors qu'il ne maîtrise pas la règle ;
- une probabilité p(D) que l'élève affichera un comportement erroné alors qu'il maîtrise la règle.

En général, les valeurs de ces paramètres peuvent être placées empiriquement et peuvent varier d'une règle à l'autre, mais Anderson et son équipe ont pu établir

des valeurs approximatives moyennes (qui ont été jugées constantes) de ces paramètres comme étant :

p(A)=0.6, p(B)=0.4, p(C)=0.2, p(D)=0.2

Mais des mécanismes sont immédiatement prévus pour ajuster ces estimations en cas de besoin et discriminer entre les différentes règles à la suite des premières mises à l'essai du système. Cette étape consiste à spécifier les tables de probabilités. Ces probabilités spécifient l'ensemble initial du système de croyances d'un élève, avant la première interaction avec le système TIDES. On assigne d'abord à chaque règle une probabilité initiale que l'élève a appliqué correctement cette règle (mal-règle). Cette probabilité évolue de manière différente pour chaque élève en fonction de la performance manifestée au cours de son apprentissage.

4.7.4.3 Évolution des probabilités en fonction de la performance

Les expériences d'Anderson et de son équipe ne fournissent pas non plus d'indications sur la formule qui détermine l'évolution d'une probabilité à la suite d'une performance donnée de l'élève. Si, au départ, il y a 60% de chances qu'un élève puisse appliquer correctement une certaine règle, que devient cette probabilité après un essai fructueux d'application? Et après un essai infructueux? Il est certain que cette probabilité augmente dans le premier cas et diminue dans l'autre mais seule l'expérience permettra de savoir dans quelle proportion.

Dans le cadre d'apprentissage évoluant dans le temps, le diagnostic vise à fournir une estimation de l'état dans lequel se trouve l'élève que l'on observe et de remettre à jour cette estimation périodiquement ou chaque fois que de nouvelles observations sont disponibles. L'état de l'élève se modifie régulièrement pour des raisons souvent liées à sa performance ou du moins des raisons non-observables (cachées). Il est donc nécessaire de pouvoir observer, au moins partiellement, l'état de cet élève et après chaque observation de pouvoir ré-estimer l'état dans lequel il se trouve. Dans le cas du diagnostic arithmétique, par exemple, les observations sont la plupart du temps les erreurs que présente

l'élève et l'état à estimer est son état d'apprentissage, à partir duquel un diagnostic peut être plus facilement déduit par le système TIDES.

Dans le système TIDES, la dynamique de modification des probabilités au cours d'une séance d'apprentissage est établie en fonction des performances de l'élève. Cette simulation est effectuée grâce à un modèle simple qui détermine l'évolution des probabilités en fonction des succès et échecs de l'élève. Concrètement le modèle prend la forme suivante : pour chaque règle de production, le tuteur établit une probabilité que l'élève connaisse cette règle. Il ajuste cette probabilité selon l'essai fructueux ou infructueux en utilisant le modèle simple suivant.

PI = probabilité initiale (que l'élève va correctement appliquer une règle)
NP = nouvelle probabilité calculée à la suite d'une certaine performance

- Après un essai fructueux, on ajoute à PI la moitié de l'écart entre PI et 1 :

  NP = PI + 0,5 * (1 - PI)

- Après un essai infructueux, on ampute PI de 0,2 fois sa valeur courante (sans toutefois descendre à une valeur inférieure à 0,4) :

  NP = Max (0,4 , 0,8 * PI)

Il faudra, bien sûr, réévaluer le bien-fondé cette formule à la suite de la mise à l'essai du système.

4.7.5 Exemple d'utilisation du module avec production d'un diagnostic d'une erreur probable

Nous présentons dans cette sous-section un exemple qui montre une autre capacité du système TIDES à diagnostiquer un autre type d'erreurs qui ne se produisent que très rarement, appelé « erreur aléatoire ». Comme le souligne Cox (1975), il y a des erreurs qui se présentent une fois ou deux sur un nombre limité de problèmes où l'élève sait au fond comment exécuter le calcul correctement, mais due aux distractions, des ennuis ou à un moment d'inattention, il commet cette erreur (voir chapitre 5 tableau 6.3). L'exemple suivant montre comment le système peut parfois trouver les causes de ce type d'erreur.

Supposons qu'un apprenant, pour résoudre le problème 704 – 286 =, ait trouvé le résultat 438 présenté dans la figure suivante (figure 4.34), selon ce résultat, le système ne peut savoir exactement comment l'apprenant arrive à cette solution : pour la première colonne et la troisième colonne, le résultat est correct, par contre il ne l'est pas pour la deuxième colonne. L'apprenant a commis une erreur non systématique, c'est une erreur aléatoire qui ne se produit pas souvent. Apparemment l'apprenant a procédé de la façon suivante : il a emprunté 1 de la troisième colonne et il a soustrait un autre 1 du chiffre du bas de la deuxième colonne, ce qui donne (10 – (8 – 1 = 7) = 3).

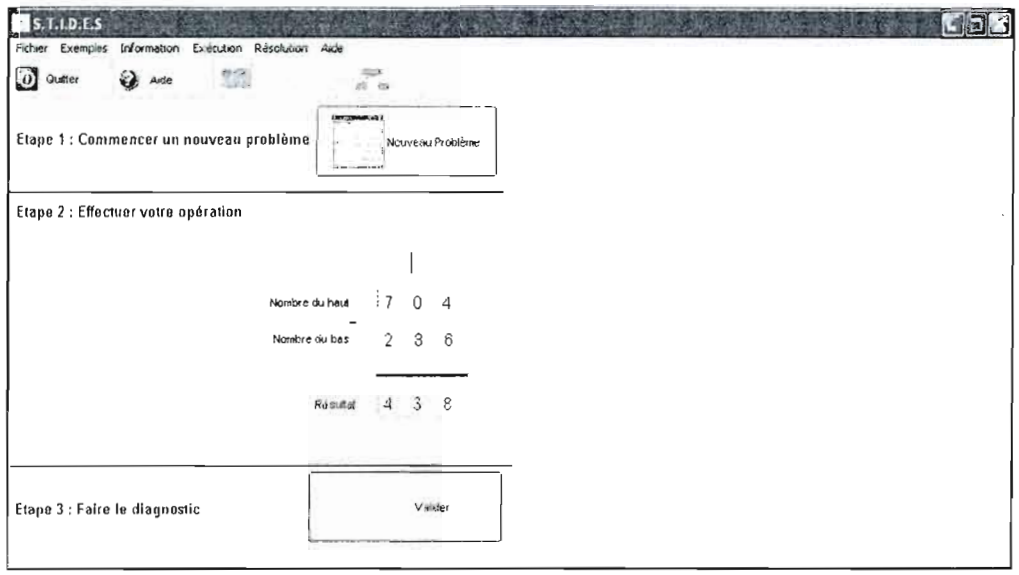


Figure 4.34. Problème de l'élève avec une erreurs aléatoire

Lorsque l'apprenant demande la validation de son résultat, le système TIDES échoue de trouver au premier coup la procédure utilisée par l'élève dans la deuxième colonne; cependant, il affiche un message avec trois hypothèses différentes (figure 4.35) :

- **première hypothèse** : quand vous empruntez d'une colonne de la forme 0-N, vous décrémentez le N au lieu de zéro ;
- **deuxième hypothèse** : quand vous empruntez d'une colonne dans laquelle le chiffre du haut est plus petit que celui du bas, vous incrémentez au lieu de décrémenter ;

- **troisième hypothèse** : quand vous empruntez, vous décrémentez le plus grand nombre dans la colonne, peu importe qu'il soit en haut ou en bas.

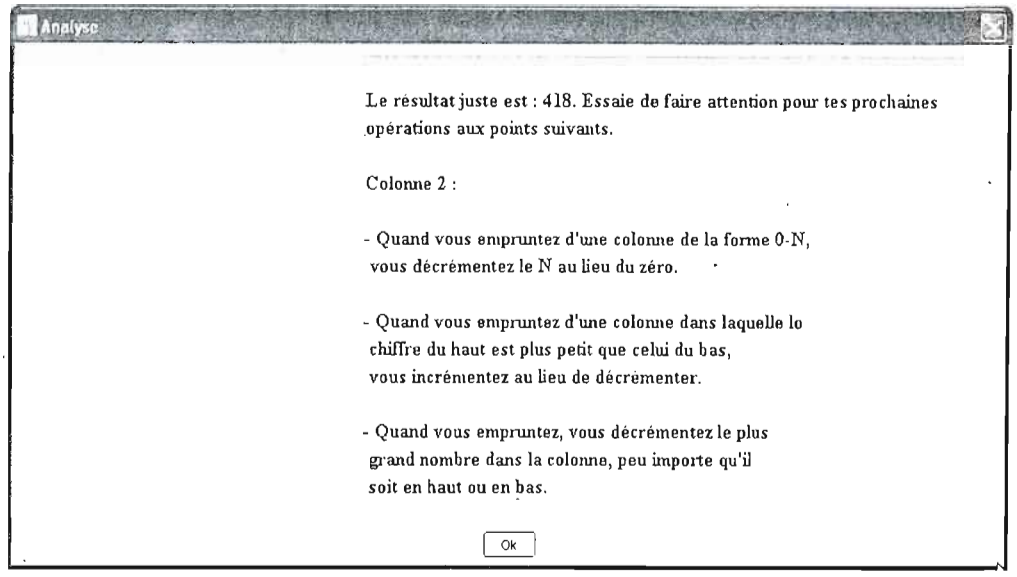


Figure 4.35. Message d'analyse avec trois hypothèses

Devant cette situation, le système génère trois problèmes tests, grâce à son moteur d'inférence, pour savoir exactement quelles procédures se trouvent derrière cette erreur.

En effet, comme on peut le constater, la première hypothèse concerne un type d'erreurs qui ne se produisent que dans une colonne de la forme 0-N. Le système doit alors proposer à l'élève un premier problème test qui comprend, dans sa deuxième colonne, cette forme afin d'éliminer cette hypothèse ou la confirmer. C'est d'ailleurs ce qu'il a fait. Il a proposé le problème test : 308-249 (figure 4.36).

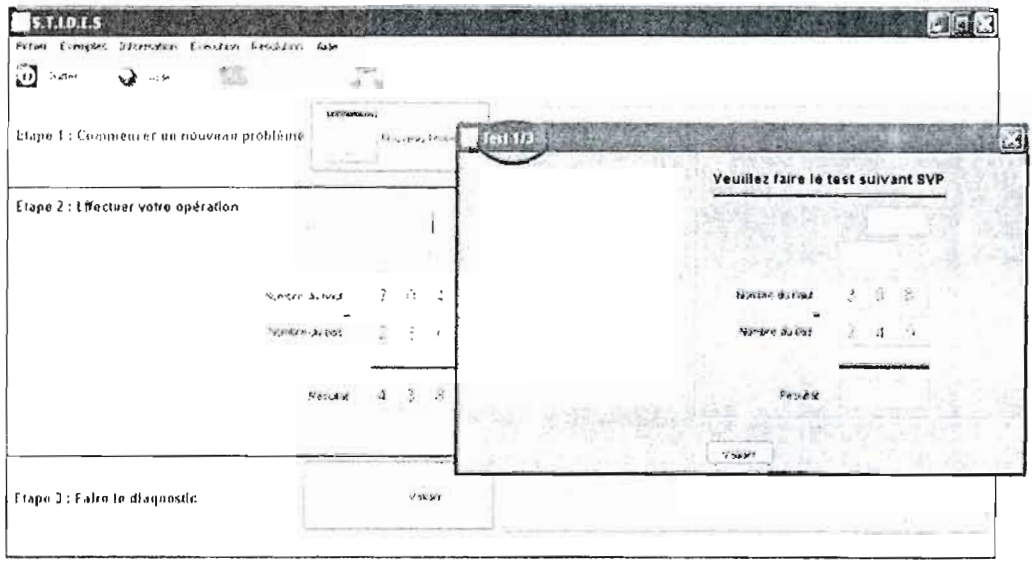


Figure 4.36. Premier problème test

On suppose que l'élève a répondu de façon correcte à ce premier problème test, comme le montre la figure 4.37. Il a décrémenté la troisième colonne (3-1), ensuite il a ajouté 10 à la deuxième colonne (donc, il n'a pas décrémenté le 4), puis il a emprunté 1 de cette colonne pour ajouter 10 à la première, ce qu'il lui donne un résultat correct. Donc, la première hypothèse est alors écartée et les erreurs liées à cette situation sont éliminées, c'est-à-dire que l'élève n'a pas décrémenté le N.

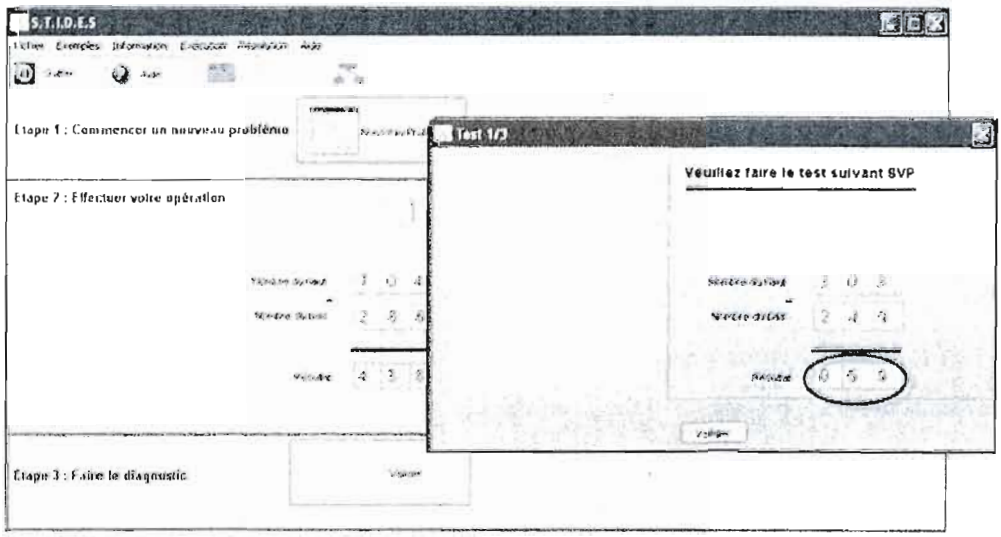


Figure 4.37. Résultat correct du premier problème test

Comme le résultat de l'élève est correct, le système génère un deuxième problème test dans le but d'écarter ou confirmer la deuxième hypothèse (figure 4.38).

Cette fois-ci, le système a changé le type de problèmes afin de cerner un peu plus le problème de l'erreur et de bien avancer le diagnostic. En effet, il a proposé le problème 833-277 (colonne 2, le chiffre du bas est plus grand que le chiffre du haut) pour savoir si l'élève va décrémenter ou incrémenter le chiffre du haut lorsqu'il va faire l'emprunt.

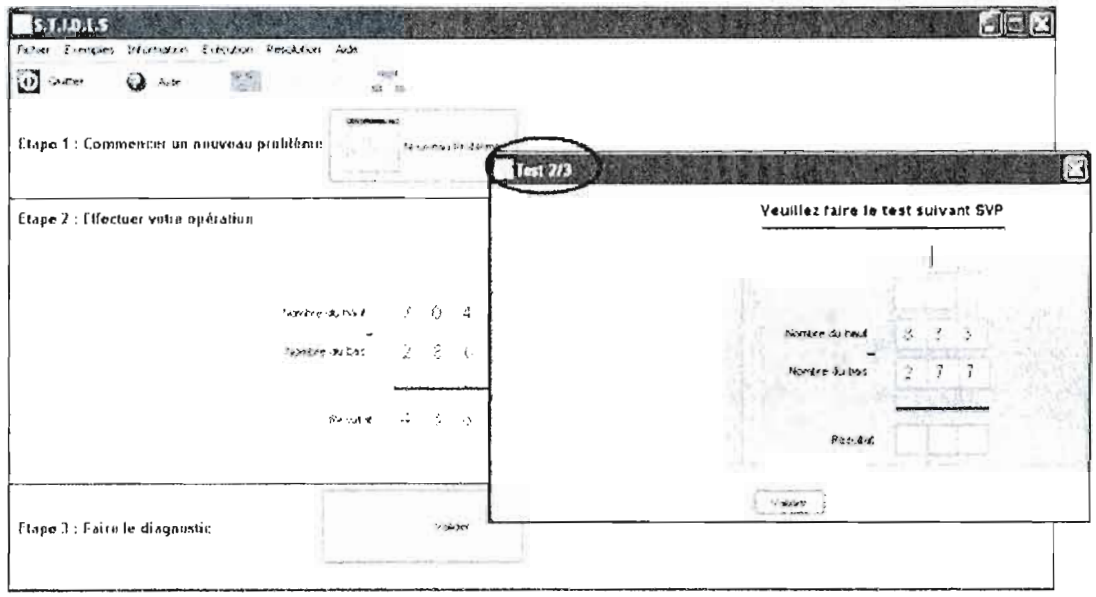


Figure 4.38. Deuxième problème test

Supposons une autre fois que le résultat de l'apprenant soit correct (pour ce deuxième test) (figure 4.39). La deuxième hypothèse est écartée, l'élève a bien décrémenté le chiffre du haut, malgré que ce chiffre soit plus petit que celui du bas. Les erreurs liées à ce type de problèmes sont éliminées : comme par exemple, l'élève incrémente le chiffre du haut au lieu de le décrémenter.

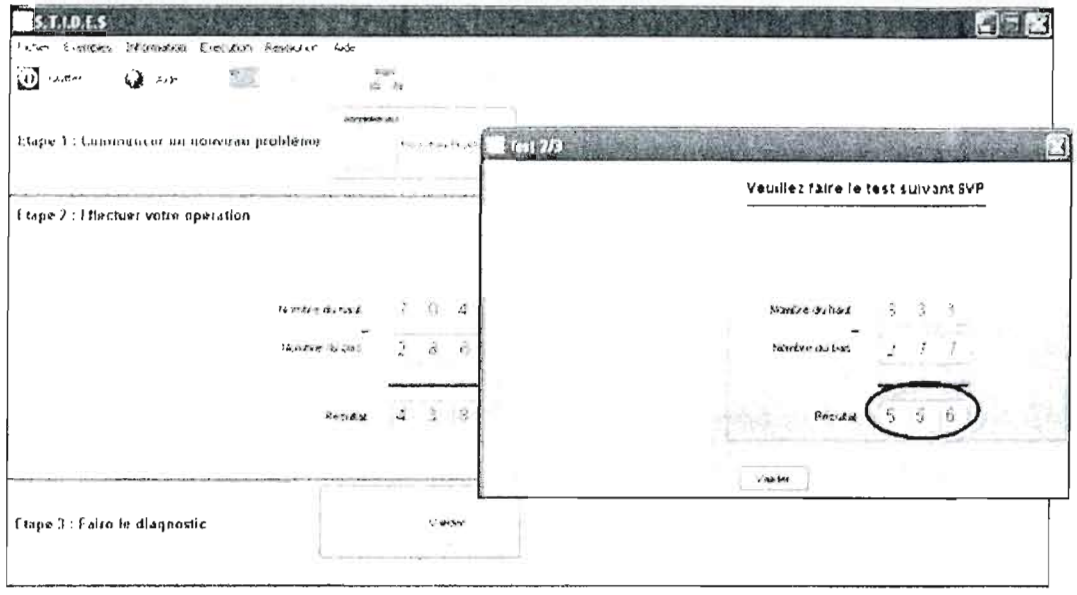


Figure 4.39. Résultat correct du deuxième problème test

Enfin un dernier problème test est proposé à l'apprenant 872-294 (figure 4.40)

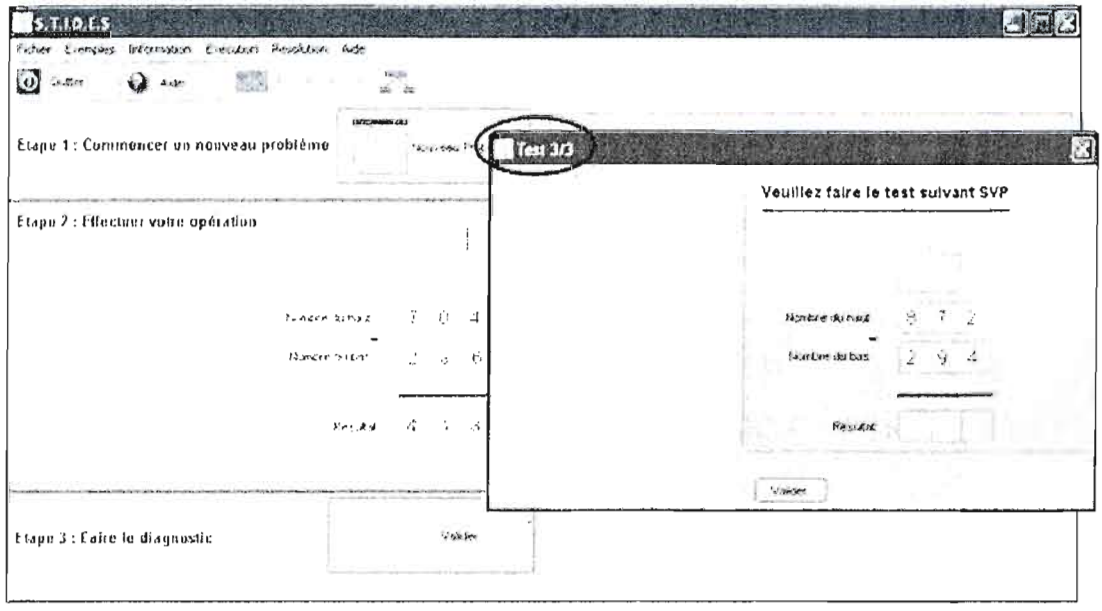


Figure 4.40. Troisième problème test

Deux cas sont possibles maintenant : soit l'apprenant trouve encore une solution correcte, le système dans ce cas ne peut avancer aucune hypothèse ; soit l'élève,

cette fois, donne une réponse fausse, le système peut aller plus loin et propose un diagnostic possible de l'erreur produite dans le problème initial.

- **Premier cas** : la solution de l'apprenant au problème test est correcte

Si la réponse de l'apprenant à ce dernier problème test est correcte, comme l'indique la figure suivante (figure 4.41), alors le système écarte la dernière hypothèse qui reste.

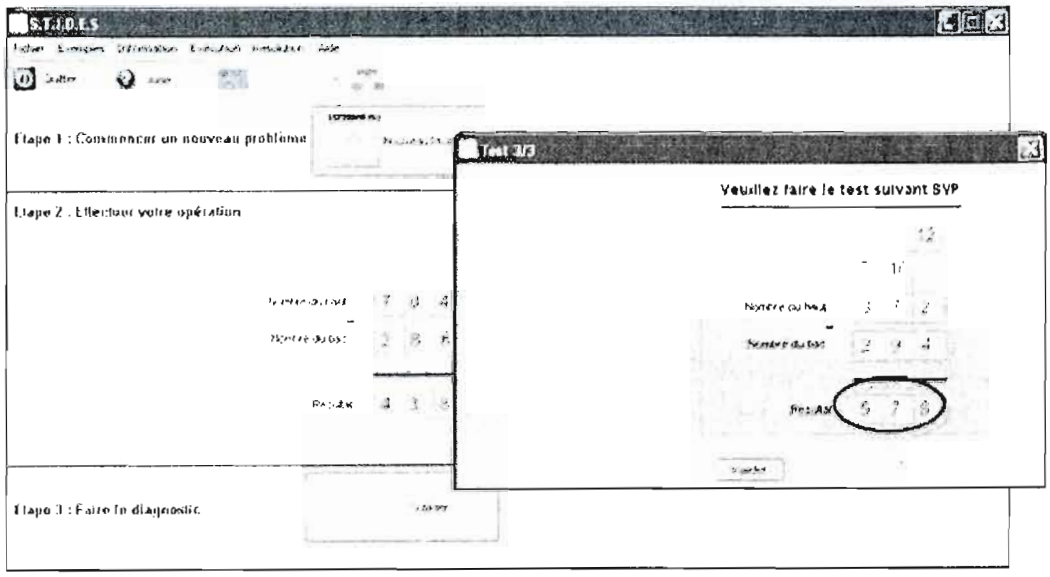


Figure 4.41. Solution correcte du troisième problème test

Ensuite, il affiche le message suivant (figure 4.42) «*je ne peux trouver aucune explication relative à vos erreurs, faites attention au moment où vous entrez votre réponse* ».

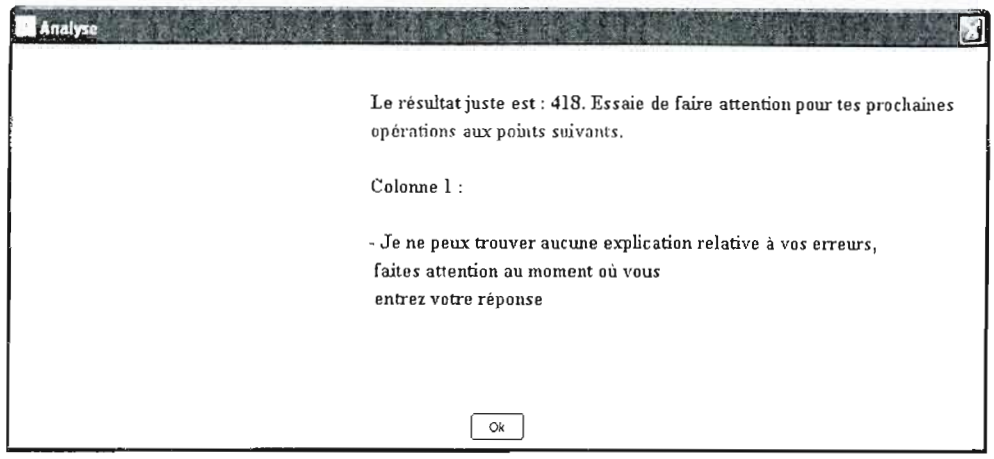


Figure 4.42. Message après la solution correcte du troisième problème test

- **Deuxième cas** : la solution de l'apprenant au troisième problème test est incorrecte

Si la réponse de l'apprenant au troisième problème test est fausse, comme l'indique la figure suivante (figure 4.43), le système dans ce cas garde la dernière hypothèse qui reste, puisqu'elle s'accorde avec la réponse erronée du problème initial.

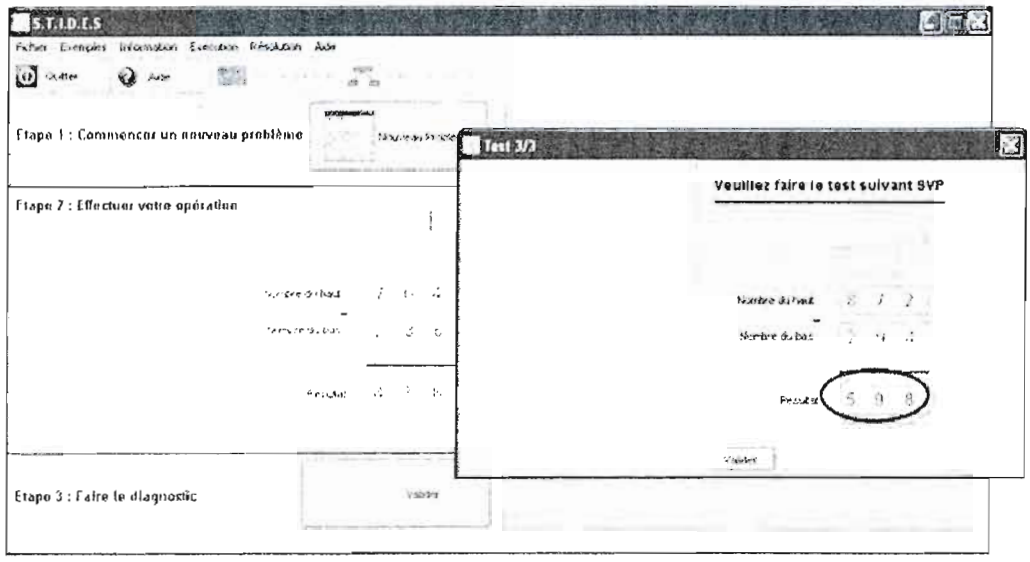


Figure 4.43. Solution incorrecte du troisième problème test

Le système peut maintenant identifier la nature de l'erreur et par conséquent il peut afficher un premier message d'analyse (figure 4.44).

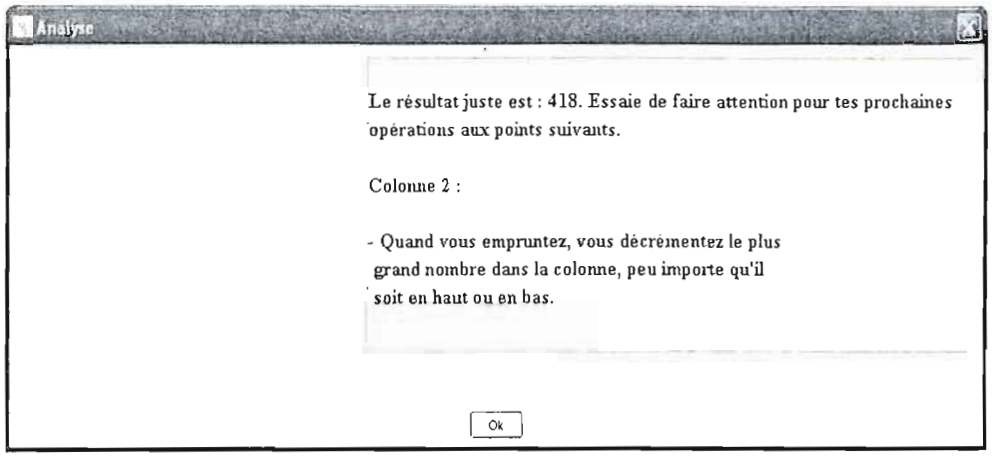


Figure 4.44. Message d'analyse après les problèmes tests

Cependant, un message de diagnostic (figure 4.45) du deuxième module (TIDES-Diagnostic) très précis est affiché. Ce message détermine les causes réelles de cette procédure erronée.

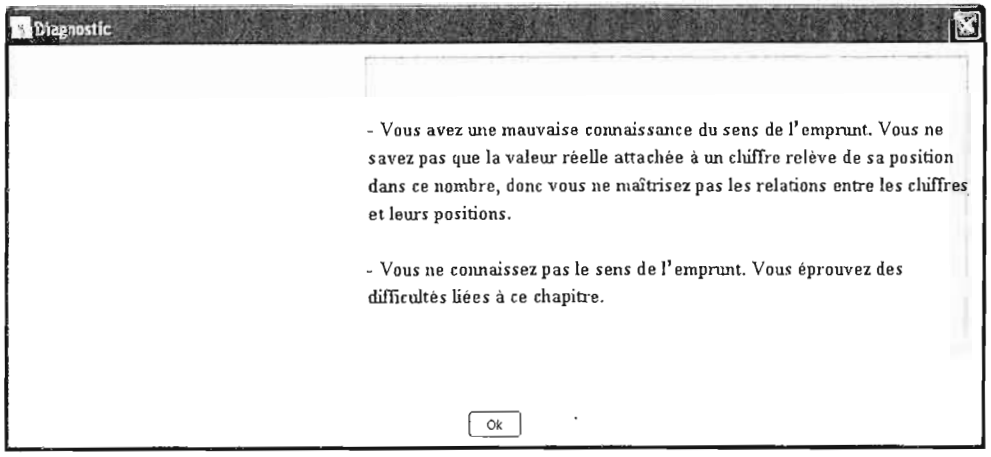


Figure 4.45. Message de diagnostic après les problème tests

On peut déterminer maintenant le profil de l'élève. En effet, l'apprenant éprouve des difficultés d'une part au niveau de l'utilisation de l'emprunt, il ne comprend pas le sens exact de l'emprunt. D'autre part il ne maîtrise pas les relations entre les chiffres et leur position.

CHAPITRE V

MISE À L'ESSAI DU SYSTÈME

Ce chapitre présente, d'une part, la méthode qui a été utilisée pour vérifier que le système d'apprentissage développé donne les résultats attendus, notamment en rapport avec le modèle d'Anderson et en ce qui a trait à la nature d'analyse, aux modalités d'explication et à la qualité du diagnostic proposé. D'autre part, les données recueillies au cours de la mise à l'essai du système TIDES sont présentées et analysées. Ces données comprennent des informations enregistrées dans le fichier LOG au cours des séances d'apprentissage (voir 2.3.1), des données recueillies à l'aide d'un questionnaire sur les perceptions des élèves après les séances d'apprentissage (voir 2.3.2 et annexe B) et quelques données pertinentes liées au rendement scolaire des élèves qui ont participé à la mise à l'essai.

6.1 Introduction

Un système tutoriel intelligent est un produit qui exige, avant sa mise en service, une validation et une évaluation comme d'autres types d'applications informatiques. La validation porte sur certains aspects de la performance du système. L'évaluation, quant à elle, porte, d'une part, sur les aspects techniques du système (comme la présentation des images d'écran, la facilité de son utilisation et les messages d'information), d'autre part, sur la valeur éducative du système, en tant qu'outil d'enseignement ou d'apprentissage en indiquant si le système réussit ou échoue par rapport à son objectif.

Dans cette perspective, les sous-questions de la recherche sont associées à la mise à l'essai du système TIDES en tant que système intelligent d'apprentissage de l'arithmétique. Ces sous-questions s'inscrivent dans l'aboutissement de la démarche de développement du système TIDES :

- L'utilisation du système TIDES par un élève donne-t-elle effectivement lieu à un apprentissage ?
- Cet apprentissage est-il conforme à celui qui était prévu en se basant sur le modèle d'Anderson ?
- Le système TIDES assure-t-il une grande efficacité de diagnostic ?
- L'utilisation d'une rétroaction à deux niveaux (analyse, diagnostic) permet-elle de concilier les avantages d'une rétroaction immédiate avec ceux d'une plus grande latitude laissée à l'élève ?

Les sections suivantes du présent chapitre développent ces questions et les critères employés pour évaluer les apprentissages dans le cadre de la mise à l'essai du système TIDES. Cette mise à l'essai implique d'abord la mise au point d'une façon de mesurer les performances des élèves afin d'évaluer les apprentissages réalisés. À cet effet, la méthode qui a été utilisée pour vérifier que le système développé donne les résultats attendus est présentée dans la prochaine section. La méthodologie utilisée lors de l'expérimentation de divers tuteurs intelligents est étudiée dans la sous-section 2.1. La mise à l'essai du système TIDES implique ensuite une façon de segmenter les données pour discriminer entre les élèves en fonction de leur utilisation des deux nivéaux de rétroaction (sous-section 2.2). La méthodologie utilisée comprend divers instruments et procédures ainsi que des méthodes concernant en particulier le choix des participants (sous-sections 2.3 et 2.4).

Ensuite, la mise à l'essai du système développé est présentée dans la troisième section. Les données recueillies au cours de cette mise à l'essai du système TIDES sont présentées et analysées. Ces données comprennent des informations enregistrées dans le fichier LOG au cours des séances d'apprentissage (voir sous-section 2.3.1), des données recueillies à l'aide du questionnaire sur les perceptions des élèves après les séances d'apprentissage (voir sous-section 2.3.2 et annexe B) ainsi que quelques données tirées des résultats des élèves qui ont participé à la mise à l'essai. Les connaissances d'arithmétique et les tâches assignées sont

clairement identifiées (sous-section 3.1). L'analyse des résultats porte sur la performance du système (section 3.2), les apprentissages réalisés (sous-section 3.3), la rétroaction à deux niveaux (sous-section 3.4) et l'accès aux autres informations pendant la résolution (sous-section 3.5).

6.2 Méthodologie de mise à l'essai du système TIDES

6.2.1 Méthodologie de mise à l'essai utilisée par Anderson

Anderson et son équipe (Anderson & al., 1985) ont réalisé les expérimentations du Lisp-Tutor visant notamment à analyser la règle de production comme unité de connaissance associée à une habileté de programmation en langage Lisp. Une séquence de touches par l'élève et de réponses par le tuteur sont les principales données à analyser. Les données recueillies sont fondées sur un cycle d'apprentissage en trois étapes :

- le tuteur propose à l'élève une tâche de codification ;
- l'élève, en utilisant le clavier, entre un code correspondant à l'activation d'une règle de production ;
- le tuteur évalue le code et émet un message approprié : (1) si le code est correct, le tuteur propose un nouveau but et le cycle recommence ; (2) si le code est incorrect, le tuteur fournit une rétroaction et repropose le même but ; (3) à la demande de l'étudiant, la réponse correcte est affichée et un nouveau but est proposé.

Dans leur expérimentation, les auteurs analysent deux aspects relatifs à l'activation d'une règle de production par l'élève :

1. le temps d'activation requis lors de l'accomplissement d'une tâche. Le temps d'activation est défini comme l'écart entre le moment où le tuteur est prêt à recevoir une entrée et le moment où l'élève complète son entrée. Il n'est mesuré que pour les buts atteints (règles utilisées correctement) du premier coup par l'élève.

2. le nombre de tentatives nécessaires pour y arriver. Il fait l'objet de deux mesures : (1) la probabilité qu'un élève réponde correctement du premier coup et (2) le nombre de tentatives successives pour arriver à une réponse correcte après une première réponse incorrecte. Les auteurs remarquent que cette deuxième mesure s'est avérée plus pertinente parce qu'une première réponse erronée peut être causée par une simple distraction alors que des erreurs répétées révèlent souvent des difficultés réelles de compréhension.

6.2.2 Conformité aux prévisions du modèle d'Anderson

La mise à l'essai effectuée auprès d'élèves qui utilisent le système TIDES se réfère dans la partie cognitive à celles réalisées par l'équipe d'Anderson, puisque les questions relatives aux apprentissages réalisés impliquent de comparer certaines données recueillies lors de la mise à l'essai du système TIDES (voir sous-section 3.3) avec certaines observations relevées notamment pendant la mise à l'essai du LISP-TUTOR. Pour être en mesure d'effectuer cette comparaison et ainsi de vérifier *la conformité aux prévisions du modèle*, il faut, d'une part identifier, parmi les résultats obtenus par Anderson, ceux qui permettent d'établir des liens avec les éléments-clés de son modèle sur le plan des apprentissages réalisés, et d'autre part s'assurer que la nature et les modalités de cueillette des données permettent une telle comparaison.

Anderson a identifié, parmi les résultats obtenus lors de l'expérimentation de ses systèmes, ceux qui établissent, dans une certaine mesure, les résultats que des chercheurs sont *en droit* d'attendre quand ce modèle est appliqué. Ces études se rapportent aux caractéristiques des règles de production et de leur apprentissage. Il s'agit plus spécifiquement des deux conclusions suivantes (Anderson, Conrad et Corbett, 1989) :

- *la règle de production est une unité de connaissance apprise indépendamment des autres unités de connaissance* ; cette caractéristique est fondée sur la régularité des courbes déterminées par l'apprentissage des

règles de production et sur le fait que ces courbes ne montrent aucune dépendance des règles similaires dans une analyse factorielle ;

- *il y a une fonction de puissance entre la performance et la pratique* ; on a pu observer en effet une relation linéaire entre le logarithme de la performance et le logarithme de la pratique, du moins à partir de la deuxième occasion d'appliquer une règle.

6.2.3 Choix des activités et des données à recueillir

L'objectif de cette sous-section est de déterminer les activités qui feront l'objet d'observations durant la mise à l'essai ainsi que les données à recueillir lors de chacune de ces activités. Pour y arriver, il s'agit le plus souvent d'indiquer comment ces données permettent de répondre aux questions posées précédemment (voir section 1) en discutant, s'il y a lieu, les traitements à effectuer.

Pour encadrer cette démarche, les étapes d'un cycle d'apprentissage dans les systèmes d'Anderson et dans le système TIDES sont utilisées. De cette façon, il est plus facile de dégager les éléments à respecter pour assurer la comparabilité des résultats. Comme il a déjà été mentionné dans la sous-section 2.1, les données recueillies lors des expérimentations du Lisp-Tutor conduites par Anderson sont fondées sur un cycle d'apprentissage en trois étapes : une étape *assignation*, une étape *input de l'élève*, et une étape *rétroaction du tuteur*. Ces trois étapes sont bien sûr précédées d'une *phase instruction* durant laquelle l'élève prend connaissance du contenu d'apprentissage de la leçon qu'il aborde à l'ordinateur,

Le cycle d'apprentissage du système TIDES comporte plusieurs similitudes avec les tuteurs d'Anderson, notamment le Lisp-Tutor. On retrouve en effet dans TIDES les trois mêmes étapes ainsi qu'une phase instruction. Pour pouvoir définir et situer chacun des événements ayant fait l'objet d'observations durant la mise à l'essai du système TIDES et pour pouvoir expliquer la façon d'assurer une comparabilité des données recueillies (par l'équipe d'Anderson) avec celles

relatives au Lisp-Tutor, il est maintenant utile de revenir sur chacune des étapes décrites mais cette fois, en fonction du système TIDES.

6.2.3.1 Phase instruction

Durant la phase instruction, qui doit précéder l'étape de mise à l'essai, un bref exposé est présenté oralement aux élèves sur la structure du système TIDES, son mode de fonctionnement, les méthodes de contrôle, son interface et ses objectifs. Puis deux problèmes sont présentés et résolus avec les élèves selon le mode de travail du système. Une fois la phase instruction terminée, l'essentiel de ces informations ainsi que les exemples de problèmes avec solutions et commentaires appropriés demeurent accessibles en tout temps à l'écran, pendant la résolution d'un exercice quelconque. De plus, et pendant toute l'étape de mise à l'essai du système, leur enseignant de français-mathématique circule dans la classe et répond à toutes les questions qu'ils se posent et s'il est nécessaire, il explique les messages d'analyse et de diagnostic qui ne permettaient pas à l'élève de comprendre quelle erreur il avait commise. Les directives étaient d'aider les élèves sur les problèmes liés à l'utilisation du système et sur la compréhension des messages et de ne pas leur fournir d'assistance au niveau de la résolution des problèmes.

Afin d'appréhender l'utilisation de l'instruction dans les comportements de l'élève, chaque fois que l'élève fait une action liée à la résolution d'un problème, l'événement est enregistré dans un fichier LOG (voir sous-section 2.3.1).

6.2.3.2 Étape assignation

Dans le système TIDES, le tuteur assigne une tâche dont la résolution implique, dans le cas le plus simple (problème sans emprunt), la formulation d'une action élémentaire dans le bloc de résolution et dans le cas le plus complexe (problème avec deux emprunts, par exemple), la formulation de stratégies de résolution et d'actions non-élémentaires qui occasionnent l'utilisation des cases d'emprunts dans le bloc de résolution. On peut donc considérer que dans un tel système, l'élève s'assigne lui-même indirectement une sous-tâche chaque fois qu'il décompose la tâche principale en sous-tâches.

6.2.3.3 Étape input de l'élève

L'étape «input» de l'élève est évidemment la plus importante. Plusieurs événements sont enregistrés dans le fichier LOG durant cette étape pour recueillir des données susceptibles de fournir des réponses à différentes questions.

- Lorsque la tâche demande un « emprunt » (ou deux emprunts), les élèves pratiquent-ils la décomposition de cette tâche en sous-tâches ?
- Les élèves qui pratiquent la décomposition mettent-ils plus de temps à résoudre leurs problèmes? Font-ils moins d'erreurs ?
- Pendant la résolution d'un problème, les élèves utilisent-ils les informations auxquelles ils ont accès à l'écran et en particulier : (1) les consignes et l'aide du système TIDES ? (2) les exemples de problèmes avec solution commentée ? (3) l'exécution graphique de la solution formulée ?
- Les élèves utilisent-ils les informations pour connaître l'analyse et le diagnostic de leurs erreurs ?

Ce sont des questions particulières au système TIDES puisque, comme il a été expliqué plus haut, les tuteurs d'Anderson ne permettent ni la décomposition, ni l'accès à des informations comme celles qui ont été décrites. Le contrôle des actions des élèves fournit des réponses aux questions mentionnées :

- entrée et sortie dans l'interface de résolution (zone de travail) ;
- accès à la fenêtre d'information sur les consignes et l'aide du système, etc. ;
- accès à la fenêtre d'information sur les exemples ;
- accès à la proposition libre du problème (mode de travail) ;
- accès à l'exécution graphique de la solution formulée.

6.2.3.4 Étape rétroaction et autres interventions du tuteur

Dans le système TIDES, le tuteur n'intervient qu'après la déclaration de la *solution terminée* par l'élève (l'élève valide sa solution) alors que dans le tuteur Lisp-Tutor d'Anderson une intervention du tuteur est susceptible de se produire à

chaque caractère de *code* entré par l'élève. Cependant, les deux messages de diagnostic possibles de la part du tuteur sont donc *le code est correct* et *le code est incorrect.*

Dans le système TIDES, une action de l'élève et une réaction du tuteur peuvent porter aussi bien sur une stratégie de résolution simple que sur une décomposition d'une tâche à un niveau plus élevé. Cette différence sera sans doute utile pour évaluer la performance du système TIDES sur le plan de sa capacité d'analyser les solutions des élèves et de diagnostiquer leurs erreurs.

La dernière question posée dans la section 1 porte spécifiquement sur l'utilisation d'une rétroaction à deux niveaux. Dans le système TIDES en effet, l'élève reçoit une première rétroaction sous forme d'un message d'analyse de ses erreurs (décrit la nature d'erreurs). À sa demande, il peut également recevoir une deuxième rétroaction sous forme d'un message diagnostic plus explicite (pour savoir les causes de ses erreurs). Par rapport à ce type particulier de rétroaction, il est pertinent de savoir comment l'élève utilise sa liberté d'obtenir ou non un tel message explicite et d'établir certaines relations avec le temps requis et la performance dans les apprentissages.

