Assistance à l'enseignant dans le cadre de l'EIAH Ambre : conception d'un générateur de problèmes

Stéphanie Riot, Duclosson Nathalie, Stéphanie Jean-Daubias

▶ **To cite this version:**

Stéphanie Riot, Duclosson Nathalie, Stéphanie Jean-Daubias. Assistance à l'enseignant dans le cadre de l'EIAH Ambre : conception d'un générateur de problèmes. RR-LIRIS-2004-036, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon. 2004. hal-04365244

HAL Id: hal-04365244
https://hal.science/hal-04365244

Submitted on 27 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RAPPORT DE RECHERCHE LIRIS

ASSISTANCE À L'ENSEIGNANT DANS LE CADRE DE L'EIAH AMBRE : CONCEPTION D'UN GÉNÉRATEUR DE PROBLÈMES

Stéphanie RIOT, Nathalie DUCLOSSON et Stéphanie JEAN-DAUBIAS

suite au stage de M2 recherche de Stéphanie RIOT au LIRIS - 2004

Résumé

Nous étudions dans ce rapport de recherche les possibilités d'adaptation par les enseignants d'un EIAH (Environnement Informatique pour l'Apprentissage Humain) destiné aux apprenants. Nous souhaitons concevoir des outils pour permettre à l'enseignant d'intégrer un EIAH à son contexte d'apprentissage et à la situation pédagogique qu'il souhaite mettre en place. Répondre aux attentes spécifiques des enseignants est en effet primordial si l'on souhaite que les EIAH soient davantage utilisés dans le système éducatif. Il apparaît donc aujourd'hui comme indispensable de considérer l'enseignant comme un utilisateur à part entière d'environnements d'apprentissage, et de l'assister dans l'utilisation de l'environnement destiné à ses élèves. Dans ce rapport, nous étudions cette notion d'adaptation dans le cas particulier de l'EIAH AMBRE (Apprentissage de Méthodes Basé sur le Raisonnement à partir de l'Expérience). Nous présentons la conception d'un environnement destiné à l'enseignant et notamment l'implémentation d'un outil de cet environnement permettant de générer de nouveaux exercices, adaptés aux besoins de l'enseignant.

MOTS-CLÉS : EIAH, adaptation, enseignant, génération de problèmes

Table des matières

Introduction

Ce rapport présente un travail de recherche réalisé au sein de l'équipe Cognition, Expérience et Agents Situés du laboratoire LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information) de Lyon. Il traite de l'assistance à l'enseignant pour l'utilisation d'environnements informatiques d'apprentissage avec ses élèves. Ce travail s'inscrit dans le cadre du projet AMBRE (Apprentissage de Méthodes Basé sur le Raisonnement à partir de l'Expérience) développé au LIRIS.

Traditionnellement, les EIAH (Environnements Informatiques pour l'Apprentissage Humain) sont destinés aux apprenants. Ils les aident à apprendre à leur propre allure, en suivant un parcours personnalisé et individualisé. Ils ne laissent toutefois que peu de place aux enseignants, qui sont pourtant les premiers concernés par ces environnements, puisque ce sont eux qui décideront ou non de leur intégration dans l'enseignement. Un EIAH ne sera en effet utilisé à l'école que si l'enseignant peut l'intégrer et l'adapter à sa démarche pédagogique. Il apparaît donc aujourd'hui comme primordial de considérer l'enseignant comme un utilisateur à part entière d'environnements d'apprentissage, et de l'assister dans l'utilisation d'un EIAH dans ses classes.

Pour permettre à un enseignant d'adapter et d'intégrer l'environnement AMBRE destiné aux apprenants dans son contexte d'apprentissage, nous souhaitons mettre à sa disposition un ensemble d'outils, comprenant notamment un générateur de problèmes. Ce système doit être capable de créer de nouveaux exercices à résoudre, en tenant compte des besoins des enseignants. Un tel système devra permettre de diversifier et d'augmenter le nombre de problèmes présents dans l'environnement, et de les adapter aux attentes des enseignants, et donc finalement aux apprenants eux-mêmes.

Nous présenterons dans ce rapport le travail effectué pour concevoir un outil d'assistance à l'enseignant pour l'utilisation de l'EIAH AMBRE, et notamment pour implémenter un générateur de problèmes pour cet environnement.

Nous présenterons auparavant le cadre de ce projet, puis le domaine de recherche qu'est l'EIAH et plus particulièrement le rôle que peut tenir un enseignant dans un tel environnement. Nous étudierons par la suite les principaux types de générateurs d'exercices existants, et les différentes méthodes utilisées dans le cadre de la génération de texte. Nous présenterons ensuite le travail réalisé et les résultats obtenus.

Chapitre 1

Cadre du projet

Ce travail de recherche s'inscrit dans le cadre du projet AMBRE et plus particulièrement dans le cadre de l'EIAH AMBRE - Problèmes Additifs (AMBRE-add), destiné à l'apprentissage d'une méthode de résolution de problèmes additifs. Nous présentons donc ci-après le projet AMBRE et son implémentation dans le domaine des problèmes additifs.

1.1 Le projet AMBRE

Le projet AMBRE [NOGRY et al. 02] est une étude pluridisciplinaire dont l'objectif est de concevoir des EIAH fondés sur le Raisonnement à Partir de Cas (RàPC) [KOLODNER 93] et destinés à l'apprentissage de méthodes de résolution de problèmes. Ces méthodes, proposées dans le cadre d'études en didactique des disciplines [ROGALSKI 94], sont fondées sur un classement des problèmes et des outils de résolution. Dans un domaine donné, appliquer une méthode consiste à savoir reconnaître la classe de problèmes dont relève le problème à résoudre, afin de savoir quelle technique utiliser pour le résoudre.

Un environnement AMBRE propose à l'élève de résoudre des problèmes en se fondant sur l'apprentissage à partir d'exemples. La stratégie d'apprentissage utilisée dans un EIAH AMBRE est la suivante. On présente dans un premier temps à l'apprenant quelques exemples de problèmes résolus, qu'on appelle problèmes types, puis on lui propose une activité de résolution de nouveaux problèmes. La résolution d'un problème par l'apprenant est guidée par l'environnement en suivant les différentes étapes du cycle AMBRE, qui est inspiré du cycle de RàPC :

Reformulation L'apprenant doit construire une nouvelle formulation du problème posé en décomposant l'énoncé du problème. Il doit pour cela mettre en évidence les traits pertinents du problème à résoudre (les traits de structure).

Choix du problème type L'apprenant doit choisir parmi les problèmes types présentés celui qui lui semble le plus proche du problème à résoudre, en se basant sur les reformulations des problèmes.

Adaptation L'apprenant doit construire la solution du problème à résoudre en adaptant celle du problème type choisi précédemment. Il doit pour cela appliquer la technique de résolution du problème type au problème cible.

Classement Une fois le problème résolu, l'apprenant doit l'ajouter à la base de cas en identifiant la classe de problèmes à laquelle il se rattache. Il doit pour cela associer le nouveau problème à un problème prototype, et donc l'insérer dans un groupe de problèmes. Il a également la possibilité de le considérer comme un nouveau problème prototype, ne pouvant être rattaché à aucun groupe existant, et ainsi créer un nouveau groupe.

Cette méthode fondée sur l'apprentissage à partir d'exemples doit permettre à l'apprenant, dans un domaine donné, de construire progressivement des classes de problèmes et de leur associer des techniques de résolution. Pour cela, l'EIAH AMBRE s'appuie sur une architecture qui permet d'expliciter la méthode à enseigner : SYRCLAD (Système de Résolution de problèmes basé sur une CLAssification du Domaine). SYRCLAD permet, pour un domaine donné, de définir les connaissances de la méthode de résolution et d'obtenir un résolveur de problèmes. Ce résolveur est un système à base de connaissances qui résout les problèmes en se fondant sur les classes de problèmes et en utilisant la méthode qu'il souhaite faire acquérir à l'élève, et non les connaissances expertes du domaine.

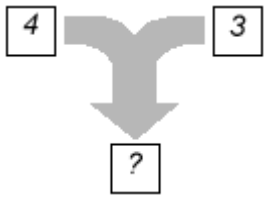
Le projet AMBRE a déjà été appliqué au domaine des problèmes de dénombrement pour les élèves de terminale scientifique. Il est maintenant mis en oeuvre dans le domaine des problèmes additifs proposés à l'école primaire.

1.1.1 AMBRE-add

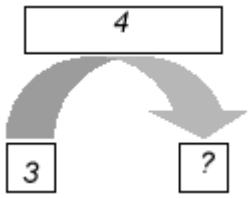
Un environnement informatique dédié au domaine des problèmes additifs a été développé en collaboration avec des enseignants, et des expérimentations de ce logiciel sont effectuées dans des classes. Le fonctionnement de ce logiciel suit bien sûr la stratégie présentée ci-dessus, même si les noms des étapes ont été modifiés, afin que les termes utilisés puissent être compris par des enfants. On parlera donc plutôt de : réécriture du problème (étape de reformulation), choix d'un modèle (étape de remémoration), rédaction de la solution (étape d'adaptation) et enfin bilan de la résolution du problème (étape de mémorisation).

Nous l'avons dit, dans un domaine donné, résoudre un problème consiste à reconnaître la classe du problème et lui associer un outil de résolution. Dans le domaine des problèmes additifs, les problèmes sont classés en trois catégories principales, elles-mêmes déclinées en sous-catégories. Le graphe de classification correspondant est présenté dans [DUCLOSSON 04]. Ces trois catégories sont :

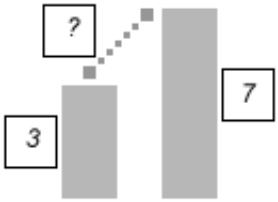
Réunion
Exemple : « Jean a 4 billes. Pierre a 3 billes. Combien ont-ils de billes ensemble ? »



Changement
Exemple : « Jean avait 3 billes. Il en a gagné 4. Combien en a-t-il maintenant ? »



Comparaison
Exemple : « Jean a 3 billes. Pierre en a 7. Combien de billes Pierre a-t-il de plus que Jean » ?



La résolution attendue des apprenants est illustrée par l'exemple suivant :

Enoncé du problème Alex avait 32 billes. A la fin de la récréation, il en a 45. Combien a-t-il gagné de billes pendant la récréation ?

Résolution attendue Le problème s'écrit : 32 + ? = 45
La solution s'écrit : ? = 45 - 32
La solution est : 13
La réponse est : Alex a gagné 13 billes.

Une présentation de l'environnement est donnée dans [DUCLOSSON 04], à travers notamment un exemple simple de résolution d'un problème additif.

Aujourd'hui, les recherches dans le cadre du projet AMBRE se tournent notamment vers les possibilités d'adaptation de l'environnement aux spécificités de l'apprenant ainsi qu'à celles du contexte d'apprentissage, par la création d'outils destinés aux enseignants. C'est dans ce cadre que s'inscrit cette recherche.

1.2 Thème de recherche

Cette recherche concerne l'assistance à l'enseignant pour l'utilisation de l'EIAH AMBRE-add. Nous souhaitons proposer à l'enseignant un environnement lui permettant de paramétrer le logiciel destiné à ses élèves : créer des séquences d'exercices destinées à l'ensemble de sa classe ou à certains apprenants, créer de nouveaux thèmes d'exercices, mais aussi et surtout générer des problèmes à résoudre.

L’enseignant est en effet celui qui connaît le mieux ses élèves, et qui est donc le plus à même d’adapter l’enseignement à chaque élève, pour mettre en place des activités en adéquation avec les besoins du terrain. Comme le souligne P. Leroux dans [LEROUX 02], « le pouvoir d’enseigner, c’est celui de former, de choisir les moyens d’apprendre pour les adapter à chaque élève, de soutenir et remédier à des apprentissages difficiles. Il existe souvent plusieurs moyens pour apprendre une même chose. Personne n’est donc plus à même d’effectuer ces choix, ce soutien qu’un enseignant. » Il est donc indispensable que l’enseignant puisse utiliser les EIAH en classe en les adaptant à la stratégie d’apprentissage qu’il a choisi, au contexte dans lequel il évolue, et à ses apprenants.