Deux autres formes de rétroaction sont également accessibles à l'élève et ont été l'objet d'observations au cours de la mise à l'essai. Ce sont l'accès à la solution du système et la possibilité de procéder à l'exécution graphique de la solution formulée par l'élève. Concernant l'exécution graphique, il faut rappeler qu'elle constitue un diagnostic global auquel l'élève a accès quand ce dernier utilise le bloc de résolution pour formuler sa solution.

6.2.3.5 Cueillette des données

Le plan de la mise à l'essai de TIDES est composé des éléments suivants :

- un traitement expérimental consistant en séances individuelles d'apprentissage à l'ordinateur par les sujets ;
- un premier groupe d'observations recueillies au cours de ces séances d'apprentissage à l'aide d'un *fichier LOG ;*

- un second groupe de données recueillies immédiatement après la séance d'apprentissage sous la forme d'un *questionnaire* avec échelle à intervalles ;
- un dernier groupe de données recueillies à partir du rendement scolaire de chaque élève[17].

L'essentiel de ce plan a été réalisé une première fois lors d'un premier test du système avec un groupe de trois élèves. Ce test nous a permis d'effectuer quelques ajustements au système et de préciser les données à recueillir.

6.2.3.6 Fichier LOG

Pendant la séance d'apprentissage à l'ordinateur, les données recueillies, pour chaque sujet et pour chaque cycle de résolution, sont :

- le niveau de la résolution (décomposition en sous-tâches ou non) ;
- les demandes de messages explicites ;
- les analyses et les diagnostics effectués par le tuteur ;
- le nombre d'essais pour arriver à une réponse correcte ;
- le nombre d'erreurs commises par un apprenant ;
- le temps d'activation défini comme la durée entre le moment où le tuteur est prêt à recevoir une entrée et le moment où l'élève complète son entrée (Anderson & al., 1989).

Les fichiers LOG constitués au cours des séances d'apprentissage ont permis de recueillir, de façon complètement transparente à l'élève, des données résumant les interactions entre l'élève et le système. De fait, des centaines d'événements ont ainsi été enregistrés dans des fichiers individuels (la nature des quinze différents types d'actions fait l'objet du Tableau 5.1).

Pour chaque événement et selon sa nature, un certain nombre de paramètres ont été enregistrés sur le comportement de l'élève. Ces données permettent d'abord

[17] Les bulletins des élèves impliqués dans la mise à l'essai sont consultés pour fin de diagnostic. La confidentialité des informations concernant chaque participant est assurée.

une analyse fondée sur des distributions de fréquences. Elles sont également fertiles pour retracer, analyser des cheminements d'élèves et diagnostiquer leur état en processus d'apprentissage.

	Temps Entrée/Sortie	Tâche globale	Verdict/ Analyse	Verdict / Diagnostic	Nombre d'essais	Nombre d'erreurs	Règles activées	Actions simple	Actions complexe
Entrée globale du système	Entrée								
Proposition libre du problème	Entrée								
Entrée dans le bloc de résolution	Entrée	globale							
Sortie du bloc de résolution	Sortie	globale							
Solution terminée	Sortie	globale	Analyse	Diagnostic	Nombre	Liste	Liste	Nombre	Nombre
Info-consignes	E+S								
Aides	E+S								
Exemple 1	E+S								
Exemple 2	E+S								
Exemple 3	E+S								
Exécution graphique	E+S	globale		Diagnostic					
Demande solution	Entrée	globale	Verdict	Verdict					
Demande analyse	Entrée	globale	Analyse						
Demande diagnostic	Entrée	globale		Diagnostic					
Sortie globale	Sortie								

Tableau 5.1. Événements et paramètres enregistrés dans le fichier LOG

Le Tableau 5.1 résume les événements qui font l'objet d'observations et établit dans chaque cas, la liste des informations recueillies pour permettre les traitements et segmentations discutés plus haut. Certains événements enregistrés nécessitent cependant quelques explications.

- *Entrée globale du système, sortie globale* : l'élève débute ou termine une séance d'apprentissage.

- *Entrée dans le bloc, sortie du bloc et solution terminée* : l'élève entre/sort du bloc de résolution ou déclare sa résolution terminée.
- *Info-consignes, aides :* l'élève demande une information concernant l'un de ces deux sujets.
- *Exemple 1, exemple 2, exemple 3* : l'élève demande à consulter l'un des exemples avec solution commentée.
- *Proposition libre du problème :* l'élève choisit la formulation libre de son problème.
- *Exécution graphique* : l'élève demande à voir, sous forme de graphe (graphe de solution), son profil.
- *Demande solution :* l'élève demande et obtient la solution du système.
- *Demande analyse :* l'élève demande un message d'analyse à la suite d'un message d'erreur.
- *Demande diagnostic :* l'élève demande un message de diagnostic cognitif après le message d'analyse.

6.2.3.7 Questionnaire

Un questionnaire (voir Annexe C) utilisant une échelle à intervalles a été administré immédiatement après la séance d'apprentissage, afin de recueillir les opinions des élèves sur différentes facettes de leur expérience d'apprentissage, notamment la rétroaction à deux niveaux en fonction de la liberté de manœuvre. Ce questionnaire permet en particulier de connaître la perception des élèves concernant l'utilité et/ou la pertinence des analyses des erreurs (premier niveau de rétroaction), des messages diagnostiques écrits (second niveau de rétroaction), de l'exécution graphique d'une solution formulée, de la décomposition en sous-tâches, de l'exposé théorique initial (phase instruction), des informations et des exemples pendant la résolution.

Il est également pertinent de savoir s'ils ont aimé leur séance d'apprentissage, s'ils croient avoir acquis certaines habiletés, et si cette séance a changé quelque chose à leur intérêt pour la soustraction. Cependant, il faut souligner que ce questionnaire est présenté et expliqué aux élèves de façon explicite, question par question pour leur faciltc la tâche.

6.2.4 Choix des élèves participant à la mise à l'essai

6.2.4.1 Critères de sélection

Étant donné que le système TIDES a comme population-cible les élèves qui ont déjà étudié la soustraction (8-9 ans), c'est d'abord ce critère qui a été utilisé pour rechercher les sujets de la mise à l'essai. C'est pourquoi le choix s'est porté sur les élèves ayant terminé la troisième année primaire (en effet, il a été observé qu'une proportion croissante des élèves de niveaux plus élevés maîtrisent bien l'opération de la soustraction).

Le fait d'avoir déjà manipulé un ordinateur, pour des programmes éducatifs, navigation sur le Web ou simplement pour des jeux, a été ajouté comme deuxième critère d'inclusion. L'addition de ce critère a pour but de minimiser les apprentissages à réaliser pour familiariser les sujets avec le clavier et les autres éléments de l'interface. De cette façon, les temps observés pour résoudre les tâches durant la mise à l'essai risquent moins d'être biaisés par les autres apprentissages liés au système lui-même.

6.2.4.2 Choix des élèves

Nous avons choisi de travailler avec des élèves de la troisième année primaire (Cycle 3), car il nous semblait que dans ce cycle, l'enfant, relativement proche du stade des opérations formelles selon Piaget, serait capable de prendre suffisamment de recul par rapport à ses actions pour pouvoir les utiliser.

Le nombre d'élèves qui ont participé à la mise à l'essai est de 25 élèves comportant 13 filles et 12 garçons. Les enfants n'ont pas l'habitude de travailler à partir de leurs évaluations sur un matériel informatique. Ces élèves étaient divisés en deux groupes, ce qui a sans doute permis à chacun de travailler tout seul devant un ordinateur, de participer davantage que si le groupe était resté dans son entier.

Le premier groupe de 13 élèves l'a fait sur une base volontaire en mai 2005. À la suite d'une autorisation du directeur de l'école primaire de la délégation de Casa-Anfa (école francophone) dans laquelle les élèves terminant leur troisième année

primaire ont été invités à participer au projet. La mise à l'essai avec ce premier groupe d'élèves s'est déroulée en une seule séance d'apprentissage.

La mise à l'essai avec l'autre groupe de 12 élèves, s'est déroulée à la fin de l'année scolaire 2004-2005 dans le cadre des activités d'un cours d'initiation à l'informatique à l'école Oum Elkora de casablanca. La moitié des élèves inscrits à ce cours ont été choisis aléatoirement pour participer à la mise à l'essai. Celle-ci a occupé deux périodes pour ce groupe.

6.2.4.3 Conditions de déroulement de mise à l'essai

Il est certain que l'échantillonnage constitué par les élèves ayant participé à la mise à l'essai comporte des faiblesses qui peuvent être sources de biais dans les données observées et qui limitent de ce fait l'utilisation qu'on peut en faire. Il faut mentionner que les conditions de mise à l'essai ont différé dans les deux groupes d'élèves. En particulier :

- la mise à l'essai s'est déroulée dans un cadre scolaire pour l'un des deux groupes et durant le début des vacances d'été pour l'autre ;
- la mise à l'essai s'est déroulée en une seule séance pour l'un des deux groupes et en deux séances pour l'autre ;
- la mise à l'essai s'est déroulée au début d'une journée scolaire pour l'un des deux groupes et à la fin d'une journée scolaire pour l'autre ;
- la mise à l'essai s'est déroulée sur une base volontaire pour l'un des deux groupes et sur une base plutôt contraignante pour l'autre.

Dans la suite nous présentons en détail les démarches de la mise à l'essai du système développé.

6.3 Mise à l'essai du système TIDES

Dans cette dernière section, les données recueillies au cours de la mise à l'essai du système TIDES sont présentées et analysées. Ces données comprennent notamment des informations liées au profil d'élève et qui sont enregistrées dans le fichier LOG (voir 2.3.1), des données recueillies à l'aide du questionnaire administré immédiatement après les séances d'apprentissage qui porte sur les perceptions des élèves (voir 2.3.2 et annexe B) et d'autres données tirées des résultats scolaires des élèves participant à la mise à l'essai.

Avant d'aborder l'analyse des résultats, les connaissances d'arithmétique et les tâches assignées sont clairement identifiées en fonction des besoins spécifiques de la mise à l'essai (sous-section 3.1). L'analyse des résultats est ensuite essentiellement divisée en quatre sections portant respectivement sur la performance du système (sous-section 3.2), les apprentissages réalisés (sous-section 3.3), la rétroaction à deux niveaux (sous-section 3.4) et l'accès aux autres informations pendant la résolution (sous-section 3.5).

6.3.1 Connaissances d'arithmétique et tâches assignées

Une partie importante de l'analyse des résultats présentés dans cette sous-section est liée aux connaissances qu'un élève doit maîtriser à la suite de sa ou ses séances d'apprentissage. Aux fins de cette analyse, l'inventaire des tâches effectivement assignées aux élèves pendant les séances d'apprentissage a d'abord été effectué : une quinzaine de tâches ont ainsi été relevées (Tableau 5.2).

Tâche	Connaissances requises pour	
	le premier niveau de décomposition	l'ensemble de la tâche
Faire un problème de soustraction sans emprunt	2	2-3
Faire un problème de soustraction avec des zéro en bas : N-0	2-4	2-3-4
Faire une soustraction qui a les mêmes chiffres dans une colonne : N-N	1-2	1-2-3
Faire une soustraction qui a un vide en bas (ex : 456 - 43)	2-4	2-4
Faire un problème de soustraction avec un seul emprunt	2-3-5	2-3-3-3-5
Faire un problème de soustraction avec deux emprunts	2-5-6	2-3-5-5-6
Faire une soustraction avec un zéro dans la première colonne : 0-N	2-3-5-7	2-3-3-3-5-6-7
Faire une soustraction avec emprunt sur zéro (deuxième colonne)	2-3-5-7	2-3-3-3-5-6-7
Faire une soustraction avec deux zéro dans les deux premières colonnes	2-5-6-7	2-3-3-4-5-5-6-7
Faire une soustraction avec deux zéro dans une même colonne	2-4-5-6	2-3-3-4-5-6-7
Faire une soustraction avec emprunt sur 1 (après l'emprunt 1 devient 0)	2-4-5	2-3-4-5-6
Faire une soustraction avec emprunt sur 0 qui surmonte un vide	2-5-7	2-3-3-3-5-6-7
Faire une soustraction avec emprunt sur 1 qui surmonte un 1 : 1-1	2-4-5	2-3-4-5-6
Faire une soustraction avec emprunt sur 1 qui surmonte un vide	2-4-5	2-3-4-5-6
Faire une soustraction avec emprunt et la première colonne a la forme : N-N	2-3-4-5	2-3-4-5-6

Tableau 5.2 - Identification d'une quinzaine de tâches

Dans un deuxième temps, les connaissances que manifeste un élève en accomplissant chacune des tâches furent identifiées : ces connaissances reprennent pour l'essentiel celles qui ont déjà été utilisées dans le modèle de l'élève et qui ont déjà fait l'objet d'une description en chapitre 4. Pour analyser les résultats de la mise à l'essai, un regroupement un peu différent de ces connaissances a été défini, comme indiqué dans le Tableau 5.3.

Identificateur	Connaissance correspondante
'1'	Action simple : compétence logique de raisonnement
'2'	Sens de l'opération : connaissance du système de numération
'3'	Capacité de calcul : procédure de comptage
'4'	Disposition algorithmique : connaissance algorithmique
'5'	Sens de l'emprunt : connaissance conceptuelle
'6'	Emprunt itératif (deux fois l'emprunt) : connaissance procédurale
'7'	Emprunt à zéro : numération positionnelle - connaissance procédurale

Tableau 5.3 - Identification des connaissances retenues pour analyser les tâches du système TIDES

Comme le montre ce tableau, les quatre connaissances reliées à l'utilisation des stratégies de calcul (identificateurs '1' à '4') ont été intégralement conservées. Ces connaissances se traduisent par plusieurs règles de production dans le système TIDES. Quant aux trois autres connaissances (identificateurs '5' à '7'), elles correspondent dans les grandes lignes aux connaissances relatives aux trois niveaux de complexité. Une description plus rigoureuse de la complexité est cependant utilisée. C'est en fonction des séquences et imbrications d'actions (complexes ou élémentaires) et de stratégies de résolution impliquées que sont déterminés les trois niveaux de complexité. Comme l'indique le tableau 5.2, le premier niveau se réfère à une action conceptuelle peut être suivie ou précédée de l'une des quatre stratégies de calcul précédentes (comme par exemple l'utilisation du concept d'emprunt). Le deuxième nécessite deux actions complexes en séquence (impliquant généralement chacune une stratégie de résolution liée à un emprunt), c'est-à-dire une stratégie pour le premier emprunt puis une autre pour le deuxième emprunt. Quant au troisième niveau, il implique également deux actions complexes (deux stratégies de résolution) mais imbriquées plutôt qu'en séquence, c'est-à-dire une stratégie pour le deuxième emprunt, puis une autre pour le premier emprunt (par exemple la résolution du problème 600 – 489 demande deux stratégies de résolution imbriquées : emprunter pour le deuxième zéro d'abord puis emprunter de cet emprunt pour le premier zéro). En général, les connaissances relatives à chacun des trois niveaux de complexité se traduisent par plusieurs règles dans le système TIDES.

L'identification des connaissances mises en pratique durant l'accomplissement de chacune des tâches a donc été réalisée. Par exemple (voir Tableau 5.4), la tâche d'*emprunt conditionnel à un zéro* implique l'utilisation des connaissances suivantes :

$$\begin{array}{r} 600 \\ -489 \\ \hline \end{array}$$

- '2' : pour savoir qu'il y a absence de groupement dans la position correspondante ;

- `3´ (à trois reprises) : capacité de faire le calcul dans les trois colonnes ;
- `5´ : le sens de l'emprunt, ce que signifie l'expression «je vais emprunter chez le voisin»;
- `6´ : pour percevoir l'équivalence des écritures décomposées lorsqu'on va emprunter deux fois ; 600 = 2c + 9d +10u (c : centaine, d : dizaine, u : unité) ;
- `7´ : pour savoir le rôle de zéro dans l'emprunt, «l'emprunt sur le 0 exige qu'on doive aller à la valeur suivante pour emprunter le groupement nécessaire».

Ce qui forme le regroupement "2-3-3-3-5-6-7".

	Connaissances manifestées	
TÂCHE	**au premier niveau de décomposition[18]**	**dans l'ensemble de la tâche**
Faire une soustraction à trois colonnes avec un emprunt	2-3-5	2-3-3-3-5
S'il y a un zéro dans l'une des deux colonnes, aller emprunter dans la colonne suivante	2-3-5-7	2-3-3-3-5-6-7

Tableau 5.4 - Identification des connaissances associées à quelques tâches du système TIDES

Le Tableau 5.4 identifie également les connaissances manipulées lorsqu'une tâche est partiellement accomplie parce que l'élève n'a réussi qu'un premier niveau de décomposition de l'une des tâches. Dans l'exemple précédent d'*emprunt conditionnel à un zéro*, les connaissances associées à un premier niveau de décomposition sont `2´ `3´ `5´ et `7´ (une fois chacune).

Le Tableau 5.2 fournit, pour chacune des tâches assignées aux élèves lors des séances d'apprentissage, la liste des connaissances associées à l'accomplissement de l'ensemble de la tâche et la liste des connaissances associées à l'accomplissement du seul premier niveau de décomposition.

[18] Décomposition seulement pour la première colonne.

6.3.2 Performance du système

6.3.2.1 Performance observée en fonction des diagnostics rendus

La première qualité d'un système intelligent d'apprentissage est la capacité de réagir intelligemment grâce à une interprétation correcte du comportement des élèves. Le système est capable de décrire la séquence d'actions observées de l'apprenant, de reconnaître ses intentions, de savoir ses stratégies et d'identifier le plan poursuivi afin d'avancer un diagnostic plus précis et fiable. De plus, dans le cas d'ambiguïté, c'est-à-dire, quand le processus de reconnaissance obtient plus d'une hypothèse, le système prend d'autres démarches (proposition des tests) pour compléter sa discrimination et tomber sur une seule possibilité. Il est inévitable que certaines actions des élèves ne soient pas comprises par le système. Cette démarche complète constitue la mesure la plus pertinente de la performance d'un système d'EIAO.

Analyse	Diagnostic	Fréquences Gr. 1	Fréquences Gr. 2	Total	%
1. Soustraire le plus petit chiffre du plus grand chiffre	Généralisation d'une conception correcte dans certains cas particuliers	40	38	78	28,9
2. Pas d'emprunt à zéro	Mauvaise connaissance du sens d'emprunt surtout en présence de zéro	34	30	64	23,7
3. Emprunt à zéro sans changement de la colonne suivante à gauche	-Mauvaise compréhension du groupement -Mauvaise connaissance du sens d'emprunt	13	17	30	11,1
4. Emprunt par dessus de zéro sans changement de zéro en neuf	-Non maîtrise de la numération positionnelle -L'emprunt est considéré comme convention formelle et non comme une transformation d'écriture réorganisant le nombre	10	9	19	7,0
5. Non décrément de zéro (emprunt à zéro sans décrément cet emprunt)	-La décomposition n'est pas faite dans l'opération -L'itération de l'opération regroupement par 10 est oubliée -Le sens de l'opération n'est pas respecté	12	14	26	9,6
6. Zéro comme réponse à une colonne qui demande l'emprunt	- Le sens de l'emprunt n'est pas respecté -Limite de la charge cognitive	7	10	17	6,3
7. Emprunt au chiffre du dessous de zéro à la place d'un emprunt à zéro	-La signification de la disposition algorithmique est oubliée -Aucune considération de la position du chiffre dans le nombre où il faut emprunt	6	9	15	5,5
8. Erreurs instables	Je ne peux trouver aucune explication relative aux erreurs précédentes, mais faites attention au moment où vous entrerez votre réponse	5	6	11	4,0
9. Interruption de l'analyse	Aucun diagnostic	6	4	10	3,7
	Sous-total diagnostics (Nos. 1 à 7)	122	127	249	95.8
	Grand total des diagnostics	133	137	270	100,0

Tableau 5.5. Distribution des diagnostics rendus par le système TIDES

Dans le tableau 5.5, la fréquence des différents diagnostics rendus à la suite d'une déclaration solution terminée par un élève est présentée en données absolues et en pourcentage du total pour les deux groupes qui ont participé à la mise à l'essai (les interruptions d'analyse (diagnostic No 9) ne sont pas prises en compte dans le calcul de ce pourcentage).

Ces données (tableau 5.5) révèlent que le système TIDES a pu rendre un diagnostic efficace dans 95,8% ; c'est donc globalement neuf diagnostics sur dix que l'on peut qualifier fiable. On parle de fiabilité ici quand le système (1) identifie la séquence d'actions de l'élève (2) reconnaît son plan et ses intentions

(3) dispose de connaissances lui permettant d'analyser ces actions et de déterminer le but poursuivi. C'est là une bonne performance pour un premier prototype de système d'EIAO. Le diagnostic 8 (Je ne peux trouver aucune explication ...) ne constitue évidemment aucun inconvénient du système puisqu'il ne représente que 4,0% des diagnostics rendus. Cependant, notre méthodologie supposait dès le départ l'existence d'un tel diagnostic, introduit pour éviter le risque que le système interprète mal la solution de l'élève et lui donne un diagnostic faux (soit avancer des hypothèses pour une solution incorrecte due aux distractions, à des ennuis ou à un moment d'inattention,) ce qui aurait été pédagogiquement inacceptable (voir l'exemple détaillé dans le chapitre 4). On peut donc affirmer que, dans 95,8% des cas, le diagnostic rendu par le système TIDES est fiable.

Les différences de performance entre les deux groupes expérimentaux ne sont pas importantes. Relativement à ces différences, il faut rappeler que (voir 2.4) :

- les ordinateurs étaient moins puissants dans le laboratoire où s'est déroulée la mise à l'essai avec le groupe 1 ;
- dans le cas du groupe 1, il n'y a eu qu'une seule séance d'apprentissage d'une durée de trois heures et celle-ci s'est déroulée pendant l'année scolaire ; en comparaison, la mise à l'essai avec le groupe 2 a eu lieu durant la fin de l'année scolaire (début des vacances d'été) et a comporté deux séances d'une heure et demie ;
- les séances initiales d'apprentissage n'ont pas eu lieu durant la même période de la journée dans le cas des groupes 1 et 2 ; ainsi le groupe 1 s'est trouvé défavorisé par le fait que sa séance s'est déroulée durant la dernière période d'une journée normale de cours alors que pour le groupe 2, elle s'est déroulée lors de la première période et donc à un moment où les élèves étaient plus reposés.

Les résultats obtenus dans le Tableau 5.5 ont été analysés en fonction du moment dans la séance d'apprentissage (Tableau 5.6), du sexe (Tableau 5.7) et du

rendement scolaire des élèves impliqués (Tableau 5.8) (Les interruptions d'analyse (diagnostic No 9) ne sont pas prises en compte dans le calcul de ce pourcentage).

Relativement au moment dans la séance d'apprentissage (à la fin ou au début), le système a rendu un diagnostic fiable dans 98,0% des cas impliquant des tâches (plus faciles) accomplies durant la première moitié de la séance d'apprentissage comparativement à 92,3% pour les tâches accomplies durant la seconde moitié.

		Début de la séance		Fin de la séance		Total	
Analyse	**Diagnostic**	**N**	**%**	**N**	**%**	**N**	**%**
1. Soustraire le plus petit chiffre du plus grand chiffre	Généralisation d'une conception correcte dans certains cas particuliers	48	29,8	30	27,5	78	28,9
2. Pas d'emprunt à zéro	Mauvaise connaissance du sens d'emprunt surtout en présence de zéro	40	24,8	24	22,0	64	23,7
3. Emprunt à zéro sans changement de la colonne suivante à gauche	-Mauvaise compréhension du groupement -Mauvaise connaissance du sens d'emprunt	18	11,2	12	11,0	30	11,1
4. Emprunt par dessus de zéro sans changement de zéro en neuf	-Non maîtrise de la numération positionnelle -L'emprunt est considéré comme convention formelle et non comme une transformation d'écriture réorganisant le nombre	12	7,4	7	6,4	19	7,0
5. Non décrément de zéro (emprunt à zéro sans décrément cet emprunt)	-La décomposition n'est pas faite dans l'opération -L'itération de l'opération regroupement par 10 est oubliée -Le sens de l'opération n'est pas respecté	16	10,0	10	9,1	26	9,6
6. Zéro comme réponse à une colonne qui demande l'emprunt	- Le sens de l'emprunt n'est pas respecté -Limite de la charge cognitive	10	6,2	7	6,4	17	6,3
7. Emprunt au chiffre du dessous de zéro à la place d'un emprunt à zéro	-La signification de la disposition algorithmique est oubliée -Aucune considération de la position du chiffre dans le nombre où il faut un emprunt	9	5,6	6	5,5	15	5,5
8. Erreurs instables	Je ne peux trouver aucune explication relative aux erreurs précédentes, mais faites attention au moment où vous entrerez votre réponse	3	1,9	8	7,3	11	4,0
9. Interruption de l'analyse	Aucun diagnostic	5	3,1	5	4,6	10	3,7
	Sous-total diagnostics (Nos. 1 à 7)	153	98,0	96	92,3	249	95.8
	Grand total des diagnostics	161	100,0	109	100,0	270	100,0

Tableau 5.6. Distribution des diagnostics rendus par le système TIDES selon le moment de l'apprentissage

Comme on pouvait s'y attendre, les diagnostics «*Généralisation d'une conception correcte*» ont été plus nombreux au début tandis que les diagnostics «*mauvaise connaissance du sens d'emprunt*» et «*je ne peux trouver aucune explication ...*» ont été rendus plus souvent durant la deuxième moitié de la séance. Il ne semble pas que l'expérience des élèves avec le système TIDES soit en cause ici. Les différences observées sont plutôt attribuables aux niveaux de difficulté des tâches.

Concernant les différences entre sexes, le tableau 5.7 indique que les filles ont eu un comportement légèrement plus conforme aux *attentes* du système : ce dernier a pu rendre un diagnostic fiable dans 96,2% des tâches impliquant les filles comparativement à 95,4% dans les tâches impliquant les garçons.

Analyse	Diagnostic	Sexe masculin		Sexe féminin		Total	
		N	%	N	%	N	%
1. Soustraire le plus petit chiffre du plus grand chiffre	Généralisation d'une conception correcte dans certains cas particuliers	41	30,1	37	27.6	78	28,9
2. Pas d'emprunt à zéro	Mauvaise connaissance du sens d'emprunt surtout en présence de zéro	30	22,0	34	25,4	64	23,7
3. Emprunt à zéro sans changement de la colonne suivante à gauche	-Mauvaise compréhension du groupement -Mauvaise connaissance du sens d'emprunt	16	11,8	14	10,4	30	11,1
4. Emprunt par dessus de zéro sans changement de zéro en neuf	-Non maîtrise de la numération positionnelle -L'emprunt est considéré comme convention formelle et non comme une transformation d'écriture réorganisant le nombre	10	7.3	9	6,7	19	7,0
5. Non décrément de zéro (emprunt à zéro sans décrément cet emprunt)	-La décomposition n'est pas faite dans l'opération -L'itération de l'opération regroupement par 10 est oubliée -Le sens de l'opération n'est pas respecté	11	8,0	15	11,2	26	9,6
6. Zéro comme réponse à une colonne qui demande l'emprunt	- Le sens de l'emprunt n'est pas respecté -Limite de la charge cognitive	8	5,9	9	6,7	17	6,3
7. Emprunt au chiffre du dessous de zéro à la place d'un emprunt à zéro	-La signification de la disposition algorithmique est oubliée -Aucune considération de la position du chiffre dans le nombre où il faut emprunt	7	5,1	8	6,0	15	5,5
8. Erreurs instables	Je ne peux trouver aucune explication relative aux erreurs précédentes, mais faites attention au moment où vous entrerez votre réponse	6	4,4	5	3,7	11	4,0
9. Interruption de l'analyse	Aucun diagnostic	7	5,1	3	2,2	10	3,7
	Sous-total diagnostics (Nos. 1 à 7)	123	95,4	126	96,2	249	95.8
	Grand total des diagnostics	136	100,0	134	100,0	270	100,0

Tableau 5.7. Distribution des diagnostics rendus par TIDES selon le sexe.

Relativement à la performance du système dans les diagnostics rendus selon les résultats scolaires (Tableau 5.8), le système a rendu une proportion très proche de diagnostics fiables (93,1% et 96,4% respectivement) dans les tâches impliquant des élèves forts et dans celles impliquant des élèves faibles. Il est à noter cependant que les élèves faibles ont un plus grand nombre d'erreurs associées à

l'utilisation d'actions complexes (deux emprunts) dans les tâches tandis que les élèves forts ont un plus grand nombre d'interruptions d'analyse.

		Élèves forts		Élèves faibles		Total	
Analyse	**Diagnostic**	**N**	**%**	**N**	**%**	**N**	**%**
1. Soustraire le plus petit chiffre du plus grand chiffre	Généralisation d'une conception correcte dans certains cas particuliers	10	14,7	39	26,5	49	22,8
2. Pas d'emprunt à zéro	Mauvaise connaissance du sens d'emprunt surtout en présence de zéro	15	22,0	28	19,0	43	20,0
3. Emprunt à zéro sans changement de la colonne suivante à gauche	-Mauvaise compréhension du groupement -Mauvaise connaissance du sens d'emprunt	7	10,3	12	8,1	19	8,8
4. Emprunt par dessus de zéro sans changement de zéro en neuf	-Non maîtrise de la numération positionnelle -L'emprunt est considéré comme convention formelle et non comme une transformation d'écriture réorganisant le nombre	6	8,8	17	11,5	23	10,7
5. Non décrément de zéro (emprunt à zéro sans décrément cet emprunt)	-La décomposition n'est pas faite dans l'opération -L'itération de l'opération regroupement par 10 est oubliée -Le sens de l'opération n'est pas respecté	8	11,7	19	13,0	27	12,5
6. Zéro comme réponse à une colonne qui demande l'emprunt	- Le sens de l'emprunt n'est pas respecté -Limite de la charge cognitive	5	7,3	14	9,5	19	8,8
7. Emprunt au chiffre du dessous de zéro à la place d'un emprunt à zéro	-La signification de la disposition algorithmique est oubliée -Aucune considération de la position du chiffre dans le nombre où il faut emprunt	3	4,4	6	4,1	9	4,2
8. Erreurs instables	Je ne peux trouver aucune explication relative aux erreurs précédentes, mais faites attention au moment où vous entrerez votre réponse	4	5,8	5	3,4	9	4,2
9. Interruption de l'analyse	Aucun diagnostic	10	14,7	7	4,7	17	7,9
	Sous-total diagnostics (Nos. 1 à 7)	54	93,1	135	96,4	189	95.4
	Grand total des diagnostics	68	100,0	147	100,0	215	100,0

Tableau 5.8. Distribution des diagnostics rendus par TIDES selon le rendement scolaire

6.3.2.2 Performance observée en fonction du nombre d'essais et terminée avec succès

Pour évaluer grossièrement la performance du système TIDES relativement à l'évaluation de la probabilité de départ et à la dynamique de modification de cette probabilité, des données ont été compilées pour déterminer le nombre de solutions

terminées avec succès en fonction du nombre d'essais complétés. Le Tableau 5.9 indique que 50,3% des résolutions ont été complétées avec succès du premier coup, 20,3% à la deuxième tentative, etc. Ces résultats montrent qu'à chaque nouvelle tentative, c'est la majorité des élèves qui n'ont pas encore terminé leur problème qui réussissent. Cependant, il convient de préciser que ces résultats portent seulement sur les diagnostics de la tâche «*Pas d'emprunt à zéro*» et ne tiennent aucunement compte des succès ou insuccès antérieurs de l'élève relativement à des tâches requérant les mêmes connaissances (voir performance en fonction de nombre d'erreurs).

	Nombre de tentatives						
Groupe	1	2	3	4	5	6 et +	Total
1	50	22	13	7	4	2	98
2	34	12	10	6	3	4	69
Total	84	34	23	13	7	6	167
Total %	50,3	20,3	13,8	7,9	4,2	3,6	100,0

Tableau 5.9. Distribution des diagnostics d'une tâche «Pas d'emprunt à zéro » selon le nombre d'essais

6.3.3 Apprentissages réalisés avec le système TIDES

6.3.3.1 Performances observées en fonction de la pratique

C'est à partir des performances manifestées par les élèves en termes de temps requis et de nombre d'erreurs commises lors de l'accomplissement d'une tâche que sont d'abord mesurés les apprentissages réalisés en utilisant le système TIDES.

La performance des élèves a d'abord été mesurée par le temps moyen requis pour résoudre correctement les tâches requérant uniquement la connaissance `6´ (Emprunt itératif : **RÉPÉTER** *deux* **FOIS** l'emprunt, si les deux chiffres du haut sont plus petits que ceux du bas) ou seulement les connaissances '6' et '7' (Emprunt à zéro : numération positionnelle – connaissance procédurale) et en fonction du nombre d'occasions de pratiquer la connaissance `6´. L'intérêt de la connaissance `6´ ici réside dans le fait qu'il s'agit d'une connaissance implantée par des règles de production particulières du système TIDES et qu'elle se

retrouve, soit seule ou soit en compagnie de la connaissance '7', dans les tâches exécutées le plus souvent par les élèves qui ont participé à la mise à l'essai du système.

Il aurait été évidemment préférable de se limiter aux seules tâches requérant uniquement la connaissance `6´ (et ainsi d'isoler parfaitement cette connaissance aux fins de l'analyse) mais l'échantillon était alors trop faible et trop concentré sur des tâches constituant une première ou une deuxième occasion de pratiquer la connaissance `6´. C'est pourquoi les tâches impliquant uniquement les connaissances '6' et '7' (ensemble) ont été ajoutées à l'échantillon. Il en résulte bien sûr un échantillon de tâches homogènes se référant à une ou deux connaissances (mais toujours la connaissance 'C') et comportant des niveaux de difficulté différents. Les résultats apparaissent au Tableau 5.10.

Par ailleurs, il convient de préciser que les titres du Tableau 5.10 sont bien précis. On rapporte le temps par « *Iéme occasion de pratiquer la connaissance `6´* » et non pas par «Ième problème correctement résolu », puisqu'on s'intéresse plus à savoir quelle connaissance est pratiquée ; par contre un apprenant peut résoudre un problème correctement sans pratiquer aucune connaissance, autrement dit, la bonne réponse ne signifie pas nécessairement que l'apprenant a bien saisi la connaissance, ou encore que le raisonnement qui l'a mené à cette réponse correcte est lui aussi correcte. .

Iéme occasion de pratiquer la connaissance « `6´ » .	N	Temps moyens requis (secondes)
1	23	184
2	20	123
3	10	160
4	8	113
5	11	102
6	9	126
Total	81	141

Tableau 5.10- Distribution des performances observées en fonction de la pratique (tous les élèves)

L'analyse des données du Tableau 5.10 nous suggère plusieurs remarques.

- Le nombre de tâches-élèves impliquant la connaissance `6′ atteint 81 mais, dans plus de la moitié des cas (43), ces tâches constituent la première ou la deuxième occasion (i=1 ou i=2) de pratiquer la connaissance en question, probablement parce qu'un grand nombre d'élèves n'ont pratiqué ces tâches qu'une ou deux fois. En conséquence, les temps moyens requis relatifs à un nombre d'occasions plus élevé sont trompeurs puisqu'ils appartiennent à des élèves qui ont manifesté une meilleure performance en complétant un plus grand nombre de tâches. Il y a donc lieu de reprendre le même tableau mais en ne retenant cette fois que les élèves qui ont pratiqué au moins un certain nombre de fois la connaissance '6'. Le nombre 5 a été retenu parce qu'il permet l'élimination des élèves qui n'ont pratiqué la connaissance ICI qu'une ou deux fois sans trop réduire l'échantillon pour les nombres d'occasions plus élevés.

- Si on combine les tâches impliquant seulement la connaissance `6′ et celles impliquant seulement les deux connaissances '6' et '7', l'objectif était de s'assurer d'un échantillon assez large pour tous les niveaux de pratiques dans un même tableau. Cependant, étant donné que les tâches impliquant seulement la connaissance `6′ (dont l'exécution requiert plus de temps) ont été surtout accomplies au début de l'expérience d'apprentissage, ce regroupement risque de biaiser les durées moyennes observées en fonction du nombre d'occasions de pratiquer la connaissance `6′. Il y a donc lieu de distinguer les deux ensembles de tâches (celles impliquant seulement la connaissance '6' et celles impliquant seulement les deux connaissances '6' et '7') pour vérifier cette hypothèse.

Le Tableau 5.11 reprend donc les observations du tableau 5.10 mais en distinguant cette fois les deux ensembles de tâches et en se limitant aux élèves ayant pratiqué au moins 5 fois la connaissance '6'.

Iéme occasion de pratiquer la connaissance « '6' »	(1) Tâches n'impliquant que la connaissance '6'		(2) Tâches n'impliquant que les connaissances '6' et '7'		(1)+(2)	
	N	Temps moyen	N	Temps moyen	N	Temps moyen
1	14	83	1	231	15	93
2	12	50	1	163	13	59
3	1	19	7	151	8	144
4	9	60	4	117	13	78
5	3	46	6	129	9	101
6	1	33	5	79	6	71
Total	40	62	24	128	64	87

Tableau 5.11- Distribution des performances observées pour deux ensembles de tâches en fonction de la pratique (élèves ayant atteint 5 occasions)

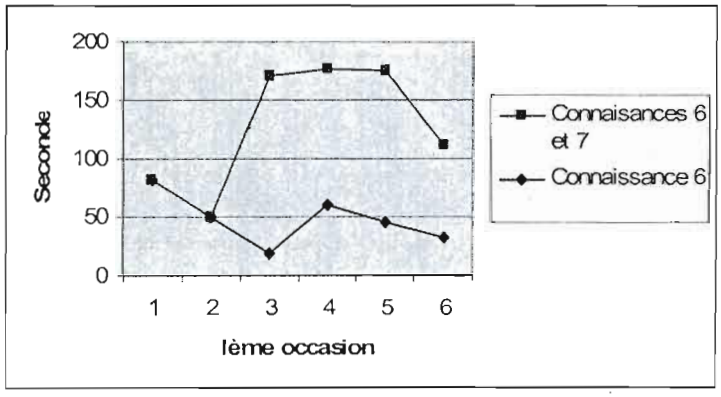


Figure 5.1 Graphe de la performance observée pour deux ensembles de tâches en fonction de la pratique (élèves ayant atteint 5 occasions)

Les données du tableau 5.11 confirment la nécessité de distinguer entre les tâches requérant seulement la connaissance '6' (Emprunt itératif : RÉPÉTER *deux* FOIS l'emprunt, si les deux chiffres du haut sont plus petits que ceux du bas) et celles requérant seulement les connaissances '6' et '7' (Emprunt à zéro : numération positionnelle – connaissance procédurale). D'abord, les élèves représentés dans ce tableau ont mis en moyenne près de deux fois plus de temps pour accomplir les tâches impliquant les connaissances '6' et '7'comparativement à celles impliquant seulement la connaissance ' 6' (128 secondes au lieu de 62 secondes). De plus, ces élèves ont en général pratiqué initialement la connaissance '6' en accomplissant des tâches n'impliquant que cette connaissance. C'est surtout à

partir de la troisième occasion de pratiquer la connaissance '6' qu'ils ont accompli des tâches impliquant les connaissances '6' et '7'.

Les données du tableau 5.11 montrent assez clairement les progrès dans l'apprentissage manifestés par la diminution du temps moyen requis en fonction de la pratique de la connaissance '6'. Cette démonstration est encore plus éloquente dans le graphe de la figure 1 où ces données sont représentées sous forme de courbes.

Ces données suggèrent également que l'on obtient des courbes assez régulières en circonscrivant la connaissance utilisée comme unité de mesure de la performance. Dans le tableau 5.10 en effet, l'augmentation de la performance était plus irrégulière quand l'unité de mesure englobait sans discernement des tâches comportant ou non une séquence. Ces observations sont en accord avec les prétentions du modèle d'Anderson à l'effet que la règle de production constitue l'unité d'apprentissage pour le LISP- TUTOR.

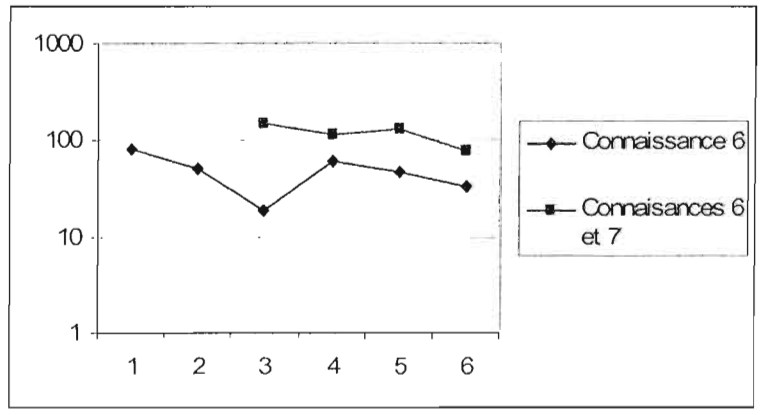


Figure 5.2 Graphe (échelles logarithmiques) de la performance observée pour deux ensembles de tâches en fonction de la pratique (élèves ayant atteint 5 occasions)

La figure 5.2 (d'ailleurs très clair par rapport à la figure 1) reprend les mêmes données que la figure 1 mais en les exprimant sur un graphe à échelles logarithmiques. Il en résulte deux courbes qui représentent une relation presque linéaire entre le logarithme de la performance et le logarithme de la pratique. À partir de ces résultats, il est permis de conclure que, sur ce plan, les apprentissages

réalisés avec le système TIDES sont conformes à ceux que l'on retrouve dans les systèmes incorporant le modèle d'Anderson. Dans les expérimentations conduites par Anderson pour démontrer que les apprentissages réalisés avec le LISP-TUTOR se déroulent selon une fonction de puissance entre la performance et la pratique, cette relation est linéaire, du moins à partir de la deuxième occasion de pratiquer une tâche.

6.3.3.2 Performance observée en fonction du nombre d'erreurs pour toutes les compétences

Dans cette section, les données présentées dans le tableau suivant (Tableau 5.12) ont pour but de montrer la performance observée des élèves pour toutes les compétences décrites précédemment, et cela en fonction du nombre d'erreurs commises par l'ensemble de ces élèves. Nous avons défini suffisamment de tâches pour circonscrire l'ensemble des connaissances retenues (voir tableau 5.2 et chapitre 3 figure 3.6). « Problème de soustraction avec un seul emprunt » constitue un exemple de ce que nous appelons une tâche. Une fois la tâche précisée, il convient d'identifier le nombre d'erreurs liées à cette tâche.

Tâche	Connaissances requises pour l'ensemble de la tâche	N	Nombre d'erreurs
Problème sans emprunt	2-3	44	4
Problème avec des zéro en bas : N-0	2-3-4	16	5
Problème avec les mêmes chiffres : N-N	1-2-3	12	6
Problème avec un vide en bas (ex : 456 - 43)	2-4	30	8
Problème avec un seul emprunt	2-3-3-3-5	64	15
Problème avec deux emprunts	2-3-5-5-6	30	26
Problème avec un zéro dans la première colonne : 0-N	2-3-3-3-5-6-7	28	20
Problème avec emprunt sur zéro (deuxième colonne)	2-3-3-3-5-6-7	18	16
Problème avec deux zéro dans les deux premières colonnes	2-3-3-4-5-5-6-7	23	19
Problème avec deux zéro dans une même colonne	2-3-3-4-5-6-7	13	12
Problème avec emprunt sur 1 (après l'emprunt 1 devient 0)	2-3-4-5-6	11	9
Problème avec emprunt sur 0 qui surmonte un vide	2-3-3-3-5-6-7	15	7
Problème avec emprunt sur 1 qui surmonte un 1 : 1-1	2-3-4-5-6	9	5
Problème avec emprunt sur 1 qui surmonte un vide	2-3-4-5-6	10	8
Problème avec emprunt, la première colonne a la forme : N-N	2-3-4-5-6	14	6
Total		337	12

Tableau 5.12. Nombre d'erreurs pour toutes les compétences

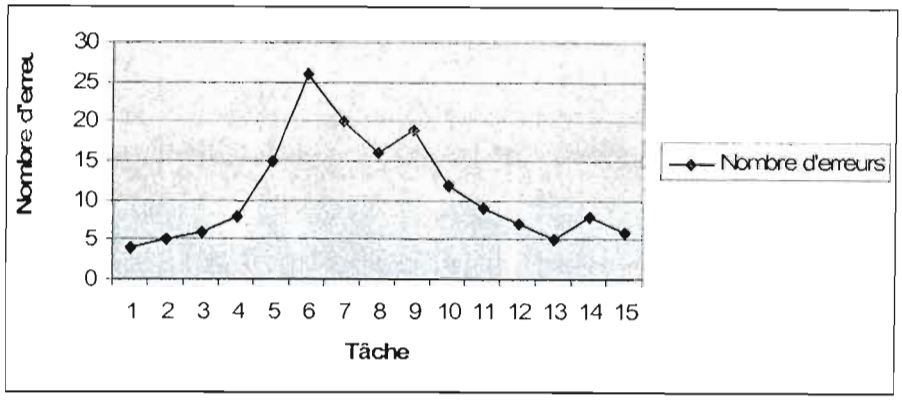


Figure 5.3. Graphe de la performance observée pour un ensemble de tâches en fonction de nombre d'erreurs

Le tableau 5.12 présente la distribution de fréquences des erreurs aux 15 tâches impliquant les sept connaissances présentées dans le Tableau 5.3.

On remarque que bon nombre d'erreurs (26) sont commises à cause de l'utilisation de deux emprunts dans un problème de soustraction ; d'ailleurs ce résultat est confirmé par plusieurs recherches faites dans le domaine de la soustraction (voir chapitre 3, section 4). De plus, cette tâche (deux emprunts) implique quatre types de connaissances. On remarque également une fréquence non négligeable (15) des erreurs liées à la tâche « un seul emprunt ». Toutefois, les élèves qui commettent ce type d'erreurs se trouvent dans une situation préoccupante. S'ils peuvent faire face à des situations relevant des quatre connaissances, les compétences qu'ils maîtrisent ne leur permettent pas de dépasser ces situations.

À l'opposé, la tâche « sans emprunt » qui implique les connaissances '2' et '3' est à peu près maîtrisée ; généralement les élèves n'éprouvent pas de difficultés à résoudre les problèmes de ce type de tâche. Les compétences impliquées sont pleinement maîtrisées.

On peut remarquer aussi que le plus grand nombre d'erreurs (96) se trouve lié à l'utilisation des emprunts, notamment ceux qui comprennent zéro. Par ailleurs, la pratique de certaines connaissances peut diminuer le nombre d'erreurs et pousse les élèves à faire des progrès dans leur apprentissage.

6.3.3.3 Cheminements d'apprentissage

À partir des mêmes événements enregistrés dans le fichier LOG, il est possible d'analyser la démarche d'apprentissage d'un élève. Le cheminement d'apprentissage de deux élèves accomplissant une tâche spécifique a ainsi été retracé. Il s'agit de cheminements pour deux tâches qui diffèrent notamment sur le plan de la complexité.

Les durées indiquées correspondent, dans le cas d'une résolution, au laps de temps écoulé entre l'entrée dans le bloc de résolution et la sortie de ce même bloc. Dans tous les autres cas (aides, analyses ou diagnostics, etc.), il s'agit du laps de temps écoulé entre le début de cette action et le début de l'action suivante. Cette durée inclut donc le temps que l'élève consacre à la réflexion avant de déclencher l'interaction suivante.

- **Pour une tâche simple**

La première tâche consiste à faire une soustraction avec un seul emprunt. L'élève A8[19] a mis 230 secondes pour la résoudre.

Dans ce premier exemple de démarche de résolution (Tableau 5.13), on peut observer qu'après avoir reçu une première indication d'erreur sous forme d'un message (action 2), l'élève a procédé à une demande de message d'analyse explicite concernant cette erreur. C'est là un comportement typique chez un grand nombre d'élèves qui préfèrent savoir l'analyse de leurs erreurs à partir du premier signe de rétroaction (message d'erreur).

[19] A8, B5,..., sont des codes d'identification des élèves qui ont participé à la mise à l'essai

No	Durée	Action	Résultat/Analyse/Diagnostic
1	90 sec.	Formulation de la solution	Dans une soustraction, soustraire toujours le petit nombre du plus grand nombre
2	15 sec.	Rétroaction	Message d'erreur : votre résultat est faux
3	32 sec.	Demande message-analyse	Vous n'empruntez pas, mais vous soustrayez le chiffre du haut du chiffre du bas
4	53 sec.	Demande message-diagnostic	-Vous avez généralisé une conception correcte dans certains cas particuliers, mais qui ne supporte pas le moindre changement de situation. Vous avez appliqué ici une règle qui dit «ne pas soustraire le grand nombre du petit nombre» -Vous n'avez pas fait la décomposition de la tâche en sous-tâches
5	20 sec.	Demande d'indication de résolution	Utilisez les règles d'emprunt
6	20 sec.	Demande de solution suggérée	Solution du système

Tableau 5.13. Démarche de résolution de la tâche « Faire une soustraction avec emprunt» par l'élève A8 dans le système TIDES

À la suite de son deuxième message d'analyse d'erreur (action 3), l'élève A8 a cependant demandé un message de diagnostic plus explicite (action 4). C'est ce qui lui a permis de savoir les causes réelles de ses erreurs. L'élève semblait avoir oublié, par l'éventualité d'un chiffre du haut plus petit que celui du bas, de se rappeler des règles d'emprunt (action 1). C'est pourquoi le système TIDES aurait dû prévoir comme seul message explicite d'analyse «*Vous n'empruntez pas, mais vous soustrayez le chiffre du haut du chiffre du bas*». Dans l'action 5, l'élève demande une sorte d'indication pour la bonne résolution de son problème.

L'élève a ensuite multiplié les vérifications, probablement par curiosité : après avoir obtenu un message de diagnostic plus explicite (action 4), il a demandé une sorte d'indication pour la résolution (action 5), ce qui constituait une autre forme de vérification ; enfin, il a demandé et obtenu la solution suggérée du système (action 6), solution qu'il a pu comparer à la sienne.

- **Pour une tâche complexe**

La deuxième démarche de résolution retenue comme exemple a été suivie par l'élève B5 ; la tâche consiste à «*RÉPÉTER deux FOIS l'emprunt, si les deux chiffres du haut sont plus petits que ceux du bas*». Cette tâche plus complexe fait l'objet d'une décomposition en deux sous-tâches et beaucoup plus de temps y est consacré. L'élève B5 a formulé une solution de premier niveau correcte dans le bloc de résolution (action 1) ainsi qu'une solution incorrecte pour la sous-tâche «*Continuer à emprunter pour l'autre colonne*» dans le même bloc de résolution (action aussi 1). Pour y arriver, il a mis cependant quelques six minutes, dont plus de cinq uniquement pour les sous-tâches. L'explication peut tenir au fait que si le problème comprend plus de zéro, les actions complexes correspondantes formulées par l'élève demandent plus de temps que si le problème ne comprend aucun zéro. Il en résulte probablement une perte de temps lors de la décomposition puisque, si l'on prend l'exemple de cette tâche, les élèves risquent de faire l'erreur (bien naturelle) d'interpréter que la première sous-tâche consiste à *Faire un emprunt pour la deuxième colonne* plutôt que le faire pour la première. Cette hypothèse est renforcée quand on observe les temps requis pour formuler d'autres tâches simples accomplies auparavant (et dont les actions ne sont pas rapportées dans le tableau 5.14). Malgré cette remarque, les observations recueillies dénotent une progression dans l'apprentissage, qui se manifeste par des temps de plus en plus courts pour la formulation de sous-tâches complexes.

No	Durée	Événement	Résultat/Analyse/Diagnostic
1	350 sec.	Formulation de la solution	SI le chiffre du haut est plus petit que celui du bas ALORS emprunter de la colonne suivante à gauche Continuer à emprunter pour l'autre colonne
2	16 sec.	Rétroaction	Message d'erreur : votre résultat est faux
3	30 sec.	Demande message-analyse	Quand vous devez emprunter deux fois, vous empruntez correctement à la colonne suivante mais vous ne changez pas cet emprunt pour la première colonne
4	54 sec.	Demande message-diagnostic	- Vous transformez correctement une centaine en dizaines, comme vous transformez correctement la dizaine en unités. Mais, vous n'arrivez pas à coordonner les deux pour réaliser une double transformation des centaines aux dizaines aux unités
5	18 sec.	Demande d'indication de résolution	L'emprunt se fait sur la colonne immédiatement à gauche Coordonner les deux transformations pour faire un emprunt correct
6	19 sec.	Demande de solution suggérée	Solution du système

Tableau 5.14. Démarche de résolution de la tâche « Emprunter deux fois» par l'élève B5 dans le système TIDES.

6.3.3.4 Performances observées en fonction de la décomposition en sous-tâches

Dans cette sous-section, les résultats analysés concernent les comportements des élèves en fonction de leur pratique de la décomposition des différentes tâches en sous-tâches. Les données recueillies permettent de comparer les performances relatives à ce facteur et de vérifier si des différences de comportement existent en fonction du genre.

Des quelques 15 tâches assignées par TIDES aux élèves qui ont participé à la mise à l'essai, la majorité étaient décomposables. C'est donc en fonction de ces seules tâches que les données ont été compilées.

La pratique de décomposition n'a pas d'équivalent chez Anderson. Ainsi dans le Lisp-Tutor, l'apprenant n'a pas besoin de décomposer ; la nature des tâches ne s'y prête pas et de toutes façons, le tuteur ne lui laisse pas le choix. C'est pourquoi les résultats rapportés dans cette section sur les pratiques de décomposition sont entièrement originaux et on n'y retrouvera aucune référence au modèle d'Anderson.

- **La décomposition en sous-tâches**

Sur la base des 126 tâches-élèves (tâche donnée accomplie par un élève donné) relevées (voir le tableau 5.15), il y a eu décomposition de la tâche principale dans 12 cas, soit dans une proportion de 9,5 %. Les données du tableau 5.11 indiquent que, 9 fois sur 10, les élèves n'ont pas utilisé la décomposition des tâches qu'ils avaient à accomplir.

Sans décomposition		Avec décomposition		Total	
N	%	N	%	N	%
114	90,5	12	9,5	126	100.0

Tableau 5.15 - Distribution des comportements relatifs à la décomposition.

Ces résultats seront commentés plus loin, mais il faut préciser que les 126 tâches-élèves dont il est question ici correspondent aux cheminements avec résultat correct ou incorrect. De plus, elles sont classées dans l'une ou l'autre des catégories de comportements relatifs à la décomposition en fonction de la dernière tentative (résultat final). Il est ainsi possible qu'un élève ait utilisé une stratégie de décomposition en cours de résolution mais ait finalement opté pour une solution n'impliquant pas de décomposition (comme par exemple, pour faire la soustraction 914 – 486, l'élève décompose la deuxième colonne qui comprend 1, après l'emprunt, ce 1 se voit changer en 0, l'élève éprouve de la difficulté pour ce 0 et arrête de continuer d'emprunter). Le cheminement inverse est également possible.

- **Les élèves qui pratiquent la décomposition**

Le tableau 5.16 reprend les données du tableau 5.15 relatives à la décomposition en distinguant, parmi les 126 tâches-élèves relevées celles accomplies par des élèves dont le rendement scolaire est élevé et celles accomplies par des élèves dont le rendement scolaire est plus faible.

Les données rassemblées dans le tableau 5.16 montrent que la pratique de la décomposition en sous-tâches est plus élevée chez les élèves qui ont un rendement scolaire élevé. C'est une observation qui n'a rien de surprenant compte tenu de ce

que, d'une part, les élèves ayant un faible rendement scolaire ont eu plus de difficultés de pratiquer cette décomposition. D'autre par, en théorie du moins, l'utilité de la décomposition croît avec la complexité relative d'une tâche et du fait que les tâches sont généralement perçues comme moins complexes par les élèves dont le rendement scolaire est élevé.

	Sans décomposition		Avec décomposition		Total	
Rendement scolaire	N	%	N	%	N	%
Élèves forts	48	84,2	9	15,8	57	100.0
Élèves faibles	66	95,7	3	4,3	69	100.0
Total	114	90,5	12	9,5	126	100.0

Tableau 5.16 - Distribution des pratiques de décomposition selon le rendement scolaire.

- **La décomposition et la performance du système**

Les données recueillies permettent de comparer la performance des élèves ayant décomposé et celle des élèves préférant tout résoudre dans le bloc de résolution sans décomposer. Le tableau 5.17 donne la distribution des durées moyennes (en secondes) requises pour les tâches décomposables traitées correctement selon les pratiques de décomposition, pour le même effectif global de 126 tâches-élèves.

Sans décomposition		Avec décomposition		Total	
N	Durée moyenne	N	Durée moyenne	N	Durée moyenne
114	163	12	351	126	181

Tableau 5.17. Performances et pratiques de décomposition.

Ce tableau montre nettement que les élèves ayant décomposé, ont mis quelque deux fois plus de temps pour résoudre un même ensemble de tâches. Concernant le nombre d'erreurs commises (Tableau 5.20), l'observation va dans le même sens et confirme la tendance observé pour le temps requis. Cette explication est renforcée par les données du tableau 5.13 indiquant que 68% des élèves ont rapporté avoir mis beaucoup de temps pour se débrouiller avec le bloc de résolution (zone de travail). Les résultats du tableau 5.18 prennent cependant en compte d'autres éléments que le bloc de résolutions comme les fenêtres, les menus, etc. (l'interface en générale).

Il m'a fallu beaucoup de temps pour me débrouiller avec le bloc de résolution, les fenêtres et tous les menus		
	N	%
Fortement en accord	7	28,0
En accord	10	40,0
En désaccord	5	20,0
Fortement en désaccord	3	12,0
Total	25	100.0

Tableau 5.18. Temps nécessaire pour maîtriser l'interface : perception des élèves

Une explication plus fondamentale à la différence observée tient au fait que, pour décomposer, l'élève fait nécessairement un plus grand nombre de manipulations. Cela est lié à l'interface, mais aussi et surtout intrinsèquement au fait de décomposer, même sur papier. L'interface n'en est finalement qu'un instrument. Il résulte de ce phénomène que l'avantage de recourir à la décomposition d'une tâche est fonction de la complexité de cette tâche.

- **Les élèves qui ont une meilleure performance**

Il se trouve que les différences observées selon le sexe sont peu significatives et elles ne sont donc pas rapportées ici. En revanche, celles relatives au rendement scolaire méritent qu'on s'y attarde. Les données du tableau 5.17 sont ventilées dans le tableau 5.19 selon le rendement scolaire des élèves.

	Sans décomposition		Avec décomposition		Total	
Rendement scolaire	N	Durée	N	Durée	N	Durée
Élèves forts	48	168	9	273	57	184
Élèves faibles	66	140	3	413	69	152
Total	114	163	12	351	126	181

Tableau 5.19. Performance (durée en seconde), pratiques de décomposition et rendement scolaire

On remarque que les élèves dont le rendement scolaire est plus élevé (élèves forts) ont mis plus de temps à résoudre les tâches décomposables que ceux dont le rendement scolaire est moins élevé (élèves faibles) quand la résolution s'est effectuée sans décomposition. Cette observation s'applique également quand la performance est exprimée par le nombre d'erreurs commises (Tableau 5.20) plutôt que par la durée de résolution des tâches décomposables. Ces résultats qui

s'appuient sur un nombre de tâches-élèves relativement élevé (48 et 66 respectivement), sont à première vue surprenants. L'explication proposée ici est qu'en moyenne, les tâches décomposables accomplies par les élèves forts étaient plus difficiles que celles accomplies par les élèves plus faibles.

Il en est ainsi parce que ces derniers ont présumément progressé plus lentement dans leurs apprentissages et cela s'est reflété sur le choix des tâches qui leur ont été assignées.

Le tableau 5.20 reprend essentiellement le même échantillon que le tableau 5.19, mais cette fois en tabulant le nombre d'erreurs commises plutôt que la durée requise. Ces résultats confirment dans l'ensemble les observations concernant les durées moyennes requises selon les pratiques de décomposition et selon le rendement scolaire.

	Sans décomposition		Avec décomposition		Total	
Rendement scolaire	N	Nombre d'erreurs	N	Nombre d'erreurs	N	Nombre d'erreurs
Élèves forts	48	0,7	9	1,7	57	0,8
Élèves faibles	66	0,9	3	2,4	69	1,0
Total	114	0,8	12	1.8	126	0,9

Tableau 5.20. Performance (nombre d'erreurs), pratiques de décomposition et rendement scolaire

Cependant, le temps requis pour accomplir une tâche et le nombre d'erreurs commises en l'accomplissant ne constituent peut-être pas les seules mesures de la performance. Certains élèves sont peut-être tout simplement incapables de résoudre des types particuliers de problèmes sans les décomposer parce que ces problèmes présentent un niveau de complexité trop élevé pour eux. Si tel est le cas, l'habileté à décomposer une tâche en sous-tâches devrait également être prise en compte dans l'évaluation de la performance.

6.3.4 Rétroaction à deux niveaux

L'une des caractéristiques essentielles dans la conception du système TIDES est qu'il incorpore une rétroaction à deux niveaux à la suite de la formulation par l'élève d'une solution dans le bloc de résolution. Comme il a été indiqué en

chapitre 4 – 5.7, le message analyse constitue la première forme de rétroaction, tandis que le message diagnostique explicite, affiché à la demande de l'élève, en constitue la seconde. Les données recueillies au cours de la mise à l'essai permettent de décrire les comportements des élèves relativement à leurs demandes de messages explicites à la suite d'une rétroaction de premier niveau.

6.3.4.1 Utilisation par les élèves du second niveau de rétroaction

Les données présentées dans les tableaux suivants (tableaux 6.16-6.20) révèlent certains comportements particuliers quant à l'utilisation des messages diagnostiques par les élèves. D'abord on observe des différences importantes dans la proportion des élèves qui ont demandé un message explicite (les huit types de message explicite sont décrits à la section 3) à la suite de la déclaration *solution terminée*. En effet, les élèves n'ont demandé un message explicite que dans des rares cas (cas simples), après une rétroaction du premier niveau. En comparaison, les élèves ont demandé un tel message plusieurs fois dans des cas complexes après un message analyse. Cette différence s'explique facilement puisque la rétroaction du premier niveau avertit l'élève qu'il a commis une ou plusieurs erreurs dont la nature peut lui être révélée s'il en fait la demande.

Il n'en demeure pas moins que dans près de la moitié des cas où le message d'erreur est affiché à la suite d'une déclaration *solution terminée*, les élèves n'ont pas jugé utile de demander un message explicite leur permettant de connaître la nature de l'erreur commise. Est-ce que ce comportement est lié à l'effort nécessaire pour demander et lire un message? Est-ce plutôt l'habitude des jeux éducatifs (dans lesquels toute l'interaction se fait à l'aide de sons et de messages iconiques) qui amène plusieurs élèves à travailler par essais et erreurs? Voilà des questions d'intérêt mais qui dépassent le cadre de cette recherche.

6.3.4.2 Demande d'un message explicite à la suite d'un message d'erreur

Dans les paragraphes qui suivent, l'intérêt est porté sur les seuls cas « *solution terminée*» où un message est donné, traduisant donc une erreur certaine. Le tableau 5.16 ventile ces cas, d'une part selon le sexe de l'élève, et d'autre part

selon qu'un message explicite a été demandé ou non à la suite de l'apparition du message d'erreurs. Ce tableau montre que la tendance à ne pas demander de message explicite à la suite d'une analyse d'erreur est un peu plus prononcée chez les élèves de sexe masculin, où elle est même majoritaire : il n'y a pas eu de demande de message explicite dans 52,4% des cas impliquant des garçons et dans 37,5% des cas impliquant des filles. Cela traduit un comportement plus rationnel des élèves de sexe féminin, mais sans qu'on puisse avancer une explication évidente à ce fait.

	Sexe masculin		Sexe féminin		Total	
	N	%	N	%	N	%
Message de diagnostic demandé	98	47,6	50	62,5	148	51,7
Message de diagnostic non demandé	108	52,4	30	37,5	138	48,3
Totaux	206	100,0	80	100,	286	100,0

Tableau 5.21. Utilisation du second niveau de rétroaction après un message d'analyse selon le sexe des élèves impliqués

De manière similaire et toujours pour les seuls cas «*solution terminée*» où un message d'erreur est signalé, le tableau 5.22 ventile les demandes et non-demandes de message explicite selon le rendement scolaire de l'élève (les pourcentages sont des pourcentages-colonne). Les valeurs de ce tableau montrent que dans la majorité (60,2%) des cas impliquant des élèves dont le rendement scolaire est élevé, il y a eu une demande de message explicite après un message d'erreur, alors que dans les cas impliquant des élèves dont le rendement scolaire était plus faible, cette proportion était minoritaire (47,3%).

Similairement, on constate donc un comportement (légèrement) plus rationnel de la part des répondants de rendement scolaire plus élevé, ...ce qui n'est pas très surprenant.

	Élèves forts		Élèves faibles		Total	
	N	%	N	%	N	%
Message de diagnostic demandé	53	60,2	80	47,3	133	51,7
Message de diagnostic non demandé	35	39,8	89	52,7	124	48,3
Totaux	88	100,0	169	100,	257	100,0

Tableau 5.22. Utilisation du second niveau de rétroaction après un message d'analyse selon le rendement scolaire des élèves impliqués

Le tableau 5.23 fournit une information concernant la perception des élèves relativement à l'utilité des messages d'analyse. Le fait que 22 élèves sur 25 (88%) ont jugé cette première forme de rétroaction utile est cohérent avec les comportements observés mais n'apporte pas d'éclairage nouveau sur les raisons de ces comportements.

Les messages d'analyse m'ont aidé dans ma démarche de résolution		
	N	%
Fortement en accord	12	48,0
En accord	10	40,0
En désaccord	2	20,0
Fortement en désaccord	1	12,0
Total	25	100.0

Tableau 5.23 - Aide apportée par les messages d'analyse : perception des élèves

6.3.4.3 Messages de diagnostic et performance

Pour ce qui est de la perception qu'en ont les élèves, la réponse à cette question de recherche est assez claire comme l'indique le tableau 5.24. Il indique en effet que 23 élèves sur 25 (92%) estiment que les messages écrits les ont aidés dans leur démarche de résolution.

Le diagnostic sous forme de messages écrits m'a aidé dans ma démarche de résolution		
	N	%
Fortement en accord	11	44,0
En accord	12	48,0
En désaccord	2	8,0
Fortement en désaccord	0	0,0
Total	25	100.0

Tableau 5.24. Aide apportée par les messages écrits : perception des élèves

Pour mesurer les différences entre la performance des élèves qui ont demandé des messages explicites et celle des élèves qui se sont contentés du message de premier niveau, une analyse des événements subséquents à chaque message d'erreur a été effectuée.

Le tableau 5.25 présente les résultats de cette analyse. On y distingue d'abord les cas où le message d'erreur a provoqué une demande de message explicite et ceux où il n'y a pas eu une telle demande. Dans chacun des cas, les données sur le

message diagnostique suivant concernant l'essai permettent d'établir si une demande de message explicite augmente les chances d'obtenir une décomposition correcte à l'essai suivant.

	Message d'analyse		Message de diagnostic		Total	
	N	%	N	%	N	%
Message explicite demandé à la suite d'un message d'erreur	78	60,0	52	40,0	130	100,0
Message explicite non demandé à la suite d'un message d'erreur	54	57,4	40	42,6	94	100,0
Totaux	132	59,0	92	41,0	224	100,0

Tableau 5.25 - Demande de message explicite en cas d'erreur

Selon le tableau 5.25 (les pourcentages indiqués sont des pourcentages-ligne), il y a eu une meilleure performance dans les cas où les élèves ont demandé un message explicite à la suite d'un message d'erreur. En effet, ces élèves ont eu un message par la suite pour le bloc de résolution dans une proportion de 60% au lieu de 57,4% pour ceux qui avaient choisi de ne pas demander de message explicite. On peut aussi voir l'effet de la demande de message explicite de la manière suivante : la visualisation d'un tel message augmente les chances d'avoir un message approprié au coup suivant (60%) et diminue celle d'avoir un message d'erreur (40%) alors que l'absence d'une telle visualisation inverse l'ordre de ces proportions, (57,4% et 42,6% respectivement), c'est-à-dire augmente les chances que l'élève persiste dans son erreur.

Parmi les observations faites concernant la question de performance, il semble que les élèves, dont le rendement scolaire est plus élevé, ont davantage tendance à demander un message explicite à la suite d'un message d'erreur, de plus ils se distinguent aussi en étant davantage capables d'utiliser efficacement ce message.

Il s'agit évidemment d'une observation importante qui nous indique que les messages diagnostiques constituent une aide véritable aux élèves dont le rendement scolaire est élevé. Mais cela suggère aussi que l'expression de ces messages devrait être améliorée afin de les rendre mieux adaptés aux besoins des élèves dont le rendement scolaire est moins élevé.

Les différences de performance observées plus haut en fonction de la capacité d'exploiter les messages diagnostiques selon le rendement scolaire des élèves se retrouvent également quand les données sont établies selon le sexe des élèves. Il a été mentionné plus haut (Tableau 5.21) que les garçons avaient moins tendance à demander des messages diagnostiques après un message d'erreur. L'observation montre en outre que, lorsqu'ils font une telle demande, les garçons profitent davantage des messages reçus ; ces messages leur permettent en effet de corriger leurs erreurs.

6.3.5 Accès aux autres informations pendant la résolution

6.3.5.1 Informations générales et exemples de résolution

Les élèves n'ont demandé d'accéder à l'une ou l'autre des informations générales (consignes, aides) qu'une fois chacun en moyenne. L'accès aux exemples a été encore moins demandé puisque chaque élève n'a jugé utile de consulter l'un des exemples de problèmes avec solution commentée qu'une demi-fois seulement en moyenne.

Ces résultats corroborent les autres comportements observés devant l'éventualité de devoir lire un texte à l'écran : un grand nombre d'élèves préfèrent les messages iconiques et une approche par essais et erreurs. Mais ces résultats concrets concordent assez mal avec les perceptions plus subjectives exprimées par les élèves. En effet, 76% jugent important de pouvoir accéder aux informations disponibles pendant la résolution (Tableau 5.26) et 80% jugent utile l'aide apportée par les exemples de problèmes avec solutions commentées (Tableau 5.27). Il faut mentionner cependant que, dans le cas des exemples de résolution, la question posée ne portait pas sur l'accès pendant la résolution mais plutôt sur l'utilité de l'information (les élèves ont tous pris connaissance des exemples de résolution pendant la phase instruction).

Le fait de pouvoir accéder aux informations sur les consignes ainsi que sur l'aide du système était important pour moi		
	N	%
Fortement en accord	9	36,0
En accord	10	40,0
En désaccord	4	16,0
Fortement en désaccord	2	8,0
Total	25	100.0

Tableau 5.26. Importance des informations disponibles : perception des élèves

Les exemples de problèmes avec solutions commentées m'ont aidé dans ma démarche de résolution		
	N	%
Fortement en accord	8	32,0
En accord	12	48,0
En désaccord	3	12,0
Fortement en désaccord	2	8,0
Total	25	100.0

Tableau 5.27. Aide apportée par les exemples de problèmes : perception des élèves

6.3.5.2 Proposition libre du problème

Comment les élèves utilisent-ils la liberté de recourir à la formulation libre de leur problème ? Les données recueillies semblent suggérer que les problèmes proposés sont largement suffisants. De fait, moins d'un élève sur deux en moyenne a eu recours à la formulation libre du problème. Ce résultat est conforme aux perceptions recueillies et tabulées dans le tableau 5.28 : plus de 64% des élèves ne trouvaient pas important de pouvoir accéder à la formulation libre.

Le fait de pouvoir utiliser la formulation libre du problème que celui proposé par le système était important pour moi		
	N	%
Fortement en accord	4	16,0
En accord	5	20,0
En désaccord	12	48,0
Fortement en désaccord	4	16,0
Total	25	100.0

Tableau 5.28. Importance de la formulation libre : perception des élèves

6.3.5.3 Exécution graphique de la solution

En revanche, les élèves ont largement utilisé l'exécution graphique après le processus de résolution. Ils pouvaient librement procéder à l'exécution de la

solution complète qu'il avait formulée. En moyenne, chaque élève s'est prévalu de cette possibilité à 7 reprises au cours de sa séance d'apprentissage. Il n'est donc pas surprenant de constater que les perceptions exprimées par les élèves sont particulièrement favorables (88,0%) en ce qui concerne l'utilité de l'exécution graphique (tableaux 5.29).

L'exécution graphique m'a permis de mieux comprendre mes erreurs et mes actions		
	N	%
Fortement en accord	16	64,0
En accord	6	24,0
En désaccord	2	8,0
Fortement en désaccord	1	4,0
Total	25	100.0

Tableau 5.29. Utilité de l'exécution graphique : perception des élèves

6.3.5.4 Solution suggérée par le système

La solution suggérée par le système constitue une autre source précieuse d'information accessible à l'élève en cours de résolution. Après la formulation de sa solution, un élève pouvait en effet demander et recevoir la solution du système correspondant à son résultat. Les données recueillies indiquent que cette option a été largement utilisée. En moyenne, chaque élève a demandé la solution du système à 8 reprises.

6.3.6 Conclusion sur la mise à l'essai du système TIDES

Plusieurs éléments d'informations ont été présentés et analysés concernant la performance du système TIDES et celle manifestée par les élèves qui l'ont utilisé au cours de la mise à l'essai. Différents comportements d'élèves durant les séances d'apprentissage ont aussi été relevés et analysés. Ces éléments ont été recoupés au besoin avec d'autres données sur le sexe et le rendement scolaire des élèves participants.

Concernant la performance générale du système, les données indiquent qu'un diagnostic fiable a pu être rendu 9 fois sur 10. Pour rendre un tel diagnostic, le système doit comprendre les actions formulées par l'élève, disposer de

connaissances permettant d'analyser cette formulation et générer un message fondé sur cette analyse.

Il y a maintenant lieu de reprendre aux sous-questions présentées au début du chapitre.

- *Le système TIDES est un système d'apprentissage et ces apprentissages sont conformes au modèle d'Anderson*

Sur le plan de la performance, les données recueillies démontrent clairement un certain progrès dans l'apprentissage manifesté d'une part, par le nombre de succès (50%, 3.2.2) des résolutions complétées, et d'autre part, la diminution de la durée moyenne requise pour utiliser correctement une connaissance particulière en fonction de la pratique de cette connaissance. Enfin, par la diminution du nombre d'erreurs en pratiquant certaines connaissances (3.3.2).

Ces observations sont conformes avec les prétentions du modèle d'Anderson appliqué au Lisp-Tutor à l'effet que la règle de production constitue l'unité d'apprentissage.

Toujours à propos de l'apprentissage, plusieurs démarches de résolution ont pu être reconstituées grâce aux actions enregistrées. Ces démarches permettent d'identifier certains comportements d'apprentissage, tel que celui retrouvé chez un grand nombre d'élèves et qui consiste à fonctionner à partir du message d'erreur donné par le système après la déclaration *solution terminée* plutôt qu'à partir des informations écrites (aides, exemples,...etc.).

- *Habiletés de calcul et d'algorithme de soustraction*

Deux autres observations importantes ont été faites à ce sujet. La première et la principale est que les élèves ont tendance à tout résoudre et donner une solution coûte que coûte dans le bloc de résolution et donc à manifester surtout des habiletés de calcul. Cette tendance est encore plus évidente quand on distingue les cas où l'emprunt (qui demande une décomposition en sous-tâche) a lieu : les données indiquent que, 9 fois sur 10, les élèves n'ont pas utilisé la décomposition des tâches décomposables qu'ils avaient à accomplir, décomposition qui leur

aurait permis de manifester des habiletés algorithmique (utilisation de l'algorithme de soustraction).

L'autre observation importante est que les élèves ayant manifesté des habiletés algorithmique en décomposant, spontanément ou non, ont mis en gros deux fois plus de temps pour résoudre un même ensemble de tâches que les élèves qui n'ont pas utilisé cette décomposition. L'explication possible la plus fondamentale à la différence observée dans les performances tient au fait que, pour décomposer, l'élève fait nécessairement un plus grand nombre de manipulations. Cela est lié à l'interface mais aussi et surtout intrinsèquement au fait même de décomposer, quelle que soit l'interface utilisée (même avec l'outil papier-crayon). Il semble que l'avantage de recourir à la décomposition d'une tâche est fonction de la complexité de cette tâche.

- *Le système TIDES incorpore une rétroaction à deux niveaux*

Plusieurs observations importantes sont rattachées à cette conception.

- Dans près de la moitié des cas où le message d'erreur est affiché, à la suite d'une déclaration *solution terminée*, les élèves n'ont pas demandé un message explicite afin de connaître la nature de l'erreur commise. Est-ce que ce comportement est lié à l'effort nécessaire pour demander et lire un message ? Est-ce plutôt l'habitude des jeux vidéo (dans lesquels toute l'interaction se fait à l'aide de sons et de messages iconiques) qui amène plusieurs élèves à travailler par essais et erreurs ?
- Il y a eu une meilleure performance dans les cas où les élèves ont demandé un message explicite à la suite d'un message d'erreur.
- Les élèves à rendement scolaire plus élevé ont non seulement davantage tendance à demander un message explicite à la suite d'un message d'erreur, mais ils sont davantage capables d'utiliser efficacement ce message.

- *Accès aux autres informations disponibles pendant la résolution*

En général, les messages écrits disponibles n'ont pas été très demandés (à l'exception de la solution suggérée par le système). Les élèves n'ont demandé à accéder à l'une ou l'autre des informations consignes, aides du système, qu'une fois chacun en moyenne. L'accès aux exemples a été encore moins en demande puisque seulement un élève sur deux (en moyenne) a consulté l'un des exemples de problèmes avec solution commentée. Ces résultats vont dans le même sens que les autres comportements observés devant l'éventualité de devoir lire un texte à l'écran : un grand nombre d'élèves préfèrent les messages iconiques et avancent par essais et erreurs avec une approche moins documentée.

Concernant la proposition libre du problème, les données recueillies semblent suggérer que les problèmes proposés par le système étaient largement suffisants. De fait, les élèves ont très peu utilisé leur liberté de recourir à la proposition libre.

En revanche, les élèves ont utilisé largement l'exécution graphique après le processus de résolution. En moyenne, chaque élève s'est prévalu de cette possibilité à 7 reprises au cours de sa séance d'apprentissage.

Les données recueillies indiquent enfin que l'option d'accès à la solution suggérée par le système a été largement exercée aussi.

CHAPITRE VI

EXTENSION DU SYSTÈME TIDES À L'AIDE D'UN RÉSEAU BAYÉSIEN

Comme il a été souligné à l'introduction, ce chapitre est dédié aux réseaux bayésiens, formalisme proposé comme extension possible du système TIDES. Il mettra en valeur la capacité de ce modèle à traiter de façon claire et efficace le problème de diagnostic des erreurs d'un apprenant dans un environnement incertain. Nous proposerons des améliorations possibles que nous présenterons comme explorées mais non encore complètement intégrées dans l'état actuel du système TIDES et aussi non évaluées. Il s'agit en fait de déterminer à quelles conditions ce modèle bayésien peut être intégré à un système d'apprentissage, en tant que système tutoriel intelligent et dont le domaine d'apprentissage est l'arithmétique. Nous montrerons aussi pourquoi nous nous intéressons particulièrement à cette approche.

6.1 Introduction

Les techniques de modélisations actuellement utilisées dans certains domaines (statistique, médecine, etc.) offrent un complément très intéressant à l'intelligence artificielle, notamment au chapitre de l'acquisition, la représentation et l'utilisation de connaissances. Ces techniques fournissent des moyens efficaces pour produire des modèles riches de l'apprenant. Bien que complexe, ces formalismes facilitent la résolution de certains problèmes difficiles que les modèles classiques n'ont pas pu les résoudre.

Il existe plusieurs méthodes pour mettre en œuvre cette modélisation. Le choix de l'une d'entre elles constitue une difficulté pour le concepteur et pour l'utilisateur. Aucune méthode n'est meilleure qu'une autre dans l'absolu. Néanmoins, les

contraintes, l'environnement, les objectifs et bien sûr les caractéristiques des méthodes doivent guider l'utilisateur dans son choix.

Parmi ces méthodes on peut citer entre autres les modèles graphiques (Lauritzen, 1996) qui constituent à l'heure actuelle un des outils principaux de modélisation et de raisonnement utilisés notamment en intelligence artificielle. Ces modèles sont aussi connus sous le nom de réseaux de croyance, réseaux d'indépendance probabilistes ou encore réseaux bayésiens (Pearl, 1988, Jensen, 1999, Cowell et al., 1999).

Selon Jordan (1998), ces réseaux sont l'association entre la théorie des graphes et celles des probabilités. Ils fournissent un formalisme naturel pour la conception et le développement de nouveaux systèmes. De plus, ces modèles conçus et développés pour certains domaines, peuvent être facilement transférés et exploités dans d'autres domaines. C'est la raison pour laquelle ce chapitre est consacré à un exposé des arguments qui justifient l'extension possible du système TIDES à l'aide des réseaux bayésiens dans la modélisation d'un apprenant. L'un des problèmes de cette modélisation se situe dans le fait que dans une situation d'apprentissage, on ne peut observer directement ce qui se passe dans la tête de l'apprenant, mais seulement l'estimer de manière très incomplète et imparfaite à travers ses actes ; de même, au seul examen de ses actions, on ne peut savoir avec certitude le but qu'il cherche à accomplir, ni le plan qu'il poursuit. C'est pour tenter de gérer ce type d'incertitudes que nous avons proposé d'utiliser les réseaux bayésiens comme extension du système TIDES (Gonzales & Wuillemin 1998 ; Conati et al., 1997, 2002 ; De Rosis & al., 2001 ; VenLehn, 2001 ; VenLehn & Niu, 2001).

Dans les sections suivantes, nous allons présenter dans un premier temps, et, sans être exhaustif, quelques définitions et notions de base de la théorie des graphes nécessaires pour la compréhension des réseaux bayésiens. Ensuite, nous définirons de façon non détaillée le formalisme bayésien. Nous allons parler de leur structure, de leur construction, de leur utilisation ainsi que de leurs

caractéristiques. Nous essaierons aussi de situer cette approche par rapport à certaines techniques, à savoir l'arbre de décision et les réseaux de neurones.

La seconde partie portera sur l'extension du système TIDES à l'aide des réseaux bayésiens. Tout d'abord, une modélisation du domaine arithmétique par ce formalisme est exposée. Ensuite nous décrirons comment nous pourrons employer les réseaux bayésiens comme extension possible du système TIDES afin de modéliser un apprenant et de diagnostiquer ces erreurs de façon plus efficace. Enfin, nous terminerons sur une conclusion et quelques perspectives.

6.2 Les réseaux bayésiens

Dans cette section, nous rappelons dans un premier temps, quelques définitions et notations de théorie des graphes, notamment le vocabulaire employé généralement pour les graphes orientés. Ensuite, nous présentons les principales définitions du formalisme bayésien. Ces définitions sont essentielles pour la compréhension du reste de ce chapitre (Becker & Naïm, 1999 ; Naïm et al., 2004).

6.2.1 Définitions et notations

6.2.1.1 Les graphes

- **Définitions**

- Un graphe $G = (V, E)$ est déterminé par la donnée :
  - d'un ensemble V dont les éléments sont appelés des sommets ou des **nœuds** ;
  - d'un ensemble $E \subset V \times V$ dont les éléments sont appelés des **couples** ordonnés d'éléments de V, ou des **arcs** de G ;
  - si $(X_i, X_j) \in E$ est un arc de G, noté $X_i \rightarrow X_j$, alors, il a pour origine X_i et pour extrémité X_j ,
  - s'il existe un arc partant de X_i vers X_j, alors X_i s'appelle le parent de X_j et X_j l'enfant de X_i.
- $G = (V, E)$ est un graphe **non orienté** $\Leftrightarrow (X_i, X_j) \in E \Rightarrow (X_j, X_i) \in E$.

- Un **chemin** est une suite d'arcs A_1, A_2, ...A_j, tel que $A_{k-1}= (X_{k-1}, X_k)$ soit un arc et pour tout k = i+1, …, j on a extrémité(A_{k-1}) = origine(A_k).
- Un **circuit** (ou cycle) est un chemin A_1, A_2, ...A_j où extrémité(A_j) = origine(A_1).
- Un graphe est **acyclique** s'il ne contient aucun cycle.

- **Graphes causaux**

Les graphes causaux sont un mode de représentation d'une structure d'influence entre divers *états, faits* ou *hypothèses* sur l'environnement (Pearl, 2000). Ainsi, la représentation graphique la plus simple de l'influence d'un événement, d'un fait ou d'une variable sur une autre, est de relier la cause à l'effet par une flèche orientée (Edwards, 2000 ; Frey, 1998 ; Becker et Naïm, 1999) (figure 6.1).

Figure 6.1 Exemple de graphe causal

Si on considère par exemple que X et Y sont des événements, la figure 6.1 peut être lue comme ceci « la connaissance qu'on a de X détermine la connaissance qu'on a de Y ». Cette détermination peut être stricte dans le sens que si on sait avec certitude que X est vrai, on peut en déduire Y avec certitude. Mais, il peut aussi s'agir d'une simple influence, c'est-à-dire que, si on connaît X avec certitude, l'opinion sur Y est modifiée, sans qu'on puisse toutefois affirmer si Y est vrai ou faux.

Cependant, il faut souligner que, malgré que la flèche soit orientée de X vers Y, elle peut fonctionner dans les deux sens, et ce même si la relation causale est stricte (Becker et Naïm, 1999).

6.2.1.2 Distribution de probabilités

La définition suivante est inspiré de Jordan (1998), Cowell & al., 1999 et Smyth et al. 1996).

On note P(X) la distribution de probabilités de X, c'est-à-dire l'ensemble des probabilités $P(X=x_i)$ où les x_i sont les valeurs possibles de X. Ces valeurs sont mutuellement exclusives. L'ensemble des valeurs d'une variable doit être exhaustif : $\sum_i P(X= x_i)=1$.

On dit que P(X) est une probabilité *a priori* si elle est donnée sans information sur les autres variables. On note P(X|Y) la distribution de *probabilité conditionnelle* de X sachant Y, c'est-à-dire l'ensemble des probabilités $P(X=x_i |Y=y_i)$. Il s'agit d'une probabilité *a posteriori* car elle prend en compte l'information $Y=y_i$. On a aussi $\sum_i P(X=x_i|Y=y_k)=1$.

Le calcul des probabilités a posteriori repose sur la règle de Bayes (voir § 2.2.1) qui met en relation les probabilités a posteriori avec les probabilités a priori et les probabilités conditionnelles.

6.2.1.3 Lien entre représentation graphique et représentation probabiliste

Soit $U=\{X_1, X_2 ,...,X_n\}$ un ensemble de variables aléatoires discrètes[20]. Chaque ensemble de variables aléatoires U peut être associé à un graphe G = (V, E). V représente l'ensemble des nœuds du graphe tel qu'il existe une bijection entre V et U. Chaque nœud de G est donc associé à une et une seule variable de U et on note $V=\{X_1, X_2, ..., X_n\}$. E désigne l'ensemble des arcs de G, i.e. $E=\{e(i,j)\}$ $1\leq i,j \leq n$ où i et j désignent les nœuds X_i et X_j }. Si les arcs sont orientés tels que e(i,j), cela signifie que l'arc est orienté du nœud i vers le nœud j.

La structure G d'un réseau probabiliste est la représentation graphique d'un ensemble de relations d'indépendance conditionnelle existant au sein d'un ensemble de variables aléatoires U (Lauritzen, 1992 ; Whittaker, 1990 ; Smyth, 1998). L'absence d'un arc e(i,j) dans G implique une certaine relation

[20] Une variable aléatoire est discrète si elle ne prend qu'un nombre fini ou dénombrable de valeurs.

d'indépendance entre les variables X_i et X_j. Ainsi le graphe G est une manière particulière de spécifier les relations d'indépendance présentes dans la distribution $P(U)=P(X_1, X_2,..., X_n)$. Inversement, une distribution particulière P(U) contient un ensemble de relations d'indépendance qui peuvent être ou non représentées de manière consistante avec un graphe.

6.2.2 Réseaux bayésiens

6.2.2.1 Règle de Bayes

Vers la fin des années 1980, grâce aux travaux de Pearl, mais aussi grâce à une équipe de chercheurs danois, les experts en intelligence artificielle découvrirent que les *réseaux bayésiens* offraient un moyen efficace de contourner l'obstacle qu'était le manque ou l'ambiguïté de certaines informations (Jensen, 1996).

L'approche bayésienne est généralement attribuée à Thomas Bayes (1702-1761). Le travail qu'il a écrit et qui est à la base de toute sa théorie a été publié en 1764 (Murphy, 2001). Dans cette approche, Bayes ne cherche pas à obtenir une probabilité précise sur la réalisation d'un événement mais sur la confiance que l'on a dans la réalisation de cet événement suite à la connaissance d'autres variables. L'objet central de cette approche est la fameuse règle de Bayes, qui dit que, si l'on a deux évènements X et Y, alors (Jensen, 2001 ; Jordan, 1998) :

$$P(X \mid Y) = \frac{P(X)P(Y \mid X)}{P(Y)}$$

Le réel P(X|Y) est la probabilité de réalisation de X sachant que Y est réalisé ; on parle de probabilité conditionnelle (probabilité *a posteriori*) ou de probabilité conditionnelle de X étant donné Y.

- **Événements indépendants**

Si on a deux événements X et Y, alors on dit que X est indépendant de Y si P(X|Y) = P(X), c'est-à-dire si le fait de savoir que Y est réalisé ne change en rien la probabilité de X.

6.2.2.2 Définition d'un réseau bayésien

Comme nous l'avons souligné précédemment, un réseau bayésien est un graphe dans lequel on peut représenter les connaissances sous forme de variables. Chaque variable désigne un nœud du graphe dont les valeurs peuvent être prises dans un ensemble discret ou continu. Le graphe est orienté et généralement acyclique. La dépendance directe entre les variables est représentée par des arcs. Ainsi, par exemple, un arc allant de X à Y exprimera le fait que Y dépend directement de X. Dans le cas où il n'y a pas de lien entre deux variables (absence d'arc) cela signifie la non-existence d'une dépendance directe. Cependant, les paramètres donnés à ces relations expriment les probabilités conditionnelles des variables sachant leurs parents ou les probabilités a priori si la variable n'a pas de parents.

6.2.3 Mécanisme d'inférence

On appelle *inférence* dans un réseau bayésien le calcul de la distribution de probabilités conditionnelles des variables constituant le modèle. Il s'agit en fait, de remettre à jour des probabilités a posteriori après une nouvelle observation des valeurs d'un certain nombre de variables. En effet, si on considère, par exemple, un ensemble V de variables et p(V) sa distribution de probabilité, alors l'arrivée d'une nouvelle observation « O » sur une ou plusieurs variables nécessite de remettre à jour les connaissances représentées dans le réseau bayésien par le biais de calcul respectant la règle de Bayes. Cette remise à jour, n'est autre que le calcul de $p(V|O)$, est appelée *inférence* dans un réseau bayésien (Bellot, 2002).