L’environnement que nous souhaitons concevoir est exclusivement destiné aux enseignants et devra donc répondre à leurs besoins, leurs attentes, et leur permettre d’intégrer et d’adapter l’EIAH AMBRE-add à leurs démarches pédagogiques.

Ce travail a pour but de concevoir un tel environnement, et d’implémenter dans ce cadre un générateur de problèmes. Ce générateur sera conçu pour l’environnement AMBRE-add, mais devra pouvoir être réutilisé dans tout autre domaine auquel le projet AMBRE pourrait être appliqué.

Chapitre 2

Etat de l'art

Dans ce chapitre, nous expliquerons tout d'abord la notion d'EIAH, en insistant notamment sur la place accordée aux enseignants dans ces environnements. Nous nous intéresserons par la suite plus particulièrement à la génération de problèmes, aux travaux existants dans ce domaine, et aux méthodes développées dans le cadre de la génération de texte.

2.1 Les EIAH

Le domaine de recherche qu'est l'EIAH a pour but la conception, le développement et l'évaluation d'environnements informatiques qui permettent à des êtres humains d'apprendre, dans l'objectif d'utiliser au mieux l'informatique pour l'enseignement. Ces outils doivent être des systèmes intelligents pour pouvoir s'adapter à l'apprenant, à sa situation et à ses compétences, et ainsi personnaliser son apprentissage. Ils s'appuient pour cela sur un modèle de l'apprenant et une représentation des connaissances. Les EIAH doivent fournir de l'aide à l'apprenant et être capables de diagnostiquer ses connaissances et ses erreurs.

C'est un domaine pluridisciplinaire, qui nécessite la collaboration de l'informatique (en tant que support), l'Intelligence Artificielle (IA) (pour la modélisation des connaissances et du raisonnement), l'Interaction Homme-Machine (IHM) (pour l'ergonomie des interfaces), la psychologie cognitive (pour la compréhension de l'apprentissage chez le sujet humain et l'évaluation des acquisitions), la didactique des disciplines et les sciences de l'éducation (pour la réflexion sur les connaissances à enseigner et les méthodes d'enseignement), et enfin les sciences de l'information et de la communication (pour la relation entre connaissances et médias).

2.1.1 La place de l'enseignant dans les EIAH

La majorité des EIAH sont aujourd'hui toujours centrés sur le duo ordinateur/apprenant et masquent donc encore le rôle de l'enseignant dans ces environnements. Ils ne considèrent pas assez l'enseignant comme utilisateur potentiel distinct de l'apprenant. On peut d'ailleurs constater que seule une très faible partie de l'intelligence des EIAH est dévolue à l'enseignant. Or il apparaît aujourd'hui comme primordial de commencer à considérer que la place de l'enseignant est distincte de celle de l'élève dans les environnements interactifs, et qu'une partie du système devrait alors être développée à son intention [BOUHINEAU 1996], [LEROUX 02].

Pour cela, il est nécessaire d'identifier les différents rôles que peut jouer un enseignant dans les EIAH. Nous nous baserons pour cela sur le travail de S. Jean-Daubias [JEAN-DAUBIAS 04], qui distingue l'enseignant concepteur, prescripteur et utilisateur.

2.1.1.1 Concepteur

Concepteur : L'enseignant peut participer directement à la conception de l'environnement. Dans ce cas, il est intégré à l'équipe projet ; on parle alors de conception participative. Cette méthode de collaboration permet d'intégrer l'enseignant, en tant qu'acteur, dans le processus de création, et ainsi de prendre en compte ses besoins tout au long du processus de développement, et non uniquement en amont de la production par l'intermédiaire d'un cahier des charges. Intégrer l'enseignant est une méthode très intéressante, car elle permet de créer un produit en adéquation réelle avec les besoins des usagers, les besoins du terrain.

Partenaire de conception : On parle dans ce cas de conception informative. L'enseignant n'est pas intégré directement à l'équipe projet, mais est considéré comme partenaire de cette équipe. On peut ainsi faire appel à lui de façon ponctuelle tout au long du processus de développement, afin de connaître son avis sur le produit en cours de création, ses besoins réels sur tel ou tel point, ou encore lui demander son aide lors de problèmes de type pédagogique.

Auteur : L'enseignant peut être lui-même concepteur de l'EIAH. Il utilise dans ce cas des outils appelés systèmes auteurs, mis à sa disposition pour lui permettre de réaliser les logiciels à vocation éducative. Les EIAH étant des logiciels extrêmement complexes, les systèmes auteurs se limitent généralement à la création de supports et de ressources de cours ou bien de tuteurs intelligents.

2.1.1.2 Prescripteur

L'enseignant a pour rôle de choisir le système qu'il fera utiliser à ses élèves, en fonction de ses besoins, ses préférences, ses choix pédagogiques personnels.

2.1.1.3 Utilisateur

Utilisateur secondaire : C'est le rôle qui est le plus souvent réservé à l'enseignant. Dans ce cas, l'EIAH est principalement centré sur l'apprenant, mais laisse tout de même une place à l'enseignant pour gérer son utilisation. Celui-ci peut ainsi adapter et paramétrer le système destiné à ses élèves, l'alimenter en situations pédagogiques, interagir avec les apprenants lors de l'utilisation de l'environnement, mais aussi faire le bilan de la session effectuée, le bilan des problèmes rencontrés par les apprenants. L'enseignant utilise ainsi l'EIAH afin de l'adapter à sa stratégie pédagogique, et à ses utilisateurs principaux, les apprenants.

Utilisateur principal : Le système s'adresse directement à l'enseignant, le but étant de lui proposer des outils qui l'aideront dans son travail. Ces types de logiciels ont ainsi une approche centrée enseignant, et non plus apprenant.

La catégorisation que nous venons de vous présenter n'est évidemment pas universelle, ni parfaite. Les noms attribués aux différents rôles peuvent ainsi, dans la littérature, être interprétés de façon différente. Nous nous en tiendrons toutefois à cette catégorisation dans ce rapport.

2.1.2 Les rôles de l'enseignant dans le cadre de notre recherche

Rappelons que l'outil que nous souhaitons mettre en place est un outil centré enseignant, qui a pour but de l'assister dans l'utilisation de l'EIAH destiné à ses élèves. D'après la classification précédente, dans ce travail de recherche, l'enseignant aura donc pour rôle majeur celui d'**utilisateur principal.** Il pourra de plus grâce à l'environnement proposé tenir en quelque sorte son rôle d' « **utilisateur secondaire** de AMBRE », en adaptant et paramétrant l'environnement destiné à ses élèves.

Nous faisons ici la distinction entre la notion d'adaptation du logiciel, qui concerne avant tout les aspects pédagogiques (modification de la démarche d'apprentissage, sélection du contenu, création de séquences en fonction du public cible, etc.), et la notion de paramétrage (ou paramétrisation), qui s'intéresse plutôt aux possibilités de modification de l'interface (choix des couleurs, de la langue, etc.).

D'autre part, certains enseignants ont joué le rôle de **partenaires de conception** pendant ce travail, pour permettre au le système que nous concevons d'être réellement adapté aux besoins du corps enseignant. On a fait appel à ces enseignants de manière ponctuelle afin d'avoir leur avis sur notre travail et connaître leurs besoins.

2.2 Générateurs de problèmes

Dans les systèmes d'apprentissage classiques, les problèmes proposés aux apprenants sont généralement issus d'une bibliothèque prédéfinie. Leur diversité et leur nombre en sont alors forcément limités. Ils ne peuvent pas non plus être adaptés exactement aux besoins des enseignants, à leur contexte d'apprentissage. La solution répondant à ce problème consiste en l'utilisation de générateurs de problèmes. Il existe comme nous allons le voir par la suite différentes catégories de générateurs : les générateurs automatiques, semi-automatiques ou manuels.

2.2.1 Historique

Les premiers programmes génératifs ont vu le jour dans les années 70 avec l'intégration dans les environnements d'apprentissage de techniques informatiques**.** Ces programmes utilisent des algorithmes permettant de générer problèmes et réponses. De tels systèmes ont la potentialité de « comprendre » le matériel enseigné. « Les systèmes génératifs sont associés à des domaines liés à des compétences en résolution de problèmes. Ces compétences s'acquièrent essentiellement par un processus d'apprentissage par l'action. Il faut disposer d'un grand nombre de problèmes et il devient coûteux de programmer à l'avance une solution à chaque nouveau problème, d'où la nécessité d'un programme génératif. » [BRUILLARD 97]. Les systèmes génératifs sont considérés comme les précurseurs des tuteurs intelligents. Certains chercheurs, dont Siklossy en 1970, cherchèrent à aller au-delà de ce type de programmes pour concevoir ce qu'ils nommaient des « tuteurs qui connaissent ce qu'ils enseignent ». Ils reprochaient en effet aux environnements d'apprentissage d'être incapables de répondre aux questions des apprenants, car ils ne connaissaient pas ce qu'ils enseignent. Les systèmes génératifs vont alors petit à petit faire place aux tuteurs intelligents.

Le cœur du tuteur artificiel, « qui connaît ce qu'il enseigne », est selon Siklossy le module de résolution. Ce module peut résoudre des problèmes et expliquer comment il les résout. Ce type de programme induit alors deux directions de recherche : l'écriture de résolveurs et l'amélioration des possibilités de dialogue. Ces directions de recherche vont alors prendre un essor important avec l'introduction de l'Intelligence Artificielle dans les EAO (Enseignement Assisté par Ordinateur). Cette intégration de l'IA dans les environnements d'apprentissage donne naissance à de nouveaux tuteurs : les tuteurs intelligents. Ces tuteurs sont capables de résoudre les problèmes, et donc de diagnostiquer plus finement les réponses de l'apprenant. Ils sont également capables d'effectuer des raisonnements, à l'aide d'un modèle de l'apprenant basé à la fois sur l'état des connaissances de l'apprenant et sur l'historique de ses interactions, ce qui leur permet d'offrir un enseignement réellement individualisé et adaptatif.

2.2.2 Typologie

2.2.2.1 Les générateurs automatiques

Les générateurs de problèmes automatiques utilisent le modèle de l'apprenant afin de générer des exercices adaptés à ce dernier. Comme l'explique G.Pecego dans [PECEGO 98], de tels générateurs sont capables de créer automatiquement toutes une batterie de problèmes axés sur une notion précise du cours que l'apprenant doit acquérir. Ils se basent pour cela sur l'état de ses connaissances et l'historique de son parcours. Dans IDEBUGGY [BURTON 82] par exemple, un problème est créé à chaque 'bug', chaque erreur de l'apprenant. Ce problème est capable de mettre en jeu la connaissance erronée en question. Dans SINT [MITROVIC et al. 96], le prochain problème à présenter à l'apprenant est choisi en fonction du modèle de l'apprenant et des connaissances pédagogiques du domaine. D'autres systèmes construisent quant à eux des problèmes dans le but de tester les connaissances acquises par l'apprenant. Les générateurs automatiques permettent de créer divers types de problèmes, QCM (Questions à Choix Multiples), exercices d'algèbre, de physique, de grammaire…

Quel que soit l'objectif des générateurs, ces systèmes ont tous un point en commun : ils sont automatiques, et n'offrent donc aucune interaction avec les enseignants. Ces derniers ne peuvent en effet pas influer sur les choix du système.