Malgré que plusieurs chercheurs (Suermondt and Cooper, 1991 ; Dawid, 1992 ; Jensen 1996) ont développé des algorithmes efficaces pour résoudre la complexité du problème d'inférence, le calcul de cette inférence reste toujours NP-difficile (Cooper, 1987, 1990 ; Dagum et Luby, 1993).

6.3 Caractéristiques des réseaux bayésiens

Les réseaux bayésiens sont un formalisme performant de gestion de l'incertitude. Dans ce domaine, ils se distinguent des autres techniques (notamment les réseaux

de neurones et les arbres de décision) par les caractéristiques suivantes (Gonzales et Wuillemin, 1998 ; Cowell, 2000 ; Meila et Heckerman, 2001 ; Bellot, 2002) :

leur présentation graphique permet de séparer aspects qualitatifs et aspects quantitatifs des modèles probabilistes ;

- ils permettent aussi de fusionner différentes sources d'informations (connaissances à partir de bases de données ou d'experts) ;
- ils ont la capacité d'introduire des nouvelles variables sur l'environnement d'étude sans nécessiter la révision de l'ensemble des réseaux ;
- ils permettent de grandes rapidités de réaction car, d'une part la propagation des informations est bidirectionnelle et, d'autre part les relations et les interactions qui lient les éléments modélisés sont explicites.

Cet outil, très structuré, offre des bonnes voies dans le domaine de la modélisation d'un apprenant, notamment dans l'évaluation de ses connaissances et dans le diagnostic de ses erreurs. C'est pourquoi notre choix s'est donc porté sur ce formalisme bayésien et nous le proposerons comme extension du système TIDES.

Dans les prochaines sections, nous allons montrer d'une part, les aspects de la modélisation bayésienne du domaine arithmétique et d'autre part, nous montrons aussi la possibilité de faire une extension du système TIDES à l'aide du modèle bayésien. Dans un exemple simple, nous montrons la nécessité de ce modèle pour l'explication diagnostique des erreurs d'un apprenant dans une situation incertaine. Nous finissons avec la présentation de quelques perspectives et une conclusion de ce chapitre. Mais tout d'abord nous définissons la modélisation de l'apprenant à l'aide des réseaux bayésiens.

6.4 Modélisation de l'apprenant à l'aide des réseaux bayésiens

La modélisation de l'élève nécessite une gestion attentive des incertitudes (Danine et al., 2006b), car elle est fondée sur un recueil de données généralement incertaines (comme par exemple la difficulté de trouver des données objectives) et incomplètes (comme la difficulté d'avoir l'ensemble des données au bon moment) (Gonzales et Wuillemin, 1998). Ces deux dernières notions sont liées aux

difficultés des observations (comme la difficulté d'observation directe de ce que l'élève connaît ou ne connaît pas) et à la fiabilité des observateurs humains (l'acquisition des connaissances, au moyen des experts humains, est généralement sujette à des erreurs ou des imprécisions). Ce sont des problèmes majeurs d'une telle modélisation qu'il faut réussir à trouver des solutions et à dépasser ces difficultés. Les approches probabilistes sont devenues indispensables dans cette situation, grâce à leur capacité à traiter des données incomplètes et incertaines. À ce titre, les réseaux bayésiens offrent une méthode de modélisation tout à fait adaptée (Millán et Luis, 2002). Cette méthode s'appuie nécessairement sur des connaissances incomplètes et incertaines pour stocker et exploiter de façon lisible et rigoureuse les connaissances génériques et spécifiques que l'on peut avoir sur l'association entre les actions et les connaissances des apprenants de façon probabiliste.

6.4.1 Modèles probabilistes et systèmes tutoriels intelligents

Les modèles probabilistes sont largement utilisés dans les domaines du diagnostic et de l'évaluation, car ils s'adaptent bien à la variabilité des observations. Ces modèles permettent, d'une part, de mieux expliciter les hypothèses utilisées sous forme d'un graphe et d'autre part, de représenter de façon réduite une distribution jointe de probabilités sur un ensemble de variables aléatoires. Par ailleurs, ils ont influencé la recherche et le développement des systèmes intelligents dans plusieurs domaines.

En effet, la réalisation d'un système tutoriel intelligent fondé sur un modèle probabiliste demeure un problème important de la recherche en EIAO. L'élaboration d'un tel système nécessite presque toujours la prise en compte des informations potentiellement incomplètes ou incertaines. Cette incertitude est d'origine multiple : difficultés d'observations, possibilité d'erreur dans les données, ambiguïté de la représentation de l'information, incertitude sur les relations entre les diverses informations. Une première approche à la modélisation d'un apprenant dans ce contexte d'incertitude est d'utiliser les modèles probabilistes. Ainsi, plusieurs études (Martin &VanLehn (1995) ; VanLehn &

Martin (1998) ; Conati & VanLehn (1996a, 1996b)) ont montré que les systèmes tutoriels élaborés sous ces modèles probabilistes sont capables de raisonner sur des faits et des règles incertains et pouvaient faire aussi bien, voire même mieux qu'un tuteur humain. Un exemple concret de modélisation de l'apprenant par ces modèles est le système tutoriel intelligent Andes (Conati et al., 1997, 2002, VanLehn et al., 2005) qui est conçu pour l'aide à l'apprentissage de résolution de problème en mécanique newtonienne.

De plus, ces systèmes, basés sur les modèles probabilistes (Pearl, 1988), présentent trois avantages par rapport aux systèmes traditionnels (Lepage et al., 1992) :

- précision dans l'acquisition des connaissances ;
- possibilité de représentation de connaissances et d'utilisation de l'incertitude ;
- clarté de représentation permettant de préciser graphiquement les dépendances parmi les propositions et les événements.

6.4.2 Construction et intégration d'un réseau bayésien

La construction d'un réseau bayésien peut être décomposée en deux parties : une partie qualitative et une partie quantitative. La partie qualitative est constituée par un graphe acyclique orienté dans lequel les nœuds représentent les variables et les arcs précisent les relations entre les variables. La partie quantitative concerne la table des probabilités associée à chaque nœud qui définit l'état du nœud étant donné l'état de ses parents. La relation entre les variables est exprimée par la présence ou l'absence d'arcs entre ces variables.

Avant d'exposer un exemple complet de cette construction (section suivante), nous fixons tout d'abord certains critères qui peuvent nous faciliter le choix et la structure raisonnable du modèle. De plus ces critères doivent être respectés dans l'intégration des réseaux bayésiens dans le système TIDES :

- le modèle doit avoir un nombre relativement faible de paramètres pour que la complexité des calculs reste abordable ;
- aucune variable discrète ne doit avoir de fils continus afin de pouvoir appliquer un algorithme d'inférence exacte ;
- des liens doivent exister entre toutes les variables (cachées ou non cachées) dans la structure du réseau afin de rendre le modèle plus complet.

Cependant, la méthode qui peut être utilisée pour la construction d'un réseau bayésien est basée sur plusieurs étapes (Corset, 2003).

1. La première étape consiste à déterminer les variables du modèle et à préciser leurs caractéristiques : le nom de la variable ; le label de la variable ; son caractère (discret ou continu) ; ses modalités dans le cas discret. Pour simplifier l'étape de l'évaluation, il faut que le nombre de modalités ne soit pas trop élevé.
2. La deuxième étape consiste à regrouper les variables par catégorie et à ordonner les groupes de variables de façon à ce qu'une variable peut être expliquée par d'autres variables. Ceci ne veut pas dire que toutes les variables sont directement reliées les unes aux autres, mais que la plupart des relations ont une interprétation en termes de causalité.
3. La troisième étape consiste à choisir d'une part, un modèle fournissant une description raisonnable des données, et d'autre part un modèle facilement interprétable.
4. La quatrième étape consiste à représenter ces variables aléatoires par des nœuds dans un graphe. Les arêtes du graphe donnent les probables dépendances ou les probables influences entre ces variables.

Le regroupement des variables par catégorie facilite la construction du graphe de manière hiérarchique et permet de simplifier et de clarifier les relations entre les variables. En effet, l'ordre des variables généralement trouvé en termes de cause à effet. Chaque variable peut être une variable réponse d'un groupe de variables

explicatives (Bellot, 2002 ; Corset, 2003). Ainsi, on peut distinguer trois groupes de variables.

1. Les variables racines du réseau sont les variables d'entrée, et sont toutes indépendantes entres elles.
2. Les variables internes ou variables intermédiaires sont les réponses des variables d'entrée ou d'autres variables intermédiaires.
3. Enfin, les variables de sortie du réseau sont les variables d'intérêt, et sont les réponses de toutes les autres variables.

Il est préférable de faire ce travail préliminaire afin que la sélection de modèle soit simplifiée. Puisque la représentation causale des groupes de variables peut permettre de réduire considérablement le nombre d'arêtes possibles. Une fois la structure du graphe déterminée, l'étape suivante consiste à évaluer les probabilités correspondantes.

Cependant, il faut noter que cette modélisation de l'apprenant, à l'aide d'un réseau bayésien, passe d'abord par la modélisation du domaine d'apprentissage en utilisant ce modèle. Ensuite, cette modélisation peut faciliter l'intégration des réseaux bayésiens à un outil qui permet de modéliser l'apprenant pour un thème (ou des thèmes) particulier(s). D'ailleurs, ce que nous allons faire dans la section suivante. Nous allons d'abord commencer par la modélisation du domaine d'arithmétique par un réseau bayésien (figure 6.3). De même nous allons aussi modéliser le même domaine d'une part, pour diagnostiquer les erreurs de soustraction d'un élève de façon générale, et d'autre part, pour l'évaluation de la connaissance de cet élève.

6.5 Modélisation du domaine arithmétique à l'aide des réseaux bayésiens

Les six facettes que nous avons décrites précédemment (cf. chapitre 4) sont intégrées dans un réseau bayésien (figure 6.3) modélisant un problème de diagnostic des erreurs de calcul arithmétique.

La construction de ce réseau prend appui sur les résultats de recherches consacrées au diagnostic et à la remédiation des erreurs de calcul arithmétique, ainsi qu'à l'enseignement du calcul arithmétique.

Comme indiqué précédemment, les erreurs commises par les élèves en arithmétique ont soulevé une attention particulière chez plusieurs chercheurs, notamment les chercheurs en didactiques des mathématiques et en sciences cognitives (Arsenault & Lemoyne, 2000). De plus, les situations d'apparition de nombreux de ces erreurs sont maintenant connues. Certains chercheurs ont tenu compte de ces situations afin de concevoir des outils d'apprentissage qui sont capables d'identifier ces erreurs, de les analyser et de les remédier. Toutefois, l'attention est davantage portée sur le processus, sur la manière dont l'élève exécute la tâche, plutôt que sur la tâche exécutée.

Si l'importance a été accordée depuis longtemps aux études des erreurs des élèves en arithmétique, c'est-à-dire à leur identification et à leur description, il faut aussi donner une importance équivalente à la recherche des sources et causes de ces erreurs ; cela exige de pouvoir poser un diagnostic approfondi en se basant sur les résultats rapportés par ces études et sur des outils fiables.

6.5.1 Construction de la structure du réseau bayésien

Nous allons dans cette sous-section modéliser le domaine arithmétique cognitive à l'aide d'un réseau bayésien.

Pour illustrer notre propos, nous présentons un modèle complet[21] de diagnostic des erreurs de calcul arithmétique. Ce domaine se prête bien à ce type de modélisation, car les réseaux bayésiens sont bien adaptés à des domaines localement structurés, de plus plusieurs données sont disponibles.

Pour commencer, il faut déterminer d'abord les variables du modèle. Pour ce faire, nous considérerons, d'ailleurs comme dans le domaine des systèmes experts (Cowell et al., 1999), que les paramètres cognitifs et le diagnostic peuvent être

[21] Ce modèle est inspiré des travaux de Cowell et al. (1999), sur le diagnostic de l'asphyxie des nouveaux-nés.

modélisés par des variables aléatoires, et ce qui nous reste c'est de spécifier une distribution de probabilités jointe sur ces différentes variables.

Trois étapes distinctes sont appliquées pour la construction d'un tel modèle (Cowell et al., 1999 ; Bellot, 2002) :

- *l'étape qualitative* : dans cette étape, seulement les relations d'influence, pouvant exister entre les variables du modèle prises deux à deux, qui sont considérées. Ceci rend possible la représentation graphique des relations entre les variables ;
- *l'étape probabiliste* : elle concerne la spécification d'une distribution de probabilités jointe définie sur les variables du modèle et que la forme de cette distribution correspond à la représentation graphique de l'étape précédente.
- *l'étape quantitative* : cette dernière étape consiste à la spécification numérique des distributions de probabilités conditionnelles.

- **Étape qualitative**

Comme il le souligne Pearl (1988), l'importance des réseaux bayésiens réside dans le fait qu'ils permettent aux experts de se concentrer sur la construction d'un modèle qualitatif avant la détermination des spécifications numériques.

6.5.1.1 Définitions des variables du modèle

L'utilisation du formalisme bayésien demande en premier lieu d'établir les paramètres du réseau. D'ailleurs c'est ce que nous allons faire dans un premier temps en définissant les variables utiles à la représentation de notre problème.

Après une recherche complète dans le domaine de l'arithmétique cognitive et en se basant sur les études faites dans ce domaine, nous avons trouvé que les variables les plus utilisées et les plus pertinentes à traiter sont les suivantes :

- Les variables d'entrée[22] (liées au domaine)

  n_1 : Compétences arithmétiques
  n_2 : Compétences logiques
  n_3 : Système de numération positionnelle
  n_4 : Connaissances de numérosité
  n_5 : Procédure de comptage
  n_6 : Procédés de l'algorithme
  n_7 : Connaissances procédurales

- Les variables intermédiaires (liées aux erreurs)

  n8 : Erreur conceptuelle
  n9 : Erreur logique de raisonnement
  n10 : Erreur de regroupement
  n11 : Erreur d'algorithme
  n12 : Erreur due à la charge cognitive
  n13 : Erreur procédurale

- Les variables de sortie (liées au diagnostic)

  n14 : Sens de l'opération n'est pas respecté
  n15 : Mauvaise compréhension de regroupement
  n16 : Non maîtrise de la numération positionnelle
  n17 : Mauvaise connaissance du sens de l'emprunt
  n18 : Mauvaise compréhension du sens de la disposition algorithmique
  n19 : Limite de la charge cognitive
  n20 : Généralisation d'une conception localement juste

Ce modèle comprend donc 7 nœuds racines (nœuds d'entrée), 6 nœuds intermédiaires et 7 nœuds terminaux (nœuds de sortie), soit au total 20 variables. Toutes les variables sont discrètes et binaires.

6.5.1.2 Structure du graphe

La structure du réseau est la représentation graphique d'un ensemble de relations d'indépendance conditionnelle existant au sein d'un ensemble de variables

[22] Nous avons supposé que les variables représentées par les nœuds (n_2,..., n_7) dans la figure 6.3 sont des vraies variables d'entrée puisqu'elles représentent le domaine lui même. De plus la variable (compétences arithmétiques) est liée directement à chacune de ses variables (domaine).

aléatoires. Pour déterminer cette structure, les variables ont été divisées en sous-groupes (figure 6.2).

- Les variables représentées par les nœuds (n_2, n_3) sont dans le groupe *compétences numériques*.
- Les variables représentées par les nœuds (n_4, n_5) font partie du groupe des *paramètres d'influence* (les résultats rapportés par plusieurs études (voir chapitre 4) soulèvent que ces variables sont souvent à l'origine des erreurs commises par les élèves).
- Les variables représentées par les nœuds (n_6, n_7) sont dans le groupe *connaissances déclaratives et procédurales*.
- Les variables représentées par les nœuds (n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}) font partie du groupe *erreurs*.
- Les variables représentées par les nœuds (n_{14}, n_{15}, n_{16}, n_{17}, n_{18}, n_{19}, n_{20}) sont dans le groupe *diagnostic*.

Ces groupes sont en relation, comme l'indique la figure 6.2.

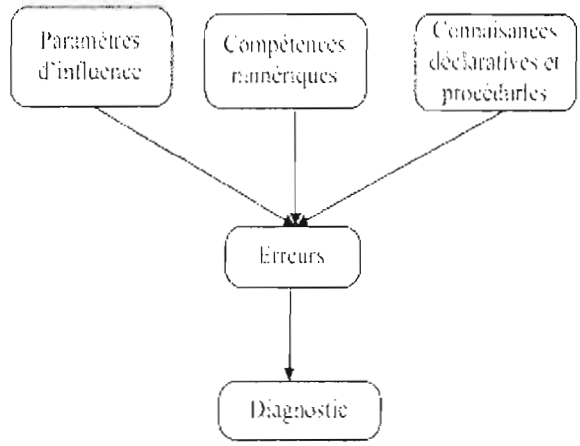


Figure 6.2 Regroupement des variables et représentation de la causalité

6.5.1.3 Le réseau bayésien

La figure suivante (figure 6.3) représente le réseau bayésien modélisant un problème de diagnostic des erreurs de calcul arithmétique. Le nœud *Compétences arithmétiques* (n_1) peut prendre deux valeurs correspondant aux situations-problèmes possibles dans ce cas précis de diagnostic. L'arc allant du nœud n_3 (en

passant par le nœud n_{10}) au nœud n_{16} exprime le fait que la non-maîtrise de la numération positionnelle par l'élève dépend directement de l'erreur logique de raisonnement de l'élève (n_8) et du regroupement (n_{10}). De même, la mauvaise connaissance du sens de l'emprunt (n_{17}) est directement influencée par l'erreur d'algorithme (n_{11}) et du regroupement aussi. Cependant, on peut remarquer l'indépendance entre le nœud *Système de numération positionnelle* (n_3) et le nœud *Représentation de numérosité* (n_4), puisque, en effet, la notion de numération positionnelle désigne que chaque chiffre occupe une place qui correspond à son ordre décimal. Dans chaque position du système décimal (composé des unités, dizaines, centaines, milliers,...), on écrit les chiffres de 0 à 9. Dès que l'on arrive à 10, la valeur de la position change, c'est-à-dire 10 unités = 1 dizaine, 10 dizaine = 1 centaines, 10 centaines = 1 millier, etc. Alors que la notion de numérosité désigne la quantité symbolisée par le numéral ou verbal (i.e. peut être représentée par les chiffres, par exemple 9, ou par les lettres, neuf). L'étude de cette notion porte sur la manière dont les quantités numériques sont représentées dans la tête de l'élève.

Certaines erreurs commises par les élèves ne peuvent en effet être réellement comprises qu'en lien avec la maîtrise de la numération positionnelle (Grégoire, 1996). Prenons l'exemple suivant d'une soustraction écrite dont le résultat est incorrect.

$$\begin{array}{r} 305 \\ -\ 52 \\ \hline 353 \end{array}$$

L'analyse de cette erreur en terme de procédure, nous montre qu'au lieu d'effectuer le calcul « 0-5 », l'élève a inversé les termes de l'opération et effectué « 5-0 ». Il n'aurait donc pas appliqué la règle correcte qui consiste à faire un emprunt de la colonne des centaines pour la colonne des dizaines. Normalement, l'élève doit être conscient que le résultat d'une soustraction de deux entiers positifs doit nécessairement être inclus dans le premier terme de l'opération. Dans l'exemple ci-dessus, le résultat doit logiquement être inclus dans 305. Par

conséquent, lorsque l'élève propose 353 comme réponse, il commet deux types d'erreurs : une erreur de procédure et une erreur de logique de raisonnement.

Le graphe de la figure 6.3 illustre donc la première étape consistant à modéliser qualitativement le problème de diagnostic et à déterminer les influences existant entre les variables.

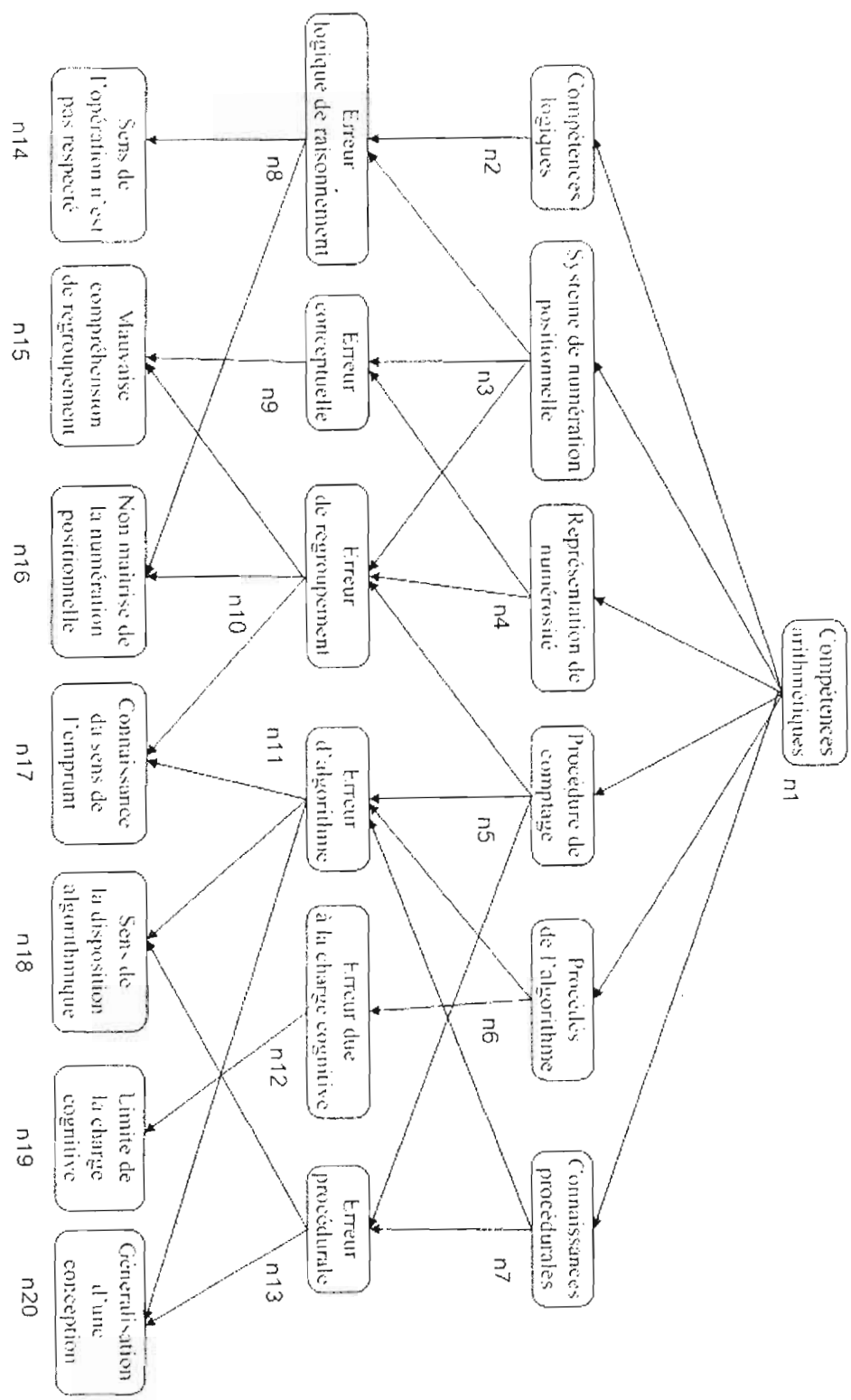


Figure 6.3. Réseau bayésien modélisant un problème de diagnostic d'arithmétique cognitive

6.5.2 Possibilité d'extension du système TIDES à l'aide d'un réseau bayésien

6.5.2.1 Le processus du diagnostic

Le processus de diagnostic dans un système tutoriel intelligent est une tâche complexe. Il ne s'agit pas de dresser un bilan des erreurs en cours d'apprentissage, mais le diagnostic vise à prendre en compte le contexte de ces erreurs et comprendre leurs origines en vue de produire une rétroaction appropriée.

Dans le domaine de l'arithmétique cognitive ce diagnostic a pour but, entre autres, l'évaluation des compétences d'un apprenant et la reconnaissance de ses stratégies de résolution.

Cependant, selon Siegler (1989), l'élève n'a pas une seule stratégie de résolution de problèmes arithmétiques, mais souvent plusieurs : certains problèmes semblent résolus par un processus de récupération directe en mémoire, d'autres par comptage ou par utilisation d'algorithme, d'autres encore par analogie aux problèmes très proches ou par décomposition en sous-problèmes, d'autres enfin en devinant. Il semblerait, en conséquence, qu'un diagnostic précis des erreurs d'un élève nécessite d'abord la prise en compte de ses stratégies de résolution de problèmes et ensuite l'identification des causes possibles des conséquences des actions de l'élève.

C'est ce que le système TIDES pourrait contribuer à faire à l'aide d'un réseau bayésien. À cet effet, on peut utiliser d'une part le moteur d'inférence de ce système pour détecter les erreurs de l'apprenant et déterminer ses stratégies de résolution, et d'autre part un réseau bayésien (notamment celui de la figure 6.3) comme extension du système TIDES pour faire un diagnostic en s'appuyant sur les résultats de ce moteur d'inférence.

Dans la suite, nous explicitons un peu cette extension du système TIDES à l'aide d'un exemple et nous proposons une démarche pour spécifier les tables de probabilités, souvent appelées paramètres du réseau bayésien.

6.5.2.2 Extension du système TIDES

Comme il a été indiqué dans le chapitre 4, le système TIDES est doté d'un moteur d'inférence (*Soustraction.plgCG*) qui joue un rôle important dans ce système. Il peut exécuter plusieurs tâches, notamment la détection des erreurs d'un apprenant ainsi que la recherche des règles et des mal-règles qui expliquent le comportement de cet apprenant. Cependant, et afin de connaître le mieux possible les causes réelles qui pourront mener l'élève à donner des réponses erronées, d'évaluer de façon précise ses compétences et de pousser un peu plus loin le diagnostic cognitif, il est possible de faire une extension du système en utilisant un réseau bayésien.

Pour mieux expliciter cette démarche, nous reprenons un exemple présenté dans le chapitre 4 (figure 4.32b). Dans cet exemple, l'apprenant doit résoudre le problème de soustraction $84-26$. Il doit commence d'abord à « soustraire 6 à 4 », et comme 6 est plus grand que 4, il doit « emprunter de la colonne suivante » afin de réaliser ce premier but. Ensuite il doit se focaliser sur la première colonne à gauche de la colonne active, décrémente cette colonne de 1 et ajoute 10 au chiffre du haut de la colonne active. Mais l'apprenant procède autrement comme l'indique sa solution :

$$\begin{array}{r} 8\,4 \\ -\,2\,6 \\ \hline 5\,2 \end{array}$$

L'élève considère les deux colonnes de la soustraction comme étant deux soustractions distinctes et commence à traiter chaque colonne indépendamment de l'autre. Il viole la contrainte sémantique de l'emprunt, qui veut que si une quantité doit être empruntée de la colonne à gauche, alors une quantité égale doit être ajoutée à la colonne active (Resnick, 1982).

Selon Young et O'Shea (1981), le bug, commis par l'élève, dérive du fait que l'élève ne s'est pas posé la question de savoir quelle était la position du chiffre le plus grand dans la colonne.

Si on se réfère à la théorie de réparation (repair theory) de Brown et VanLehn (1980), ce bug essaye de répondre à une situation face à laquelle l'élève ne peut pas suivre la procédure normale qui consiste à soustraire le chiffre du dessous du chiffre du dessus dans la colonne active. Ne sachant comment faire devant cette impasse, l'élève décide alors d'inverser les termes.

Lorsque l'apprenant termine sa résolution du problème, le moteur d'inférence analyse ses résultats colonne par colonne afin de détecter dans quelle colonne l'erreur s'est produite. Dans le cas où il trouve une erreur, il déclenche les règles ou les mal-règles fautives. Ensuite le même processus recommence jusqu'à ce qu'il active toutes les règles et les mal-règles pour conclure le comportement final de l'apprenant. Ce mécanisme d'inférence est en mesure d'utiliser la base de connaissances au fur et à mesure que les informations sur l'élève se présentent. Il apporte un aspect dynamique au système.

Dans l'exemple précédent, le moteur peut identifier l'erreur commise par l'apprenant comme *l'Erreur de Regroupement*. Cependant, pour connaître mieux le raisonnement qui a mené l'élève à commettre cette erreur et savoir exactement les causes réelles derrières ce comportement, on peut intégrer le réseau de diagnostic de la figure 6.3 dans le système TIDES.

Dans le cas de l'évaluation diagnostique (module diagnostic) le réseau bayésien de la figure 6.3 peut montrer les causes probables des erreurs commises par l'élève ainsi que les connaissances probablement manquantes.

Pour mettre ça en lumière, regardons de nouveau le résultat du problème 84 – 26 = 52. Le réseau bayésien de la figure 6.3 peut estimer alors les causes les plus probables de l'erreur de regroupement inférée par le moteur d'inférence:

- *mauvaise compréhension de regroupement* : il est très probable que l'apprenant éprouve des difficultés à dire combien il y a d'unités dans 84 ;
- *maîtrise limitée de la numération positionnelle* : l'élève ne connaît pas la signification de la disposition algorithmique, ni la pertinence de cette disposition. De ce fait, il ne respecte pas les positions des chiffres de la colonne traitée ;

- *mauvaise connaissance du sens de l'emprunt* : enfin, on peut aussi parler de la mauvaise compréhension du sens de l'emprunt qui pousse l'élève à chercher d'autres solutions.

Toutes ces causes sont probablement dues en amont à une mauvaise connaissance de la procédure de comptage, de la représentation de numérosité et du système de numérotation positionnelle.

Les probabilités a priori de maîtrise de ces connaissances sont ainsi mises à jour en fonction de l'erreur commise. Ceci entraine une mise à jour correspondante du modèle de l'élève.

Mais, il faut noter que quoique cette intégration des réseaux bayésiens dans le système TIDES, puisse être très intéressante, elle ne vient qu'en support pour maintenir le modèle de l'élève qui est basé sur certaines hypothèses cognitives et pédagogiques, les hypothèses que nous avons discutées dans les chapitres précédents.

6.5.2.3 Exemple de calcul de probabilités dans le réseau

Considérons le problème suivant : *soustraire le plus petit du plus grand*

```
7 8 2
3 9 6
-----
4 1 4
```

En première colonne (C1) : 6 - 2 = 4
En deuxième colonne (C2) : 9 - 8 = 1
En troisième colonne (C3) : 7 - 3 = 4

Au lieu d'emprunter, l'élève soustrait le plus petit chiffre du plus grand chiffre, colonne par colonne, sans tenir compte de leur position dans l'opération. Dans ce contexte, on peut retreindre le réseau précédent à celui, plus simple, de la figure 6.4.

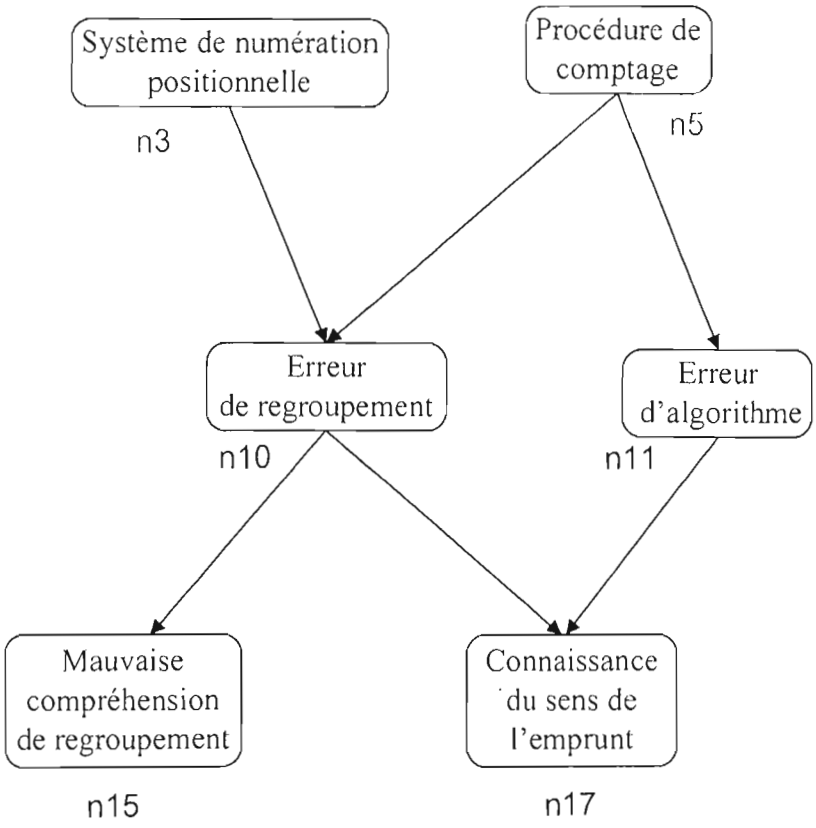


Figure 6.4. Réseau bayésien modélisant le problème : soustraire le plus petit du plus grand

Les états des variables aléatoires dans cette situation sont :

n_3 : Système de numération positionnelle, qui prend les valeurs *Oui/Non*

n_5 : Procédure de comptage, qui prend les valeurs *Oui/Non*

n_{10} : Erreur de regroupement, qui prend les valeurs *Présent/Absent*

n_{11} : Erreur d'algorithme, qui prend les valeurs *Présent/Absent*

n_{15} : Mauvaise compréhension de regroupement, qui prend les valeurs *Oui/Non*

n_{17} : Mauvaise connaissance du sens de l'emprunt, qui prend les valeurs *Oui/Non*

Pour compléter l'information sur cette partie du réseau on donne les lois de probabilité de chacune de ces variables aléatoires ; ces lois dans ce cas de variables discrètes sont des tables de probabilités ; pour les deux variables sans

parent n_3 et n_5 ce sont des probabilités *a priori* tandis que pour les autres variables ce sont des tables de probabilités étant donné les parents (probabilités conditionnelles).

On assigne d'abord à chaque variable une probabilité initiale que l'élève a appliqué correctement le concept ; cette probabilité initiale est fondée sur l'expérience et rajustée au besoin (Cox, 1975, Brown & VanLehn, 1980, VanLehn 1990).

- Pour les deux variables sans parent (n_3) et (n_5) ce sont des probabilités *a priori (pour alléger l'exposé, la probabilité complémentaire n'est pas donnée):*

  n_3 : Système de numération positionnelle : $P(n_3 = Oui) = 0.60$
  $P(n_3 = Non) = 0.40$

  n_5 : Procédure de comptage : $P(n_5 = Oui) = 0.70$
  $P(n_5 = Non) = 0.30$

- Pour les autres variables ce sont des tables de probabilités étant donné les parents (probabilités conditionnelles) :

  n_{11} : Erreur d'algorithme : $P(n_{11}\ Présent \mid n_5 = Oui) = 0.60$

  $P(n_{11}\ Présent \mid n_5 = Non) = 0.30$

  n_{10} : Erreur de regroupement :

Système de numération positionnelle (n_3)	Procédure de comptage (n_5)	Erreur de regroupement	Probabilité
Oui	*Oui*	*Présent*	*0.95*
Oui	*Non*	*Présent*	*0.75*
Non	*Oui*	*Présent*	*0.25*
Non	*Non*	*Présent*	*0.05*

Table 6.1. Loi de Erreur de regroupement conditionnellement à «Système de numération et Procédure de comptage»

n_{15} : Mauvaise compréhension de regroupement

Erreur de regroupement	Compréhension de regroupement	Probabilité
Présent	*Oui*	*0.90*
Absent	*Oui*	*0.05*

Table 6.2. Loi de Mauvaise compréhension de regroupement conditionnellement à «Erreur de regroupement»

n_{17} : Mauvaise connaissance du sens de l'emprunt

Erreur de regroupement	Erreur d'algorithme	Sens de l'emprunt	Probabilité
Présent	*Présent*	*Oui*	*0.90*
Présent	*Absent*	*Oui*	*0.60*
Absent	*Présent*	*Oui*	*0.80*
Absent	*Absent*	*Oui*	*0.05*

Table 6.3. Loi de Sens de l'emprunt conditionnellement à «Erreur de regroupement» et Erreur d'algorithme.

6.6 Conclusion et perspectives quant à l'utilisation des réseaux bayésiens

Nous avons présenté dans ce chapitre les démarches possibles de l'intégration d'un réseau bayésien dans un système tutoriel intelligent dédié au diagnostic des erreurs de la soustraction.

Malgré les difficultés de la modélisation de l'élève, notamment la difficulté d'observation directe de ce qu'il connaît ou ne connaît pas, ou la difficulté de savoir avec certitude le but qu'il cherche à accomplir. Nous avons montré que les réseaux bayésiens se prêtent particulièrement bien à cette modélisation dans un environnement d'apprentissage.

Pour les fins de cette modélisation, les réseaux bayésiens peuvent être intégrés dans le système TIDES pour préciser les causes probables derrières les actions de l'élève et donner un diagnostic efficace à ses actions.

Enfin, et toujours aux fins de la modélisation de l'élève, une méthode d'estimation des probabilités qu'une compétence soit maîtrisée et d'évolution de ces probabilités en fonction de la performance de l'élève pourra être développée.

Par ailleurs, certains problèmes peuvent se poser lors de la mise à jour et de la construction du réseau, notamment la taille et la topologie de ce réseau qui peuvent rendre difficile cette opération en temps réel. Nous verrons, lors de l'expérimentation, si ces problèmes se posent en réalité et nous examinerons alors les solutions possibles dont celle de réseaux générés dynamiquement en fonction des problèmes posés.

CONCLUSION GÉNÉRALE

Dans la présente recherche, le but poursuivi était de mieux comprendre le processus cognitif d'un élève en situation d'apprentissage, et implicitement de mieux cerner le rôle d'un tuteur au cours de ce processus. Ce but a été atteint par la conception et l'élaboration du système TIDES, prototype de système d'EIAO appliquant le modèle d'Anderson, et par l'observation du comportement du système et des élèves au cours de sa mise à l'essai.

Les premières questions de la recherche portent sur les *conditions d'intégration du modèle d'Anderson à un système d'apprentissage en tant que système tutoriel intelligent et dont le domaine d'apprentissage est l'arithmétique*. Selon les écrits recensés, les systèmes d'EIAO se distinguent généralement des systèmes experts par les connaissances particulières dont ils sont dotés pour remplir leurs fonctions pédagogiques. Le développement du système TIDES a permis d'identifier spécifiquement des connaissances à transformer ou à ajouter à celles d'un système d'EIAO dédié à l'apprentissage d'arithmétique et qui se prête à une application de la théorie ACT-R d'Anderson. Dans un tel système d'EIAO :

- les connaissances du domaine doivent être découpées en unités correspondant à celles manifestées durant le processus de résolution par l'élève et représentant la solution du système, la solution de l'élève, ainsi que différentes erreurs de compréhension ;
- à cause de la liberté de manœuvre exigée par le domaine d'apprentissage, l'élève doit pouvoir utiliser librement sa stratégie de solution, sans crainte

d'être interrompu par le système ;

- on doit retrouver des connaissances permettant d'analyser avec puissance (la solution comporte potentiellement plusieurs erreurs de type stratégique ou superficiel) et avec précision (l'analyse peut porter sur un premier niveau de décomposition, sur une solution finale ou sur toute résolution partielle) les solutions des élèves ;
- les connaissances associées aux interventions tutorielles doivent permettre au tuteur de tenir compte dans ses interventions (analyse, diagnostic) du niveau courant de résolution de l'élève et de sa manière de s'exprimer ;
- on doit retrouver des connaissances permettant de tracer le modèle de l'élève en fonction des résultats de l'analyse du comportement de l'élève et d'assigner des tâches en fonction de ce modèle ;
- on doit incorporer des connaissances permettant de gérer une interface entre l'élève et le système dans laquelle l'élève, d'une part, peut expliciter sa démarche de résolution de son problème en se concentrant sur la stratégie de cette résolution plutôt que sur les problèmes de vocabulaire et de formulation et d'autre part, dispose d'une information adéquate sans que cela impose un fardeau trop lourd à sa mémoire de travail.

Grâce à l'expérience de TIDES, il est davantage possible d'apprécier les forces et les faiblesses, de la théorie cognitive ACT-R d'Anderson en tant que méthode d'élaboration d'un système d'EIAO.

- La théorie ACT-R et son application à des systèmes intelligents d'apprentissage ont démontré les influences mutuelles que pouvaient avoir les recherches en psychologie cognitive et celles sur les tutoriels intelligents. À ce sujet, il faut souligner que l'idée même d'appuyer le design d'un système d'EIAO sur une théorie unifiée de l'acquisition de connaissances est importante et constitue la base d'une cohérence pédagogique dans le système à développer. Aucun doute possible, le système TIDES a largement profité du modèle d'Anderson comme guide

général d'élaboration.

L'organisation des connaissances en une structure hiérarchique de tâches est un élément de la théorie ACT-R qui s'est avéré utile au cours du développement de TIDES, notamment pour procéder à la décomposition des différentes tâches en sous-tâches (et en sous-sous-tâches, s'il y a lieu) et aussi pour mieux adapter les messages d'intervention du tuteur au niveau de concentration ou de décomposition courant.

La distinction entre la mémoire à long terme et la mémoire de travail, et en particulier la reconnaissance du caractère limité de cette dernière constituent un autre élément très positif de la théorie ACT-R. C'est pour tenir compte de la taille limitée de la mémoire de travail que furent explorées, dans le design de TIDES, différentes façons de minimiser le fardeau sur cette mémoire.

Quant au traçage de modèle, il constitue à la fois une force et une faiblesse de la théorie ACT-R. Dans les systèmes où il a été appliqué par Anderson, notamment dans le Lisp-Tutor et le Geometry-Tutor, il constitue véritablement une méthodologie efficace sur laquelle repose l'analyse des comportements des élèves et les rétroactions intelligentes du tuteur. Mais le traçage de modèle (du moins dans sa version originale) comporte une limite importante : il oblige l'élève à suivre de très près le modèle du système. Il intervient dès que l'élève entre une information s'éloignant de la solution correcte connue du tuteur. Cela est possible avec un système relativement directif. Avec l'arithmétique comme domaine d'apprentissage et la nécessité pour l'élève de pouvoir utiliser librement sa démarche de solution, le traçage de modèle est beaucoup plus difficile à appliquer dans sa version originale. À ce sujet, l'élaboration de TIDES a permis d'adapter le traçage de modèle au domaine d'arithmétique, avec la possibilité que l'élève garde le contrôle absolu sur le déroulement de la session d'apprentissage. Il peut formuler sa solution et corriger son résultat sans faire appel au tuteur ni être interrompu par celui-ci.

> Le modèle d'Anderson fournit des fondements cognitifs solides pour procéder à l'élaboration d'un tutoriel intelligent mais, même si on retrouve plus de souplesse dans les versions plus récentes des théories cognitives d'Anderson, c'est un modèle qui demeure relativement contraignant sur le plan des interactions entre le tuteur et l'élève.

Au cours de la mise à l'essai du système TIDES, quelques points forts et quelques faiblesses de ce prototype de système d'apprentissage ainsi que divers *ajouts et améliorations* à y apporter ont pu être identifiés :

- la liberté consentie à l'élève de choisir son mode de travail et la formulation d'une solution complète avec ou sans décomposition de sa tâche demeure un point fort du système TIDES même si, lors de la mise à l'essai, les élèves ont surtout opté pour la non-décomposition ;
- la capacité du système d'analyser correctement des solutions formulées par les élèves et de donner un diagnostic fiable est également une force, compte tenu du fait que ces solutions correspondent à une solution complète ou partielle et comportent potentiellement plusieurs erreurs ;
- l'accès après le processus de résolution à une exécution graphique de la solution constitue une autre force du système ;
- la rétroaction à deux niveaux s'est révélée être un élément important du système TIDES en donnant lieu à des observations intéressantes relatives aux comportements d'apprentissage ;
- enfin, l'enregistrement discret et automatique des actions d'apprentissage constitue l'élément majeur qui a transformé le système TIDES en un instrument d'observation du processus d'apprentissage des élèves.

Parmi les principales faiblesses observées, il faut mentionner que :

- dans sa version actuelle, le système TIDES ne peut porter de jugement sur les bonnes réponses d'un élève : il se contente d'en reconnaître l'exactitude. Par contre, dans ses diagnostics touchant les réponses fausses, il fournit les

pistes qui permettront ultimement de déterminer les causes profondes des erreurs, en signalant les trous dans les connaissances de l'élève ou les diverses conceptions erronées qui peuvent expliquer l'erreur commise ;

- le nombre de tâches est trop limité dans la version actuelle et il en résulte des séquences de problèmes insuffisamment variés.

D'autres ajouts et suggestions d'améliorations ont été relevés au passage dans les chapitres 4, 5 et 6. Iles incluent :

- l'incorporation d'un processus d'analyse plus raffiné afin de discriminer davantage des solutions équivalentes à la solution correcte en apparence mais qui, en tenant compte du contexte, s'avèrent différentes sur le plan stratégique (voir chapitre 4- 7.3) ;
- la mise au point d'un analyseur plus raffiné dans le cas où l'élève peut être félicité (en termes de connaissances réputées maîtrisées) par le fait qu'il utilise un plan spécifique pour accomplir une tâche donnée, parce que ce plan déclenche davantage de règles (voir chapitre 4- 7.3) ;
- l'ajout d'une option permettant à l'élève de gérer son espace-travail en choisissant le mode d'affichage intégral ou restreint en fonction du bloc de résolution, ainsi que l'espace dont il dispose sur l'écran (voir chapitre 4 – 4.3) ;
- l'enrichissement et l'affinement du vocabulaire proposé à l'élève (voir chapitre 4- 4.5) ;
- l'amélioration (en plusieurs occasions et particulièrement en cas d'erreur) de la formulation du message complémentaire accompagnant le verdict dans les messages analyses (premier niveau de rétroaction), les messages diagnostiques (deuxième niveau de rétroaction) et la suppression de certains messages non pertinents (voir chapitre 4 : 4.4 et 6.5).
- l'amélioration possible du système TIDES en proposant une extension bayésienne de ce dernier et que l'on qualifierait d'explorée mais non encore

complètement intégrée dans l'état actuel du système (voir chapitre 6).