2.2.2.2 Les générateurs semi-automatiques

Les générateurs semi-automatiques sont similaires aux générateurs automatiques, dans le sens où ils construisent eux-mêmes les énoncés des problèmes, à la différence près qu'ils laissent intervenir l'utilisateur dans le processus de création. Ceux-ci ont donc la possibilité avec de tels systèmes d'influer sur les problèmes qui seront générés, en spécifiant un ensemble de contraintes sur l'exercice à générer. SYGEP [PECEGO 98] par exemple, est un système à base de connaissances qui crée des énoncés de problèmes dans des domaines divers, qui utilisent des dispositifs ou des expérimentations, c'est-à-dire des systèmes qui sont à la base du problème et donnent lieu à des questions (circuits

électriques, expériences de mécanique…). Les domaines d'application actuels de SYGEP sont l'électricité, la mécanique, la thermodynamique ou encore la chimie. Dans ce système, l'utilisateur (humain ou système pilote) peut fixer ses objectifs pédagogiques en spécifiant un ensemble de contraintes (tel que le niveau de difficulté) qui calibreront le problème généré. SYGEP permet la création de l'énoncé du problème et de sa solution, sous forme textuelle. De la même façon, le système CEP [GIROIRE 89] permet d'engendrer des exercices liés au monde réel, nécessitant l'introduction de directives par l'utilisateur et permettant d'engendrer des énoncés en langue naturelle.

2.2.2.3 Les générateurs manuels

La dernière catégorie de générateurs de problèmes est celle des générateurs manuels, qui, à l'inverse des deux catégories précédentes, ne construisent pas eux-mêmes les énoncés des problèmes, ni leur solution. C'est à l'enseignant que revient cette tâche. On trouve aujourd'hui de nombreux logiciels éducatifs permettant de générer des exercices manuellement. Ils peuvent générer des exercices divers, tels que les exercices à trous, les reconstitution de textes ou encore les appariements, des exercices sous forme de jeux (mots mêlés, croisés, puzzle, quiz, etc.), ou encore des exercices multimédias (lecture et écoute). Ils peuvent souvent être utilisés pour créer des énoncés dans divers domaines d'enseignement.

Dans la catégorie des générateurs manuels apparaissent notamment des systèmes auteurs. Un système auteur est un environnement permettant de créer un logiciel, en guidant l'auteur dans sa démarche. Dans le domaine particulier des EIAH, ils se limitent en général à la création de supports de cours numériques ou de tuteurs intelligents. Il existe différentes catégories de systèmes auteurs (sept selon [MURRAY 99]) : certains permettent de séquencer et organiser des cours magistraux en plusieurs unités d'apprentissage, selon une hiérarchie de cours, modules, leçons… ; d'autres encore permettent de créer des exercices divers de type fermé (exercices à trous, à choix multiples…) qui proposent une correction automatique, ou de type ouvert (question ouverte avec réponse à formuler directement à l'ordinateur ou sur une copie, présentation de ressources multimédias qui peuvent éventuellement servir de base à une réponse sur feuille de copie…). GenEval [2] par exemple, est un outil auteur qui permet de créer des exercices hypermédias ayant tous la même structure. L'enseignant saisit l'énoncé de l'exercice, les questions et les réponses, ces dernières pouvant être données selon trois niveaux progressifs. Les apprenants quant à eux prennent connaissance des énoncés, questions et réponses via l'ordinateur mais résolvent les exercices proposés sur papier.

D'autres logiciels de génération manuelle d'exercices sont quant à eux spécifiques à un environnement centré sur l'apprenant. L'environnement Creexo [3] par exemple, permet de créer tout type d'exercices, dans des disciplines diverses (mathématiques, géographie, etc.), et adaptés aux niveaux des apprenants. L'outil proposé à l'enseignant consiste en un éditeur d'énoncés. L'utilisateur va construire l'interface correspondant à l'exercice qu'il souhaite créer. Il choisit pour cela les éléments qu'il souhaite utiliser à l'interface (zones de saisies, de texte, case à cocher, etc.), les place, spécifie leur utilisation, et les diverses réponses attendues. Spécifier les réponses permettra au système de corriger directement le travail de l'apprenant, et lui indiquer si certaines réponses sont erronées. Les apprenants résolvent les exercices ainsi créés dans l'environnement apprenant associé, Readexo.

Les logiciels de cette catégorie sont donc assez simples d'utilisation : l'enseignant construit chaque énoncé, en précisant la ou les réponses attendues. Le système ne possède quant à lui aucune connaissance de l'exercice qu'il propose. Il n'est pas apte à résoudre lui-même les problèmes qu'il permet de créer, les réponses étant préenregistrées, et n'offre pas de fonctionnalités d'aide ou de diagnostic automatique des réponses de l'apprenant. Le seul diagnostic possible consiste à dire à l'apprenant s'il a répondu juste ou faux aux questions posées. L'aide quant à elle doit être préalablement définie par l'enseignant.

2.2.3 Conclusion

Le générateur à mettre en place dans le cadre de notre travail doit permettre à l'enseignant d'adapter les problèmes produits à son contexte d'apprentissage, à sa démarche pédagogique. Il est donc important que celui-ci puisse caractériser les exercices, c'est-à-dire saisir un ensemble de contraintes qui spécifieront ces problèmes. Notre générateur entre ainsi dans la catégorie des générateurs que nous avons appelés semi-automatiques. Les produits de génération rencontrés dans

cette catégorie ne semblent toutefois pas réellement appropriés à nos besoins, car ils permettent de créer des énoncés et non des problèmes au sens où nous l'entendons, c'est-à-dire associés à des connaissances et à une méthode de résolution. Pour AMBRE, il ne suffit pas de créer un énoncé mais il faut de plus que celui-ci ait du sens, pour l'apprenant bien sûr, mais aussi et surtout pour le résolveur de problèmes.

Nous voulons toutefois, comme c'est le cas dans la démarche utilisée pour créer SYGEP, permettre la création de problèmes indépendamment du domaine d'application, afin que notre générateur puisse être utilisé dans les différents domaines qui sont ou seront appliqués à AMBRE. Nous souhaitons ainsi mettre en place une architecture qui soit la plus générique possible.

2.3 Génération de textes

Pour générer des énoncés, il est nécessaire de s'intéresser à la problématique de génération de texte, problématique qui est souvent abordée dans les travaux d'IA. Les systèmes de génération de texte ont des approches différentes en fonction de leur finalité : certains permettent d'engendrer des histoires, d'autres de traduire des textes, d'autres encore des commentaires, des courriels de réponses, etc. Les applications à la génération de texte sont donc nombreuses et variées. Comme le précise G. Pecego dans [PECEGO 98], « l'important est de savoir ce que l'on souhaite faire exprimer par un système informatique pour déterminer l'approche à adopter. »

De nombreuses recherches et travaux ont été réalisés dans le domaine de la génération de texte. Nous en donnerons un aperçu rapide ici, afin de voir quelle méthode sera la plus efficace, la plus appropriée à la génération des énoncés dans le cadre des problèmes additifs. Nous nous baserons pour cela principalement sur le travail de S. Beauregard [BEAUREGARD 01].

2.3.1 Le modèle de génération

La génération de texte est souvent assimilée à un processus linéaire, prenant en entrée les buts communicatifs visés, qui seront finalement convertis vers du langage écrit, en sortie. On divise habituellement ce processus en trois modules :

La partie stratégique « Décider quoi dire » : on sélectionne les connaissances qui seront utilisés dans le discours.

La partie tactique « Décider comment le dire » : on utilise un processus de réalisation linguistique : utilisation de fonctions d'agrégation (combinaison de clause, réduction des conjonctions, etc.), de règles d'expressions référentielles (utilisation de pronoms, etc.), ou autres.

La partie moteur « Décider comment l'écrire » : on applique des règles morphologiques (conjugaison des verbes, accords des pluriels, etc.), d'orthographe (élision, contraction, ponctuation, etc.), et de formatage (police de caractères, paragraphes, etc.).

2.3.2 Niveaux de complexité de la génération

Texte figé Cette technique de génération de textes est la plus simple. Le texte est totalement figé, autant du point de vue de la morphologie, de l'orthographe et du formatage. Rapide à implémenter, cette méthode impose tout de même de prévoir toutes les phrases qui seront utiles dans le programme, dans toutes les situations. Le texte n'est donc en aucun cas paramétré par le système, et est donc totalement spécifique à l'application pour laquelle il a été implémenté.

Patrons à trous Les patrons à trous possèdent comme leur nom l'indique des trous à remplir avec du texte variable. Ils sont implémentés dans la plupart des logiciels de traitement de texte, et permettent par exemple de créer des relances à partir d'un fichier d'adresses clients. Ces environnements utilisant les partons à trous ne permettent toutefois pas d'implémenter de notions proprement linguistiques. Les accords en genre et nombre ne sont par exemple pas effectués par le système.

Systèmes à base de phrases lexicales On utilise dans ce type de systèmes des (morceaux de) phrases et non plus simplement des mots dans le lexique. Les divers morceaux d'une phrase peuvent alors être combinés assez simplement en implantant des règles syntaxiques.

Génération linguistique profonde C'est le niveau de génération le plus complexe. Il implémente toutes les étapes que nous avons présentées dans le modèle de génération. On les implante souvent via des langages de recherche puissants tels que Prolog ou Lisp. Ce type de génération permet d'obtenir une qualité améliorée des textes générés. Ils imposent toutefois de créer ce générateur, aucun artefact logiciel bien conçu, robuste et complet n'étant pour l'instant disponible sur le marché. Il est alors nécessaire de disposer de connaissances en IA et en linguistique formelle. De plus, ces générateurs requièrent des entrées complexes, et sont assez lourds à mettre en place.

Techniques hybrides Les techniques hybrides sont un mélange de génération profonde et de surface, qui permettent une bonne fluidité dans le texte de sortie. Certains systèmes permettent par exemple de combiner des éléments variables et des éléments prédéfinis. D'autres mélangent les patrons à trous et la génération de groupes nominaux.

2.3.3 Conclusion

Les systèmes de patrons à trous ou de texte figé ne peuvent être adaptés à notre domaine, étant donné leur rigidité. La génération profonde est quant à elle lourde à mettre en place mais a l'avantage de donner des résultats très bons. Une solution pour nous est d'utiliser une technique hybride, mélangeant génération profonde et de surface. On peut ainsi créer des phrases de qualité à partir d'éléments variables et de quelques morceaux de phrases figés.

2.4 Problématique

Le premier problème auquel nous nous confrontons dans ce travail est la notion d'adaptation d'un EIAH par un enseignant. Nous savons qu'un EIAH ne sera utilisé à l'école que si l'enseignant peut l'intégrer et l'adapter à sa démarche pédagogique. Il est donc primordial de tenter de répondre au plus près aux attentes des enseignants dans ce domaine si l'on souhaite que les EIAH soient davantage utilisés dans le système éducatif. Nous allons donc concevoir un outil d'assistance à l'enseignant pour l'utilisation de l'EIAH AMBRE. Mais peut-on réellement permettre à un enseignant d'adapter un EIAH à son contexte d'apprentissage, à sa démarche pédagogique ? Et si oui, comment ? Quel type d'outils peut-on lui proposer pour l'assister dans l'utilisation de l'EIAH destiné aux apprenants ?

Selon nous, un des outils qui peut répondre en partie à ce problème est un générateur de problèmes, qui permettra à l'enseignant de créer les exercices de son choix. Nous avons vu dans l'état de l'art qu'il n'existait pas actuellement de générateur d'exercices permettant de construire un problème dans le sens de AMBRE, c'est-à-dire associé à des connaissances et une méthode de résolution. Les logiciels permettent généralement uniquement de créer les énoncés des problèmes. Or, nous l'avons dit précédemment, pour AMBRE, il ne suffit pas de créer un énoncé mais il faut de plus que celui-ci ait du sens, pour l'apprenant bien sûr mais aussi et surtout pour le résolveur de problèmes. Or un énoncé en langage naturel n'a pas de sens pour ce dernier. Nous allons donc devoir créer nous-mêmes ce générateur sous forme d'un système à base de connaissances, afin que les problèmes générés puissent ensuite être interprétés par le module de résolution existant. Ceci constitue le second problème de cette recherche. Une des principales difficultés rencontrées dans ce cadre est la notion de généricité du générateur. Comment construire un système de génération de problèmes qui puisse être utilisé dans des domaines variés (tels que les dénombrements, les problèmes additifs, voire même des problèmes autres que mathématiques) ? Comment construire une architecture indépendante du domaine qui puisse servir à générer des problèmes linguistiquement corrects destinés à l'apprenant et interprétables par le résolveur ?

Chapitre 3

Conception de l'environnement enseignant

Dans le cadre de l'environnement AMBRE-add, EIAH destiné à l'apprentissage de méthodes de résolution de problèmes additifs, nous souhaitons proposer à l'enseignant un environnement lui permettant de paramétrer le logiciel destiné à ses élèves, créer des séquences destinées à l'ensemble de sa classe ou à certains profils d'apprenants, et plus particulièrement générer des problèmes à résoudre. Cet environnement, exclusivement destiné aux enseignants, doit donc répondre à leurs besoins, et leur permettre d'intégrer et d'adapter l'EIAH AMBRE-add à leurs démarches, à leurs stratégies pédagogiques, mais aussi aux contextes dans lesquels ils évoluent.

Pour cela, la première partie de mon travail a été de réfléchir à la mise en place d'un outil qui répondrait au mieux aux besoins des enseignants. Un outil à la fois utile, et utilisable (adapté à des non experts). La mise en place d'une telle application soulève alors deux questions fondamentales : Que faut-il mettre à disposition des enseignants (Quels sont leurs besoins ? Et quel type d'assistance peut fournir un tel outil ?) ? Et comment leur présenter les divers éléments de notre outil (Quelle organisation utiliser à l'interface ?) ?

3.1 Acteurs de la conception

Pour avoir des connaissances plus riches et plus proches de la réalité, nous avons souhaité travailler avec des enseignants. Il était en effet impératif de pouvoir collaborer avec eux, qui sont, ne l'oublions pas, les premiers concernés par notre outil. Connaître leurs attentes, leurs besoins mais aussi leurs habitudes de travail et le comportement de leurs élèves en matière de problèmes additifs. Nous avons eu la chance de pouvoir travailler avec deux enseignants qui exercent ou ont exercé en école primaire. L'un connaissait mieux les élèves de CE, l'autre ceux de CM, ce qui nous a permis de prendre en compte les besoins et méthodes adaptées à ces différents niveaux. Les difficultés ne sont bien sûr pas les mêmes face à un problème de type additif lorsque l'on est en CE1 ou en CM1. Les difficultés de langue et de structure de phrase sont notamment plus accentuées dans les petites classes. Ces différences sont importantes à prendre en compte dans notre logiciel, afin qu'il puisse fournir des exercices adaptés aux petits comme aux plus grands.

Ces enseignants ont donc eu dans notre travail un rôle de partenaires de conception. On parle dans ce cas de conception informative [JEAN-DAUBIAS 04]. Les enseignants n'étaient pas intégrés directement à l' « équipe projet », mais bien considérés comme partenaires de cette équipe. On a ainsi pu faire appel à eux de façon ponctuelle tout au long du processus de conception, afin de connaître leur avis et leurs idées sur le produit en cours de création, leurs besoins réels sur tel ou tel point, ou encore leur demander de l'aide, des conseils, lors de problèmes de type pédagogique.

3.2 Expression des besoins

Notre première tâche dans ce travail a été d'identifier le type d'assistance que pouvait fournir l'environnement enseignant que nous souhaitons concevoir. Nous avons dans cet objectif élaboré une liste des activités qui nous semblaient utiles voire nécessaires à l'enseignant pour lui permettre une intégration la plus totale possible du logiciel AMBRE-add dans son environnement de travail :

- La génération de problèmes
- La création de séquences d'apprentissage (ensemble d'activités destinés aux l'apprenants)
- La création de thèmes d'exercices
- Le paramétrage de l'environnement élève
- La création de classes
- Et la distribution du travail (séquences, exercices) aux classes, aux élèves

Cette liste a ensuite été présentée et discutée avec des enseignants.

Nous ne traiterons pas par la suite des trois derniers points, qui sont faciles à implémenter et ne posent pas de problèmes ni d'intérêts particuliers d'un point de vue recherche. Le paramétrage de l'environnement élève consiste simplement en l'adaptation du logiciel élève, c'est-à-dire la modification de son interface (choix des couleurs, de la langue, etc.).

Nous allons maintenant aborder les trois autres activités, en expliquant les besoins et les choix effectués pour chacune d'entre elles. Précisons que notre travail pour cette recherche est principalement tourné vers la première activité, la génération de problèmes, pour laquelle nous avons conçu un système de génération. Notre travail concernant les activités de création de séquences et de thèmes d'exercices est quant à lui limité à l'étude des besoins des enseignants.

3.2.1 La génération de problèmes

C'est l'outil qui nous intéresse le plus, et qui constitue le cœur de notre application. Nous voulons permettre à un enseignant de pouvoir créer ses propres problèmes, et de pouvoir les adapter au public à qui ils les destinent. Il nous semblait de plus important de pouvoir lui permettre de créer avec exactitude un énoncé souhaité, ou, à l'inverse, de pouvoir en créer un complètement « au hasard ». Pour arriver à une telle souplesse, et réussir à contrôler la difficulté des problèmes produits, nous avons décidé d'utiliser l'approche de la génération à partir de contraintes. L'outil proposé devra donc permettre à l'enseignant de créer automatiquement des problèmes à résoudre, à partir d'un ensemble de contraintes qu'il pourra définir, lui permettant d'adapter au maximum les problèmes générés à ses besoins.

Afin que la tâche de création soit moins laborieuse et surtout moins longue pour l'enseignant, nous voulions lui éviter d'avoir à créer un à un les nouveaux énoncés. Nous avons donc décidé qu'un même jeu de contraintes pourrait permettre la création de plusieurs exercices. Moins il y aura de contraintes plus les problèmes générés seront variés.

Une batterie d'exercices (créé par les concepteurs) sera bien sûr initialement fournie à l'enseignant avec le logiciel Ambre, afin qu'il puisse commencer à l'utiliser avec ses élèves, sans être obligé de créer tout de suite lui-même ses exercices. La génération de problèmes à résoudre doit en effet être un plus pour les enseignants, et non une obligation. Ce passage semble toutefois nécessaire pour augmenter et diversifier la batterie d'exercices, et adapter les énoncés au public à former.

Les contraintes que l'enseignant pourra définir ont été partagées en différentes catégories :

Structure La structure des problèmes à générer comprend le type du problème (réunion, changement ou comparaison) et la place de l'inconnue.

Traits de surface Les traits de surface sont les éléments qui permettent d'« habiller » les énoncés produits. L'enseignant peut choisir certains éléments de cette catégorie : les thèmes, les objets, et les personnages. Les objets peuvent de plus être déclinés en catégories pour augmenter la difficulté des problèmes. Nous allons l'illustrer par un exemple : « Julie a 3 tulipes. Sophie a 4 roses. Combien ont-elles de fleurs ensemble ? » L'apprenant n'additionne pas ici que des fleurs mais bien des catégories de fleurs, ce qui lui demande un plus grand effort de compréhension et d'analyse, notamment lors de l'étape de réécriture du problème.

Valeurs L'enseignant peut choisir les valeurs qui seront utilisées dans les problèmes, en ce qui concerne les éléments connus (et non l'élément à déterminer), selon deux modes : le mode *Intervalle* dans lequel il peut saisir un intervalle global de valeurs pour les éléments, et l'écart (sous la forme aussi d'un intervalle) souhaité entre les deux valeurs retenues ; ou le mode *Valeurs* où l'enseignant peut fixer une valeur ou un intervalle pour chaque élément connu de l'énoncé. Dans ces deux modes, l'enseignant peut également, s'il veut simplifier les opérations, interdire l'utilisation de la retenue dans les calculs.

Complication C'est cette partie qui a le plus nécessité l'intervention des enseignants, dont nous avions besoin de connaître l'avis et les besoins. Cette catégorie concerne toutes les options qui permettront de compliquer l'énoncé du problème, et donc de les adapter au niveau des élèves. On propose ici différents types de difficultés à intégrer dans les exercices: des complications de langue, et de l'énoncé lui-même. Les complications de langue prennent en compte la difficulté du vocabulaire employé et de la situation décrite dans l'énoncé. On entend notamment par difficulté de la situation le choix des tournures des phrases utilisées. Par exemple, l'énoncé « Jean a 2 roses. Paul en a 5 de plus que lui. Combien Paul a-t-il de roses

? » est relativement simple car les termes ‘de plus’ indiquent l’opération à effectuer. Par contre, on peut compliquer cette situation en écrivant « Jean a 2 roses. Il en a 5 de moins que Paul. Combien Paul a-t-il de roses ? ». L’énoncé est plus difficile dans ce sens car il utilise les termes ‘de moins’ alors que l’opération à faire est une addition. Les complications de l’énoncé comprennent quant à elles : l’écriture des nombres en lettres, la modification de l’ordre des propositions de l’énoncé, et l’ajout de phrases inutiles, introduisant éventuellement dans le problème des données non pertinentes (c’est-à-dire inutiles pour la résolution). Ceci pour accroître bien sûr la difficulté des exercices : plus de texte à lire mais aussi une distinction à faire entre les données utiles à la résolution du problème et les données inutiles. Nous avons pour cette dernière option de complication mené une réflexion sur les phrases qu’il était possible d’ajouter à un énoncé, et sur la difficulté qu’elles pouvaient introduire pour l’élève. Nous avons créé des catégories de phrases inutiles, en fonction de la difficulté ajoutée.

Précisons qu’aucune des contraintes citées précédemment n’est obligatoire pour la création d’exercices. L’enseignant peut donc choisir de n’en définir aucune, juste quelques-unes, ou toutes s’il a une idée précise d’énoncé en tête. Il est donc totalement libre dans cette phase de création, et peut ainsi générer les exercices à tout moment. Les contraintes non spécifiées par l’enseignant seront définies aléatoirement par le système. L’enseignant peut de plus attribuer aux problèmes crées un nom, un niveau (par exemple CE1 1er trimestre) et une description, afin de les retrouver plus facilement lors de la création de séquences.

On constate, à travers ces différentes contraintes, que tout est mis en place pour permettre à l’enseignant de créer des exercices variés, en jouant notamment sur les nombreuses possibilités de complication qu’il existe, que ce soit l’utilisation de catégories d’objets dans l’énoncé ou l’ajout de phrases dans l’énoncé. Nous avons essayé de tenir compte de l’ensemble des techniques utilisées par les enseignants rencontrés pour diversifier et augmenter la difficulté des problèmes additifs présentés aux apprenants. Le nombre de contraintes peut paraître important pour des exercices qui paraissent a priori simples, mais cet éventail de contraintes doit permettre à chaque enseignant d’adapter les problèmes à ses élèves, à leur niveau, à toute sa classe ou à un profil particulier.

3.2.2 La création de séquences d’apprentissage

Une séquence d’apprentissage est définie comme un « ensemble continu ou discontinu de séances articulées entre elles dans le temps et organisées autour d’une ou plusieurs activités en vue d’atteindre des objectifs fixés par les programmes d’enseignement » [1]. Nous avons choisi de travailler avec les séquences et non en terme de séances, afin de pouvoir nous adapter à toutes les configurations d’utilisation du logiciel. En effet, si certaines classes consacrent des séances entières à la résolution de problèmes via l’EIAH AMBRE, d’autres peuvent utiliser le logiciel de façon plus ponctuelle et plus libre. Citons par exemple les exercices de « rattrapage » pour revoir certaines notions qui peuvent être faits pendant n’importe quelle séance d’enseignement, lorsque l’enfant a un moment de libre. Il ne fera donc peut être pas tous les exercices en une fois, mais étalés sur plusieurs heures, jours ou semaines.