La recherche effectuée a donné naissance à un système d'apprentissage mais aussi à un instrument d'observation du processus cognitif d'un élève en situation d'apprentissage. La mise à l'essai du système TIDES (il s'agit d'une recherche-développement) a permis de démontrer que ce système n'existe pas seulement sur papier mais fonctionne véritablement et en conformité avec les résultats attendus.

L'instrument d'observation que constitue TIDES adopte une configuration particulière : l'ACT-R d'Anderson comme modèle d'apprentissage cognitif, notamment le traçage de modèle comme formalisme de diagnostic, l'arithmétique comme domaine d'apprentissage, les messages explicites comme type de rétroaction, etc. Cette configuration actuelle du système TIDES et des actions qu'il recueille, a été établie pour pouvoir répondre aux questions spécifiques de la présente recherche. Il faut cependant être conscient que le système TIDES pourrait également, avec cette même configuration, constituer l'outil de base aux fins de d'autres recherches en résolution de problèmes. Il serait par ailleurs possible d'exploiter davantage, les données recueillies pour examiner d'autres aspects du processus d'apprentissage.

La mise à l'essai du système TIDES a également permis de relever différents comportements des élèves pendant les séances d'apprentissage et de comparer ces comportements avec des résultats attendus. Il faut rappeler cependant que le type de recherche privilégié et notamment l'absence de groupe-contrôle, imposent des limites quant aux possibilités de généralisation à partir des résultats observés.

Sur le plan de la performance, les données recueillies démontrent clairement les progrès dans l'apprentissage, manifestés par la diminution de la durée moyenne requise pour utiliser correctement une connaissance particulière en fonction de la pratique de cette connaissance. Ces observations sont en conformité avec les prétentions du modèle d'Anderson à l'effet que la règle de production constitue l'unité d'apprentissage dans le Lisp-Tutor.

Concernant les différences de comportements d'apprentissage entre les habiletés de calcul et celles d'algorithme de soustraction, les données recueillies lors de la mise à l'essai se ramènent essentiellement à deux observations :

- les élèves ont tendance à tout résoudre et donner une solution coûte que coûte dans le bloc de résolution et donc à manifester surtout des habiletés de calcul ;
- les élèves ayant manifesté des habiletés algorithmiques, en décomposant une tâche, ont mis environ deux fois plus de temps pour résoudre un même ensemble de tâches que les élèves qui n'ont pas utilisé cette décomposition bloc.

Concernant les comportements des élèves vis-à-vis de la rétroaction à deux niveaux, trois observations ont été effectuées au cours de la mise à l'essai :

- les élèves ont tendance à se contenter du premier message d'erreur et à ne pas consulter les messages explicites disponibles ;
- ceux qui demandent à voir ce message font moins d'erreurs par la suite ;
- les élèves dont le rendement scolaire est élevé sont ceux qui profitent le plus de ces messages explicites.

L'analyse des résultats de la mise à l'essai a également permis de formuler certaines explications à des comportements observés, explications qui peuvent constituer autant de pistes pour des recherches ultérieures :

- la résolution d'une tâche par décomposition en sous-tâches implique plusieurs manipulations, et par conséquent, (1) l'avantage de recourir à une telle décomposition est fonction de la complexité de la tâche à résoudre et (2) il existe une limite de ce côté-ci duquel cet avantage disparaît et peut même devenir négatif ;
- l'habitude des jeux éducatifs ou vidéo dans lesquels toute l'interaction se fait à l'aide de sons et de messages iconiques amène plusieurs élèves à

travailler en devinant, par essais et erreurs plutôt que par la lecture de textes à l'écran ;

- la plus grande capacité d'abstraction liée à la lecture chez les élèves dont le rendement scolaire est élevé est le facteur principal expliquant que ces derniers sont davantage capables d'exploiter les messages diagnostiques.

Le développement et la mise à l'essai du système TIDES démontrent qu'il est possible de laisser un plus grand contrôle à l'élève (du moins en comparaison des systèmes appliquant la version moins récente du modèle d'Anderson) quant aux pistes de solution à utiliser, sans perdre la possibilité de lui fournir une rétroaction intelligente après avoir analysé son comportement. Il est important de poursuivre les recherches dans cette direction car le compromis idéal se situe certainement dans un système qui incorpore l'assignation de certains objectifs d'apprentissage (pour lesquels le tuteur peut servir de guide) et une liberté laissée à l'élève concernant la démarche d'apprentissage.

La conception et l'élaboration d'un système d'EIAO sont des opérations longues et laborieuses. Le développement de TIDES n'a pas échappé à cette réalité. Il est certain que la perspective que de tels systèmes soient développés et utilisés dans les écoles implique l'existence de coquilles de système d'EIAO dans lesquelles des contenus d'apprentissage et une stratégie tutorielle pourraient être incorporés sans avoir à développer chaque fois l'ensemble du système. TIDES va dans cette direction, dans la mesure où plusieurs de ses composantes sont généralisables à d'autres domaines d'apprentissage.

BIBLIOGRAPHIE

Abdi, H. (1994). Les Réseaux de Neurones. Éditions PUG (Presses Universitaires de Grenoble), Collection : Sciences et Technologies de la Connaissance.

Allal, L. (1999). Impliquer l'apprenant dans le processus d'évaluation: promesses et pièges de l'autoévaluation, in C. Depover, B.Noel (eds.): L'évaluation des compétences et des processus cognitifs. Bruxelles: De Boeck Université, 35-56.

Allal, L. (2001). La métacognition en perspective. Dans G. Figari, & M. Achouche (éds.), L'activité évaluative réinterrogée. Regards scolaires et socioprofessionnels, 142-146. Bruxelles : De Boeck.

Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B., & Wood, D. (2003). REDEEM: Simple Intelligent Tutoring Systems from Usable Tools, In Murray, Ainsworth, & Blessing (eds.), Authoring Tools for Adv. Tech. Learning Env., 205-232.

Anandeep, S. Pannu . (1995). Search strategy for exhaustive generation of parameters for a student model. Intelligent Systems Technical Report ISP-95-1.

Anderson, J.R. (1983). The architecture of cognition. Hillsdale, NJ : Lawrence Erlbaum.

Anderson, J.R. (1989). A theory of origins of Human Knowledge. Artificial Intelligence, 40 , 313-351.

Anderson, J. R., Boyle, C. F., Corbett, A., and Lewis, M. W. (1990). Cognitive modelling and intelligent tutoring. Artificial Intelligence, 42,7-49.

Anderson, J.R. (1993). Rules of the mind. Hillsdale, NJ : Lawrence Erlbaum.

Anderson, J.R. (1996). ACT. A simple theory of complex cognition. American Psychologist, 51, 355-365.

Anderson, J.R., Boyle, C.F., Corbett, A.T. & Lewis, M.W. (1990). Cognitive modelling and intelligent tutoring. Artificial Intelligence, 42, 7-49.

Anderson, J.R., Corbett, A.T., Koedinger, K.R. & Pelletier, R. (1995). Cognitive tutors : Lessons learned. Journal of the Learning Sciences, 4, 167-207.

Anderson, J.R. (1992). "Intelligent tutoring and high school mathematics". In ITS'92, Frasson, C., Gauthier, G. & McCalla, G. (dirs.). Springer Verlag.

Anderson, J., Reder, L. et Simon, H. (1996). Situated Learning and Education. Educational Researcher, 25 (4), 5-11.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. Science, 228, 456-462.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1987). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), Modeling Cognition, Wiley.

Anderson, J. R., Boyle, C. F., & Yost, G. (1986). The geometry tutor. The Journal of Mathematical Behavior, 5-20.

Anderson, J. R. & Lebiere, C. (1998a). The atomic components of thought. Erlbaum, Mahwah, NJ.

Anderson, J.R. & Lebiere, C. (1998b). The Newell test for a theory of Mind. Behavioral and Brain Sciences. In press.

Anderson, J.R. (1987). Skill acquisition: Compilation of weak-method problem solutions. Psychological Review, 94, 192-210.

Anderson, J. R., Kline, P. J. & Beasley C.M Jr. (1980). Complex learning processes. In R.E. Snow, P.-A. Federico & W.E. Montague (eds.) : Aptitude, learning, and Instruction, 2, pp. 199-235. Lawrence Erlbaum Associates, Hillsdate, N.J.

Anderson, J.R., and Reiser, B. J. - (1985) The LISP tutor, Byte, vol.10, n° 4, 159-175.

Anderson J. R., Language, memory and thought. Erlbaum Associates, Hillsdale, New-Jersey, 1976.

Anderson J. R. & Bower G. H., Human associative memory, Winston and Sons, Washington, 1973.

Anderson, J.R. (1986). Cognitive Modeling and Intelligent Tutoring. National Science Foundations, Washington, D.C.

Anderson, J. R., Budiu, R. & Reder, L. M. (2001). A theory of sentence memory as part of a general theory of memory. Journal of Memory and Language, 45, 337-367

Arroyo. I. (2000). Animalwatch : an arithmetic ITS for elementary and middle school students. "Learning Algebra with the Computer" Workshop. Fifth International Conference on Intelligent Tutoring Systems. Montreal, Canada.

Arsenault, C. & Lemoyne, G. (2000). Une introduction non classique aux algorithmes d'addition et de soustraction. Educational Studies in Mathematics 42 : 269–296. Kluwer Academic Publishers. Printed in the Netherlands.

Ashcraft, M.A., & E.P. Kirk. (2001). The relationships among working memory, math anxiety, and performance. Journal of Experimental Psychology : General, 130 (2), 224-237.

Attardi, G. et Simi, M. (1986) A Description-Oriented Logic for Building Knowledge Bases, in Proceedings of the IEEE, vol. 74, n°. 10, pp.1335-1344.

Attisha, M. et Yazdani. M. (1993). A micro-computer based tutor for teaching arithmetic skills.Instruction Science, 12, p. 333-342.

Baker, L. (1991). Metacognition, reading and science Education. In C. Santa et D. D. Alvermann (dir.), Science learning: Processes and applications (p. 2-13). Neward.

Baker, M., & Lund, K. (1997). Promoting reflective interactions in a CSCL environment. Journal of Computer-Assisted Learning, 13, 175-193.

Baker, M. (2003). Les dialogues avec, autour et au travers des technologies éducatives. Orientation scolaire et professionnelle, vol. 32, n° 3, 2003, p. 359-397.

Balacheff, N. (1994). Didactique et Intelligence Artificielle, in Didactique et Intelligence Artificielle, Balacheff & Vivet (eds), La Pensée Sauvage, p. 9-42.

Baron, M.(1994). EIAO : quelques repères. Revue Terminal n° 65, 67-84.

Barr A. et Feigenbaum E. A. (1986). Manuel de l'intelligence artificielle. Tome I. Paris. Eyrolle.

Barrouillet, P. et Camos, V. (2002). Savoirs, savoir-faire arithmétiques, et leurs déficiences. Programme cognitique, école et sciences cognitives. Ministère de la recherche, France

Beal, C., Woolf, B., Beck, J., Arroyo, I., Schultz, K. and Hart, D. (2000).Gaining Confidence in Mathematics : Instructional Technology for Girls. In the of Proceedings of International Conference on Mathematics/Science Education and Technology. pp. 57-64.

Beck. J.E, et Sison, J. (2004). Using knowledge tracing to measure student reading proficiencies.

Becker et Naïm, 1999 ; Les réseaux bayésiens : modèles graphiques de connaissance Eyrolles

Bell, B. (1998). Investigate and decide learning environments: Specializing task models for authoring tools design. J. of the Learning Sciences, 7(1) pp. 65-106.

Besnard, P. (1989). Logiques formelles et raisonnement de bon sens. Annales des télécommunications, vol. 44, n° 5–6, pp. 242– 250.

Bednarz, N. & Janvier, B. (1984). La Numération. Les difficultés suscitées par son apprentissage. Revue N, no 33, pp. 5-31.

Benoît, J. (2000). La communauté de pratique en réseau: une source d'apprentissage collectif. http://www.tact.fse.ulaval.ca/ang/html/cp/strategies.htm

Bideaud, J. & Houdé, O. (1991). Cognition et développement. Boîte à outils théoriques. Berne : Peter Lang.

Bideaud, J. (1991). Les chemins du nombre. Confrontations et perspectives. In J. Bideaud, C.

Blessing, S.B. (1997). A programming by demonstration authoring tool for model tracing tutors. Int. J. of Artificial Intelligence in Education. Vol. 8 , No. 3-4, pp 233-261.

Blin, F & Donohoe, R. (2000). Vers un apprentissage collaboratif dans une classe virtuelle bilingue. Alsic.org, Vol. 3, numéro 1.

Bishop, C. (1997). Classification and Regression. In : Handbook of Neural Computation (section B6.2). E. Fiesler and R. Beale (Eds.) Institute of Physics and Oxford University Press. New York, NY - U.S.A.

Borgida, A. (1992). From Type Systems to Knowledge Representations : Natural Semantics Specifications for Description Logics, International Journal on Intelligent and Cooperative Information Systems, vol.1 n°.1, World Scientific Publ. Singapore.

Braten, I. & Throndsen, I. S. (1998). Cognitive strategies in mathematics, Part II : teaching a more advanced addition strategy to an eight-year-old girl with learning difficulties. Scandinavian Journal of Educational Research, vol. 42, 2.

Brecht, B.J., MacCalla, G.I., Greer, J.E. & Jones M. (1989). Planning the Content of Instruction. Actes de Artificial Intelligence and Education (AI&Ed), pp. 32–41.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). Classification and Regression Trees. Belmont, CA : Wadsworth.

Brien, R., Bourdeau, J. et Rocheleau, J. (1999). L'interactivité dans l'apprentissage : la perspective des sciences cognitives. Revue des sciences de l'éducation, vol. 25, n° 1, 1999, p. 17-34.

Brousseau, G. (2001). Les erreurs des élèves en mathématiques. Étude dans le cadre de la théorie des situations didactiques. Petit x, 57, pp. 5-30.

Brown, J. S. & VanLehn, K. (1980). Repair theory, a generative theory of bugs in procedural skills. Cognitive Science , vol. 4, p. 426-479.

Brown, J. S., Collins, A. & Duguid, P. (1989). Situted cognition and culture of learning. Educational Researcher, 18(1), 32-41.

Brown, J., and Burton, R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science 2 : 155--192.

Brown, J. S. & VanLehn, K. (1982). Towards a generative theory of bugs. In C.P. Carpenter, J.M. Moser and T.A. Romberg (Eds.) Addition and Subtraction : a cognitive perspective. Hillsdale, NJ : Erlbaum, p. 117-135.

Brown, J.S. (1989) Situated learning : toward a new epistemology of learning, Institute for research on Learning, Palo-alto.

Brown, J.S., Burton, R.R., and Bell, A.G. (1975). SOPHIE : a step towards a reactive learning environment, International Journal of Man-Machine Studies, 7, 675-696.

Bruillard, E. et Péreira, I. (1987). Quelques problèmes d'apprentissage avec Logo et Prolog. Actes du Congrès Francophone sur l'Enseignement Assisté par Ordinateur, pp. 267-276

Bruillard, É. (1997). Les machines à enseigner. Paris : Hermès.

Budiu, R. & Anderson, J. R. (2000). Integration of background knowledge in sentence processing : A unified theory of metaphor understanding, semantic illusions, and text memory. In Proceedings of the Third International Conference on Cognitive Modeling, pp. 50-57. Groningen, Netherlands : Universal Press.

Bull S., Mangat M., Mabbott A., Abu Issa A.S. & Marsh J., "Reactions to Inspectable Learner Models: Seven Year Olds to University Students". In Workshop Learner Modelling for Reflection, AIED'05, Amsterdam, p. 1-10, 2005.

Bull, S. et Pain, H. (1995). Did I say what I think I said, and you agree with me? Inspecting and questioning the student model. In Proceedings of Artificial Intelligence in Education 1995 (p. 501-509).Washington, DC: AACE.

Bundy A., (2001). Diagnosing Arithmetic Errors using Algorithmic Debugging, Blue Book Note 1396, Edinburgh.

Bundy, A. (2002). Adapting Shapiro's Algorithmic Debugging Technique to Diagnose Student's Faulty Arithmetic Procedures.

Burton, R. R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman and J. S. Brown (Eds.), Intellligent Tutoring Systems. New York : Academic Press, p. 79-98.

Burton, R.R. (1982). Diagnosing bugs in a simple procedural skill. In Sleeman, D.H. and Brown, J.S. (Eds.) Intelligent Tutoring Systems. Academic Press, London. 157183.

Burton, J., Moore, D.M. & Magliano, S. (1996). Behaviorism in instructional technology. In D. Jonassen (Ed.), Handbook of Research for Educational Communications and Technology (pp. 46-73). New York, NY : Macmillan.

Caelen, J. (1995a). Vers une logique et dialogue. Conférence invitée au Séminaire international de Pragmatique, Jérusalem.

Caelen, J. and Villasenor, L. "Dialogue homme-machine et apprentissage" in Apprentissage par l'interaction, EuropIA. productions éd. Paris, K. Zreik coord., pp. 83-118, 1997

Carbonell, J.R. (1970). AI in CAI : artificial intelligence approach to computer-assisted instruction. IEEE transactions on Man-Machine Systems, 11, 4, 190-22.

Carpenter, T. P., Moser, J. M. & Romberg, T. A. (1982). Addition and subtraction: a cognitive perspective. Hillsdale, N.J., L. Erlbaum Associates.

Carpenter, T. P., Hiebert, J. et Moser, J.M. (1981). Problem structure and first-grade-children's initial solution processes for simple addition and subtraction problems. Journal for Research in Mathematics Education, 12(1), p.27-39.

Casta, J.-F., et Prat, B. (1994), Approche connexionniste de la classification des entreprises : contribution au traitement d'informations incomplètes, Document dactylographié, CEREG, Université de Paris-Dauphine.

Cayrol, M., Farreny, H. et Prade, H. (1982) Fuzzy Pattern Matching, Kybernetics vol.11, pp.103-116.

Chance, P. (1994). Learning and behavior. Pacific Grove, CA : Brooks/Cole.

Charlier, P. (1999). Interactivité et interaction dans une modélisation de l'apprentissage. Revue des sciences de l'éducation, vol. 25, n° 1, 1999, p. 61-85.

Chaudet, H. & Pellegrin, L. (1998). Intelligence artificielle et psychologie cognitive, Dunod.

Chauvin, Y. et Rumelhart, D.E. (1995). Backpropagation : theory, architectures, and applications. Lawrence Erlbaum Associates.

Chentouf, R. (1997). Construction de Réseaux de Neurones Multicouches pour l'Approximation. Thèse de Doctorat en Sciences Cognitives, Laboratoire TIRF - INPG, Grenoble - France, Mars.

Chiu, B., Geoffrey I. Webb, G. and Kuzmycz, M. (1997). A Comparison of First-Order and Zeroth-Order Induction for Input-Output Agent Modelling .In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.) User Modeling : Proceedings of the Sixth International Conference, UM97 (pp. 347-358). Vienna, New York : Springer Wien New York.

Clancey, W. & Joerger, K. (1988). rA Practical Authoring Shell for Apprenticeship Learning." Proceedings of ITS-88, 67-74. June 1988, Montreal.

Clancey, W.J. (1990) Why today's computers do not learn the way people do, Annual meeting of AERA, 16-20 April, Boston.

Clancey, W. J. (1997). Situated Cognition : On Human Knowledge and Computer Representations, Cambridge University Press.

Clark, H.H. & Marshall, C.R. (1981). Definite reference and mutual knowledge, in A.K.

Cole, M. & Engeström, Y. (1991). A cultural-historical approach to distributed cognition. In Distributed cognition, Salomon G. (dir.). Cambridge : Cambridge University Press, pp 1-47.

Collins, A., Brown, J.S. & Newman, S.E. (1989). Cognitive apprenticeship : teaching the crafts of reading, writing and mathematics, in L. Resnick (ed.) : Knowing, learning and instruction, LEA, Hillsdale, New-Jersey.

Collins A. (1977). Processes in Acquiring Knowledge. In Anderson R. C., Sprio R. J., Montague W. E. (eds). Schooling and the Acquiring of Knowledge, LEA, Hillsdale.

Collins A. M., Quillian M. R. (1969). Retrieval time from semantic memory. Journal of Verbal Learning and Verbal Behavior Vol. 8, pp. 240-247.

Collins, A. and Stevens, A. (1982). Goals and methods for inquiry teachers. In Glaser, R., editor, Advances in Instructional Psychology, Vol. 2. NJ: Lawrence Erlbaum Associates, Hillsdale.

Colmerauer, A., Kanoui, H., Van Caneghem, M. (1983). PROLOG, bases théoriques et développements actuels, TSI, vol. 2, n°. 4, pp.255-292.

Comiti, C. (1980). Les premières acquisitions du concept de nombre chez l'enfant entrant en cycle préparatoire. Eductional Studies in Mathematics, 4, 164-198.

Conati, C., and VanLehn, K. (1996a). POLA : a student modeling framework for Probabilistic On-Line Assessment of problem solving performance. In Proceedings of the 5th International Conference on User Modeling, 75–82.

Conati, C., and VanLehn, K. (1996b). Probabilistic plan recognition for cognitive apprenticeship. In Proceedings of the 18th Annual Conference of the Cognitive Science Society, 403–408.

Conati, C., Gertner, A., VanLehn, K., Drudzel, M. J. (1997). On-line student modeling for coached problem solving using bayesian networks. User Modeling : Proceedings of the Sixth International Conference, UM'97, Jameson, A., Paris, C., et Tasso, C., éditeurs, Vienna, New York. Springer.

Conati C., Gertner A. and VanLehn K. (2002). Using Bayesian Networks to Manage Uncertainty in Student Modeling . To appear in : Journal of User Modeling and User-Adapted Interaction, vol. 12(4), Winner of the 2002 James Chen Annual Award for Best UMUAI Paper.

Conne, F. et Brun, J. (1991). Une analyse des brouillons de calcul d'élèves confrontés à des items de divisions écrites. *Proceedings Fifteenth PME Conference*, June 29–July 4, 1991, Vol. I, pp. 239–246.

Contandriopoulos, A.-P., Champagne, L., Potvin, L., Denis, J.-L. & Boyle, P. (1990). Savoir préparer une recherche : la définir, la structurer, la financer. Les presses de l'Université de Montréal, Montréal.

Conway, J. (1997). Educational Technology's effect on models of instruction. En ligne : http ://copland.udel.edu/~jconway/ed.

Cooper, G. F. (1990). The computation complexity of probabilistic inference using Bayesian belief networks, Artificial intelligence, vol. 42, 393-405.

Cooper, G. F. (1987). Probabilistic inference using belief networks is np-hard. Knowledge Systems Laboratory, 42 :393--405.

Corbett, A. & Anderson, J. (1995). Knowledge tracing : Modeling the acquisition of procedural knowledge. User modeling and user-adapted interaction. 4 : p. 253-278.

Corbett, A., Koedinger, K. & Anderson, J. (1999). Intelligent computer tutors : out of the research lab and into the classroom. Papert presented as part of a special symposium at the American Educational Research Association annual conference, Montreal.

Corbett, A. et Trask, H. (2000). Instructional Interventions in Computer-Based Tutoring : Differential Impact on Learning Time and Accuracy. *Proceedings of ACM CHI'2000 Conference on Human Factors in Computing Systems*, 97-104.

Core, M. G., Moore, J. D., & Zinn, C. (2003). The role of initiative in tutorial dialogue. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL) (pp. 67–74). Morristown, NJ: Association of Computation Linguistics.

Corset, F. (2003). Aide à l'optimisation de maintenance à partir de réseaux bayésiens et fiabilité dans un contexte doublement censure. Thèse de doctorat.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999). Probabilistic Networks and Expert Systems. Springer-Verlag.

Cowell, R. G. (2000). Advanced inference in bayesian networks. Statistics, 19 :301-312.

Cox, L.S. (1974). Analysis, classification and frequency of systematic error computationnal patterns in the Add., Sub., Mult. Et div. In vertical Algorithm for Grades 2-6 Special Education classes. Kansas University, Research in Education., 9, p. 180-191.

Cox, L.S. (1975). Diagnosing and remediating systematic errors in addition and subtraction computations. Arithmetic Teacher 22, p. 151-157.

Cox, L.S. (1975). Systematic error in the four vertical algorithms in normal and handicapped populations. Journal for Research in Mathematics Education, p. 202-220.

Crick, F. (1989). The recent excitement about neural networks, Nature, 337 :129-132.

Crinon et al. (2000) Les effets des systèmes et des outils multimédias sur la cognition, l'apprentissage et l'enseignement. Cognition et Didactique du

Texte. Rapport final du CNCRE (Comité national de Coordination de la Recherche en Éducation).

Dagum P. and Luby M. (1993). Approximating probabilistic inference in Bayesian belief network is NP-hard. Artificial Intelligence, vol. 60, pp. 141-153.

Danine, A., Lefebvre, B. & Mayers, A. (2006.a). TIDES - Using Bayesian Networks for Student Modeling. Proceedings of 6th IEEE International Conference on Advanced Learning Technologies (ICALT 06), July 5-7, Kerkrade, Netherlands, 2006.

Danine, A., Lefebvre, B. & Mayers, A. (2006.b). Utilisation des réseaux bayésiens pour la gestion de l'incertitude en modélisation de l'étudiant. Conférence sur les Systèmes Intelligents : Théories et applications. 16-17 Novembre. Mohammedia, Maroc.

Davalo E. & Naïm P.(1989). Des réseaux de neurones. Editions Eyrolles.

Dawid, A.P. (1992). Application of a general propagation algorithm for probabilistic expert systems. Statistics and Computing. 2 25-36.

De Corte, E., Verschaffel, L., & Schrooten, H. (1991). Computer simulation as a tool in studying teacher's cognitive activities during error diagnosis in arithmetic. In P. Goodyear (Ed.), Teaching knowledge and intelligent tutoring (pp. 367-378). Norwood, NJ : Ablex. (91/9).

Depover, C., Quintin, J. & Delièvre, B. (2000). La conception des environnements d'apprentissage : de la théorie à la pratique/de la pratique à la théorie. Alsic, Vol 3, (1) pp.3-18

De Rosis, F., Covino, E., Falcone, R., and Castelfranchi, C.. (2001). Bayesian Cognitive Diagnosis in Believable Multiagent Systems. International Belief Revision NMR7 Workshop, Trento, Italy. Also in "Frontiers of Belief Revision", M.A. Williams, H. Rott (eds.), Kluwerg Academic Press.

Dewey, J. (1933). How We Think, éd. révisée, Boston, Heath.

Dillenbourg, P. & Self, J. (1991). A Framework for Learner Modelling. Interactive Learning Environments, vol. 2, n° 2, pp. 111–137.

Dillenbourg, P., Baker, M.,Blaye, A. et O'Malley, C.(1996). The Evolution of Research on Collaborative Learning. Dans E. Spada et P. Reiman (éds) Learningin Humans and Machine :Towards an Interdisciplinary Learning Science. Oxford, Elsevier, p. 189-211.

Dillenbourg, P. & Schneider D. (1995). Collaborative learning and the Internet. Proceedings of ICCAI.

Dillenbourg, P. & Traum, D. (1999). Does a shared screen make a shared solution ? International Conference on Computer Supported Collaborative Learning, Stanford, décembre.

Dillenbourg, P., Baker, M., Blaye, A. & O'Malley, C. (1995). The evolution of research on collaborative learning. In E. Spada & P. Reiman (Eds.), Learning in Humans and Machines : Towards an Interdisciplinary Learning Science (pp. 189-211). Oxford : Elsevier.

Dillenbourg P., Poirier, C. & Carles, L. (2003). Communautés virtuelles d'apprentissage : e-jargon ou nouveau paradigme ? In A. Taurisson et A. Sentini. Pédagogies.Net. Montréal, Presses Universitaires du Quebec.

Dillenbourg, P. & Jermann, P. (1996). Le paradoxe de la machine sociale. Interface.

Dillenbourg, P. (1993). Évolution épistémologique en EIAO. Ingenierie Educative. Sciences et Techniques Educatives. 1 (1), 39-52.

Dimitrova, V., Self, J.A. and Brna, P. (2000). Involving the Learner in Diagnosis – Potentials and Problems, tutorial to be presented at Web Information Technologies : Research, Education and Commerce, Montpellier, France, 2-5 May.

Dimitrova V. (2001). Interactive Open Learner Modelling, PhD thesis, Computer Based Learning Unit, Leeds University.

Dimitrova V. (2003). StyLE-OLM: Interactive open learner modelling". International Journal of Artificial Intelligence in Education, v13, 35-78.

Dion, P. et Lelouche, R. (1992). Application de la méthodologie du traçage de modèle à un environnement d'apprentissage utilisant une stratégie pédagogique non directe. In ITS'92, Frasson, C., Gauthier, G. & McCalla, G. (dirs.). Springer Verlag.

Dionne, J. (1986). Effet de l'analyse conceptuelle sur la perception des mathématiques, de leur apprentissage et leur enseignement chez les enseignants du primaire. Thèse de Doctorat, Université de Montréal.

Doise, W. & Mugny, G. (1984). Le développement social de l'intelligence. Paris : InterEditions.

Dubinsky E. (1991). Reflexive abstraction. In Tall D. (ed) Advanced mathematical thinking, Kluwer. pp-95-123.

Dubinsky E. (1992). Utilisation de l'ordinateur à partir d'une théorie de Piaget sur l'apprentissage de concepts mathématiques. In Cornu B. (ed), L'ordinateur pour enseigner les Mathématiques, Nouvelle Encyclopédie Diderot, Presses Universitaires de France, Paris, pp. 237-269.

Dubois, L. & Dagau, P. (2000). Les modèles de l'apprentissage et les mathématiques. http://tecfa.unige.ch/~laurent/didact/theories.htm

Duffy, T. & Cunningham, D.J. (1996). Constructivism : Implications for the design and delivery of instruction. In D.H. Jonassen (Ed.), Handbook of research for educational communications and technology (pp. 170-198). New York, NY : Macmillan.

Edwards, D. (2000). Introduction to Graphical Modelling. 2nd ed. Springer-Verlag.

Ellerton, N. F. (1986). Children's made up mathematics problems : A new perspective on talented mathematicians Educational Studies in Mathematics, 17, pp. 261-271.

Elman, J.L. (1993). Learning and Development in Neural Networks : The Importance of Starting Small. Cognition, 48(1993), pp. 71-99.

Emond, B. (1997). Modeling natural language comprehension and anaphora resolution with ACT-R. In Proceedings of the Fourth Annual ACT-R Workshop (pp. 1-8). Pittsburgh, PA : Department of Psychology, Carnegie Mellon University.

Engelhandt, J. M. et Weibe, J. H. (1979). Computational Errors as Indication of Numeration Concept Misunderstandings. Paper presented at the Annual Conference of the Research Council for Diagnostic and Prescriptive Mathematics. Tampa, April.

Engelhardt, J. M. (1982). Using Computational errors in diagnostic teaching. Arithmetic Teacher vol. 5, 4, April, p. 16-19.

Fabiani, P. (1996). Représentation dynamique de l'incertitude et stratégie de prise d'information pour un système autonome en environnement évolutif. Thèse de doctorat. Centre d'Études et de Recherche de Toulouse de l'ONERA.

Fayol, M. (1990).L'enfant et le nombre. Neufchâtel/Paris : Delachaux et Niestlé.

Fayol, M. , Camos, V. & Roussel, J.L. (2000). Acquisition et mise en œuvre de la numération par les enfants de 2 à 9 ans. *In* M. Pesenti et X. Seron (Eds.) , *Neuropsychologie des troubles du calcul et du traitement des nombres* (pp. 33-58). Marseille : Solal.

Ferrero, B., M. Martín, A. Alvarez, M. Urretavizcaya, and I. Fernández-Castro, (2005). Authoring and Diagnosis of Learning Activities with the KADDET Environment. J. of Univ. Computer Science. 11(9): p. 1530-1542.

Fiesler, E. & Beale, R. (1997). Handbook of Neural Computation. Institute of Physics and Oxford University Press. New York, NY. U.S.A.

Flavell, J.H. (1979). Metacognition and cognitive monitoring. A new area of cognitive developmental inquiry, American Psychologist, 34(10), p. 906-911.

Fletcher, J.D. (1984). Intelligent Instructional System in Training, dans S. A. Andriole (ed), Applications in Artificial Intelligence, Princeton University Press.

Foss, C.L. (1987). Learning from errors in ALGEBRALAND. Technical Report IRL87 003. Institute for Research on Learning, Palo Alto, California.

Frey, B. (1998). Graphical models for machine learning and digital communication, MIT Press.

Frye, D., Braisby, N., Lowe, J., Maroudas, C., & Nicholls, J. (1989). Young children's understanding of counting and cardinality, Child Development, 60, 1158-1171.

Fuson, K.C., & Hall, J.W. (1983). The acquisition of early number word meanings. In H. Ginsburg (Ed.), The development of children's mathematical thinking (pp. 49-107), New-York: Academic Press.

Gagné, E.D. (1985a). The Cognitive Psychology of School Learning. Little, Brown and Company, Boston.

Gagné, R., M. (1985b). The conditions of learning and the theory of instruction. CBS College Publishing, 4e édition.

Gagné, R., M., Briggs, L. & Wager, W.(1992). Principles of instructional design. Harcourt Brace Jovanovich, Orlando, FL, 4e édition.

Gallistel, C.R. & Gelman, R. (1992). Preverbal and verbal counting and computation. *Cognition, 44*, 43-74.

Gaonac'h, D. (1991). Théories d'apprentissage et acquisition d'une langue étrangère, Paris : Hatier-Didier.

Gardner, H. (1985). The mind's new science. A History of the cognitive revolution. New York : Basic Books

Gaynor, P. (1981) The effect of feedback delay on retention of computer-based mathematical material. Journal of computer-based instruction, Nov., pp. 28-34.

Geary, D.C., Hamson, C.O., Chen, G., Liu, F., Hoard, M.K., & Salthouse, T.A. (1997). Computational and reasoning abilities in arithmetic : Cross-generational change in China and the United States. Psychonomic Bulletin and Review, 4, 425-430.

Geary, D. C., Hoard, M. K., & Hamson, C. O. (1999). Numerical and Arithmetical Cognition: Patterns of functions and deficits in children at risk for a mathematical disability. Journal of Experimental Child Psychology, 74, 213-239.

Giardina, M.et Laurier, M. (1999). Modélisation de l'apprenant et interactivité. Revue des sciences de l'éducation, vol. 25, n° 1, 1999, p. 35-59.

Gick, M. & Holyoak, K.J. (1980): "Analogical problem solving". Cognitive Psychology 12, p.306-355.

Gick, M. & Holyoak, K.J. (1983) : "Schema induction and analogical transfer". *Cognitive Psychology* 15, p. 1-38.

Glasersfeld, E. (1994). Pourquoi le constructivisme doit-il être radical ? Revue des sciences de l'éducation, 20(1), 21-28.

Glaserfeld, E. (1995). A constructivsim approach to teaching. In L. Steffe & J. Gale (Eds.), Constructivism in education (pp. 3-16). Hillsdale, NJ : Lawrence Erlbaum.

Goldstein, I.P., and Carr, B. (1977). The computer as a coach: an athletic paradigm for intellectual education. Proc. Of 1977 ACM annual conference, Seattle, p. 227-233.

Goldstein, I.P., (1982). The genetic graph : a representation for the evaluation of procedural knowledge. Intelligent Tutoring Systems, D.Sleeman et J.S.Brown (eds), Academic Press, New York, 1982, p. 51-78.

Gonzales, C. and Wuillemin P.H. (1998). Réseaux bayésiens en modélisation d'utilisateurs. Sciences et Techniques éducatives Volume 5, n°2, page 173. Laboratoire d'informatique de Paris 6 (LIP6).

Goupil, G. & Lusignan, G. (1993). Apprentissage et enseignement en milieu scolaire. Québec: Gaëtan Morin Editeur.

Graesser, A. C., N. K. Person, & J. P. Magliano (1995). Collaborative dialogue patterns in naturalistic one-on-one tutoring. Applied Cognitive Psychology, vol. 9, 495-522.

Graesser, Arthur C., Kurt VanLehn, Carolyn P. Rosé, Pamela W. Jordan, & Derek Harter (2001). Intelligent Tutoring Systems with Conversational Dialogue. AI Magazine 22(4): 39-52.

Graesser, A. C., Jackson, G. T., Mathews, E. C., Mitchell, H. H., Olney, A., Ventura, M., Chipman, P., Franceschetti, D., Hu, X., Louwerse, M. M., & Person, N. K. (2003). Why/AutoTutor: a test of learning gains from a physics tutor with natural language dialog. In R. Alterman, & D. Hirsh (Eds.), Proceedings of the 25th Annual Conference of the Cognitive Science Society. Boston, MA: Cognitive Science Society, 1-5.

Graesser,A. C., Lu, S., Jackson, G. T., Mitchell, H. H.,Ventura, M., Olney, A., et Louwerse, M. (2004). AutoTutor: A tutor with dialogue in natural language. Behavioral Reseach Methods, Instruments and Computers, 36, 180–193.

Grandbastien, M. (1999). Teaching Expertise is at the Core of ITS Research. International Journal of Artificial Intelligence in Education, 10, 335-349.

Grangeat, M. (1999). Processus cognitifs et différenciation pédagogique. Dans C., Depover, & B., Noël, (Éds). L'évaluation des compétences et des processus cognitifs, modèles, pratiques et contextes (pp. 115-127). Bruxelles : De Boeck.

Greeno, J.G. & Moore, J.L (1993) Situativity and Symbols: response to Vera and Simon, Cognitive Science, vol. 17, 1, 49-60.

Greeno, J. (1991). Number sense as situated knowing in a conceptual domain. Journal for Research in Mathematics Education, 22, 170-218.

Grégoire, J. (1999). Que peut apporter la psychologie cognitive à l'évaluation formative et à l'évaluation diagnostique ? In: L'évaluation des compétences et des processus cognitifs, B. Noël & Ch. Depover ed., Bruxelles, De Boeck, 1999, 17-33.

Grégoire, J. (2001). Evaluer les troubles du calcul. In: Les troubles du calcul et les dyscalculies chez l'enfant. A. Van Hout & C. Meljac ed., Paris, Masson, 309-329.

Grégoire, J., Van Nieuwenhoven, C. (1999). Le développement du comptage et son rôle dans les troubles numérique. Confrontations Orthophoniques : Les activités numériques. Opérations logiques et formulations langagières, 3, 53-83.

Guin, N. (1997). Reformuler et classer un problème pour le résoudre. Thèse de doctorat, Université de paris 6, France.

Halpern , J. Y. , Pearl, J. (2001a). Causes and explanations : A structural model approach. Part I: Causes. Proc. of the17th Conf. on Uncertainty in Artificial Intelligence (UAI'2001), (J. Breese, D. Koller, eds.), Seattle, Wa., Aug. 2-5, Morgan Kaufmann Publ., San Francisco, 194-202, 2001.

Halpern , J. Y. , Pearl, J. (2001b). Causes and explanations : A structural model approach. Part II! : Explanations. Proc. of the 17th Inter. Joint Conf. on Artificial Intelligence (IJCAI'2001), Seattle, Wa., Aug. 4-10, Morgan Kaufmann Publ., San Francisco, 27-34, 2001.

Halverson, C.A. & Rogers, Y. (1995). Combining the social and the cognitive - Distributed cognition theory and apllication, Tutorial presented at ECSCW'95, Stockholm.

Hammond, K.J. (1990). Case-based planning: A framework for planning from experience. Cognitive Science 14, p. 385-443.

Hannafin, M.J., Hannafin, K.M., Land, S. & Oliver, K. (1997). Grounded practice and the design of constructivist learning environments. Educational Technology Research and Development (ETR&D), 45 (3), 101-117.

Haton J.P, Bouzid N., Charpillet F., Haton M., Léasri B., Léasri H., Marquis P., Mondot T. & Napoli A. (1991). Le raisonnement en intelligence artificielle. Modèles, techniques et architectures pour les systèmes à bases de connaissances. InterEditions.

Hebb, D.O. (1949). The organization of behavior, New York : Wiley.

Hennessy, S. (1990). Why Bugs Are not Enough. In Elsom-Look (Eds.). Guides Discovery learning Paul Chapman Publishing.

Henri, F. (2002). Apprentissage collaboratif en mode virtuel. Centre de recherche LICEF Télé-université Montréal INRP-GRAME-MSH : 20 novembre 2002. http://www.inrp.fr/Tence/docseminaires/fhenri021120.pdf

Hirashima,T. Nakano, A. and Takeuchi, A. (2000). A Diagnosis Function of Arithmetical Word Problems for Learning by Problem Posing Proc of PRICAI2000, pp. 745-755.

Holt, P., Dubs, Jones, M., and Greer, J. (1994). The State of Student Modelling. In J. Greer & G. McCalla (Eds.) Student Modelling : Key to Individualized

Knowledge-based Instruction. Berlin: Springer-Verlag, NATO ASI Series, 3-35

Hoover, M.L. (1997). Effects of textual and cohesive structure on discourse processing. Discourse Processes, 23 (2), 193-220.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the USA, 79(8) :2554–2558.

Højsgaard, H. (1996). Learning structure from data and experts. Mathematics and Computer in simulation, 42 : 143-152.

Hsieh, P., Halff, H, Redfield, C. (1999). Four easy pieces: Developing systems for knowledge-based generative instruction. Int. J. of Artificial Intelligence in Education. (this issue)

Huang, C. and Darwiche, A. (1996). Inference in Belief Networks : A procedural guide. Intl. J. Approximate Reasoning, 15(3) :225-263.

Hutchins E. (1989). The technology of Team Navigation, in Galegher J., Kraut B., Egido C. (eds.), Teamwork : Social and Technical Bases of Collaborative work, Lawrence Erlbaum Associates, Hillsdale.

Hutchins, E. (1991). How a cockpit remembers its speed, UCSD research report, San Diego.

Hutchins, E. (1995). Cognition in the wild, Bradford Books-MIT Press, Cambridge MA.

Ibrahim, B., Laustsen, B., Aubord, A. et Tepper, M. (1996). Techniques de génie logiciel pour l'EAO, Université de Genève.

ITS 2004 Workshop on Dialog-Based Intelligent Tutoring Systems: State of the Art and New Research Directions. Designing Computational models of collaborative learning interaction. http://www.cscl-research.com/Dr/ITS2004Workshop/index.htm

Jacobson, M.J., Maouri, C., Mishra, P. & Kolar, C. (1995). Learning with hypertext learning environments : theory, design and research., Journal of Educational Multimedia and Hypermedia, 4, 321-364.

Javabayes (2003) www.cs.cmu.edu/~javabayes/

Jean, S. (2000). PÉPITE : un système d'assistance au diagnostic de compétences, Thèse de doctorat de l'Université du Maine

Jensen, F. (1996). An Introduction to Bayesian Networks, Springer-Verlag, North America.

Jensen, F. (1999). An introduction to Bayesian Networks. UCL Press.

Jensen, F. (2001). Bayesian Networks and Decision Diagrams. Springer.

Johnson, T. R. (1998). Acquisition and transfert of declarative and procedural knowledge, In Ritter F. E. and Young R. M. (Eds), Proceedings of the second

conference on cognitive modelling, Nottingham University Press, Nottingham, pp. 15-22.

Johsua, S. & Dupin, J.-J. (1993). Introduction à la didactique des sciences et des mathématiques. Paris : PUF.

Jonassen, D.H. & Murphy, L.R. (1999). Activity theory as a framework for designing constructivist learning environments. Educational Technology Research and Development (ETR&D), 47 (1), 61-79.

Jonassen, D.H., Campbell, J. & Davidson, M.E. (1994). Learning with media : restructuring the debate, Educational Technology Research and Development (ETR&D), 42 (2), 31-39.

Jong, T. de & vanJoolingen, W.R. (1998). Scientific discovery learning with computer simulations of conceptual domains. Review of Educational Research, Vol. 68 No. 2, pp. 179-201.

Jongman G. M. (1998). How to fatigue ACT-R? In Ritter F. E. and Young R. M. (Eds), Proceedings of the second conference on cognitive modelling, Nottingham University Press, Nottingham, pp. 52-57.

Jordan, M.I. (1998). Learning in Graphical Models. MIT Press.

Kabbaj A. et al. (2006). AminePlatform. http://sourceforge.net/projects/amine-platform.

Kamii, C. (1990). Les enfants réinventent l'arithmétique. Berne : Peter Lang.

Kasabov, N. (1997). Learning Fuzzy Rules and Approximate Reasoning in Fuzzy Neural Networks and Hybrid Systems. Fuzzy Sets and Systems. Special Issue, pp.1-19.

Kass, R. (1987). The Role of User Modelling in Intelligent Tutoring System. Rapport technique, Moore School, Université de Pennsylvanie.

Katz, S., Lesgold, A., Eggan, G. et Gordin, M. (1994). Modelling the Student in Sherlock II. Actes des journées Student Modelling : The Key to Individualized Knowledge-Based Instruction, éd. par Greer (J. E.) et Mac Calla (G. I.). pp. 99–126. – Springer- Verlag.

Katz, S., Lesgold, A., Hughes, E., Peters, D., Eggan, G., Gordin, M., & Greenberg, L. (1998). Sherlock 2 : An intelligent tutoring system built on the LRDC framework (pp. 227-258). In C. P. Bloom & R. B. Loftin (Eds.), Facilitating the development and use of interactive learning environments. Mahwah, NJ : Erlbaum.

Kay, J. (1995). The um toolkit for cooperative user modeling. User Modeling and User-Adapted Interaction, 4, 149-196.

Kayser D. (1997) La représentation des connaissances. Paris : Hermès.