Nous voulons offrir la possibilité à l’enseignant de créer de telles séquences en utilisant le matériel pédagogique (les problèmes) qu’il a créé. Cette activité lui permettra de mettre en place sa stratégie, sa démarche pédagogique, en intégrant à sa façon les problèmes dans une séquence. On propose pour cela à l’utilisateur différentes manières de créer une séquence : la création manuelle et la création « automatisée ».

Création manuelle L’enseignant sélectionne ici les exercices qu’il souhaite intégrer dans la séquence qu’il crée. Il peut choisir de les faire apparaître de façon chronologique ou de façon aléatoire, selon sa stratégie. Il peut par exemple vouloir créer une séquence où le niveau de difficulté augmente au fur et à mesure des problèmes à résoudre. Dans ce cas, il choisira lui-même l’ordre de présentation des problèmes de la séquence. L’enseignant peut ensuite définir le comportement des exercices de la séquence : nombre d’essais autorisé par étape de l’exercice (et que faire si ce nombre est atteint ?), nombre d’accès à la vérification ou à l’aide autorisé pendant l’exercice, fonctionnement du diagnostic (toujours diagnostiquer ou laisser l’apprenant se tromper à certaines étapes de la résolution), etc. On définit par défaut ce

comportement pour tous les exercices de la séquence. Toutefois, si l'enseignant souhaite intégrer une progression au niveau de ce comportement dans la séquence, il a la possibilité de définir un comportement différent en fonction des exercices (il peut ainsi par exemple mettre un diagnostic maximum en début de séquence, et le diminuer au fur et à mesure).

Création automatisée Dans ce cas, l'enseignant ne choisit que le(s) dossier(s) d'exercices dans lequel le système pourra « piocher », et le nombre d'exercices qu'il souhaite mettre dans la séquence. Charge ensuite au système de choisir aléatoirement les exercices. L'enseignant pourra tout de même s'il le souhaite modifier les choix du système (supprimer, ajouter ou remplacer un problème), ou tout simplement ordonner les exercices, définir un comportement…

3.2.3 La gestion des traits de surface

Afin que l'enseignant puisse réellement adapter les problèmes générés à son environnement, au contexte dans lequel ses élèves évoluent, nous lui offrons ici la possibilité de gérer lui-même les traits de surface qui seront utilisés dans les exercices. Il pourra donc créer/modifier/supprimer des thèmes d'exercices, et des personnages.

A chaque thème est associé un ensemble d'objets et de personnages qui lui est spécifique. A cela s'ajoute également un ensemble d'actions qui peuvent être effectuées sur les objets du thème (action de 'gagner' pour des billes), des événements de temps et quelques données utiles au processus de complication. L'enseignant peut ici ajouter des objets, des personnages et/ou des actions, et ainsi modifier les thèmes et les adapter à ses élèves.

On peut également utiliser dans les exercices des personnages globaux, c'est-à-dire qui ne sont pas spécifiques à un thème, mais plutôt transverses à tous les thèmes, et qui peuvent donc a priori être utilisés dans chacun d'entre eux, tels que des prénoms (Julie, Paul). Il peut en effet être intéressant pour un enseignant d'intégrer ses élèves comme personnages des problèmes qu'ils auront à résoudre. Les enfants aiment en effet voir leur propre prénom ou ceux de leurs copains dans les énoncés. Cela les amuse, et peut peut-être même les motiver, leur donner encore plus envie de résoudre les exercices proposés.

3.3 Travail sur l'interface

Nous avons voulu, pour cet environnement, travailler un maximum sur l'interface proposée aux enseignants. Il faut en effet, si l'on souhaite que notre application soit utilisée à l'avenir, concevoir une interface conviviale, la plus claire et la plus compréhensible possible, dans le but que les utilisateurs puissent se l'approprier facilement et rapidement. Éviter la lourdeur de l'interface demande de trouver une manière d'organiser et de présenter les options disponibles à l'enseignant la plus simple et la plus agréable possible. Ce qui n'est évidemment pas une tâche facile... Les opinions, les avis des personnes du corps enseignant sur l'environnement proposé sont indispensables à la réalisation d'un outil à la fois utile et utilisable.

L'environnement enseignant a été développé avec Delphi 7 Professionnel, outil également utilisé pour le logiciel destiné aux apprenants.

Nous ne présenterons dans cette partie que l'outil de génération de problèmes, qui, comme nous l'avons précisé, a été le cœur de notre travail pour cette recherche.

Pour ne pas alourdir l'interface de génération de problèmes malgré le grand nombre de contraintes que l'utilisateur peut définir, nous avons choisi de travailler avec des onglets. Chaque catégorie de contraintes présentée précédemment correspond ainsi à un onglet. Cette présentation permet non seulement d'harmoniser l'interface avec celle du logiciel destiné aux apprenants, mais aide surtout l'enseignant à comprendre et à distinguer les différents types de contraintes qu'il peut définir.

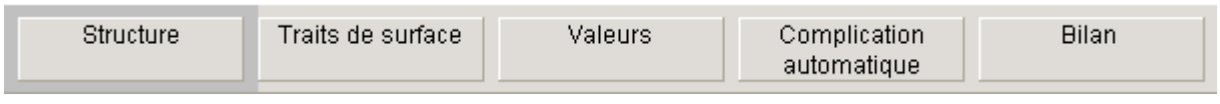


Figure 1 : Onglets de l'interface du générateur de problèmes

A ces quatre onglets est ajouté un dernier onglet de bilan, qui permet de présenter et de rappeler à l'enseignant l'ensemble des choix qu'il a effectués. Ce jeu de contraintes (ou trame du problème) peut alors être enregistré afin d'être réutilisé ultérieurement, pour créer de nouveaux problèmes.

Nous l'avons dit précédemment, aucune des contraintes proposées n'est obligatoire pour la création de problèmes. De ce fait, aucun onglet de la fenêtre n'est obligatoire, ni d'ordre imposé pour y accéder. Les onglets sont présentés à l'interface dans l'ordre utilisé ci-dessus, qui n'est pas, nous l'avons dit, un ordre d'accès imposé à l'enseignant, mais plutôt un ordre logique, nous paraissant approprié dans le cas d'un processus de définition de contraintes d'un problème : d'abord définir la structure de l'exercice, puis les traits de surface et les valeurs qui y seront utilisés. Une fois ceci fait, envisager la complication de l'énoncé. Donc si l'ordre d'accès aux onglets n'est pas imposé, il est toutefois fortement conseillé.

D'autre part, tout au long de la phase de génération de problèmes, un aperçu d'énoncés créés à partir des contraintes déjà définies est accessible à l'enseignant. Cela lui permet de mieux comprendre, de mieux visualiser le résultat de ses actions, de ses choix de contraintes sur les problèmes à générer. Il peut en effet, s'il ouvre cet aperçu, voir l'évolution d'un énoncé au fur et à mesure de la définition de contraintes. Cet aperçu affiche uniquement un exercice et sa technique de résolution.

Une fois que l'enseignant a défini les contraintes de son choix (éventuellement aucune), il peut générer les exercices répondant à ses besoins.

L'interface actuelle est présentée en annexe. Précisons que cette interface n'est pas le produit final, et sera donc soumis à de nouvelles améliorations, afin de faciliter encore son utilisation par les enseignants.

Chapitre 4

Implémentation : générateur de problèmes

Dans ce chapitre, nous présentons l'architecture de notre système de génération de problèmes et le fonctionnement du cœur de cette architecture, qui sont indépendants de tout domaine d'application. Rappelons que cette notion d'indépendance, de généricité du système est très importante car ce générateur doit pouvoir être réutilisé dans le cadre des différents domaines que traite ou traitera AMBRE. Il ne doit donc surtout pas être spécifique au domaine des problèmes additifs, dans lequel nous travaillons plus particulièrement.

4.1 Architecture de l'application

Nous avons conçu entièrement le générateur de problèmes pour AMBRE, étant donné qu'aucun outil existant ne répondait correctement à nos besoins. Il a été implémenté en Prolog, pour pouvoir communiquer aisément avec le résolveur de problèmes de AMBRE, lui-même implémenté dans ce langage.

Notre système de génération est un processus linéaire qui prend en entrée le jeu de contraintes spécifié par l'enseignant (et présenté dans la partie précédente) pour générer un problème, et qui fournit en sortie deux éléments : l'énoncé du problème généré en langue naturelle destiné aux apprenants, et un modèle de ce problème, appelé modèle descriptif, qu'il doit fournir au résolveur de problèmes de AMBRE, Syrclad [GUIN-DUCLOSSON 99].

Le modèle descriptif est une représentation du problème qui se veut la plus proche possible de l'énoncé en langage naturel. Il décrit une situation concrète, qui est celle présentée dans l'énoncé : on y retrouve non seulement les valeurs connues de l'énoncé et la question à laquelle il faut répondre, mais également les traits de surface de celui-ci : objets et personnages, ainsi que leurs relations.

L'architecture de notre système de génération est présentée sur la figure 2 ci-après.

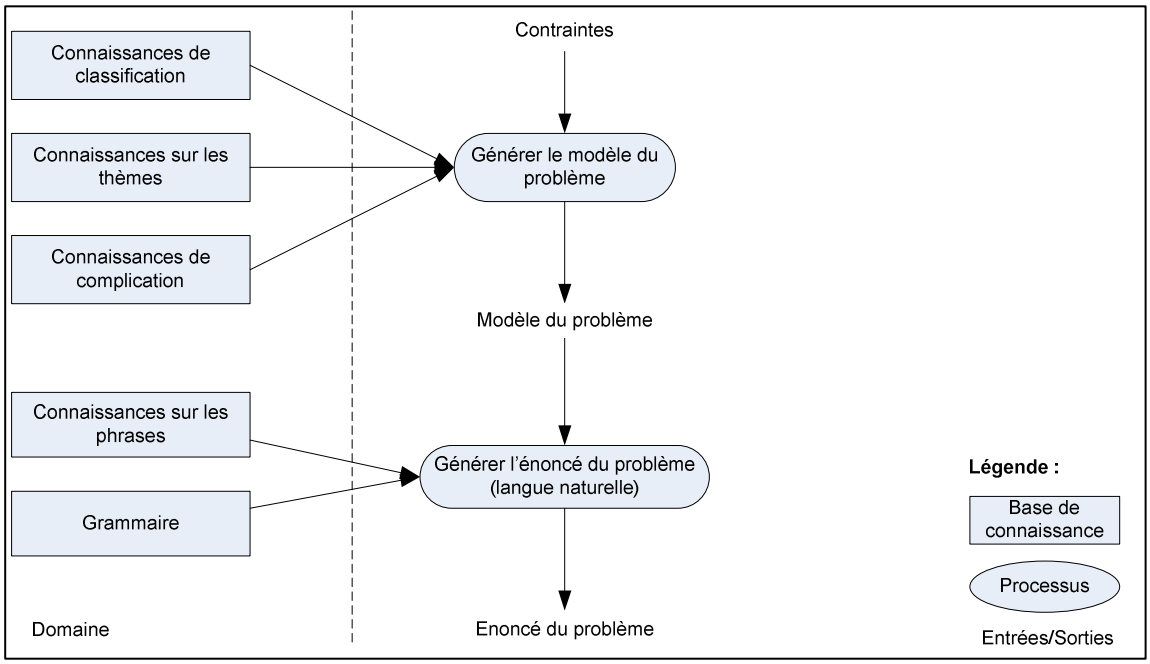


Figure 2 : Architecture du système de génération

L'architecture de ce système de génération et le processus de génération sont indépendants du domaine. Seul le contenu des bases de connaissances est modifié en fonction du domaine pour lequel le générateur est utilisé. Ainsi, pour utiliser cette architecture dans un domaine autre que celui des problèmes additifs, il faut qu'un expert du nouveau domaine fournisse l'ensemble des bases de connaissances présentées sur le schéma (connaissances de classification, connaissances sur les thèmes, connaissances de complication, connaissances sur les phrases et grammaire), adaptées à son domaine d'expertise en respectant un formalisme unique quel que soit le domaine. Ce formalisme permet au noyau du générateur d'utiliser les connaissances fournies.