Kimball, R. (1982). A self-improving tutor for symbolic integration. In Sleeman D., Brown J.S. (eds), Intelligent Tutoring System, Academic Press, p. 283-307.

Kiyama, M., Ishiuchi, S., Ikeda, K., Tsujimoto, M. & Fukuhara, Y. (1997). Authoring Methods for the Web-Based Intelligent CAI System CALAT and its Application to Telecommunications Service. In the Proceedings of AAAI-97 , Providence, RI.

Klahr, D. & Wallace, J. (1976). Cognitive Development, an Information Processing View. Lawrence Erlbaum Associates, Hillsdate, N.J.

Klahr D. & Robinson M. (1981). Formal assessment of problem-solving and planning processes in preschool children, Cognitive psychology, vol. 13, pp. 113-148.

Larochelle, M. &Bednarz, N. (1994). À propos du constructivisme et de l'éducation. Revue des sciences de l'éducation, 20(1), 21-27

Lauritzen, S. (1992). Propagation and probabilities, means and variances in mixed graphical association models. JASA, 87 :1098 -1108.

Lauritzen, S. (1996). Graphical Models, Oxford.

Lave, J. & Wagner, E. (1991). Situated learning : Legitimate peripheral participation. New York : Cambridge University Press.

Lave, J. (1988). Cognition in practive : Mind, mathematics, and culture in everyday life. New York : Cambridge University Press.

Lave, J., Murtaugh, M., Delarocha, O. (1984). The Dialectic of Arithmetic in Grocery Shopping, in : Rogoff Barbara & Lave jean (dir)- everyday cognition : its Development en Social Context- Cambridge Massachussets, and London England : Harvard University Press- p. 67-95.

Laveault, D. (2000). La régulation des apprentissages et la motivation scolaire, Québec, Ministère de l'Éduction du Québec.

Lebiere, C. (2001). A theory-based model of cognitive workload and its applications. In Proceedings of the 2001 Interservice/Industry Training, Simulation and Education Conference. Orlando, FL.

Lepage, E. & al. (1992). Système d'aide à la décision fondé sur un modèle de réseau bayésien application à la surveillance transfusionnelle. Informatique et Santé. Volume 5, 78-87.

Lewis, A. B. & Mayer, R. E. (1987). Students miscomprehension of relational statements in arithmetic word problems. Journal of Educational Psychology, 79(4), 363-317.

Lewis, R. (1997). An Activity Theory framework to explore distributed communities. Journal of Computer Assisted Learning, 13, pp. 210-218.

Litman, D. and Silliman, S. (2004). ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In Proceedings of the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL) (Companion Proceedings), Boston, MA.

Major, N., Ainsworth, S. & Wood, D. (1997). REDEEM: Exploiting symbiosis between psychology and authoring environments. International J. of Artificial Intelligence in Education. Vol. 8 , No. 3-4, pp. 317-340.

Makatchev, M., Jordan, P., etVanLehn, K. (2004). Abductive Theorem Proving for Analyzing Student Explanations and Guiding Feedback in Intelligent Tutoring Systems. Journal of Automated Reasoning and Theorem Proving in Education, 32(3), 187-226.

Marguin, J. (1994). Histoire des instruments et machines à calculer : trois siècles de mécanique pensante, 1642-1942. Paris : Hermann.

Marino, O. (1993). Raisonnement classificatoire dans une représentation à objets multi-points de vues. Thèse de doctorat, Université Joseph Fourrier, Grenoble1.

Markman, E.M. (1979). Classes and collections: Conceptual organization and numerical abilities. Cognitive Psychology, 11, 395-411.

Martin, J., & VanLehn, K. (1995). A Bayesian approach to cognitive assessment. In Nichols, P., Chipman, S., and Brennan, R. L., eds., *Cognitively Diagnostic Assessment*. Hillsdale, NJ : Erlbaum.

Masson, M.H. and Dubuisson, B. 1993. A Neural Architecture for Diagnosis. IEEE International Conference on Systems, Man and Cybernetics 5 :737-742.

Maurice, J. (2002) Apprentissage et développement. http://corinne.vigean.free.fr/depp/cours

Mayers, A. (1997). Miace : Une architecture théorique et computationnelle de la cognition humaine pour étudier l'apprentissage. Thèse de Doctorat, Université de Montréal. Canada.

McClellan, H. (1996). Situated learning perspectives. Englewood Cliffs, NJ : Education Technology Publications.

Meila, M. and Heckerman, D. (2001). An experimental comparison of several clustering and initialization methods. Machine Learning, 42 :9 -29.

Mendelsohn P. et Dillenbourg P. (1991). Le développement de l'enseignement intelligemment assisté par ordinateur. In : J.F. Le Ny (Ed), Intelligence. Naturelle et Intelligence Artificielle (233-258). Paris : PUF.

Mendelsohn, P. (1995). EIAO et psychologie cognitive. Sciences et Techniques éducatives, vol. 2, 1, pp. 9–29.

Mendelson, P. (1994). Le transfert des connaissances. Conférence à l'Université de Lyon II.

Mendelson, P. (1996). Le concept de transfert, in P., Meirieu et M., Develay, C., Durand et Y., Mariani, (dir.), (1996). Le transfert des connaissances en formation initiale et continue. Lyon : Centre régional de documentation pédagogique, pp. 11-19.

Mendelsohn P. (1998) Quand les technologies éducatives nous aident à repenser la question de l'efficacité de l'enseignement In S. Hannart – L'efficacité de nos système de formation. Delachaux et Niestlé.

Merrill, D. (2003). Using Knowledge Objects to Design Instructional Learning Environments. In T. Murray, S. Blessing & S. Ainsworth (Eds.), Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software. Dordrecht: Kluwer Academic Publishers.

Merrill, M. D. & ID2 Research Team. (1993). Instructional Transaction Theory: knowledge relationships among processes, entities, and activities. Educational Technology, 33(4), 5-16.

Merrill, M., D., et ID2-Research-Group. (1998). ID Expert: A Second Generation Instructional Development System. Instructional Science, 26, 243-262.

Millán, E and LUIS, J. R. (2002). A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation. User Modeling and User-Adapted Interaction 12 : 281-330, Kluwer Academic Publishers. Printed in the Netherlands.

Mizoguchi R., et Bourdeau J. 2000. «Using Ontological Engineering to Overcome Common AI-ED Problems». International Journal of Artificial Intelligence in Education, IJAIED. vol. 11, no 2, p. 107-121. En ligne. http://cbl.leeds.ac.uk/ijaied/

Morais, J. (1994). L'art de lire. Paris : Odile Jacob.

Mousty, Ph., Leybaert, J., Alégria, J., Content, A. & Morais, J. (1994). BELEC. Une batterie d'évaluation du langage écrit et de ses troubles. In J. Grégoire & B. Piérart (Eds.). Evaluer les troubles de la lecture (pp.127-145). Bruxelles : De Boeck.

Murphy, K. (2001). An introduction to graphical models. http ://http.cs.berkeley.edu/~murphyk/

Murray T. 1999. «Authoring intelligent tutoring systems: an analysis of the state of the art». In International Journal of Artificial Intelligence in Education (IJAIED): Special Issue on Authoring Systems for Intelligent Tutoring Systems, Murray T. et Blessing S., p. 98-129.

Murray T., Blessing S. et Ainsworth S. (2003). Authoring Tools for Advanced Technology Learning Environments. Dordrecht, Hardbound, Kluwer Academic Publishers: 571 p.

Murray, T. (1998). Authoring knowledge-based tutors: Tools for content, instructional strategy, student model, and interface design. J. of the Learning Sciences, 7(1) pp. 5-64.

Nakano, A., Hirashima, T & Takeuchi, A. (2000). A Learning Environment for Problem posing in Simple Arithmetical Word Problem. Proc of ICCE2000, pp. 91-98.

Nakano A., Hirashima T., and Takeuchi, A. (2002). An Evaluation of Intelligent Learning Environment for Problem Posing. S.A. Cerri, G. Gouardères, and F. Paraguaçu (Eds.) :ITS 2002,LNCS 2363, pp.861 –872, 2002. Springer-Verlag Berlin Heidelberg.

Naïm, P. et al. (2004). Les réseaux bayésiens. Groupe Eyrolles

Neapoliton, R. (1990). Probabilistic Reasoning in Expert Systems. John Wiley & Sons.

Newell A., Simon, H.A. (1972). Human Problem Solving. Prentice Hall, Englewood Cliffs, N.J.

Newell A., Rosenbloom P.S. (1981). Mechanism of skill acquisition and the law of practice. In J.R. Andderson(eds) : Cognitive skills and their acquisition, pp. 1-55. Hillsdale, N.J.

Newell A., Simon, H.A. (1963). GPS, a program that simulates human thought, dans Computers and Tought, Feigenbaum & Feldman (éd.), McGraw Hill, N.Y., pp.279- 93.

Nicaud, J. et Vivet, M. (1988). Les tuteurs intelligents, réalisations et tendances de recherche, TSI, Vol. 7, n°1, pp. 21-45.

Nicolson, R. I. (1988). The SUMMIT intelligent arithmetic tutor. In the fifth International Conference on Technology and Education CEP Consultants, Ltd, vol. 1, pp. 348-351.

Niederhauser, D.S., Salem, D.J. & Fields, M. (1999). Exploring teaching, learning and instructional reform in an introductory technology course. Journal of Technology and Teacher Education, 7 (2), 153-172.

Nikolopoulos, C. (1997). Expert Systems - Introduction to First and Second Generation and Hybrid Knowledge Based Systems. Marcel Dekker Inc. Press.

Nkambou, R. (1996). Modélisation des connaissances de la matière dans un système tutoriel intelligent : modèle, outils et applications. Thèse de la Facult. des études supérieures de l'Université de Montréal.

Nkambou R., Frasson C. et Gauthier G. 2003. «CREAM-Tools: An Authoring Environment for Knowledge Engineering in Intelligent Tutoring Systems». In Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software, Murray T., Blessing S. et Ainsworth S., p. 93-138. Dordrecht: Kluwer Academic Publisher.

Nkambou, R., Gauthier, R., & Frasson, M.C. (1996). CREAM-Tools: an authoring environment for curriculum and course building in an ITS. In Proceedings of the Third International Conference on Computer Aided Learning and Instructional Science and Engineering. New York: Springer-Verlag.

Noël B. (1997). La métacognition, De Boeck Université, France.

Nunes, T., Schliemann, A.D. & Carraher, D.W. (1993). Street mathematics and school mathematics. Cambridge : Cambridge University Press.

Nunes, T. & Bryant, T. (1996). Children doing mathematics. Oxford : Blackwell.

O'Shea, T. & Self, J.A. (1983). Learning and Teaching with computers. Prentice-Hall, Englewood Clifs, N.J.

Ohlsson, S. (1992). Constraint-Based Student Modelling. Artificial Intelligence in Education (AIEd), vol. 3, 4, pp. 429–447.

Orey, M.A. & Burton, J.K. (1990). Process Oriented Subtraction Interface for Tutoring. Journal of Artificial Intelligence in Education, 1,2, pp. 77-85.

Orsier, B. (1995). Etude et Application de Systèmes Hybrides NeuroSymboliques. Thèse en Informatique, Laboratoire LIFIA-IMAG, UJF - Grenoble.

Osório, F. S. (1998). INSS : un système hybride neuro-symbolique constructif. Thèse de doctorat. Institut National Polytechnique de Grenoble. Laboratoire Leibniz – IMAG.

Paiva A. & Self, J. (1994). A Learner Model Reason Maintenance System. Actes de European Conference on Artificial Intelligence (ECAI), pp. 193–196.

Papert, S. (1980). Mindstorms : Children, computers and powerful ideas. New York : Basic Books.

Paquette, G. (1999). Les environnements d'apprentissage intelligents. Texte tiré de la note de cours INF5100 *L'intelligence artificielle.* Télé-université, Montréal.

Paquette, G. Pachet, F. Giroux, S. (1994). EpiTalk, un outil générique pour la construction de systèmes conseillers. Techniques et Sciences Educatives, Vol. 1, n. 3, pp. 305-336.

Pea, R. (1993). Practices of distributed intelligence and design for education in G., Salomon, (éd.), Distributed cognition, (p. 134-148). Cambridge, MA : Cambridge University Press.

Pearl, J. (1997). Probabilistic Reasonning in intelligent systems - Network of Plausible Inference. Morgan Kaufmann, San Francisco, CA, deuxième edition.

Pearl, J. (2000). Causality. Cambridge.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference. Morgan Kaufmann.

Perkins, D. & Simmons, R. (1988). Patterns of misunderstanding : an integrative model of science, math, and programming. Review of Educational Research, 58, 303-326.

Perkins, D. (1995). L'individu-plus : une vision distribuée de la pensée et de l'apprentissage. Revue française de pédagogie (111), 57-71.

Person, N., Graesser, A., Kreuz, R., Pomeroy, V., etGroup, T. R. (2001). Simulating human tutor dialog moves in AutoTutor. International Journal of Artificial Intelligence in Education, 12, 23-39.

Philips, D. (1995). The good, the bad, and the ugly : the many faces of constructivism. Educational Researcher, 14, 11-17.

Piaget, J. (1977). The development of thought : equilibration of cognitive structures. New York, NY : Viking.

Pon-Barry, H., Clark, B., Owen Bratt, E., Schultz, K. & Peters, S. (2004a). Evaluating the Effectiveness of SCoT: a Spoken Conversational Tutor. Proceedings of ITS 2004 Workshop on Dialogue-based Intelligent Tutoring Systems: State of the Art and New Research Directions. Maceio, Brazil. 23–32.

Proust, J. (1995). Espace et représentation. Archives de Philosophie, (58) :563-575.

Psyché, V. (2007). Rôle des ontologies en ingénierie des EIAH : cas d'un système d'assistance au design pédagogique. Thèse de doctorat, Université du Québec à Montréal, Canada.

Py, D. (1998). Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève, Sciences et techniques éducatives, Vol. 5, n°2, pp. 123-140, Hermès.

Py, D. (2001), Environnements interactifs d'apprentissage et géométrie, Habilitation à diriger des recherches.

Pylyshyn, Z.W. (1989). Computing in cognitive science, Foundations of Cognitive Science, Posner Editor, The M IT Press, 133-159.

Pynadath, D. V., & Wellmann, M. P. (1995). Accounting for context in plan recognition, with application to traffic monitoring. In P. Besnard & S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 472–481). San Francisco : Morgan Kaufmann.

Rankin, R.J. & Trepper, T. (1978). Retention and delay of feedback in a computer-assisted instruction task. Journal of experimental education, 46, 4, pp. 67-70.

Rational Rose (2003) www.ibm.com/software/fr/rational/

Renaudie, D. (2005). Méthodes d'apprentissage automatique pour la modélisation de l'élève en algèbre. Thèse de doctorat. INPG, France

Renkl, A. (1997). "Learning from worked-out examples : a study on individual differences". *Cognitive Science* 21 (1), p. 1-29.

Rézeau, J. (2001). Modélisation et médiation dans un environnement multimédiat: le cas de l'apprentissage de l'anglais en Histoire de l'art à l'université. Thèse de doctorat, Université Bordeaux 2, France.

Resnick, L.B. (1989). Knowing, Learning and Instruction, LEA, Hillsdale N.J.

Resnick, L.B. (1991). Shared cognition : thinking as social practice, in L.B. Resnick, J.M. Levine & S.D. Teasley (eds) : Perspectives on Socially Shared Cognition, American Psychological Association, Washington D.C.

Resnick, L. B. (1982). Syntax and semantics in learning to subtract. In C. P. Carpenter, J. M. Moser and T. A. Romberg (Eds.) Addition and subtraction : a cognitive perspective. Hillsdale, NJ : Erlbaum, p. 135-156.

Reuchlin, M. (1991). La psychologie cognitive, In H. Bloch, R. Chemama, A. Gallo, P. Leconte, J-F. Le Ny, J. Postel, S. Moscovic, M. Reuchlin, E. Vurpillot (Eds), Grand Dictionnire de la Psychologie (pp. 139-1410). Paris :Larousse.

Rich, E. (1983) : Users are individuals : individualizing user models. International Journal of Man-Machine Studies, Vol 18, pp 199-214.

Ripley, B.B. (1996). Pattern Recognition and Neural Networks. Cambridge Univ. Press.

Roblyer, M.D., Edwards, J. & Havriluk, M.A. (1997). Integrating educational technology into teaching. Merill, Upper Saddle River, NJ.

Rogers, Y. (1993). Coordinating Computer-Mediated Work. Computer Supported Cooperative Work (CSCW), 1, 295-315.

Rogers, Y. & Ellis, J. (1994). Distributed cognition : an alternative framework for analysing and explaining collaborative working, Journal of Information Technology, 9, 119-128.

Rojas S.A., Ramírez J., and Romero, O. (2002). Intelligent tutoring system for Mathematics using an evolutionary student model, Proceedings of the second IASTED conference on Artificial Intelligence and Applications, Málaga, España, Septiembre.

Roschelle, J. & Teasley, S.D. (1995). Construction of shared knowledge in collaborative problemsolving. In C. O'Malley (Ed.), Computer-supported collaborative learning. New York :Springer-Verlag.

Rosé, C. P., Di Eugenio, B., and Moore, J. D. (1999). A dialogue based tutoring system for basic electricity and electronics. In Proceedings of the Ninth World Conference on Artificial Intelligence in Education.

Rosé, C. P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., etWeinstein, A. (2001). Interactive Conceptual Tutoring in Atlas-Andes. Paper presented at the Proceedings of AIED2001.

Rosé, C. P. and Torrey, C. (2005): Interactivity and Expectation: Eliciting Learning Oriented Behavior with Tutorial Dialogue Systems. In: Proceedings of IFIP INTERACT05: Human-Computer Interaction 2005. pp. 323-336.

Rosenblatt, F. (1958). The Perceptron : A probabilistic Model for Information Storage and Organization in the Brain, Psychological Revue, n° 65, pp 386-408.

Ross, P. (1987). Intelligent tutoring systems. Journal of Computer Assisted Learning, 3, p.194-203.

Rouchier A. (1992). Logo : exemple générique ou cas particulier ? In B.Cornu (ed), L'ordinateur pour enseigner les Mathématiques, Nouvelle Encyclopédie Diderot, Presses Universitaires de France, Paris, pp. 299-328.

Russell, S. and Norvig, P. (1995). Artificial Intelligence : A Modern Approach. Prentice Hall.

Saint-Pièrre, L. (1991). La cognition. Pédagogie collégiale, déc. 1991, vol. 5, no 2, p. 15.

Salembier, P. (1996). Cognition(s) : Située, Distribuée, Socialement Partagée, etc., etc. Bulletin du LCPE, 1, Ecole Normale Supérieure, Paris.

Salembier, P. (2000). Cadres conceptuels et méthodologiques pour l'analyse, la modélisation et l'instrumentation des activités coopératives situées. GRIC-IRIT, UMR 5505 CNRS http://www.irit.fr/ACTIVITES/GRIC/personnel/salembier

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. Cognitive Systems Research, 1(4), 201-220.

Saporta, G. (1990). Probabilités analyse des données et statistiques. Paris: Editions Technip.

Scaife, M. & Rogers, Y. (1995). External cognition : how do graphical representations work ? Cognitive Science Research Papers 335, University of Sussex at Brighton.

Schegloff, E.A. (1991). Conversation analysis and socially shared cognition, in L.B.

Scribner S. (1986). Thinking in action : some characteristics of practical thought, in Sternberg R.J., Wagner R.K. eds., Practical intelligence : origins of competence in the everyday world, Cambridge Univ. Press, Cambridge, pp. 13- 30.

Self. J. (1992). Cognitive Diagnosis for Tutoring Systems. Actes de European Conference on Artificial Intelligence (ECAI), éd. par Neumann (B.), pp. 699–703.

Self, J. (1974). Student models in Computer-Aided Instruction. International Journal of Man-Machine Studies, 6, p261-276.

Self, J. (1994). Formal Approaches to Student Modelling. Actes des journées Student Modelling : The Key to Individualized Knowledge- Based Instruction, éd. par Greer (Jim E.) et Mac Calla (Gordon I.). pp. 295–354. – Springer-Verlag.

Self, J. (1988). Bypassing the Intractable Problem of Student Modelling. In C. Frasson and G. Gauthier (Eds.), Intelligent Tutoring Systems : At the crossroads of AI and Education.

Self, J. (1993). Model-based Cognitive Diagnosis. User Modeling and User-Adapted Interaction, 3, pp. 89-106.

Self J. (1987). Students Models : What Use Are They? Actes de IFIP/TC3, pp. 73-85.

Scron, X. (1991). Du diagnostic neuropsychologique à l'évaluation cognitive et pragmatique des troubles. Revue Suisse de Psychologie, 50, 186-197.

Sethi, I.K. and Kain, A.K. 1991. Artificial Neural Networks and Statistical Pattern Recognition. Machine Intelligence and Pattern Recognition Series, vol 11. North Holland.

Shute, V. (1996). Student modeling : Cognitive approaches. Troisième conférence internationale sur les systèmes tutoriels intelligents. Montréal.

Siegler, R. S., & McGilly, K. (1989). Strategy choices in children's time-telling. In I. Levin and D. Zakay (Eds.) *Time and human cognition : A life span perspective*, (pp. 185-218). The Netherlands : Elsevier Science Publishers.

Site Web. (2002). Common errors in written subtraction. Teaching and Learning about Whole Numbers. http ://online.edfac.unimelb.edu.au/485129/wnproj/subtract/errors.htm

Skinner, B.H. (1963). L'avenir des machines à enseigner. Psychologie française, 8 (3), 170-180.

Sleeman, D. & al. (1989). Moore Studies of diagnosis and remediation with high school algebra students. Cognitive Science, 13, p. 551-568.

Sleeman, D. and Brown, J. S., (Eds.) (1982). Intellligent Tutoring Systems. New York : Academic Press.

Sleeman, D. (1982). Assessing aspects of competence in basic algebra. In Sleeman and Burton (Eds) Intelligent Tutoring Systems, Academic Press, London.

Smith, J.B. (1994). Collective Intelligence in Computer-Based Collaboration. Hillsdale, NJ : Lawrence Erlbaum.

Smyth, P., Heckman, D. and Jordan M. (1996). Probabiliste Independence Networks for HMM. M.I.T, A.I Memo n° 1565.

Smyth, P. (1998). Belief networks, hidden Markov models, and Markov random fields : a unifying view, Pattern Recognition Letters.

Sowa, J.F. (1991). Toward the Expressive Power of Natural Language, in Principles of Semantic Networks : Exploration in the Representation of Knowledge, J.F. Sowa (éd.), chapitre 5, Morgan Kaufmann Publishing.

Sowa, J.F. (1984). Conceptual Structures. Information Processing in Mind and Machine, Addison-Wesley Publishing.

Sparks, R. Dooley, S., Meiskey, L. & Blumenthal, R. (1999). The LEAP authoring tool: supporting complex courseware authoring through reuse, rapid

prototyping, and interactive visualizations. Int. J. of Artificial Intelligence in Education (this issue).

Sperber, D. & Wilson, D. (1986). Relevance. Oxford : Blackwell.

Staub, F. C., Stebler, R., Reusser, K., & Pauli, C. (1994). Improving Understanding and Solving of Math Story Problems Through Collaborative Use of a Computer Tool (HERON). Paper prensented at the Annual Meeting of the American Educational Research Association, New Orleans, 4-8 Avril.

Stevens. A, Collins, A. & Godin, S.E. (1982). Misconceptions in students' understanding. In Sleeman, D. and Brown, J.S. (Eds.) Intelligent Tutoring Systems, p. 13-24.

Sturges, P.T. (1978). Delay of informative feedback in computer-assisted testing. Journal of educational psychology, 70, 3, pp. 378-387.

Suchman, L.A. (1986). What is a plan? ISL Technical Note 20708, April.

Suchman, L. (1987) Plans and situated actions: the problem of human/machine communication, Cambridge University Press.

Suermondt, H. J. and Cooper, G. F. (1991). A combination of exact algorithms for inference on bayesian belief networks. int. J. Approximate Reasoning, 5 :521–542.

Swartz, M., L. (1992). Introduction to Intelligent Tutoring Systems for Foreign Language Learning. The Bridge to International Communication, éd. par Swartz (M. L.) et Yazdani (M.). pp. 1–6, Springer-Verlag.

Sweller, J. et Chandler, P. (1991) Evidence for cognitive load theory, Cognition and Instruction, 8, 351-362.

Taatgen, N. A. & Anderson, J. R. (2002). Why do children learn to say «broke»? A model of learning the past tense without feedback, Cognition, 86 (2), 123-155.

Taatgen, N., Lebiere, C. & Anderson, J. (2004). Modeling paradigms in ACT-R. Ch.1. Carnegie Mellon University

Tardif, J. (1992) Pour un enseignement stratégique : l'apport de la psychologie cognitive. Éditions Logiques, Montréal.

Tardif, M., Gauthier, C. (1996). L'enseignant comme « acteur rationnel » : quelle rationalité, quel savoir, quel jugement ? in : L. Paquay, M. Altet, É. Charlier, P. Perrenoud (éds) Former des enseignants professionnels (13-26). Bruxelles, De Boeck.

Tardif, J. (1998). Intégrer les nouvelles technologies de l'information, ESF éditeur. p.44.

Tardif, J. (1999). Le transfert des apprentissages. Montréal : Éditions Logiques.

Tchetagni, J., Nkambou, R., etBourdeau, J. (2005a). Supporting Student Reflection in an Intelligent Tutoring System for Logic Programming. Paper

presented at the International Conference on Artificial Intelligence in Education (To appear).

Tchétagni, J. (2005). Une approche pro-pédagogique du diagnostic cognitif dans les STI: conception, formalisation et implantation. Thèse de doctorat, Université du Québec à Montréal, Canada.

Tchétagni, J., Nkambou, R. et Bourdeau, J. (2007). « Explicit Reflection in Prolog-Tutor ». International Journal on Artificial Intelligence in Education, vol. 17, no 2, p. 169-217.

Tessmer, M. & Richey, R. (1997). The role of context in learning and instructional design. Educational Technology Research and Development (ETR&D), 45 (2), 29-45.

Thayse A. et al. (1988). Approche logique de l'Intelligence Artificielle. Vol 1. Paris Dunod.

Theureau, J. (1999). Théories et méthodes d'analyse de l'action et ingénierie. Cours 4 UTC/SHT.

Toma, T. (1996). L'enseignant face au multimédia, Paris : Pedagog International.

Towell, G. (1991). Symbolic Knowledge and Neural Networks : Insertion, Refinement and Extraction. Ph.D. Thesis, Computer Science Dept., University of Wisconsin-Madison, U.S.A.

Turenne, N. (2000). Apprentissage statistique pour l'extraction de concepts à partir de textes. Application au filtrage d'informations textuelles. Thèse en informatique de l'Université Louis-Pasteur. France.

UML (2008). Unified Modeling Language, Objet Mamagement Group, http://www.uml.org/

Utgoff, P.E. (1989). Incremental Induction of Decision Trees. In : Machine Learning, 4, pp.161-186. Kluwer Academic Publishers, Boston, MA - U.S.A..

VanLehn K (1982). Bugs are not enough : Empirical studies of bugs, impasses and repairs in procedural skills. Journal of Mathematical Behavior 3, pp. 3-72.

VanLehn, K. (1983). The representation of procedures in repair theory. In H.P. Ginsberg, ed. The Development of mathematical thinking. Hillsdale, NJ : Erlbaum.

VanLehn, K. (1987). Learning One Subprocedure per Lesson. Artificial Intelligence, vol. 31, n° 1, pp. 1–40.

VanLehn, K. (1988). Toward a theory of impasse-driven learning. In H. Mandl & A. Lesgold (Eds.) Learning Issues for Intelligent Tutoring Systems, pp. 19-41. New York, NY : Springer.

VanLehn, K., Ball, W. & Kowalski, B. (1989). Non-LIFO execution of cognitive procedures. Cognitive Science, 13, pp. 415-465.

VanLehn, K. (1990). Mind Bugs : The origins of procedural misconceptions. Cambrigne, MA : The MIT Press.

VanLehn, K., & Martin, J. (1998). Evaluation of an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence in Education, 8*(2).

VanLehn, K., & Niu, Z. (2001). Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. International Journal of Artificial Intelligence in Education, 12, in press.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A. & Shelby, R. (2005). The Andes Physics Tutoring System: Lessons Learned. *International Journal of Artificial Intelligence in Education, 15 (3).*

VanLehn, K., Jordan, P. W., Rosé, C. P., Bhembe, D., B¨ottner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Ringenberg, M., Roque, A., Siler, S., Srivastava, R., and Wilson, R. (2002). The architecture of Why2-Atlas: A coach for qualitative physics essay writing, in Proc. Intelligent Tutoring Systems Conference.

Van Joolingen W.R. et De Jong T. (2003). "SIMQUEST : Authoring Educational Simulations", in Authoring tools for advanced technology educational software Eds T Murray, S Blessing and S Ainsworth, Dordrecht.

Van Nieuwenhoven, C. (1999). Le comptage. Vers la construction du nombre. Bruxelles, De Boeck.

Van Nieuwenhoven, C. (2001). Pourquoi tu joues? Rôle du jeu dans le développement de l'enfant. Louvain-la-Neuve, Presses Universitaires de Louvain.

Vera A.H. & Simon H.A. (1993). Situated action: a symbolic interpretation, Cognitive Science, 17, 7-48.

Vergnaud, G. (1982). Cognitive and developmental psychology and research in mathematics education: Some theoretical and methodological issues. For the Learning of Mathematics, vol. 3, n° 2, p. 31-41.

Vergnaud, G. (1991). La théorie des champs conceptuels, *Recherches en didactique des mathématiques*, 10 (2.3), 133–170.

Vernant, D. Modèle projectif et structure actionnelle du dialogue informatif. In Du dialogue, Recherches sur la philosophie du langage, Vrin éd., Paris, n°14, p. 295-314, 1992.

Visetti Y.M. (1989). Compte rendu de "Plans and Situated Actions" de L. SUCHMAN, Intellectica, 1, n° 7, pp. 67- 96.

Vygotski, L. (1978). Mind in Society: The development of higher psychological processes. Cambridge, MA : Harvard University Press.

Vygotski, Lev. (1997). Pensée et Langage, Paris : La Dispute, 3e éd. (édition originale en russe publiée en 1934).

Webb, G. (1993). Feature Based Modelling. Actes de Artificial Intelligence and Education (AI&Ed), pp. 497–504.

Weil-Barais, Annick (dir.). (1993). L'homme cognitif, Paris : PUF (4e édition mise à jour, 1998).

Wenger, M.J. & David, G.P. (1996). Comprehension and retention of nonlinear text : considerations of working memory and material-appropriate processing. The American Journal of Psychology, 109, 93-130.

Wenger E. & Lave, J. (1990). Situated Learning : Legitimate Peripheral Participation, New York NY : Cambridge University Press.

Wenger, E. (1998). Communities of practice : Learning, meaning, and identity. Cambridge : Cambridge University

Wenger, E., McDermott, R. & Snyder, W. M. (2002). Cultivating communities of practice. Boston : Harvard Business School Press.

Wenger, E. (1987). Artificial Intelligence and Tutoring Systems. Morgan Kaufman.

Wertsch, J. (1985). Vygotsky and the social formation of mind. Cambridge, MA : Harvard University.

Whittaker, J. (1990). Graphical Models in Applied Multivariate Statistics. Wiley.

Widrow, B.& Lehr, M. (1990). 30 Years of Adaptive Neural Networks : Perceptron, Madaline, and Back-Propagation. Proceedings of the IEEE, New York, Vol.78, N.9, pp.1415-1441. September.

Widrow, B. and Hoff, M.E. (1960). Adaptative Switching Circuit. IRE Western Electric Show and Convension Record, part 4 :96-104.

Winograd T. & Flores, F. (1986). Understanding computers and cognition, Ablex, Norwood.

Wong, W.K. & Chan, T.W. (1997). A Multimedia authoring system for crafting topic hierarchy, learning strategies, and intelligent models. International J. of Artificial Intelligence in Education, Vol. 8, No 1, pp. 71-96.

Wu Yao Kuang R., « La robustesse des systemes auteurs multimedias : contribution théorique et mise en oeuvre », Thèse de doctorat de l'Université Paris 8, 2000.

Yazdani, M. (1987). ITS : an Overview. Lawler R., Yazdani M. (eds.) Artificial Intelligence an education : learning environments and tutoring system, vol. 1, Norwood, NJ : Ablex, pp. 183-201.

Young, R. (1976). Strategies and the structures of cognitive skill. In g. Underwood : Strategies of Information Processing. Academic Press, London.

Young, R. & O'Shea, T. (1981). Errors in children's subtraction. Cognitive Science, 5, 153-177.

Zadeh, L. A. (1965). Fuzzy Sets, Information and Control, Vol. 8, pp. 338-353.

Zapata-Rivera, J.-D., etGreer, J. (2003). Analyzing Student Reflection in The Learning Game. Paper presented at the Proceedings of AIED2003, Australia.

ANNEXES

ANNEXE A : LISTE DES BUGS

1. Quand une colonne comprend un 1 qui se voit changé en un 0 par un emprunt, l'apprenant écrit 0 comme réponse à cette colonne.

```
 9 1 4
-4 8 6
-------
 5 0 8
```

2. Quand une colonne comprend un 1 qui se voit changé en un 0 par un emprunt, l'apprenant écrit le chiffre du bas comme réponse à cette colonne.

```
 5 1 2
-1 3 6
-------
 4 3 6
```

3. Si, au départ de l'opération, une colonne comprend des 1 en haut comme en bas et qu'il y effectue un emprunt, l'apprenant écrit 1 comme réponse à cette colonne.

```
 8 1 2
-5 1 8
-------
 3 1 4
```

4. Quand il emprunte d'un 1, l'apprenant considère ce 1 comme s'il était 10 et le décrémente à 9.

```
 3 1 6
-1 3 9
-------
 2 6 7
```

5. Quand il emprunte d'un 1, l'apprenant change ce 1 en 10 au lieu de 0 .

```
 4 1 4
-2 7 7
-------
 2 3 7
```

6. Quand un emprunt est causé par un 1, l'apprenant change le 1 en 10 au lieu de lui ajouter 10 .

```
 7 1
- 3 8
--------
 3 2
```

7. Quand il emprunte d'un 1, l'apprenant traite ce 1 comme un 0, le changeant en 9 et décrémentant le nombre à sa gauche.

```
 3 1 3
- 1 5 9
--------
 1 4 4
```

8. Au lieu de décrémenter un 1, l'apprenant le change en un 11.

```
 3 1 4
   - 6
--------
 3 11 8
```

9. Au lieu d'emprunter sur un 0 au-dessus d'un autre 0, l'apprenant ne change pas le premier 0, mais décrémente plutôt la colonne suivante à gauche.

```
 8 0 2
- 3 0 4
--------
 3 0 8
```

10. Au lieu d'emprunter sur un 0, l'apprenant change le 0, mais ne décrémente aucune colonne à gauche.

```
 3 0 7
- 1 0 8
--------
 2 1 9
```

11. Quand il emprunte sur un 0 qui surmonte un autre 0, l'apprenant saute par-dessus le premier 0 et diminue le chiffre suivant à gauche.

```
 3 0 5
- 1 0 7
--------
 1 0 8
```

12. Quand il emprunte sur un 0, l'apprenant change le 0 en 10 et ne décrémente aucun chiffre à gauche. Cependant, si le 10 ainsi créé est au-dessus d'un 0, l'apprenant fait une retenue au lieu d'essayer d'écrire 10 dans la réponse.

```
 6 0 4
- 2 0 5
--------
 5 0 9
```

13. Quand il emprunte sur un 0 surmontant un vide, l'apprenant saute à la colonne suivante pour décrémenter.

```
  4 0 2
  - 6
--------
  3 0 6
```

14. Quand il emprunte sur un 0 au-dessus d'un vide, l'apprenant considère la colonne où il y a ce 0 comme si elle n'existait pas.

```
  5 0 5
  - 7
--------
    4 8
```

15. L'apprenant ne décrémente pas une colonne qui a un chiffre au-dessus d'un vide.

```
  3 2 9
  - 9 7
--------
  3 3 2
```

16. Quand la colonne à l'extrême gauche de l'exercice comprend un 1 au-dessus d'un vide, l'apprenant ignore la colonne.

```
  1 4 3
  - 2 2
--------
    2 1
```

17. Quand le nombre du bas comporte moins de chiffre que celui du haut, l'apprenant arrête l'exercice à la colonne où s'arrête le nombre du bas.

```
  4 3 9
  - 4
--------
    5
```

18. Quand il y a des vides dans le nombre du bas l'apprenant soustrait le chiffre le plus à gauche du nombre du bas des chiffres du haut des colonnes où se trouvent ces vide.

```
  4 3 6 9
  - 2 2
----------
  2 1 4 7
```

19. Quand il y a des vides dans le nombre du bas, l'apprenant soustrait 1 des chiffres du haut.

```
  5 4 8
  - 2
--------
  4 3 6
```

20. L'apprenant ignore toute colonne de la forme 0 sur un vide.

```
  9 0 7
```

```
  - 5
--------
  9 2
```

21. Lorsque l'apprenant écrit un exercice, il aligne les colonnes de gauche à droite au lieu de droite à gauche.

```
  6 3
  - 2
-------
  4
```

22. L'apprenant emprunte toujours de la colonne à l'extrême gauche plutôt que de celle immédiatement à gauche.

```
  7 3 3
-2 1 6
--------
  4 2 7
```

23. L'apprenant emprunte la différence entre le chiffre du haut et celui du bas de la colonne où il opère. En d'autres mots, il emprunte juste assez pour faire la soustraction dans cette colonne où il obtient toujours un 0.

```
  8 6
-2 9
------
  3 0
```

24. L'apprenant arrête l'exercice dès qu'un emprunt est requis.

```
  8 3 3
-2 6 2
--------
    1
```

25. Quand un emprunt est requis, l'apprenant saute simplement par-dessus la colonne et va à la colonne suivante.

```
  4 2 5
-2 8 3
--------
  2 2
```

26. Quand il emprunte, l'apprenant décrémente correctement, puis, colonne par colonne, il soustrait le plus petit chiffre du plus grand comme s'il n'avait pas emprunté.

```
  7 3
- 2 4
-------
  4 1
```

27. Quand une colonne demande un emprunt, l'apprenant décrémente correctement, mais écrit 0 comme réponse.

```
 6 5
-4 8
-------
 1 0
```

28. Quand il emprunte, l'apprenant saute les colonnes dans lesquelles le nombre du haut et du bas sont égaux.

```
 9 2 3
-4 2 7
-------
 4 0 6
```

29. L'apprenant pense que N – 0 = 0.

```
 5 7
-2 0
-------
 3 0
```

30. Quand une colonne est de la forme N-N, l'apprenant écrit 9 comme réponse et décrémente la colonne suivante à gauche bien que l'emprunt ne soit pas nécessaire .

```
 9 4
-3 4
-------
 5 9
```

31. L'apprenant change le nombre pour lequel il faut emprunter en 10 au lieu de lui ajouter 10.

```
 8 3
-2 9
-------
 5 1
```

32. Quand il emprunte, l'apprenant décrémente la quantité en bas de la colonne extrême, c'est-à-dire le résultat de la colonne extrême.

```
 8 7 4
-2 5 9
-------
 4 2 5
```

33. Quand il emprunte, l'apprenant essaie de trouver une colonne dans laquelle le nombre du haut est plus petit que celui du bas. S'il y en a un, il le décrémente. Sinon, il emprunte correctement.

```
 9 3 8 3
-3 5 6 6
---------
 5 6 2 7
```

34. Quand il emprunte sur un 0, l'apprenant saute le 0 pour emprunter de la colonne suivante. Si cela le force à emprunter deux fois, il diminue le même chiffre à deux reprises.

```
  9 0 4
- 2 3 7
-------
  5 7 7
```

35. Quand il emprunte sur un 0 et que cet emprunt est causé par un autre 0, l'apprenant change le zéro de droite en 9 au lieu de le changer en 10.

```
  6 0 0
- 1 4 2
-------
  4 5 7
```

36. Quand il doit emprunter sur un ou des 0, l'apprenant n'ajoute pas 10 à la colonne où il le devrait, mais ajoute plutôt 10 moins le nombre de 0 par-dessus lesquels il a emprunté.

```
  3 0 0 8
- 1 3 5 9
---------
  1 6 4 7
```

37. Quand il emprunte sur des 0, l'apprenant change le 0 le plus à droite en un 9, le 0 suivant en un 8, etc.

```
  8 0 0 2
- 1 7 1 4
---------
  6 1 8 8
```

38. Au lieu d'emprunter sur un 0, l'apprenant arrête de faire l'exercice.

```
  8 0 3 8
- 2 6 6 5
---------
        3
```

39. La première colonne qui demande un emprunt est décrémentée avant de soustraire.

```
  8 3 2
- 2 6 5
-------
  5 6 6
```

40. Au lieu d'emprunter sur les 0, l'apprenant les change tous en 9, mais ne va pas emprunter de la colonne à leur gauche.

```
  3 0 0 6
- 1 8 0 7
---------
  2 1 9 9
```

41. Quand il emprunte sur deux 0 ou plus, l'apprenant change le 0 le plus à gauche en 9, mais transforme les autres en 10.

```
 1 0 0 3
 - 9 5 8
 ---------
 1 0 5 5
```

42. Quand il doit emprunter sur un 0 et que cet emprunt est causé par un autre 0, l'apprenant n'emprunte pas. Il écrit plutôt le nombre du bas comme réponse à la colonne. Il emprunte correctement dans la colonne suivante dans les autres circonstances.

```
  4 0 0
 - 2 4 8
 -------
  1 6 8
```

43. Quand il doit emprunter sur un 0 et que cet emprunt est causé par un autre 0, l'apprenant change le 0 duquel il emprunte en un 10 au lieu d'un 9.

```
  7 0 0
 - 2 5 8
 -------
  4 5 2
```

44. Au lieu d'emprunter de plusieurs 0 à la suite, l'apprenant ajoute 10 à la colonne où il effectue son calcul, mais ne change aucune colonne à gauche.

```
 4 0 0 4
     - 9
 ---------
 4 0 0 5
```

45. Quand il emprunte d'une colonne de la forme 0-N, l'apprenant décrémente le N au lieu du zéro.

```
  6 0 8
 - 2 4 9
 -------
  3 7 9
```

46. Quand il emprunte sur un 0, l'apprenant change le 0 en 10 et ne décrémente aucun chiffre à gauche.

```
  6 0 4
 - 2 3 5
 -------
  4 7 9
```

47. Au lieu d'emprunter sur un 0, l'apprenant change le 0 en 9, mais ne continue pas pour emprunter de la colonne de gauche. Cependant, si le

chiffre à gauche du 0 est au-dessus d'un vide, l'apprenant procède alors de manière correcte.

```
  3 0 6
- 1 8 7
-------
  2 1 9
```

48. Quand il emprunte sur un 0, l'apprenant change le 0 en 10 au lieu de 9. Cependant, si ce 0 est au-dessus d'un autre 0, l'apprenant fait une retenue au lieu d'écrire 10 comme réponse à cette colonne.

```
  5 0 6
- 3 0 8
--------
  2 0 8
```

49. L'apprenant n'emprunte pas sur un 0, soustrayant plutôt le plus petit chiffre du plus grand dans la colonne où l'emprunt serait requis.

```
  3 0 6
- 1 4 8
-------
  1 6 2
```

50. Au lieu d'emprunter de 0, l'apprenant ajoute 10 à la colonne où il effectue son calcul, mais ne décrémente aucune colonne à gauche.

```
  4 0 4
- 1 8 7
-------
  2 2 7
```

51. Au lieu d'emprunter sur un 0, l'apprenant change ce 0 en 9, mais ne continue pas pour aller emprunter de la colonne à gauche.

```
  3 0 6
- 1 8 7
-------
  2 1 9
```

52. L'apprenant n'emprunte pas d'un 0, écrivant plutôt 0 comme réponse à la colonne exigeant l'emprunt.

```
  7 0 2
- 3 4 8
--------
  3 6 0
```

53. Au lieu d'emprunter sur un 0, l'apprenant ne change pas ce 0, mais décrémente plutôt la colonne suivante à gauche. De plus, s'il lui faut emprunter d'une colonne commençant par un 0 qui devrait alors être modifié, l'apprenant écrit 0 comme réponse à cette colonne.

```
  8 0 2
- 3 2 4
--------
  5 0 8
```

54. Au lieu d'emprunter sur un 0, l'apprenant ne change pas ce 0, mais décrémente plutôt la colonne suivante à gauche. De plus, s'il lui faut emprunter d'une colonne commençant par un 0 qui devrait alors être modifié, l'apprenant écrit le chiffre du bas comme réponse à cette colonne.

```
  8 0 2
- 3 2 4
--------
  5 2 8
```

55. Au lieu d'emprunter sur un 0, l'apprenant ne change pas ce 0, mais décrémente plutôt la colonne suivante à gauche. De plus, s'il lui faut emprunter d'une colonne commençant par un 0 qui devrait alors être modifié, l'apprenant ajoute 10 au 0 mais ne décrémente rien.

```
  8 0 2
- 3 2 4
--------
  5 8 8
```

56. L'apprenant n'emprunte pas. Pour les colonnes de la forme 0-N, il écrit N comme réponse et pour celles du type petit nombre – (moins) grand nombre, il met 0.

```
  3 0 4
- 1 7 9
--------
  2 7 0
```

57. Quand un emprunt est causé par un 0, l'apprenant n'ajoute pas correctement 10. Il ajoute plutôt 10 plus le chiffre dans la colonne suivante à gauche.

```
  5 0
- 3 8
------
  1 7
```

58. Quand l'apprenant rencontre une colonne de la forme 0-N, il n'emprunte pas, mais écrit plutôt 0 comme réponse dans cette colonne.

```
  4 0
- 2 1
------
  2 0
```