Nous allons présenter ci-après l'ensemble des bases de connaissances et des processus utilisés dans notre système de génération de problèmes. Nous expliquerons ainsi le déroulement du processus de génération d'un exercice, en insistant sur les éléments qui sont indépendants du domaine d'application.

Connaissances de classification

Dans chaque domaine d'application, un expert fournit à Syrclad, et donc par extension à notre générateur, un graphe de classification de problèmes. Ce graphe est utilisé dans le résolveur pour classer le problème. Le graphe de classification est bien évidemment dépendant du domaine, mais sa structure de représentation et d'utilisation est la même quel que soit le domaine. Cette classification est une hiérarchie de classes, dans laquelle les sous-classes d'une classe sont différenciées par la valeur d'un seul attribut du problème, appelé attribut discriminant de la classe. Ce graphe permet ainsi de connaître l'ensemble des spécificités et attributs d'une classe donnée.

Connaissances sur les thèmes

Pour pouvoir générer un problème, nous avons besoin de connaître le thème et donc les traits de surface qui lui sont associés. De telles connaissances sont bien évidemment dépendantes du domaine pour lequel elles sont utilisées. Par contre, leur représentation est elle indépendante du domaine. Les thèmes par exemple sont introduits à l'aide du prédicat `theme`, les objets à l'aide du prédicat `objet`. Ce dernier prédicat permet de définir un objet (premier argument), son genre (deuxième argument) et son pluriel (troisième argument), et le (ou les) thème(s) au(x)quel(s) il appartient (quatrième argument).

Exemple :
```
theme (jeu). /* soit jeu un thème */
objet(bille, feminin, billes, jeu). /* soit bille un objet du thème
jeu, qui est de genre féminin et dont le pluriel est billes */
```

Actuellement, on a pu le constater sur l'exemple ci-dessus, les connaissances sur les thèmes sont également utilisées comme lexique pour la grammaire, étant donné qu'elles définissent, pour chaque élément du thème, les accords, genre ou nombre utiles à la génération linguistique de l'énoncé. On pourrait toutefois imaginer de séparer les connaissances lexicales des connaissances purement thématiques.

Connaissances de complication

Pour pouvoir compliquer un exercice, et donc son énoncé, ce qui revient principalement dans le cadre des problèmes additifs à changer l'ordre des phrases et ajouter des phrases inutiles, il est nécessaire de fournir des connaissances de complication au système de génération. Ici, ces connaissances sont : comment et dans quel cas est-il possible de changer l'ordre des phrases utiles (au nombre de trois) du problème ? Quelles phrases inutiles peut-on ajouter aux problèmes, et où les placer ?

```
complication -> melange, phrases_inutiles
```

Exemple pour le mélange : on emploie le prédicat melange qui pour une classe donnée (argument 1) et à partir d'une liste de phrases à mélanger (argument 2) spécifie le ou les mélanges possibles sous forme de listes de phrases ordonnées (argument 3). On se place pour cet exemple dans le cas de la classe reunion où l'inconnue est le résultat (le nombre d'objets des deux personnages ensemble). Dans cette classe, le seul mélange possible est de placer la question en tête de l'exercice.

```
melange(classe, [phrase1,phrase2,question], [question,phrase1,phrase2]) .
```

Processus : Générer le modèle du problème

Pour permettre la génération d'un problème dans un domaine donné, un expert doit avoir fourni au système un graphe de classification de problèmes du domaine, des connaissances de complication et des connaissances sur les thèmes. Un utilisateur de l'outil enseignant peut alors fournir au processus de génération du modèle les contraintes (de structure, surface, valeur et complication) qu'il a définies à l'interface et qui permettent de spécifier le problème à créer.

Le processus en question doit alors créer ce que nous appellerons le modèle du problème. Ce modèle est en fait un modèle descriptif étendu, puisqu'il contient le modèle descriptif du problème mais précise également quelques informations absentes du modèle descriptif, tels que la classe du problème ou encore son thème. Pour construire ce modèle, le processus a pour objectif de compléter les contraintes définies par l'enseignant, en choisissant notamment (aléatoirement) des « valeurs » pour celles qui n'ont pas été précisées. Rappelons que l'enseignant peut ne spécifier aucune contrainte, ou seulement quelques-unes sur un problème. Une fois créé, le modèle du problème sera fourni au processus suivant, qui l'utilisera pour générer l'énoncé en langue naturelle.

Grammaire

L'expert du domaine doit fournir une grammaire au système de génération, c'est-à-dire un ensemble de structures de phrases qui pourront être utilisées dans le domaine. Dans le cadre des problèmes additifs, cette grammaire permet de générer des phrases affirmatives comme interrogatives, à partir d'un ensemble de mots. Les unités syntaxiques de base sont les groupes sujets, les verbes, et les compléments. La structure simplifiée d'une phrase pourrait être décrite comme suit :

```
phrase -> sujet, verbe, complements.
```

Un groupe sujet peut être un groupe nominal, une conjonction de groupes nominaux, ou un pronom personnel. Les compléments sont quant à eux déclinés sous forme de compléments d'objets directs, indirects ou encore prépositionnels. La structure simplifiée d'une question pourrait quant à elle être décrite comme suit :

```
question -> mot interrogatif, sujet, verbe interrogatif ,complement.
Ou : question -> mot interrogatif, complement,sujet, verbe interrogatif.
```

Les structures utilisées sont en réalité un peu plus compliquées, étant donné que certaines phrases utilisent des compléments de temps (dans les exercices de changement), des adverbes, dont la place dans la phrase peut changer en fonction des contraintes de complication. Par contre, nous n'utilisons pas de propositions relatives, d'adjectifs, ou autres qui pourraient être nécessaires dans d'autres domaines d'application. La grammaire est donc dépendante du domaine.

Connaissances sur les phrases

Les énoncés générés, notamment la structure de leurs phrases, dépendent de la classe à laquelle le problème appartient. Il est donc nécessaire pour pouvoir générer l'énoncé en langue naturelle, de savoir quelles phrases ou plutôt quelles structures de phrases (de la grammaire) pourront être utilisées pour le problème à générer. Ce sont ces informations que contiennent les connaissances sur les phrases. Elles permettent d'associer à la classe du problème les structures de phrases utilisables, et les éléments du problème qui seront associés à ces structures. Elles permettent donc de répondre aux questions suivantes : Quelles (structures de) phrases peut-on utiliser pour ce problème ? Et quels éléments (objets, personnages) du modèle descriptif va-t-on utiliser dans ces phrases ?

Processus : Générer l'énoncé du problème (en langue naturelle)

Pour générer un énoncé en langue naturelle, le processus doit récupérer le modèle du problème, créé auparavant. Ce modèle contient en effet toutes les informations utiles (valeurs, objets…) de l'énoncé. Le processus va ainsi pouvoir générer l'énoncé en langue naturelle, en utilisant pour cela les connaissances de phrases et la grammaire du domaine. Les connaissances de phrases permettent au système de prendre les décisions d'ordre conceptuel (« Décider quoi dire ») : on spécifie ici les phrases qui seront générées (leur structure), en affectant à chacune d'entre elles les éléments qui la composeront (qui proviennent du modèle descriptif du problème : objets, personnages, etc.). Une fois les décisions d'ordre conceptuel prises, le processus passe à l'étape de génération de texte, et met donc en place différents mécanismes linguistiques : traitements syntaxiques (« Décider comment le dire »), lexicaux et morphologiques (conjugaison des verbes, accord des pluriels, etc.) (« Décider comment l'écrire »). On applique notamment des règles d'expressions référentielles (comme l'utilisation de pronoms pour réduire les répétitions d'entités), et des règles d'orthographe telles que l'élision (le → l'), la contraction (de + les → des) ou la ponctuation. Ces traitements permettent de rendre le texte plus fluide, plus naturel et de meilleure qualité. En sortie de ce processus, on obtient alors l'énoncé en langue naturelle du problème généré.

Notre architecture peut être représentée sur trois niveaux, qui permettront de distinguer plus facilement les éléments dépendants du domaine de ceux qui sont indépendants, comme nous pouvons le voir sur la figure 3. Le formalisme (légende) utilisé pour la figure 2 est réutilisé ici.

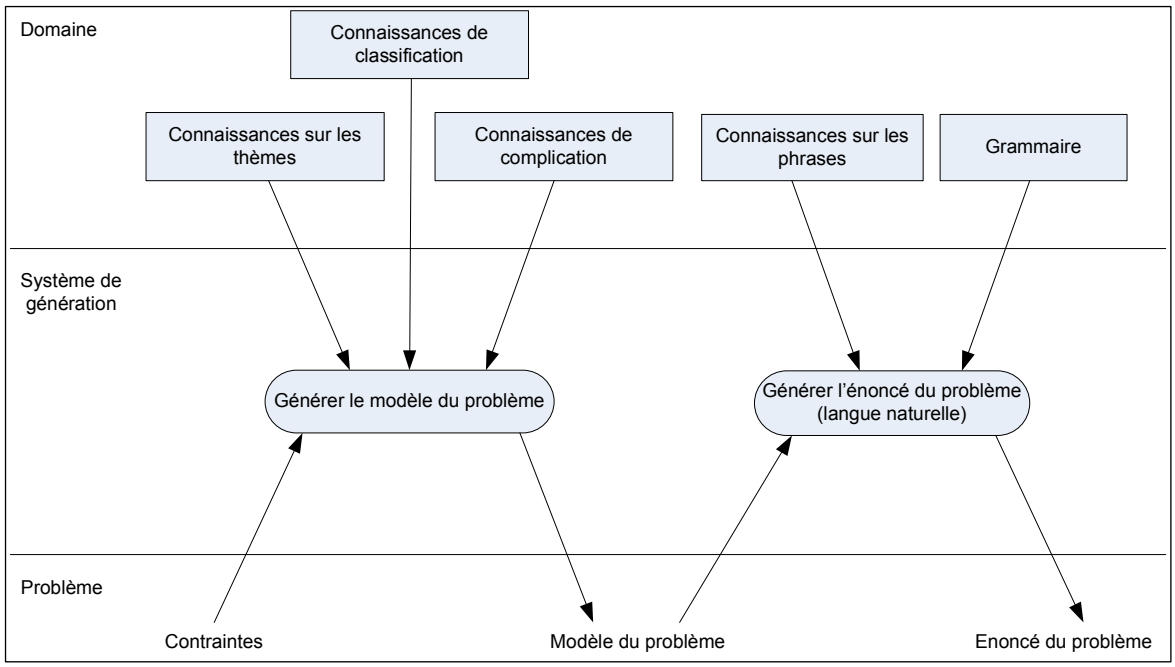


Figure 3 : Architecture sur trois niveaux du générateur

Nous venons de le voir, le noyau de notre système de génération de problèmes est indépendant du domaine. Ce noyau est conçu et implémenté par le concepteur du système.

Pour l'utiliser dans un domaine particulier, un expert du domaine doit fournir les connaissances de classification, les connaissances de complication, les connaissances sur les thèmes, les connaissances de phrases et la grammaire qui seront utilisées par le noyau. Les connaissances sur les thèmes sont actuellement fournies par l'expert, elles seront ensuite créées directement par l'enseignant lui-même, via l'activité de gestion des traits de surface qui est incluse dans notre outil d'assistance à l'enseignant.