59. Quand l'apprenant rencontre une colonne de la forme 0-N, il n'emprunte pas, mais écrit plutôt N comme réponse dans cette colonne.

```
 8 0
-2 7
------
 6 7
```

60. Quand un emprunt est causé par un 0, l'apprenant n'emprunte pas, mais traite le 0 comme s'il était un 9.

```
 3 0
 - 4
------
 3 9
```

61. Quand un emprunt est causé par un 0, l'apprenant lui ajoute 10 correctement, mais ne change aucune colonne à gauche.

```
 4 0
-2 7
------
 2 3
```

62. L'apprenant n'emprunte qu'une fois par exercice. Ensuite, il soustrait le plus petit nombre du plus grand dans chaque colonne, peu importe leur position dans la colonne.

```
 7 1 2 7
-2 3 8 9
----------
 5 2 7 8
```

63. Quand il y a deux emprunts à la suite, l'apprenant fait le premier emprunt correctement, mais dans le second, il ne décrémente pas (mais ajoute 10 correctement).

```
 1 4 3
 - 8 8
--------
 1 5 5
```

64. L'apprenant n'emprunte qu'une fois par exercice. Si un autre emprunt est nécessaire, l'apprenant ajoute le 10 correctement, mais ne décrémente pas. S'il y a emprunt sur 0, il change le 0 en 9, mais ne décrémente pas le chiffre à gauche du 0.

```
 4 0 8
-2 3 9
--------
 2 6 9
```

65. Quand il emprunte d'une colonne dans laquelle le chiffre du haut est plus petit que celui du bas, l'apprenant incrémente au lieu de décrémenter.

```
  8 3 3
- 2 7 7
--------
  5 7 6
```

66. Quand il emprunte, l'apprenant ajoute 10 correctement, mais ne change aucune colonne à gauche.

```
  6 2
- 4 4
------
  2 8
```

67. Quand une colonne est de la forme 1-N, l'apprenant écrit 1 comme réponse à cette colonne.

```
  5 1
- 2 7
------
  3 1
```

68. Chaque fois qu'une colonne présente le même chiffre en haut et en bas, l'apprenant écrit ce chiffre comme réponse.

```
  8 3
- 1 3
------
  7 3
```

69. L'apprenant ne décrémente pas un chiffre qui est au-dessus d'un vide.

```
  3 4 7
    - 9
--------
  3 4 8
```

70. Quand il emprunte d'un 0 au-dessus d'un vide, l'apprenant augmente le 0 au lieu de le diminuer.

```
  4 0 2
    - 6
--------
  4 1 6
```

71. S'il a déjà emprunté d'une colonne de la forme N-9, l'apprenant, lorsqu'il arrive à cette colonne, soustrait 1 au lieu de soustraire 9.

```
  8 3 4
- 7 9 6
--------
  1 2 7
```

72. Si l'apprenant emprunte d'une colonne de la forme N-N, il écrit ensuite 1 comme réponse à cette colonne.

```
 9 4 4
-3 4 8
-------
 6 1 6
```

73. Une fois que l'apprenant a emprunté, il continue à le faire pour chacune des colonnes qui restent dans l'exercice.

```
 4 8 8
-2 2 9
-------
 1 1 5 9
```

74. L'apprenant n'emprunte pas ; dans chaque colonne, il soustrait le plus petit chiffre du plus grand, peu importe leurs positions respectives.

```
 8 1
-3 8
------
 5 7
```

75. L'apprenant soustrait toujours le chiffre du haut de celui du bas. Si le chiffre du bas est le plus petit, il décrémente le chiffre du haut et ajoute 10 à celui du bas avant de soustraire. Cependant, si le chiffre du bas est 0, il écrit le chiffre du haut comme réponse. Si le chiffre du haut dépasse celui du bas par 1, il écrit alors 9 comme réponse.

```
 4 7 2 3
-3 0 8 5
---------
 9 7 4 2
```

76. L'apprenant ne soustrait pas ; il copie plutôt les chiffres de l'exercice dans l'espace réservé à la réponse, retenant le chiffre le plus à gauche du nombre du haut et les autres chiffres du nombre du bas.

```
 6 4 8
-2 3 1
-------
 6 3 1
```

77. L'apprenant n'emprunte jamais, écrivant plutôt 0 comme réponse à toute colonne exigeant un emprunt.

```
 4 2
-1 6
------
 3 0
```

78. L'apprenant additionnc au lieu de soustraire, mais il soustrait la retenue au lieu de l'ajouter.

```
 5 4
-3 8
------
 7 2
```

79. Au lieu de décrémenter, l'apprenant ajoute 1 à la colonne de laquelle il emprunte, effectuant au besoin une retenue sur la colonne à gauche.

```
 8 6 3
-1 3 4
--------
 7 4 9
```

80. L'apprenant additionne au lieu de soustraire.

```
 5 0 2
-3 2 4
--------
 8 2 6
```

81. L'apprenant additionne au lieu de soustraire. Si une retenue est nécessaire, il n'additionne pas le chiffre retenu.

```
 5 2 7
-3 2 4
--------
 8 4 1
```

82. Quand il emprunte d'une colonne dans laquelle le chiffre du haut est plus petit que celui du bas, l'apprenant ajoute 10 au chiffre du haut, décrémente la colonne d'où il emprunte et emprunte de la colonne suivante à gauche. De plus, l'apprenant évite toute colonne comportant un 0 sur un 0 ou sur un vide dans le processus d'emprunt.

```
 1 8 3
 -9 5
------
 9 7
```

83. L'apprenant emprunte pour chacune des colonnes, que cela soit ou non nécessaire.

```
 4 8 8
-2 2 9
--------
 1 1 5 9
```

84. L'apprenant ne décrémente pas une colonne si le chiffre du haut est plus petit que celui du bas.

```
 7 3 2
-4 8 4
------
 2 5 8
```

85. L'apprenant ne décrémente pas une colonne sauf si le nombre du bas est plus petit que celui du haut.

```
 7 3 2
-4 8 4
------
 2 5 8
```

86. l'apprenant emprunte de la rangée du bas au lieu de celle du haut.

```
 8 2 7
-2 0 8
------
 8 3 9
```

87. Quand il emprunte, l'apprenant décrémente le plus grand nombre dans la colonne, peu importe qu'il soit en haut ou en bas.

```
 8 7 2
-2 9 4
------
 5 9 8
```

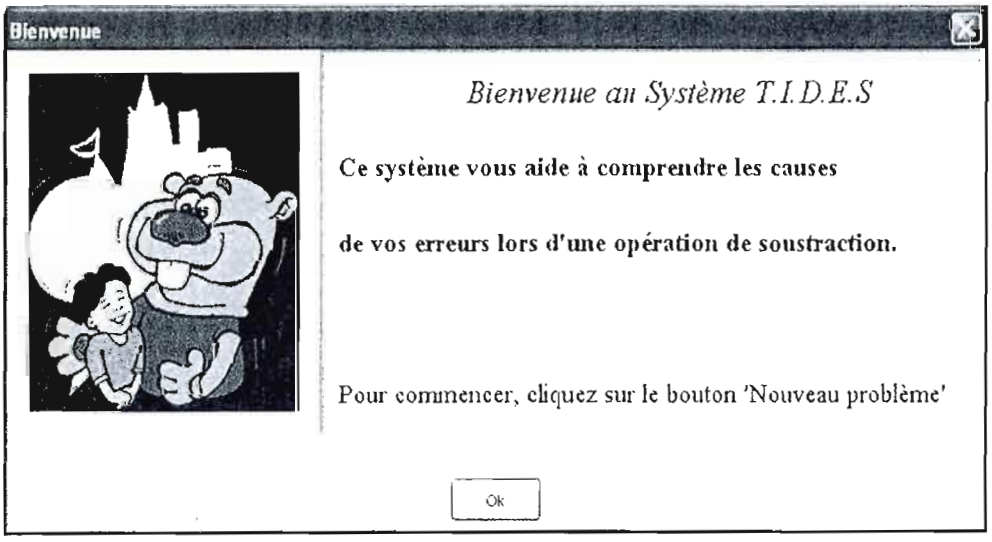
88. Au lieu de décrémenter, l'apprenant ajoute 1 à la colonne de laquelle il emprunte. Si le résultat atteint 10, l'apprenant ne fait pas la retenue, mais écrit simplement les deux chiffres dans le même espace.

```
 8 9 3
-1 0 4
------
 7109
```

ANNEXE B : UTILISATION DU SYSTÈME TIDES : SESSION D'APRENTISSAGE

Manuel d'utilisation du système TIDES

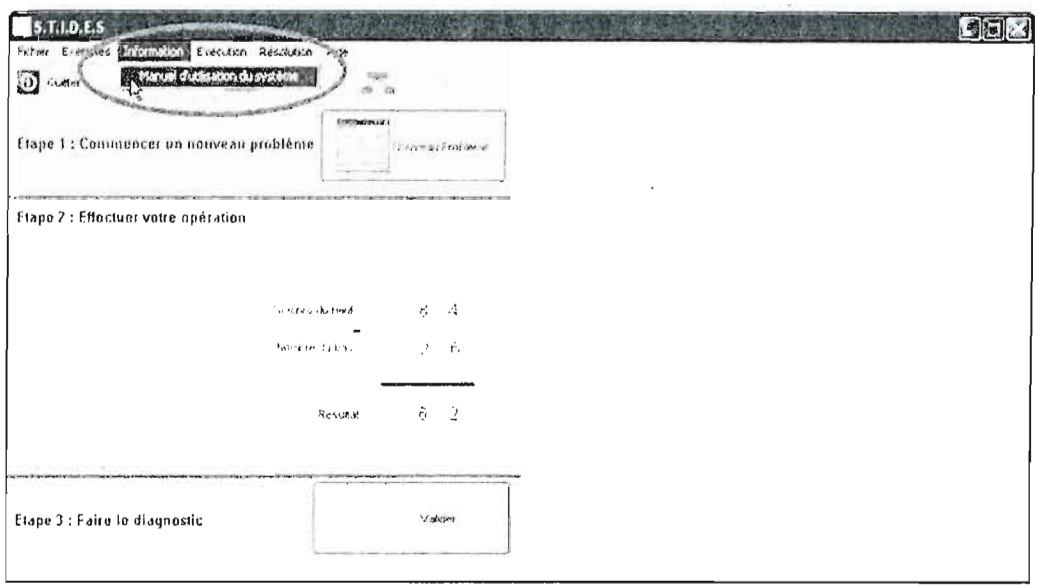
Le présent manuel a pour but de vous faciliter l'utilisation du système TIDES. Les schémas d'explication exposent et structurent la succession des interventions possibles du système, pour la justification d'une étape de résolution. Ils ont pour rôle d'assurer la cohérence du dialogue explicatif entre le système et son utilisateur.
Le système commence par souhaiter la bienvenue à l'utilisateur et l'invite à bien lire les instructions pour connaître son fonctionnement.



Message de bienvenue du système TIDES

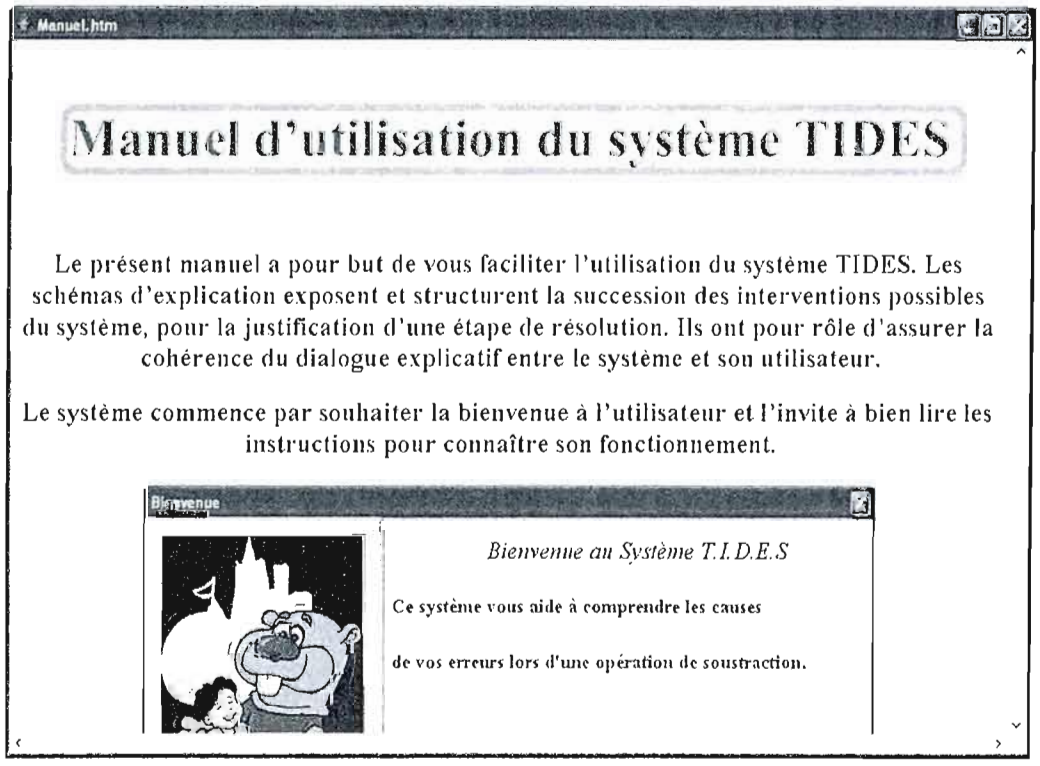
Un clic sur « OK » fait apparaître l'écran principal du système. Cet écran contient un certain nombre d'icônes dont l'utilisateur doit se servir au cours de son utilisation du système TIDES.

Par exemple :
L'utilisateur peut consulter le « manuel d'utilisation du système » pour savoir comment le système fonctionne. Pour ce faire, il doit cliquer sur le menu « Information » puis, il doit choisir le sous-menu « manuel » comme le montre la figure suivante :



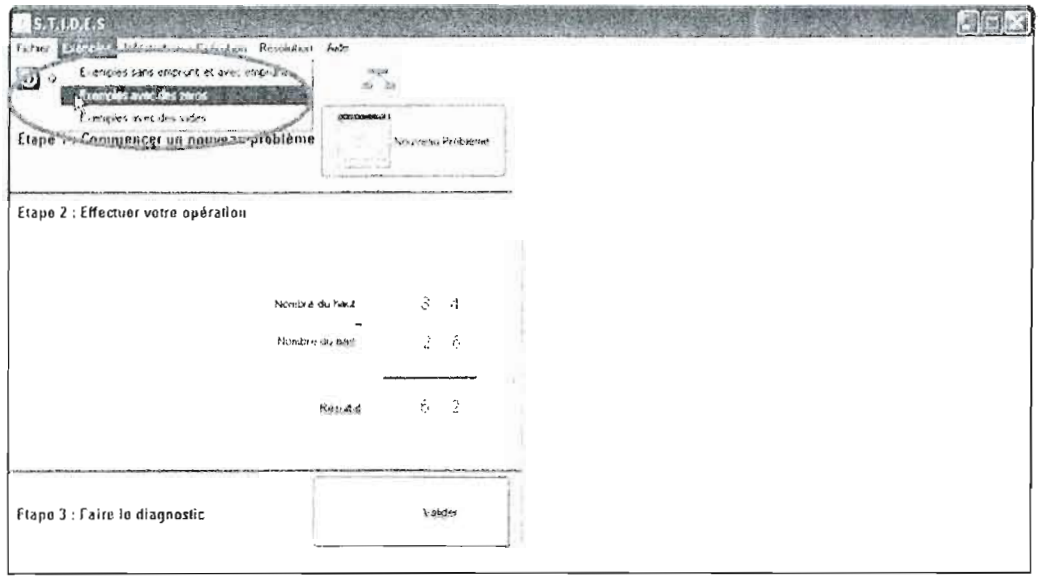
Choix du manuel d'utilisation du système TIDES

Et le manuel s'affiche :



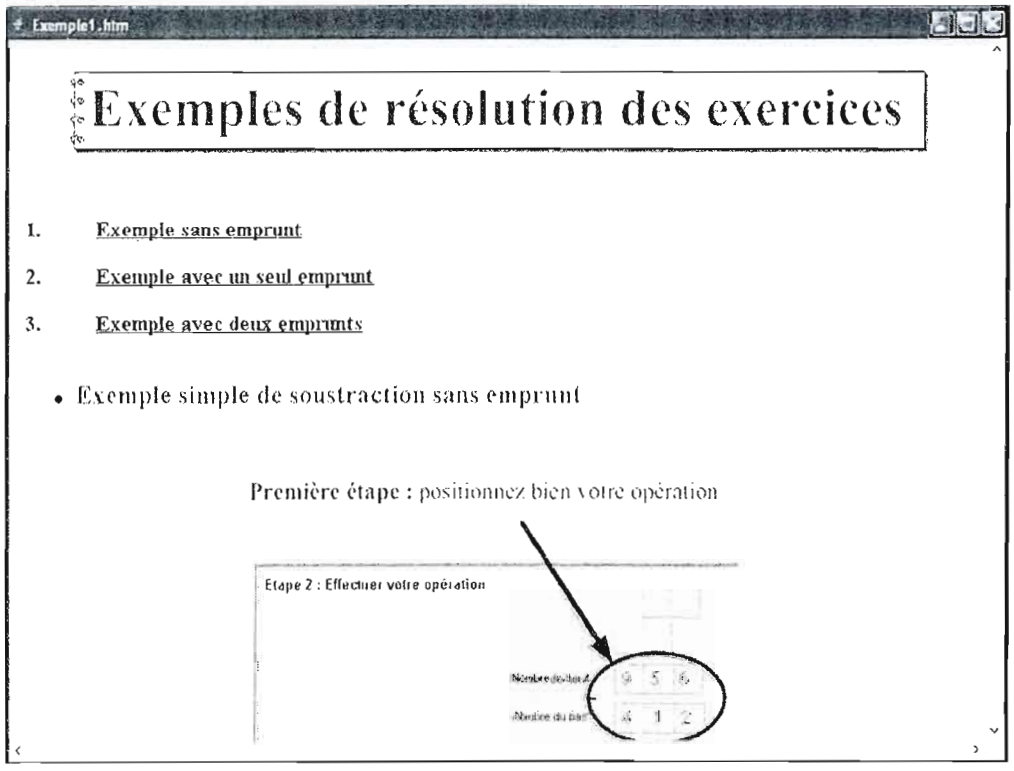
Manuel d'utilisation du système TIDES

L'utilisateur peut aussi consulter certains exemples types pour bien comprendre comment le système traite les problèmes :



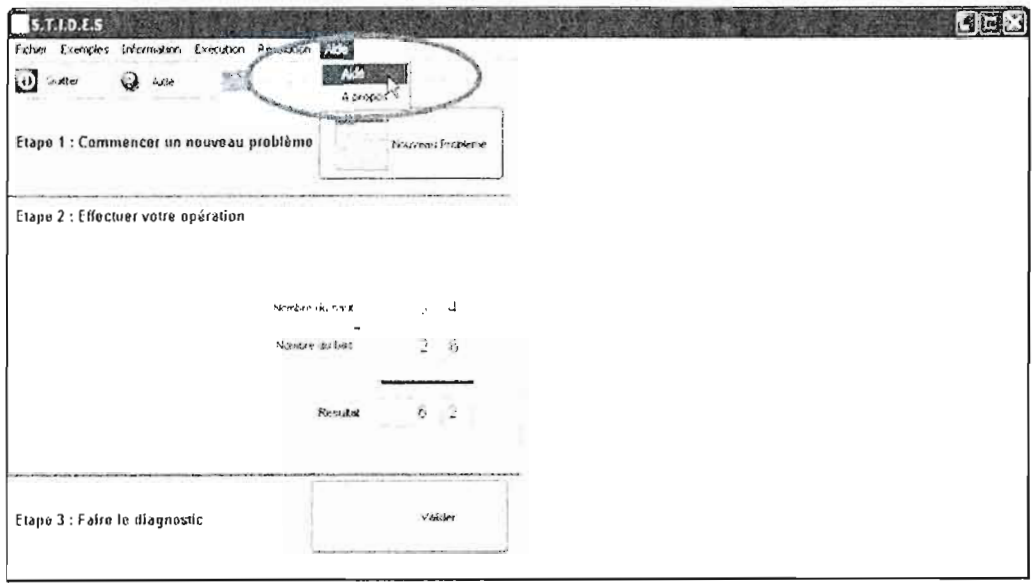
Choix des exemples

Un deuxième exemple est choisi pour cette démonstration :



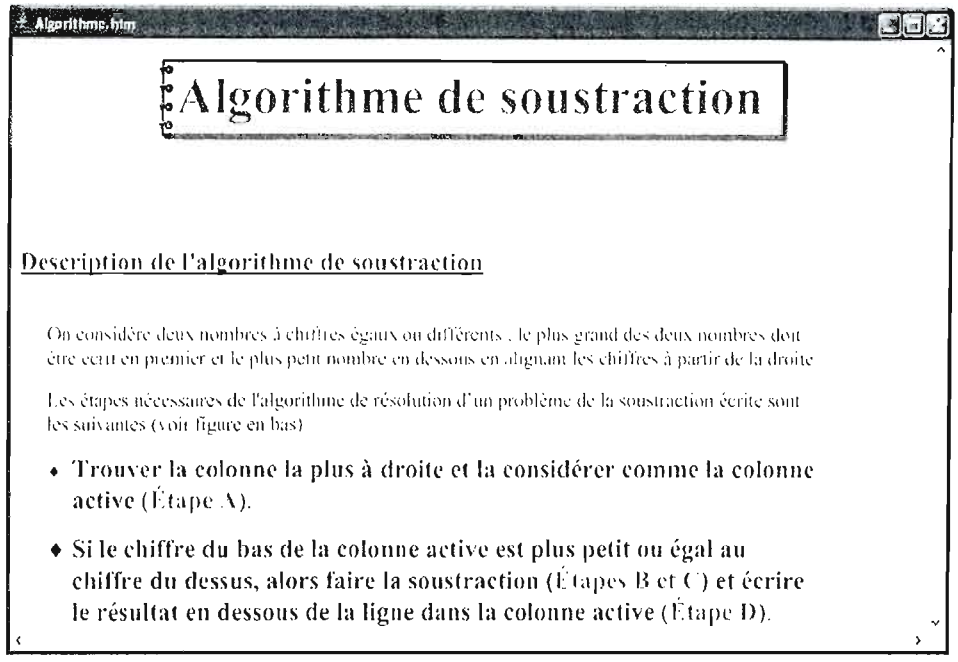
Affichage des exemples type 2

Enfin l'utilisateur peut demander de l'aide pour les procédures de la solution :



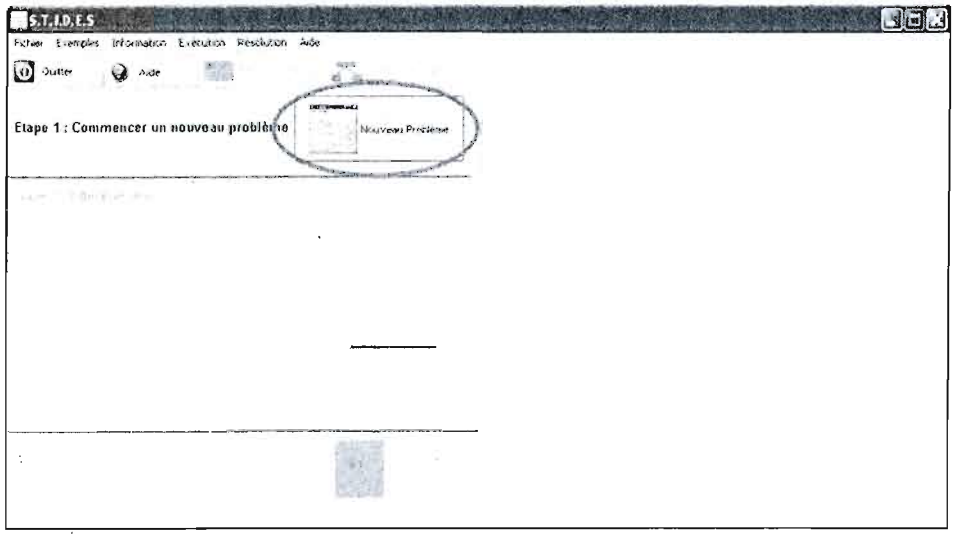
Demande de l'aide

Et le système affiche l'aide à l'utilisateur avec toutes les étapes à suivre pour résoudre un problème de soustraction quel que soit son type :



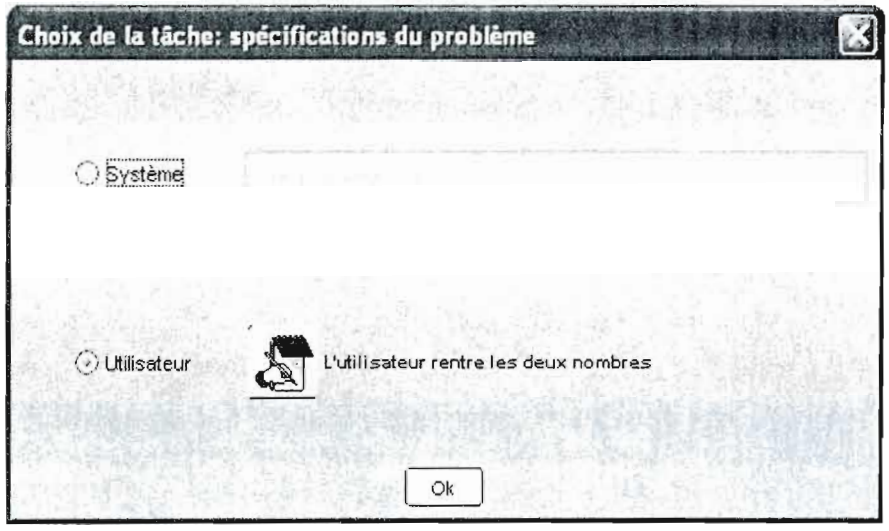
Affichage de l'aide

Pour commencer, l'utilisateur doit cliquer sur le bouton « Nouveau Problème ».



L'écran principal du système

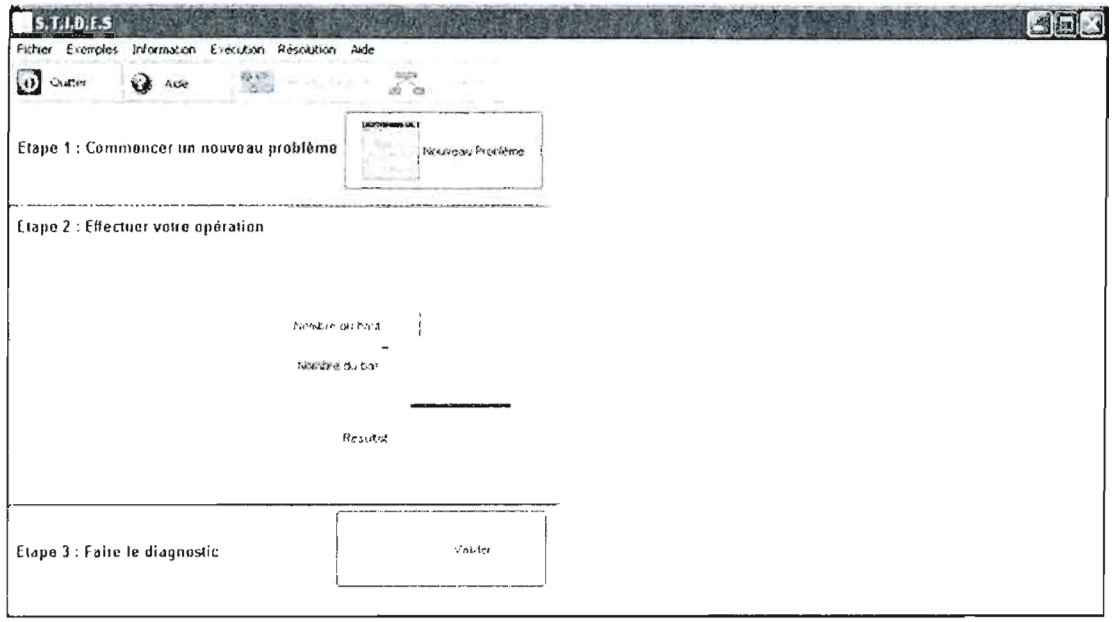
Un message l'invite alors à choisir son mode de travail, c'est-à-dire s'il désire entrer son propre problème ou s'il accepte celui proposé par le système.



Choix de mode de travail

Mode Utilisateur

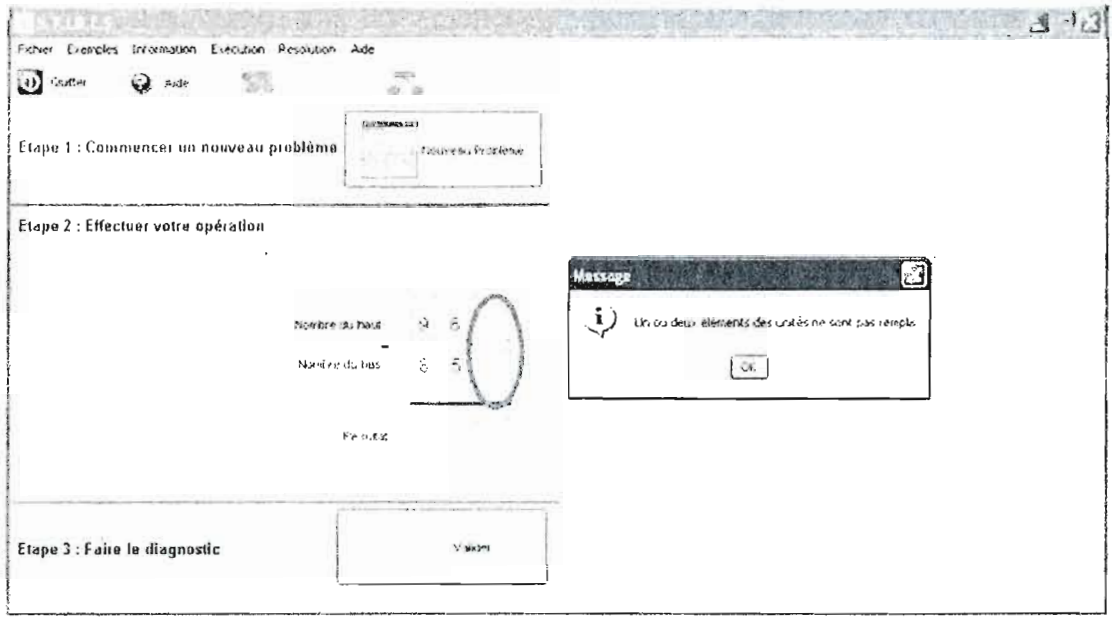
Si l'utilisateur répond en cliquant sur l'icône « Utilisateur » le système l'invite à entrer son problème, en lui permettant de donner les deux termes, comme l'indique la figure suivante.



Entrée de deux termes

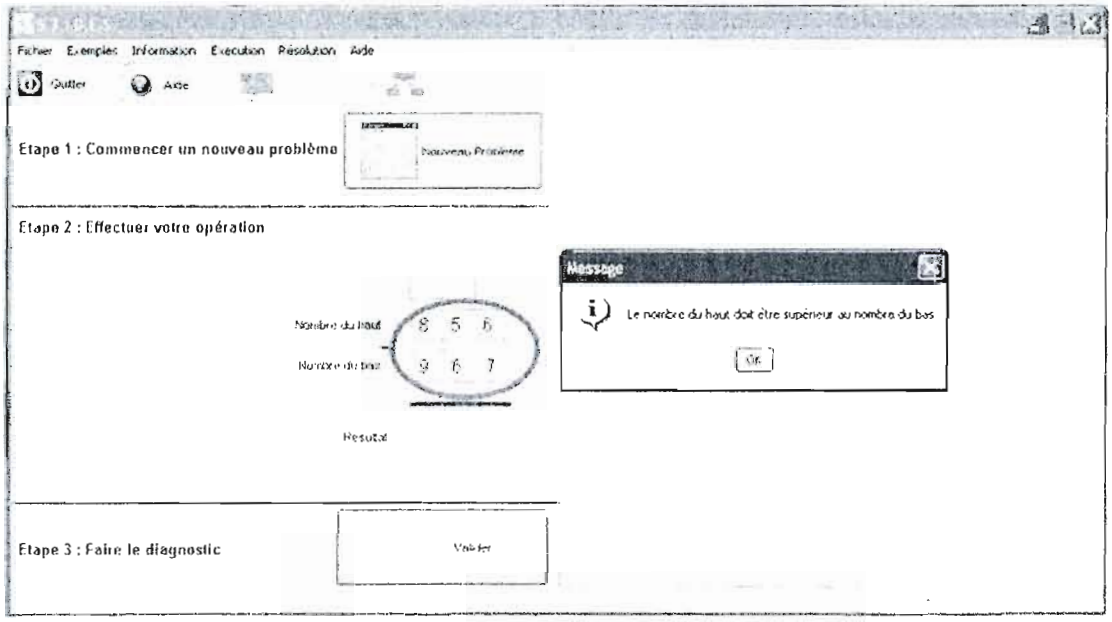
Seuls les caractères numériques (c'est-à-dire les chiffres) sont acceptés dans les cases, l'utilisateur ne peut entrer d'autres caractères.

Lorsque l'utilisateur entre ses termes, le système contrôle les positions des chiffres, et si l'utilisateur pose son opération de façon incorrecte dans les cases, il reçoit le message suivant :



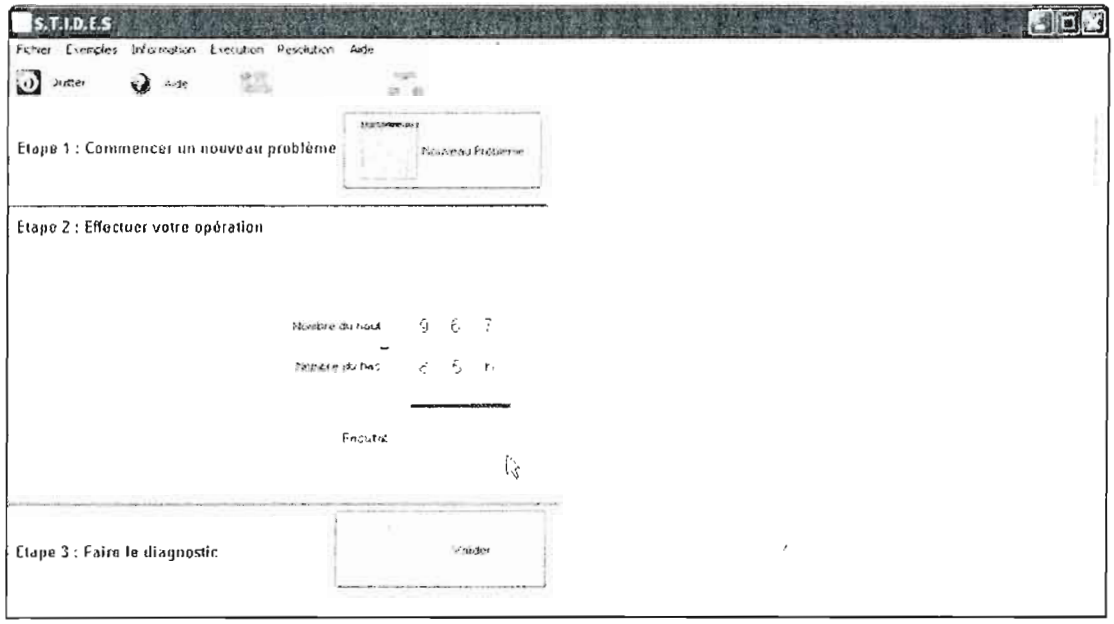
Message de contrôle de position des chiffres

De même, si l'utilisateur intervertit les termes, c'est-à-dire change l'ordre des termes, il reçoit le message suivant :



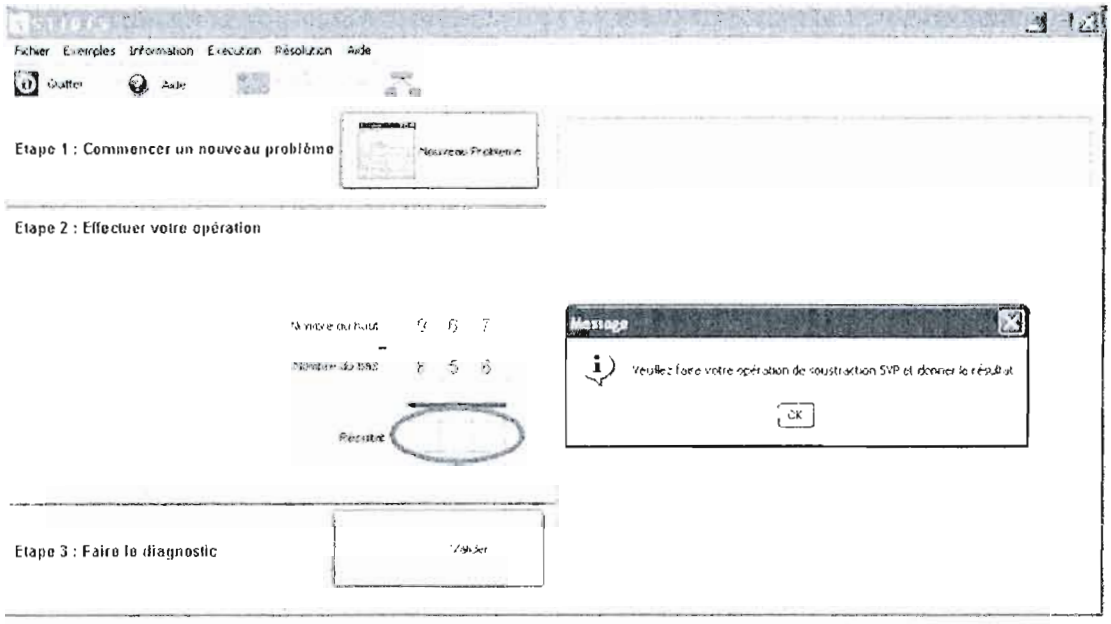
Message de contrôle de l'ordre des termes

Ensuite, si l'utilisateur entre de façon correcte son problème, il peut commencer sa résolution.



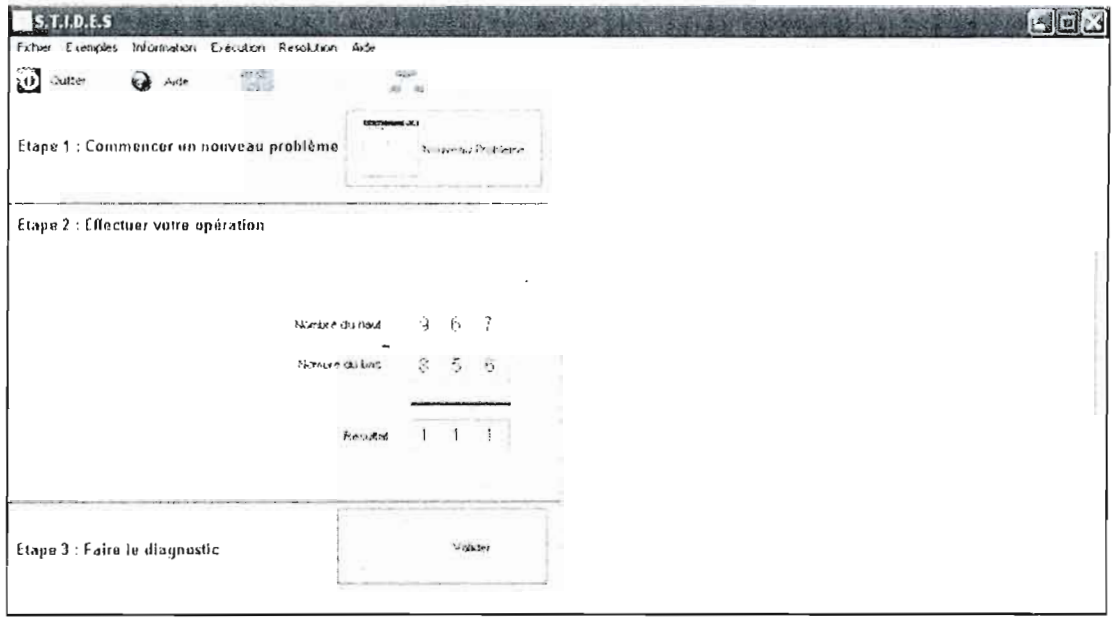
Entrée correcte des deux termes

Après l'entrée correcte des deux termes, l'utilisateur doit commencer à résoudre son problème et donner son résultat colonne par colonne, lequel sera affiché comme réponse finale de l'opération. Si par exemple l'utilisateur valide son résultat sans qu'il inscrive aucune réponse à une colonne, le système lui adresse le message suivant :



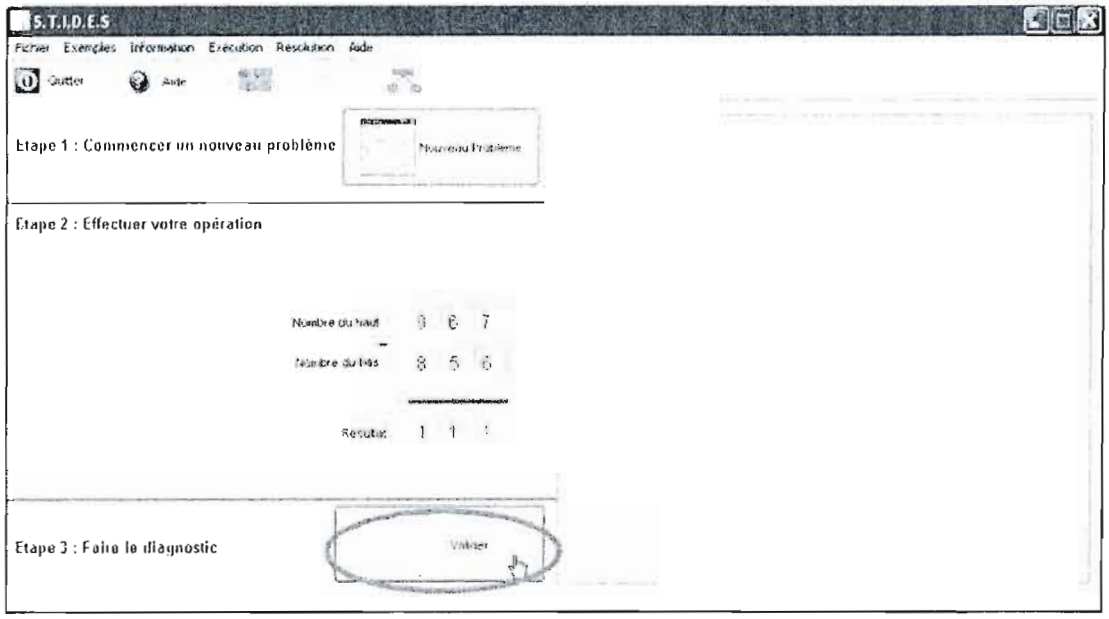
Message de contrôle du résultat

L'utilisateur entre correctement les résultats de chaque colonne, comme le montre le schéma suivant :



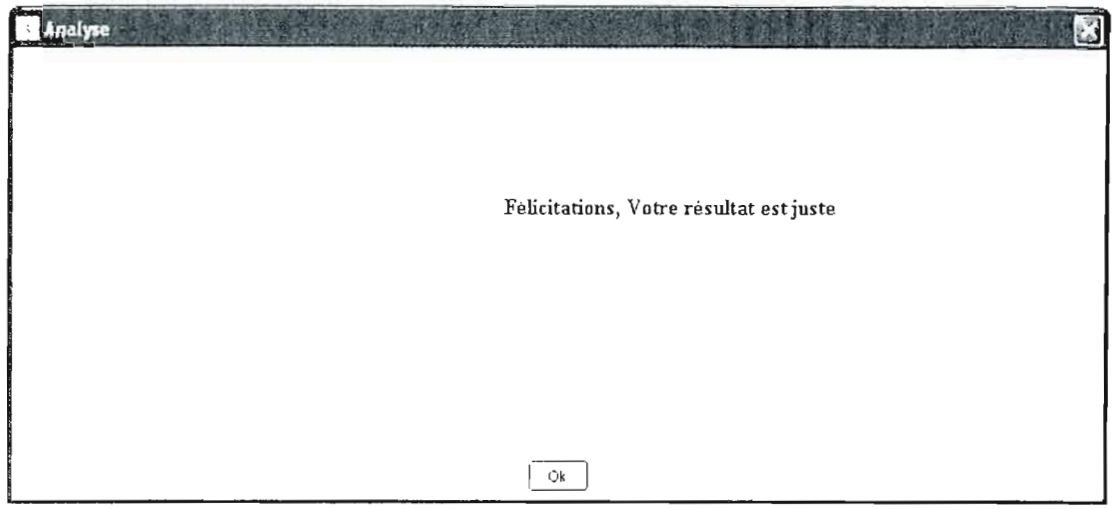
Calcul de résultat de chaque colonne

Ensuite, après l'entrée du résultat de chaque colonne, l'utilisateur clique sur le bouton « Valider » pour valider sa réponse.



Validation de la réponse

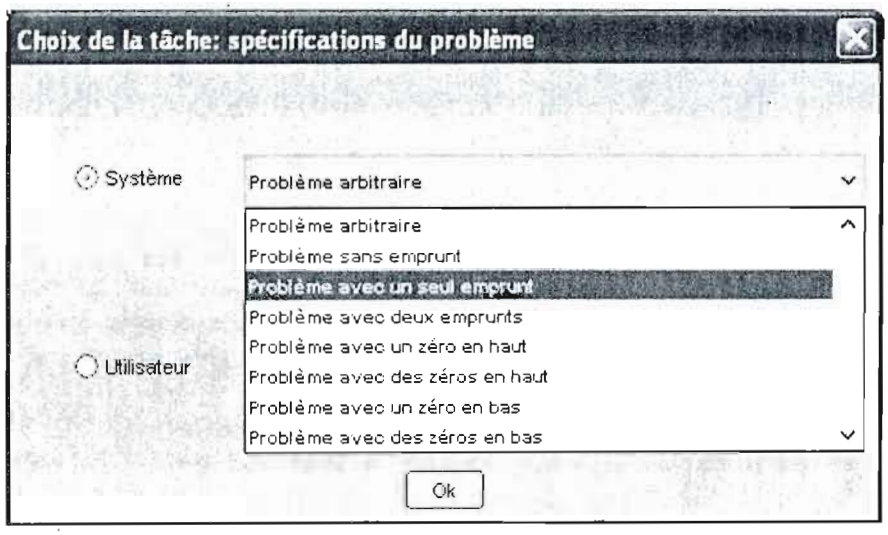
Dans l'étape suivante, le système contrôle la justesse de la réponse de l'utilisateur. Si le résultat du problème est correct, le système affiche un message de félicitations à l'utilisateur.



Message de félicitations

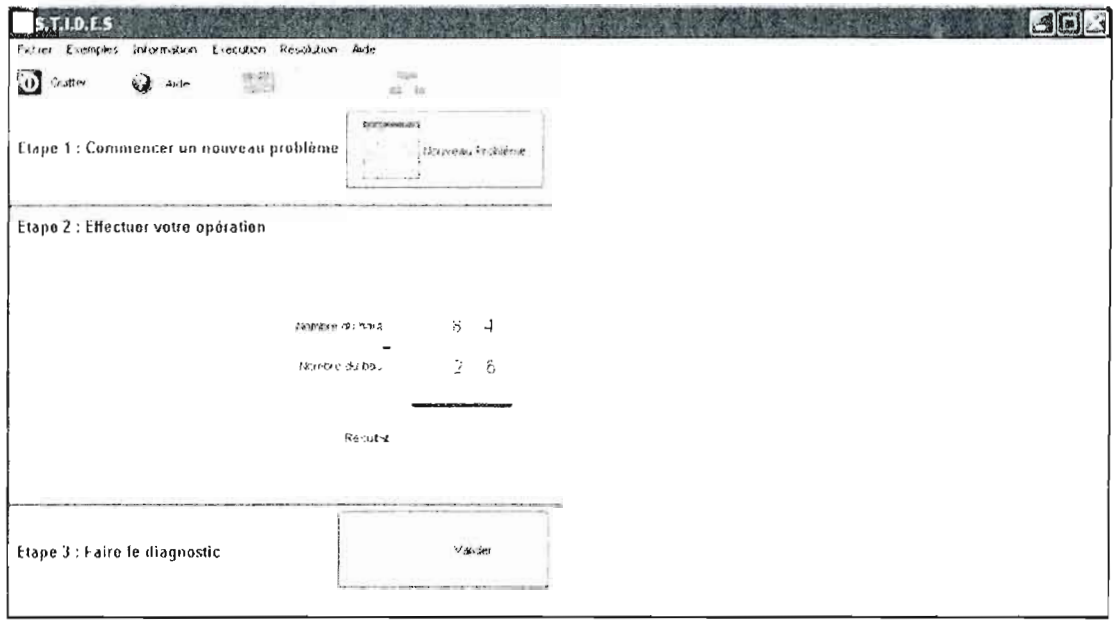
Mode Système

Si l'utilisateur répond en cliquant sur l'icône « Système », une liste de tâches est proposée à l'utilisateur, et ce dernier peut spécifier son problème selon son niveau d'apprentissage. Le schéma suivant montre que l'utilisateur a choisi un problème avec un seul emprunt.



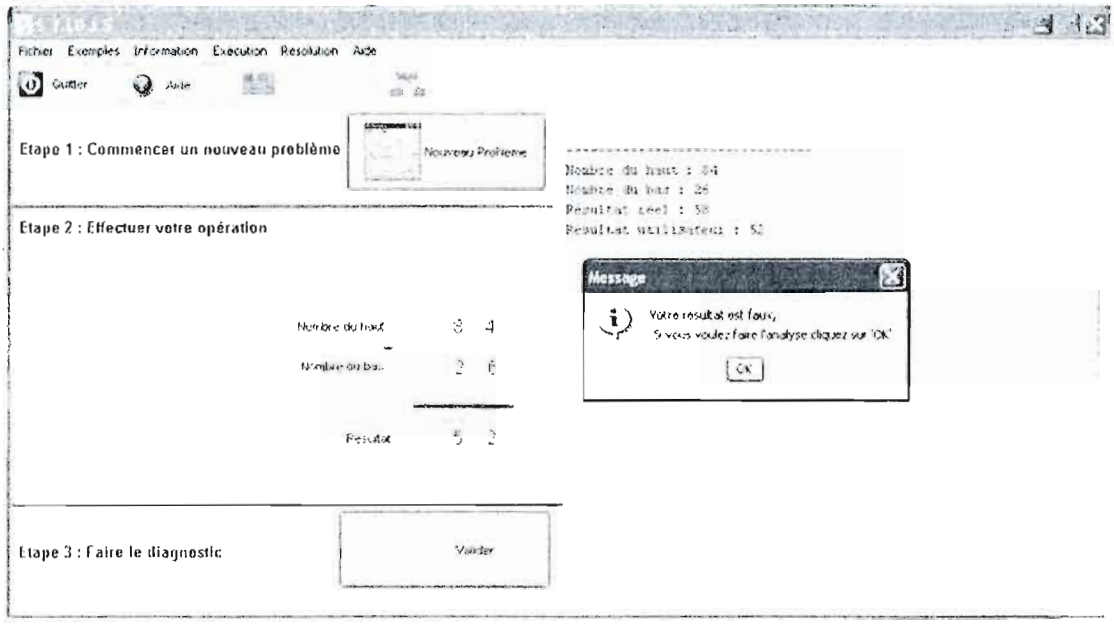
Choix d'une tâche particulière

Après le choix de l'utilisateur, le système propose un problème à ce dernier tout en respectant son choix et attend que l'utilisateur commence à résoudre ce problème.



Affichage d'un problème par le système

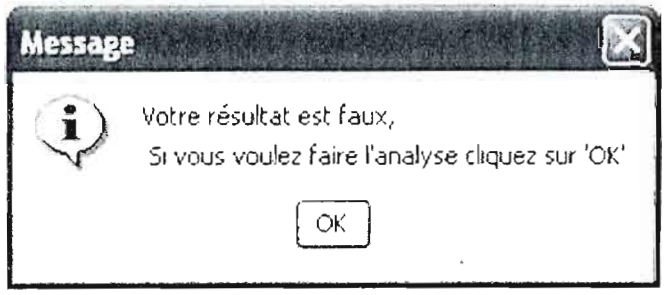
Ensuite l'utilisateur doit entrer sa solution dans les cases « Résultat ».



L'utilisateur entre sa solution et valide son résultat

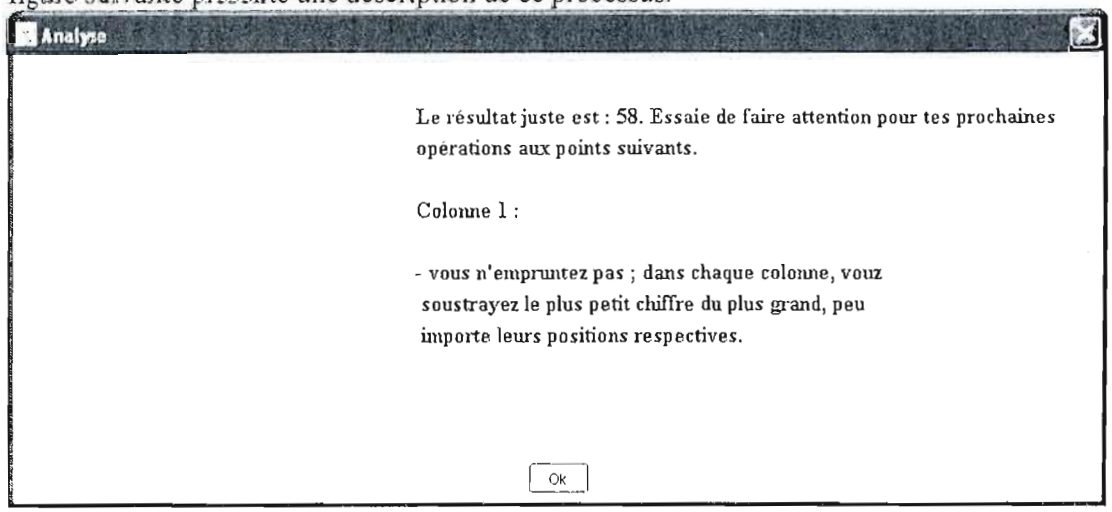
Lorsque l'utilisateur a fini d'entrer sa solution et valide son résultat, le système identifie un ensemble de règles dont l'activation reproduit son comportement. Il est capable de générer deux messages, l'un pour l'analyse de l'erreur et l'autre pour le diagnostic.

Après l'analyse de la réponse de l'utilisateur, le système affiche le message suivant.



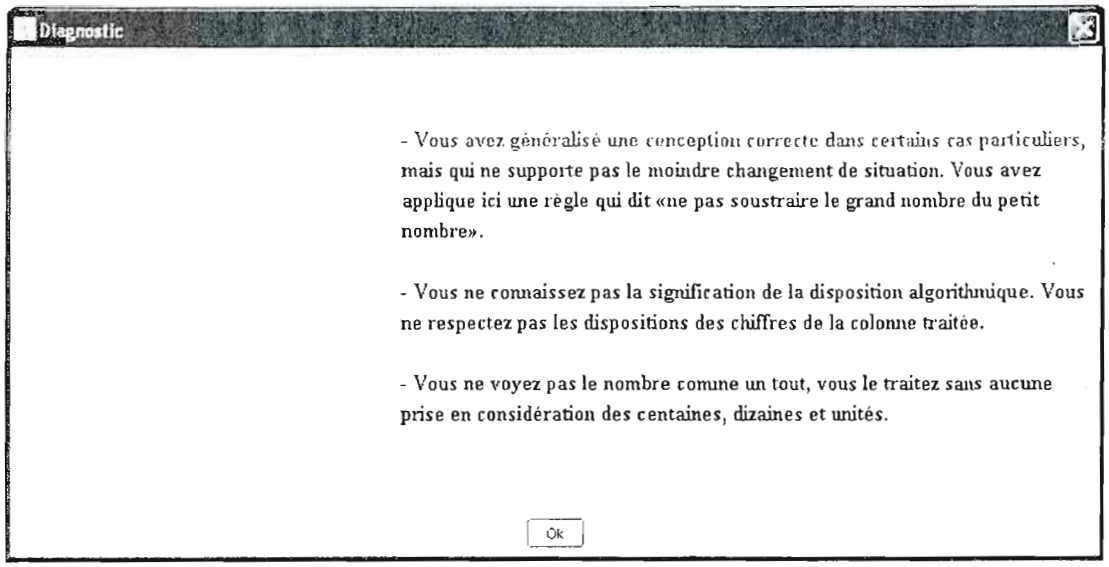
Message erreur

Une réponse affirmative conduirait le système à analyser une autre fois le résultat de l'utilisateur et à produire des explications précises pour sa conception erronée en déclenchant la mal-règle (ou mal-règles) qui a produit la mauvaise réponse. La figure suivante présente une description de ce processus.



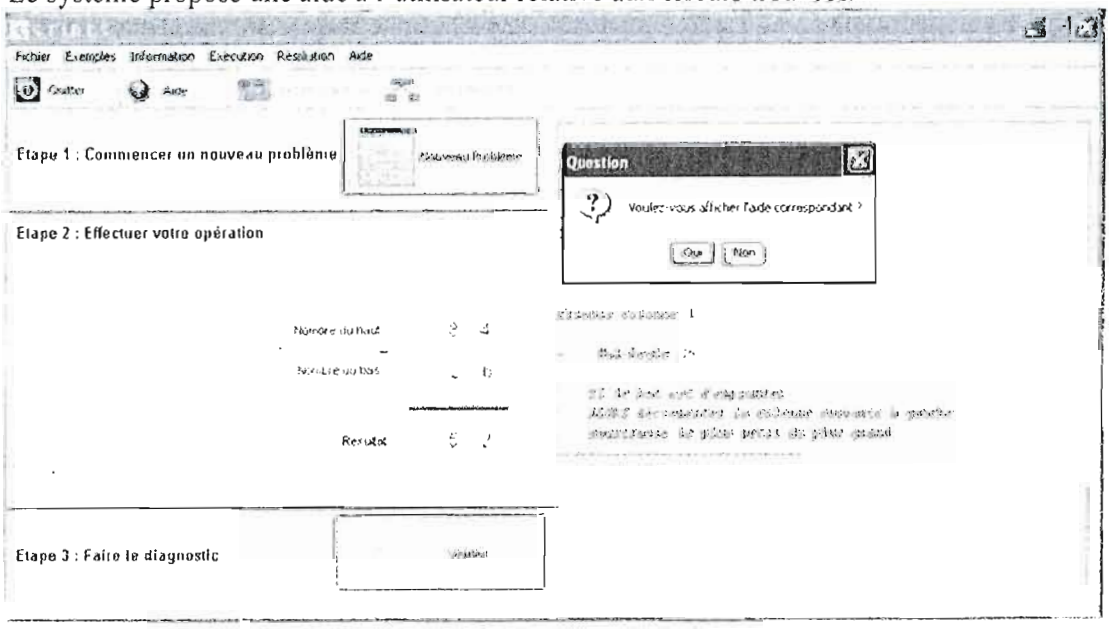
Message de l'analyse de l'erreur

Après ces hypothèses, le système propose un diagnostic relatif à cette erreur sous forme d'un autre message, selon le type d'erreur.



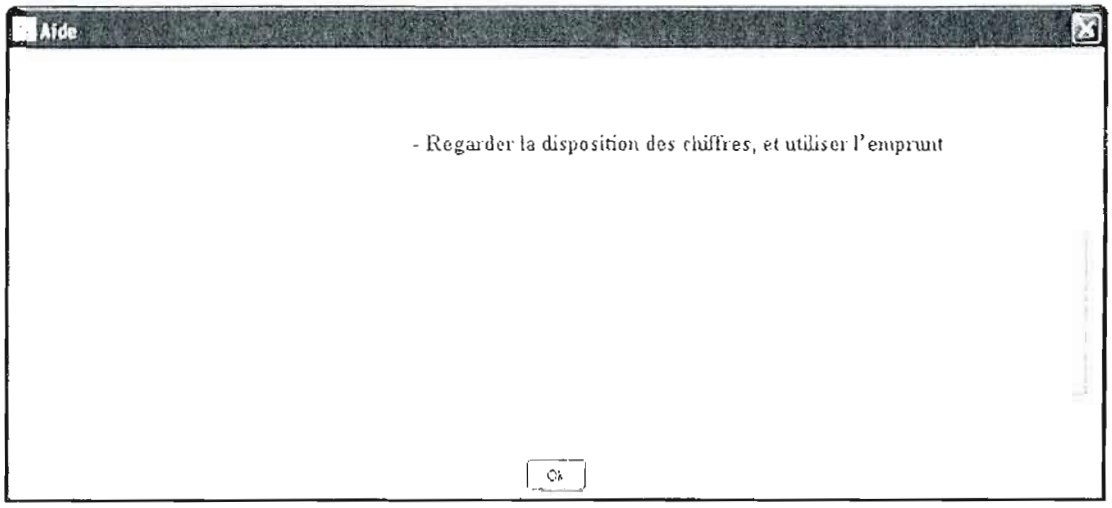
Message de diagnostic de l'erreur

Le système propose une aide à l'utilisateur relative aux erreurs trouvées.



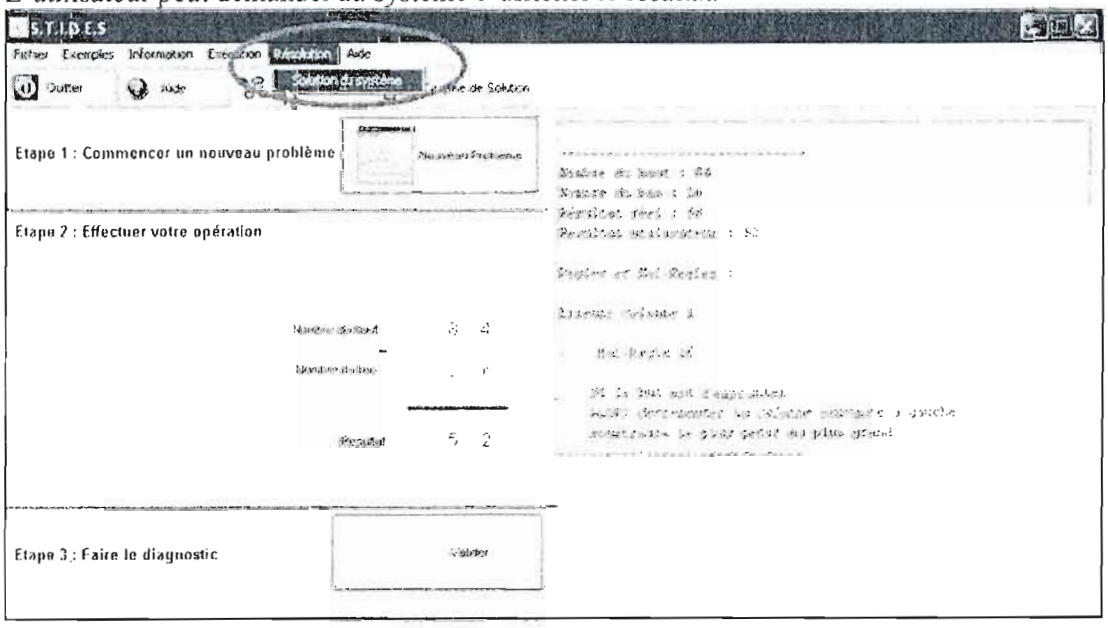
Message propose de l'aide à l'utilisateur

Le système affiche l'aide demandée par l'utilisateur



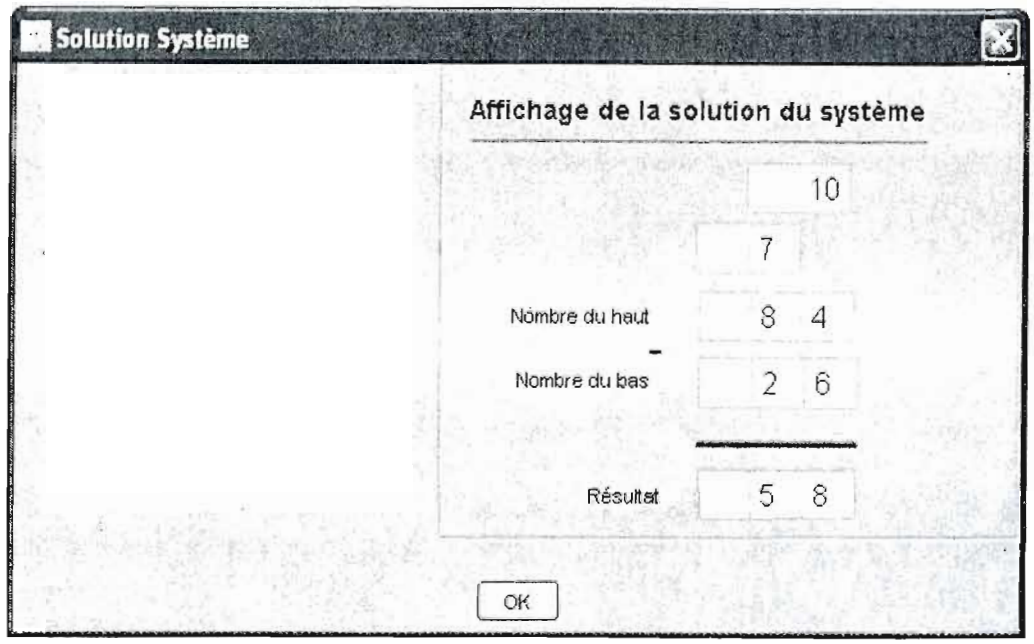
Message d'aide

L'utilisateur peut demander au système d'afficher le résultat.



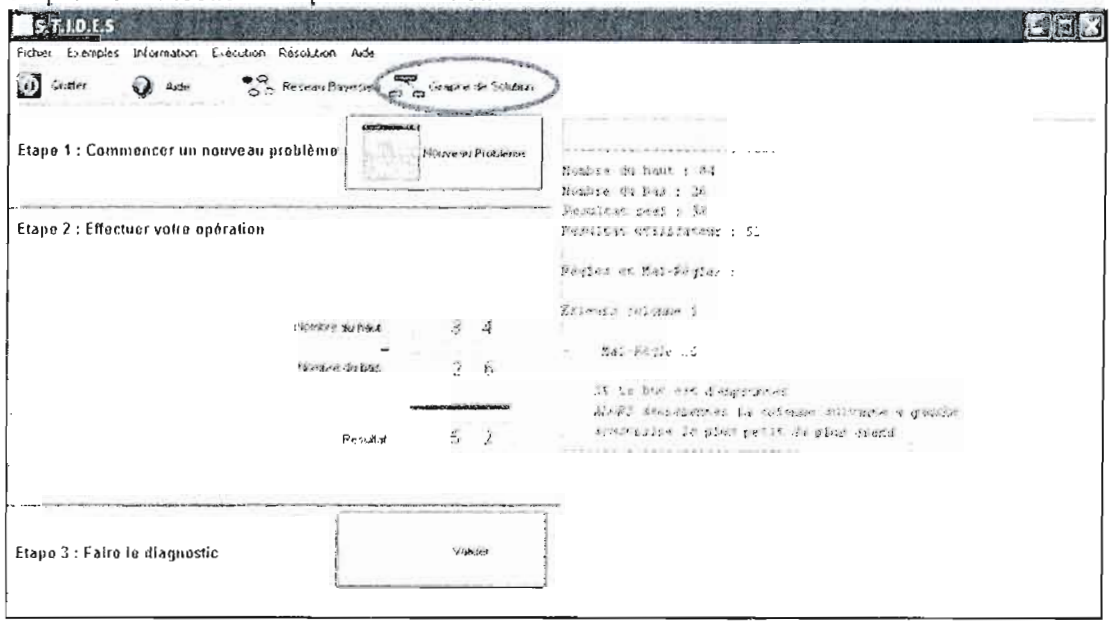
Demande de la solution du système

Le système affiche la solution correcte avec les décréments et les emprunts



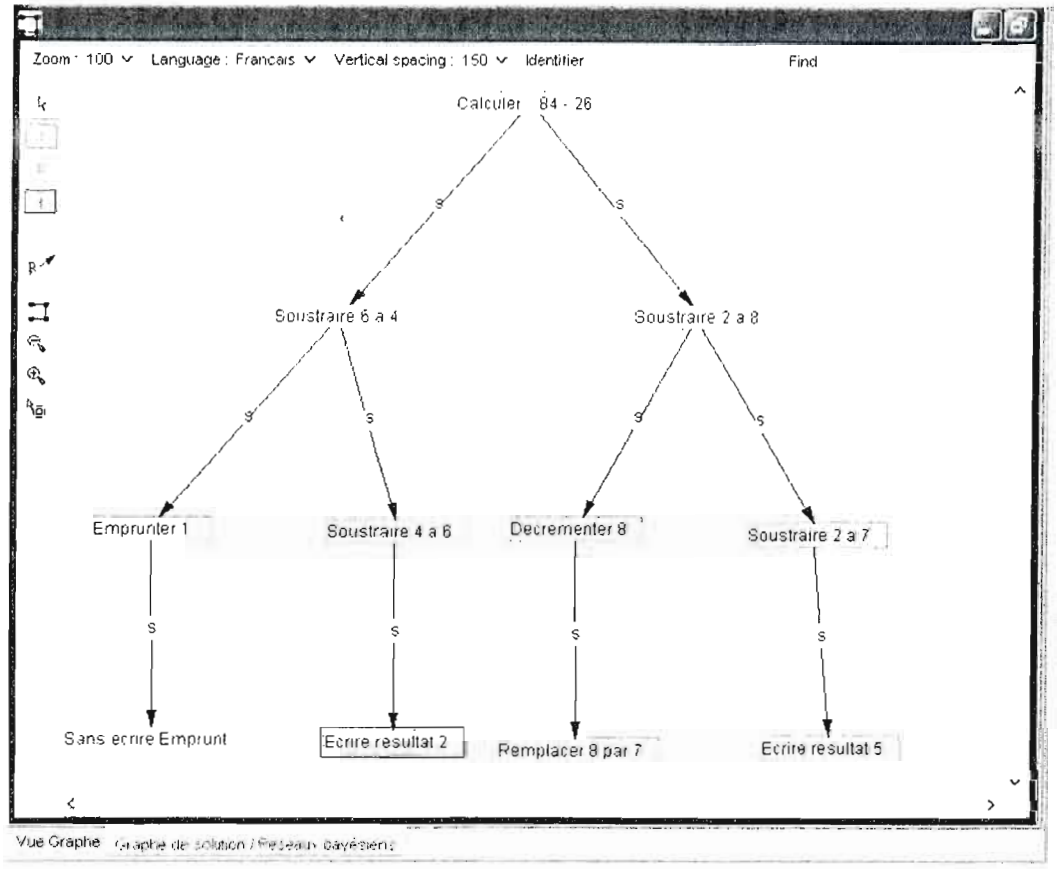
Affichage de la solution du système

L'utilisateur peut demander au système d'afficher le graphe de solution en cliquant sur l'icône « Graphe de Solution ».



Demande d'affichage du graphe de solution

Dans un premier temps, le système affiche le graphe de la solution :



Affichage du graphe de solution

ANNEXE C - QUESTIONNAIRE SYSTÈME TIDES

Fortement en accord = A
En accord = B .....
En désaccord = C
Fortement en désaccord = D

Code de l'élève.....

1.	Le système TIDES a stimulé mon intérêt pour l'arithmétique
2.	Le système TIDES m'a permis d'acquérir des habiletés de calcul et d'algorithme de soustraction
3.	L'exposé théorique qui a précédé la séance d'apprentissage m'a aidé à résoudre les problèmes
4.	La formulation des actions et des stratégies de résolution était facilitée par les fenêtres.
5.	Les messages d'analyse m'ont aidé dans ma démarche de résolution
6.	Les diagnostics sous forme de messages écrits m'ont aidé dans ma démarche de résolution
7.	L'exécution graphique sous forme de réseau bayésien m'a permis de mieux comprendre les causes de mes erreurs
8.	Le fait de pouvoir utiliser la formulation libre du problème que celui proposé par le système était important pour moi
9.	Le fait de pouvoir accéder aux informations sur les consignes de résolution ainsi que sur l'aide du système était important pour moi
10.	Les exemples de problèmes avec solutions commentées m'ont aidé dans ma démarche de résolution
11.	Il m'a fallu beaucoup de temps pour me débrouiller avec le bloc de résolution, les fenêtres et les menus
12.	D'une façon générale, j'ai aimé participer à cette expérience d'apprentissage

Améliorations suggérées au système TIDES

.....

.....

.....

Améliorations suggérées à l'exposé qui précède l'utilisation de TIDES

.....

.....

.....

Commentaires généraux

.....

.....

.....

.....

ANNEXE D – ENCODAGE DES RÈGLES EN PROLOG

Le mécanisme d'inférence analyse les colonnes une par une pour savoir dans quelle colonne l'erreur s'est produite (FindErrors). Si le résultat d'une colonne est différent de celui du système, il déclenche les règles correspondantes (Rules).

```
FindErrors(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1, _ListErrorsAll,
_ListErrorsCol1, _ListErrorsCol2, _ListErrorsCol3) :-
        FindErrorsAll(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsAll),
        FindErrorsCol1(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol1),
        FindErrorsCol2(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol2),
        FindErrorsCol3(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol3).

FindErrorsAll(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1, _ListErrorsAll) :-
        FindErrorsPerCol(1, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsAll).

FindErrorsCol1(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol1) :-
        FindErrorsPerCol(1, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol1).

FindErrorsCol2(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol2) :-
        FindErrorsPerCol(1, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol2).

FindErrorsCol3(_X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol3) :-
        FindErrorsPerCol(1, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_ListErrorsCol3).

FindErrorsPerCol(88, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
(88)) :-
        Error(88, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1),
        /.

FindErrorsPerCol(88, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1,  R3, _R2, _R1, _U3, _U2, _U1, ())
:-
        not (Error(88, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1)),
        /.

FindErrorsPerCol(_Error, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
(_Error | _List)) :-
        Error(_Error, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1),
        val(_NextError, add(_Error, 1)),
        FindErrorsPerCol(_NextError, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1,
_U3, _U2, _U1, _List),
        /.
```

```
FindErrorsPerCol(_Error, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1,
_List) :-
        not (Error(_Error, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2,
_U1)),
        val(_NextError, add(_Error, 1)),
        FindErrorsPerCol(_NextError, _Col, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1,
_U3, _U2, _U1, _List),
        /.

Error(1, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        eqv(_U2, 0).

Error(2, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        not(eqv(_Y2, 1)),
        eqv(_U2, _Y2).

Error(3, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        eqv(_Y2, 1),
        eqv(_U2, 1),
        not(eqv(_U2, _R2)).

Error(4, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        eqv(_U2, sub(9,_Y2)),
        eqv(_U3, sub(_X3,_Y3)),
        not(eqv(_U3, _R3)).

Error(5, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        eqv(_U2, sub(10,_Y2)),
        not(eqv(_U3, _R3)).
```

```
Error(6, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_X1, 1),
        inf(_X1, _Y1),
        eqv(_U1, sub(10, _Y1)).
Error(6, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_X2, 1),
        not(inf(_X1, _Y1)),
        inf(_X2, _Y2),
        eqv(_U2, sub(10,_Y2)).

Error(7, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        inf(_X2, _Y2),
        eqv(_X2, 1),
        eqv(_U2, sub(9,_Y2)),
        eqv(_U3, sub(sub(_X3, 1), _Y3)),
        eqv(_U3, _R3).

Error(8, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        eqv(_U2, 11).

Error(8, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 1),
        eqv(_U2, 11).

Error(9, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(0, _Y2),
        eqv(_U2, 0),
        eqv(_U3, sub(sub(_X3, 1), _Y3)).

Error(9, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
```

```
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(0, _Y2),
        eqv(_U2, 0),
        eqv(_U3, sub(sub(_X3, 2), _Y3)).

Error(10, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_Y2, 0),
        eqv(_U2, 1),
        eqv(_U3, sub(_X3, _Y3)).

Error(11, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_Y2, 0),
        eqv(_U3, sub(sub(_X3,1), _Y3)).

Error(12, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_Y2, 0),
        eqv(_U2, 0),
        eqv(_U3, sub(add(1, _X3), _Y3)).

Error(13, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, _R1),
        eqv(_U2, 0),
        eqv(_U3, sub(_X3, 1)).

Error(14, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, empty, _U1) :-
        not(eq(_U3, empty)),
```

```
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_U3, sub(_X3, 1)).

Error(15, 3, _X3, _X2, _X1, empty, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        sup(_X1, _Y1),
        eqv(_U1, _R1),
        inf(_X2, _Y2),
        eqv(_U2, _R2),
        eqv(_U3, _X3).

Error(16, 3, _X3, _X2, _X1, empty, _Y2, _Y1, _R3, _R2, _R1, empty, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_U1, _R1),
        eqv(_U2, _R2),
        eqv(_X3, 1).

Error(17, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, empty, empty, _U1) :-
        eqv(_R1, _U1).

Error(17, 3, _X3, _X2, _X1, empty, _Y2, _Y1, _R3, _R2, _R1, empty, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_R2, _U2),
        eqv(_R1, _U1).


Error(18, 1, _X3, _X2, _X1, empty, _Y2, _Y1, _R3, _R2, _R1, empty, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_R2, _U2),
        eqv(_R1, _U1).

Error(19, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        sup(_X1, _Y1),
        eqv(_R1, _U1),
        eqv(_U2, sub(_X2, 1)),
        eqv(_U3, sub(_X3, 1)).

Error(19, 3, _X3, _X2, _X1, empty, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        sup(_X1, _Y1),
        eqv(_R1, _U1),
        eqv(_U3, sub(_X3, 1)).

Error(20, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, empty, _U1) :-
        not(eq(_U3, empty)),
```

```
        sup(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_U3, _X3).

Error(22, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_R1, _U1),
        sup(_X2, _Y2),
        eqv(_U2, sub(_X2, _Y2)),
        eqv(_U3, sub(sub(_X3, 1), _Y3)).

Error(23, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, sub(add(_X1, sub(_Y1, _X1)), _Y1)),
        not(eqv(_U1, _R1)),
        eqv(_U2, sub(sub(_X2, sub(_Y1, _X1)), _Y2)),
        not(eqv(_U2, _R2)).

Error(24, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, empty, empty, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        sup(_X1, _Y1),
        inf(_X2, _Y2),
        eqv(_U1, _R1).

Error(25, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        sup(_X1, _Y1),
        inf(_X2, _Y2),
        eqv(_U2, sub(_X3, _Y3)),
        not(eqv(_U2, _R2)).

Error(26, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        inf(_X1, _Y1),
        not(eqv(_U1, _R1)),
        eqv(_U1, sub(_Y1, _X1)),
        eqv(_U2, _R2).

Error(27, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, 0),
        eqv(_U2, _R2).

Error(27, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
```

```
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_R1, _U1),
        inf(_X2, _Y2),
        eqv(_R2, 0),
        eqv(_U3, _R3).

Error(28, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_R1, _U1),
        not(eqv(_X2, 0)),
        eqv(_X2, _Y2),
        eqv(_U2, sub(_X2, _Y2)).

Error(29, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        not (eqv(_X1, 0)),
        not(eqv(_X1, 0)),
        eqv(_Y1, 0),
        eqv(_U1, 0).

Error(29, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        not (eqv(_X2, 0)),
        eqv(_Y2, 0),
        eqv(_U2, 0).

Error(29, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        not (eqv(_X3, 0)),
        eqv(_Y3, 0),
        eqv(_U3, 0).

Error(30, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_X1, _Y1),
        eqv(_U1, 9),
        eqv(_U2, sub(_R2, 1)).

Error(30, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
```

```
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_X2, _Y2),
        eqv(_U2, 9),
        eqv(_U3, sub(_R3, 1)).

Error(31, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        not(eqv(_X1, 1)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        not(eqv(_X2, 0)),
        eqv(_U1, sub(10, _Y1)),
        not(eqv(_U2, _R2)).

Error(32, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        inf(_X1, _Y1),
        sup(_X2, _Y2),
        eqv(_U1, _R1),
        eqv(_U2, sub(_X2, sub(_Y1,_X1))).

Error(34, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        sup(_Y2, _X2),
        eqv(_U2, sub(10, _Y2)),
        eqv(_U3, sub(sub(_X3, 2), _Y3)).

Error(35, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_X1, 0),
        eqv(_X2, 0),
        eqv(_U1, sub(9, _Y1)),
        eqv(_U2, sub(9, _Y2)).

Error(36, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, sub(add(9, _X1), _Y1)),
        not(eqv(_U1, _R1)),
        eqv(_U2, _R2),
```

```
        eqv(_U3, _R3).

Error(37, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U2, sub(add(8, _X2), _Y2)),
        eqv(_U1, _R1),
        not(eqv(_U2, _R2)),
        eqv(_U3, _R3).

Error(38, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        eq(_U3, empty),
        eq(_U2, empty),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, empty).

Error(39, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        inf(_X2, _Y2),
        eqv(_U1, sub(add(10, sub(_X1, 1)), _Y1)),
        eqv(_U2, _R2),
        eqv(_U3, _R3).

Error(42, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_X1, 0),
        eqv(_X2, 0),
        eqv(_U1, _Y1),
        eqv(_U2, sub(10, _Y2)),
        eqv(_U3, _R3).

Error(43, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_X1, 0),
        eqv(_X2, 0),
        eqv(_U1, _R1),
        eqv(_U2, sub(10, _Y2)),
        not(eqv(_U2, _R2)),
```

```
        eqv(_U3, _R3).

Error(44, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_U2, 0),
        eqv(_U3, _X3),
        not(eqv(_U3, _R3)).

Error(45, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, _R1),
        eqv(_U2, sub(10, sub(_Y2,1))),
        eqv(_U3, _R3).

Error(46, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U2, sub(add( _X2, 10), _Y2)),
        eqv(_U3, sub(_X3, _Y3)).

Error(47, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U2, sub(9, _Y2)),
        eqv(_U3, sub(_X3, _Y3)).

Error(48, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_Y2, 0),
        eqv(_U2, 0),
        eqv(_U3, sub(_X3, _Y3)),
        not(eqv(_U3, _R3)).
```

```
Error(49, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        eqv(_X2, 0),
        eqv(_U1, sub(_Y1, _X1)).

Error(50, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        not(eqv(_Y2, 0)),
        eqv(_U2, sub(10, _Y2)),
        eqv(_U3, sub(sub(_X3, 1), _Y3)).

Error(51,3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, _R1),
        eqv(_U2, sub(9, _Y2)),
        eqv(_U3, sub(_X3, _Y3)).

Error(52, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, 0),
        eqv(_U2, sub(10, _Y2)),
        eqv(_U3, _R3).

Error(53, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        eqv(_X2, 0),
        not(eqv(_Y2, 0)),
```

```
        eqv(_U2, 0),
        eqv(_U3, sub(_X3, _Y3)).


Error(54, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, _R1),
        not(eqv(_Y2, 0)),
        eqv(_U2, _Y2),
        eqv(_U3, sub(_X3, _Y3)).

Error(55, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U2, sub(add(_X2,10), _Y2)),
        eqv(_U3, sub(_X3, _Y3)).

Error(56, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, 0),
        eqv(_U2, _Y2),
        eqv(_U3, sub(_X3, _Y3)),
        not(eqv(_U3, _R3)).

Error(57, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X1, 0),
        eqv(_U1, sub(add(add(_X1,10),_X2), _Y1)),
        eqv(_U2, sub(sub(_X2,1), _Y2)).

Error(58, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_X1, 0),
        not (eqv(_Y1, 0)),
        eqv(_U1, 0).

Error(59, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
```

```
        not(eq(_U2, empty)),
        eqv(_X1, 0),
        not (eqv(_Y1, 0)),
        not (eqv(_X2, 0)),
        eqv(_U1, _Y1).

Error(59, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_X2, 0),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_U1, _R1),
        eqv(_U2, _Y2),
        not(eqv(_U3, _R3)).


Error(60, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U2, empty)),
        eqv(_X1, 0),
        not (eqv(_Y1, 0)),
        eqv(_U1, sub(add(_X1, 9), _Y1)).

Error(61, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        not (eqv(_Y1, 0)),
        eqv(_X1,0),
        not(eqv(_Y2, 0)),
        eqv(_U1, _R1),
        not(eqv(_U2, _R2)).


Error(61, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        not(eq(_Y3, empty)),
        not(eq(_U3, empty)),
        eqv(_X2,0),
        not (eqv(_Y1, 0)),
        not(eqv(_Y2, 0)),
        eqv(_U2, _R2),
        not(eqv(_U3, _R3)).

Error(62, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_U3, empty)),
        inf(_X1, _Y1),
        inf(_X2, _Y2),
        not(eqv(_X1,0)),
        not(eqv(_X2,0)),
        eqv(_U1, _R1),
        eqv(_U2, sub( sub(1, _X2), _Y2)),
        not(eqv(_U3, _R3)).
```

```
Error(63, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        inf(_X2, _Y2),
        eqv(_U2, _R2),
        eqv(_U3, _X3).

Error(64, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_X2, 0),
        eqv(_U2, sub (add(_X2, 9), _Y2)),
        eqv(_U2, _R2),
        eqv(_U3, sub(_X3, _Y3)),
        not(eqv(_U3, _R3)).

Error(65, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        inf(_X2, _Y2),
        eqv(_U2, sub(add(_X2, 11), _Y2)).

Error(66, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        eqv(_U1, _R1),
        sup(_X2, _Y2),
        eqv(_U2, sub(_X2, _Y2)).

Error(66, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eqv(_X1, 0)),
        not(eqv(_X2, 0)),
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        eqv(_U2, sub(add(10, _X2), _Y2)).

Error(67, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X1, 1),
        not(eqv(_Y1, 1)),
        eqv(_U1, 1).
```

```
Error(68, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        not(eqv(_X1,0)),
        eqv(_X1, _Y1),
        eqv(_U1, _X1).

Error(69, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        eqv(_U2, _X2),
        not(eqv(_U2, _R2)),
        eqv(_U3, _X3),
        eqv(_U3, _R3).

Error(70, 2, _X3, _X2, _X1, empty, empty, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        eq(_Y3, empty),
        eq(_Y2, empty),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_X2, 0),
        eqv(_U1, _R1),
        eqv(_U2, add(_X2, 1)),
        eqv(_U3, _X3).

Error(71, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_Y2, 9),
        eqv(_U2, sub(_X2, 1)).


Error(72, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X2, 0)),
        not(eqv(_X2, 1)),
        eqv(_X2, _Y2),
        eqv(_U2, 1).

Error(73, 3, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        sup(_X2, _Y2),
        eqv(_U2, _R2),
        eqv(_U3, sub(sub(_X3, 1), _Y3)),
```

```
        not(eqv(_U3, _R3)).

Error(74, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)).
        inf(_X1, _Y1),
        eqv(_U1, sub(_Y1, _X1)).

Error(74, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, sub(_Y1, _X1)).
Error(74, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X2, _Y2),
        eqv(_U2, sub(_Y2, _X2)).

Error(76, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        eqv(_U1, _Y1),
        eqv(_U2, _Y2),
        eqv(_U3, _X3).

Error(77, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        eqv(_U1, 0),
        eqv(_U2, sub(_X2, _Y2)).

Error(77, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        inf(_X2, _Y2),
        eqv(_U1, 0),
        eqv(_U2, 0),
        eqv(_U3, sub(_X3, _Y3)).

Error(78, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, add(_X1, _Y1)),
        not(eqv(_U1, _R1)),
        eqv(_U2, sub(add(_X2, _Y2), 1)).

Error(79, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
```

```
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X2, 1)),
        not(eqv(_Y2, 0)),
        eqv(_U2, add(1, sub(_X2, _Y2))),
        not(eqv(_U2, _R2)).

Error(80, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        val(_N1, add(_X1, add(mul(_X2, 10), mul(_X3, 100)))),
        val(_N2, add(_Y1, add(mul(_Y2, 10), mul(_Y3, 100)))),
        val(_N3, add(_U1, add(mul(_U2, 10), mul(_U3, 100)))),
        eqv(_N3, add(_N1,_N2)).

Error(81, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(add(_X1, _Y1), 10),
        eqv(_U1, add(_X1, _Y1)),
        inf(add(_X2, _Y2), 10),
        eqv(_U2, add(_X2, _Y2)),
        eqv(_U3, add(_X3, _Y3)).

Error(81, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        not(inf(add(_X1, _Y1), 10)),
        eqv(_U1, sub(10, add(_X1, _Y1))),
        inf(add(_X2, _Y2), 10),
        eqv(_U2, add(_X2, _Y2)),
        eqv(_U3, add(_X3, _Y3)).

Error(81, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq( Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        not(inf(add(_X1, _Y1), 10)),
        eqv(_U1, sub(10, add(_X1, _Y1))),
        not(inf(add(_X2, _Y2), 10)),
        eqv(_U2, sub(10, add(_X2, _Y2))),
        eqv(_U3, add(_X3, _Y3)).

Error(81, 0, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
```

```
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(add(_X1, _Y1), 10),
        eqv(_U1, add(_X1, _Y1)),
        not(inf(add(_X2, _Y2), 10)),
        eqv(_U2, sub(10, add(_X2, _Y2))),
        eqv(_U3, add(_X3, _Y3)).

Error(82, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y2, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, sub(add(10, sub(_X1, 1)), _Y1)),
        not(eqv(_U1, _R1)),
        inf(_X2, _Y2),
        eqv(_U2, sub(add(10, _X2), _Y2)),
        not(eqv(_U2, _R2)).

Error(84, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        not(eqv(_X2, 0)),
        eqv(_U1, _R1),
        inf(_X2, _Y2),
        eqv(_U2, sub(add(10,_X2), _Y2)),
        not(eqv(_U2, _R2)),
        eqv(_U3, _R3).

Error(86, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1,  Y1),
        eqv(_U1, _R1),
        sup(_X2, _Y2),
        eqv(_Y2, 0),
        eqv(_U2, sub(add(1, _X2), _Y2)),
        not(eqv(_U2, _R2)),
        not(eqv(_U3, _R3)).

Error(87, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        inf(_X1, _Y1),
        eqv(_U1, _R1),
        inf(_X2, _Y2),
        eqv(_U2, sub(add(_X2, 10), sub(_Y2, 1))),
        not(eqv(_U2, _R2)),
```

```
        eqv(_U3, sub(sub(_X3, 1),_Y3)),
        eqv(_U3, _R3).

Error(88, 2, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        not(eq(_U2, empty)),
        sup(_X2, _Y2),
        eqv(_Y2, 0),
        not(eqv(_U2, _R2)),
        eqv(_U3, _R3).

Error(89, 1, _X3, _X2, _X1, _Y3, _Y2, _Y1, _R3, _R2, _R1, _U3, _U2, _U1) :-
        not(eq(_Y3, empty)),
        not(eq(_U2, empty)),
        not(eq(_Y2, empty)),
        not(eq(_U3, empty)),
        inf(_X1, _Y1),
        not(eqv(_X1, 0)),
        eqv(_R1, _U1),
        inf(_X2, _Y2),
        not(eqv(_U2, _R2)).
```