L'utilisateur peut ensuite soumettre à notre générateur les contraintes sur le problème à résoudre.

Précisons que seul le générateur est indépendant du domaine. L'interface de ce système, présentée dans le chapitre 3 de ce rapport dans le cadre des problèmes additifs, est quant à elle dépendante du domaine, et devra donc être redéfinie pour chaque domaine auquel le générateur sera appliqué.

Chapitre 5

Bilan

5.1 Résultats

Le premier résultat significatif de notre travail est relatif à l'analyse des besoins des enseignants pour la conception d'un environnement les assistant dans l'utilisation de l'EIAH AMBRE-add destiné aux apprenants. Notre collaboration avec des enseignants dans ce travail nous permet aujourd'hui de mieux connaître leurs besoins et leurs attentes, et ainsi de savoir comment il est possible d'adapter le logiciel apprenant au contexte d'apprentissage des enseignants. Nous proposons donc pour cela un ensemble d'outils : le paramétrage de l'environnement, la création de séquences d'apprentissage, la génération de problèmes, la création de nouveaux thèmes d'exercices et la distribution du travail aux élèves.

Le deuxième résultat de notre travail concerne l'activité de génération de problèmes. Nous avons conçu et implémenté un système qui crée des problèmes (énoncé en langue naturelle + modèle du problème utile au résolveur) qui satisfont les contraintes définies à l'interface par l'utilisateur, qui sont linguistiquement corrects, et conformes aux attentes des enseignants. Nous avons en effet présenté les résultats obtenus aux enseignants participant au projet, qui nous ont affirmé que les problèmes générés correspondaient à leurs attentes, à leurs méthodes et habitudes de travail. La notion de complication implémentée dans ce générateur semble être l'élément qui les a le plus satisfait et intéressé, car elle permet d'adapter vraiment les exercices au niveau des élèves, à leurs facilités ou difficultés. Elle permet également de diversifier largement les énoncés créés. Notre générateur permet de créer des énoncés difficiles tel que « *Julie a six billes. Damien est dans le parc avec Julie. Il a trente-six tulipes. Il a dix-sept fleurs de moins que Julie. Combien de fleurs Julie a-t-elle ?* », comme des énoncés beaucoup plus simples du style « *Damien a 36 fleurs. Julie en a 17 de plus que Damien. Combien Julie a-t-elle de fleurs ?* ». Il permet de plus de créer aussi bien des problèmes très contraints que des problèmes « libres », c'est-à-dire dont peu de contraintes ont été définies par l'enseignant. Définir peu (voire aucune) contraintes permet d'augmenter rapidement la base de problèmes du système, avec des exercices variés (car le système choisit aléatoirement les éléments non spécifiés).

D'autre part, ce système est générique, donc indépendant du domaine pour lequel il est utilisé. Il a été implémenté dans le cadre des problèmes additifs, mais il peut être réutilisé dans tout autre domaine d'application de AMBRE. Il faut pour cela qu'un expert fournisse au système l'ensemble des connaissances du domaine que nous avons présentées dans le chapitre 4.

Les limites de notre travail sont l'interface et l'évaluation de l'environnement. L'interface est en effet encore actuellement un peu compliquée à comprendre et à utiliser car elle n'est pas suffisamment intuitive. Cela est notamment dû à la richesse des contraintes de notre système, qui, si elle permet d'adapter au mieux les problèmes aux apprenants, et par la même aux besoins des enseignants, rend l'interface plus complexe et donc plus difficile à utiliser sans l'aide d'un expert. Celle-ci devra donc être retravaillée afin d'être encore améliorée. Il est en effet indispensable que l'interface de notre environnement de génération de problèmes additifs soit la plus intuitive, la plus claire et la plus compréhensible possible, afin que les utilisateurs puissent se l'approprier facilement et rapidement, et donc ne la rejettent pas. Nous intégrerons pour cela un module d'aide dans l'environnement. Cette interface est présentée en annexe de ce rapport.

Le second point qui modère nos résultats est le manque d'évaluation de notre logiciel : évaluation de la génération comme de l'interface. Nous l'avons dit, l'environnement créé répond a priori aux attentes des enseignants. Mais pour l'évaluer concrètement, il est indispensable de mener une expérimentation réelle avec des enseignants, évaluation que nous n'avons pas encore pu effectuer. Une telle expérimentation est en effet le seul moyen de connaître réellement le résultat de notre travail, en terme de génération surtout. Elle devra donc être menée prochainement, une fois que le générateur sera totalement terminé, notamment lorsque l'interface aura été quelque peu modifiée.

5.2 Perspectives

La première perspective est, nous venons de le souligner, l'évaluation de l'environnement créé : évaluation de son utilité (Y a-t-il adéquation entre les fonctions fournies par le système et celles nécessaires à l'utilisateur pour atteindre ses objectifs ?), de son utilisabilité (L'environnement est-il maniable, aisé à prendre en main et à utiliser ?), et de son acceptabilité (L'environnement est-il compatible avec les valeurs, la culture, l'organisation dans lesquelles on veut l'insérer ?), qui sont les trois critères principaux d'évaluation des EIAH [TRICOT et al. 03].

La deuxième perspective dans le cadre de la création de l'outil d'assistance à l'enseignant pour l'utilisation de l'environnement d'apprentissage AMBRE est évidemment l'implémentation des outils, activités, autres que la génération de problèmes, que nous souhaitons mettre en place dans ce logiciel, notamment la création de séquences comportant des exercices de types différents et permettant d'instaurer une progression dans ces séquences, et la création de nouveaux thèmes d'exercices.

Un ensemble de perspectives peut également être dégagé de l'environnement apprenant. Ce sont donc surtout des perspectives du logiciel destiné aux élèves, mais leur mise en place aura bien évidemment des répercussions sur les outils de l'enseignant.

Afin d'adapter le logiciel destiné aux apprenants à des enfants plus âgés, ou même à des adultes (notamment dans le cadre d'un projet d'utilisation de AMBRE pour l'alphabétisation d'adultes), il serait intéressant de pouvoir implémenter la notion de métriques dans les problèmes à résoudre (3 kg de…, 10 cm), et donc dans les problèmes à générer. Ceci n'est toutefois pas faisable directement, et demande une réflexion pour modifier la structure des problèmes actuels, et peut être leur résolution.

Il serait également intéressant de pouvoir intégrer un modèle de l'apprenant (connaissances acquises, difficultés rencontrées, problèmes déjà résolus…) dans l'environnement AMBRE sur lequel le système de génération pourrait se fonder pour créer automatiquement des exercices adaptés aux apprenants. AMBRE deviendrait alors un système encore plus adaptatif. Notre générateur pourrait ainsi créer des exercices automatiquement (pendant l'utilisation de l'environnement apprenant), et semi automatiquement (à partir des contraintes définies par l'enseignant utilisateur).

Enfin, une étude est actuellement menée dans le projet AMBRE-add sur la mise en place d'activités complémentaires pour les élèves autour de cet environnement. On pourrait alors imaginer de permettre à un apprenant de générer lui-même un problème, en tant qu'activité complémentaire. Il faudrait bien sûr dans ce cas réfléchir aux contraintes qui pourraient être accessibles à l'apprenant, et compréhensibles par lui. Il faudra également à terme que le générateur soit capable de générer, en plus des problèmes à résoudre, les activités complémentaires qui seront intégrées à AMBRE.

La perspective la plus intéressante d'un point de vue recherche est la création de nouveaux thèmes d'exercices, activité qui sera implémentée dans l'environnement enseignant. La mise en place de cet outil pose en effet d'intéressants problèmes de représentation des connaissances et d'intelligence artificielle.

Bibliographie

[BEAUREGARD 01] BEAUREGARD S., *Génération de texte dans le cadre d'un système de réponse automatique à des courriels,* Mémoire présenté en vue de l'obtention du grade Maître ès sciences en informatique, Université de Montréal, mars 2001.
http://rali.iro.umontreal.ca/Publications/URL/memoireSB.pdf.

[BRUILLARD 97] BRUILLARD E., *Les machines à enseigner*, Éditions Hermès, Paris, 1997. ISBN 2-86601-610-6.

[BURTON 82] BURTON R.R., *Diagnosing bugs in a simple procedural skill*., Intelligent Tutoring Systems, Academic Press, London, 1982.

[DEUFF 03] DEUFF D., *Structuration et représentation de contenus multimédias pour une application dans le domaine de l'éducation*, Thèse de l'Université de Rennes 1, 2003. http://www.irisa.fr/bibli/publi/theses/2003/deuff/deuff.html

[DUCLOSSON 04] DUCLOSSON N., *Le projet AMBRE*, Séminaire e-Praxis, INRP, 21 avril 2004. www.inrp.fr/rencontres/seminaires/2004/praxis/ambre.pdf

[GIROIRE 89] GIROIRE H., *Un système à base de connaissances pour la génération d'exercices dans des domaines liés au monde réel*, Thèse de doctorat de l'Université Paris 6, 1989.

[GUIN-DUCLOSSON 99] GUIN-DUCLOSSON N., *SYRCLAD : une architecture de résolveurs de problèmes permettant d'expliciter des connaissances de classification, reformulation et résolution.*, Revue d'Intelligence Artificielle, 13-2, p. 67-94, Paris : Hermès, 1999.

[HENRY 04] HENRY N., *Mémoire sur les Systèmes Auteurs*, mémoire de cours d'option de DEA, février 2004.

[JEAN-DAUBIAS 04] JEAN-DAUBIAS S., *De l'intégration de chercheurs, d'experts, d'enseignants et d'apprenants à la conception d'EIAH*, TICE'2004.

[KOLODNER 93] KOLODNER J., *Case Based Reasoning*, Morgan Kaufmann Publishers, San Mateo, 1993.

[LEROUX 02] LEROUX P., *Machines partenaires des apprenants et des enseignants – Étude dans la cadre d'environnements supports de projets pédagogiques*, Habilitation à Diriger des Recherches en Informatique de l'Université du Maine, Le Mans, 2002. http://www.inrp.fr/atief/hdr.htm

[LESTA et al. 04] LESTA L., YACEF K., *An Intelligent Teaching Assistant System for Logic*, ITS'2004.

[MITROVIC et al. 96] MITROVIC A., STOINEMOV L., DJORDJEVIC-KAJAN S., *INSTRUCT : Modelling Students by asking questions*, User Modelling And User-Adapted Interaction, vol.6, Iss.4, p. 273-301.

[MURRAY 99] MURRAY T., *Authoring ITS : An analysis of the State of the Art*, International Journal of Artificial Intelligence in Education, vol.10, p. 98-129, 1999.

[NOGRY et al. 02] NOGRY S., JEAN-DAUBIAS S.,GUIN-DUCLOSSON N., *La psychologie cognitive au service de la conception de l'environnement d'apprentissage AMBRE*, TICE'2002, INSA de Lyon, 13-15 novembre 2002.

[OULD AHMED LIMAM et al. 02], OULD AHMED LIMAM M., GAIO M., *Description textuelle de schémas géographiques. Génération de texte guidée par Focus.*, Actes du colloque CIDE'5, CFD 2002, Hamamet, Tunisie, 20-23 octobre 2002.
http://infodoc.unicaen.fr/publications/moal.2002.cide.pdf.

[PECEGO 98] PECEGO G., *SYGEP, un Système de Génération d'Énoncés de Problèmes dans des domaines variés*, Thèse de l'Université Pierre et Marie Curie, Paris VI ,12 juin 1998.

[ROGALSKI 94] ROGALSKI M., *Les concepts de l'EIAO sont-ils indépendants du domaine ? L'exemple d'enseignement de méthodes en analyse*. Recherches en Didactiques des Mathématiques, vol.14 n°1.2, p. 43-66, 1994.

[TRICOT et al. 03] TRICOT A., PLÉGAT-SOUTJIS F., CAMPS J.-F., AMIEL A., LUTZ G., MORCILLO A., *Utilité, utilisabilité, acceptabilité : interpréter les relations entre trois dimensions de l'évaluation des EIAH*, EIAH 2003, Strasbourg, 2003. http://archive-edutice.ccsd.cnrs.fr/edutice-00000154

[VAN LABEKE 1999] VAN LABEKE N., *Prise en compte de l'usager enseignant dans la conception des EIAO – Illustration dans Calques 3D*, Thèse de l'Université Henri Poincaré – Nancy 1, 1999. www.psyc.nott.ac.uk/staff/nvl/docs/these_nvl.pdf

[YACEF 02] YACEF K., *Intelligent Teaching Assistant Systems*, International Conference on Computers in Education 2002, New Zealand, 2002.

Logiciels et pages web :

[1] Ruffin de la REBERDIERE
http://www.acguadeloupe.fr/Cati971/Prem_Degre/Bouillante/SEQUENCE.htm#II

[2] GenEval (GENérateur d'Exercices pour auto-EVALuation) est un outil auteur développé à Grenoble, au CAFIM, dans le cadre d'un projet européen de création de documents pédagogiques hypermédias et de développement d'outils de développement de tels documents, le projet ARIADNE.
http://www.ac-grenoble.fr/champo/site/prepa/bcpst/Ariadne/ariadne.htm
Pour plus d'informations sur ARIADNE :
http://www-clips.imag.fr/arcade/projets/ARIADNE/ARIADNE.html.

[3] Creexo est un logiciel gratuit de créations d'exercices dans divers domaines, créé par F.Courtillat. Il est téléchargeable sur http://perso.wanadoo.fr/creexo/ .

Annexe

Présentation de l'interface de génération

Nous présentons l'interface de l'outil de génération de problèmes de notre environnement enseignant, interface créée pour le domaine des problèmes additifs. Nous la détaillons par onglet ci-après, qui, nous l'avons dit, sont au nombre de cinq : Structure, Traits de Surface, Valeurs, Complication et Bilan.

Onglet Structure On définit la classe de l'exercice. L'enseignant peut sélectionner un type d'exercice, puis éventuellement placer l'inconnue sur le schéma correspondant, afin de spécifier encore plus la classe choisie (descendre dans le graphe de classification).

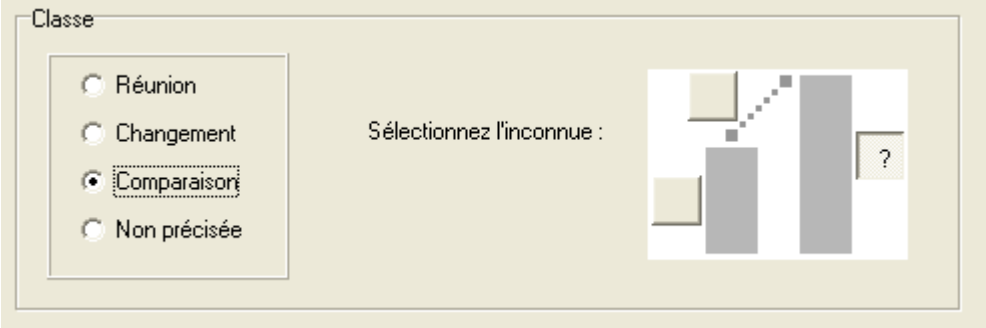


Figure 4 : Interface de l'onglet structure

Onglet Traits de surface Cet onglet permet de définir les traits de surface de nos problèmes : choix d'un thème, d'objets, de personnages.

Dans cet onglet (figure 5), nous avons choisi le thème de la promenade, et l'objet fleur, en précisant que l'on souhaitait utiliser différentes sortes d'objets dans un même énoncé, c'est-à-dire utiliser des catégories d'objets dans le problème. Nous avons également décidé que nos personnages s'appelleraient Damien et Julie. Nous aurions pu sélectionner l'option 'Utiliser une sorte d'objets…' qui signifie que les catégories d'objets peuvent être utilisées, mais sans être mélangées dans un même énoncé. Le système pourra dans ce cas utiliser un sous objet de fleur, le même dans tout l'énoncé.

Dans le choix des personnages, il sera possible d'utiliser le nom de l'élève qui résout l'exercice dans l'énoncé. Ce n'est pas encore implémenté.

D'autre part, il est possible d'affecter directement les objets et/ou personnages choisis aux éléments du schéma choisi auparavant. Ceci revient à les placer dans l'énoncé, ce qui fige bien sûr beaucoup plus l'énoncé produit, et permet donc à l'enseignant de créer exactement l'énoncé souhaité.

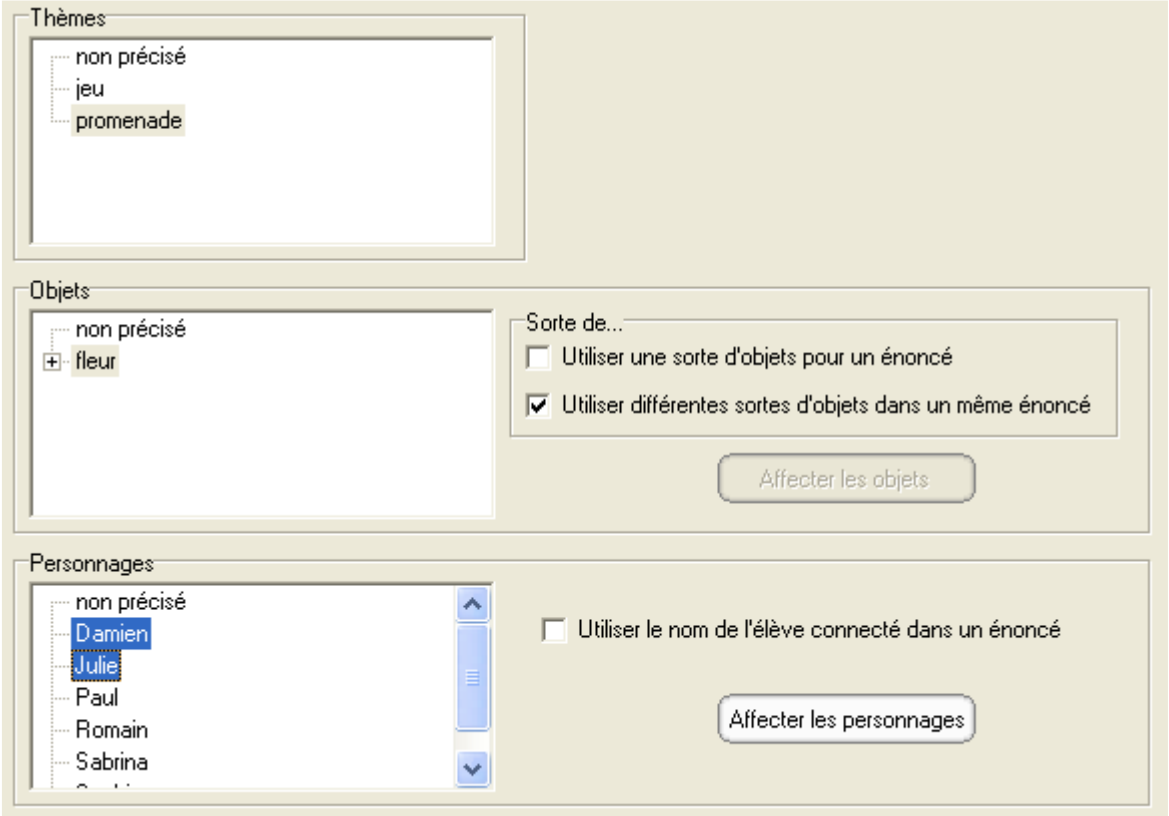


Figure 5 : Interface de l'onglet traits de surface

Onglet Valeurs Nous avons ici la possibilité de choisir entre le mode Intervalle et le mode Valeurs, et d'autoriser ou non l'utilisation de la retenue dans les calculs.

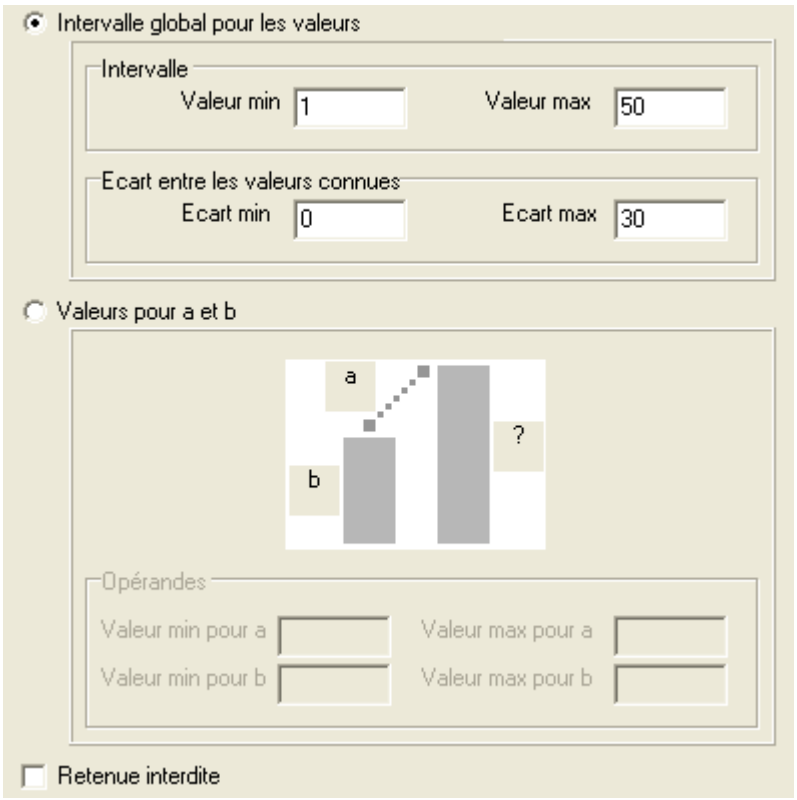


Figure 6 : Interface de l'onglet valeurs

Onglet Complication On précise ici les difficultés à intégrer dans l'énoncé. Dans le cas de la complication de langue, un niveau 1 signifie 'aucune complication'.

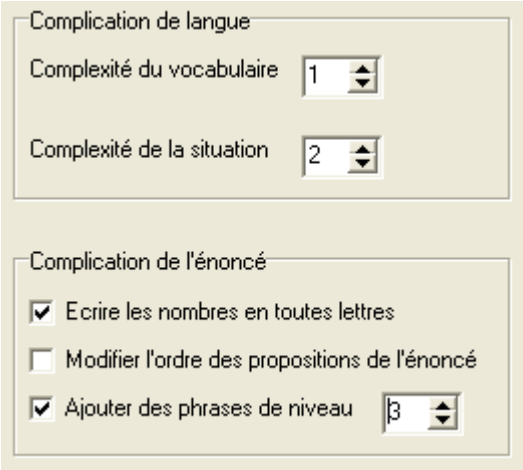


Figure 7 : Interface de l'onglet complication

Onglet Bilan Nous présentons ici l'écran complet de l'interface de notre application. L'onglet Bilan permet de récapituler les choix effectués par l'enseignant, et donc de créer la trame du problème. Cette trame, nous l'avons dit, pourra être enregistrée pour être réutilisée. Nous avons de plus ouvert l'aperçu, qui est accessible dans n'importe quel onglet, et nous avons obtenu l'énoncé suivant : « *Julie a six billes. Damien est dans le parc avec Julie. Il a trente-six tulipes. Il a dix-sept fleurs de moins que Julie. Combien de fleurs Julie a-t-elle ?* ».

Notons qu'il reste deux cases vides dans l'aperçu, sous l'énoncé. Celles-ci devront contenir le schéma et le plan de résolution de l'exercice, une fois que notre application sera complètement reliée au module de résolution.

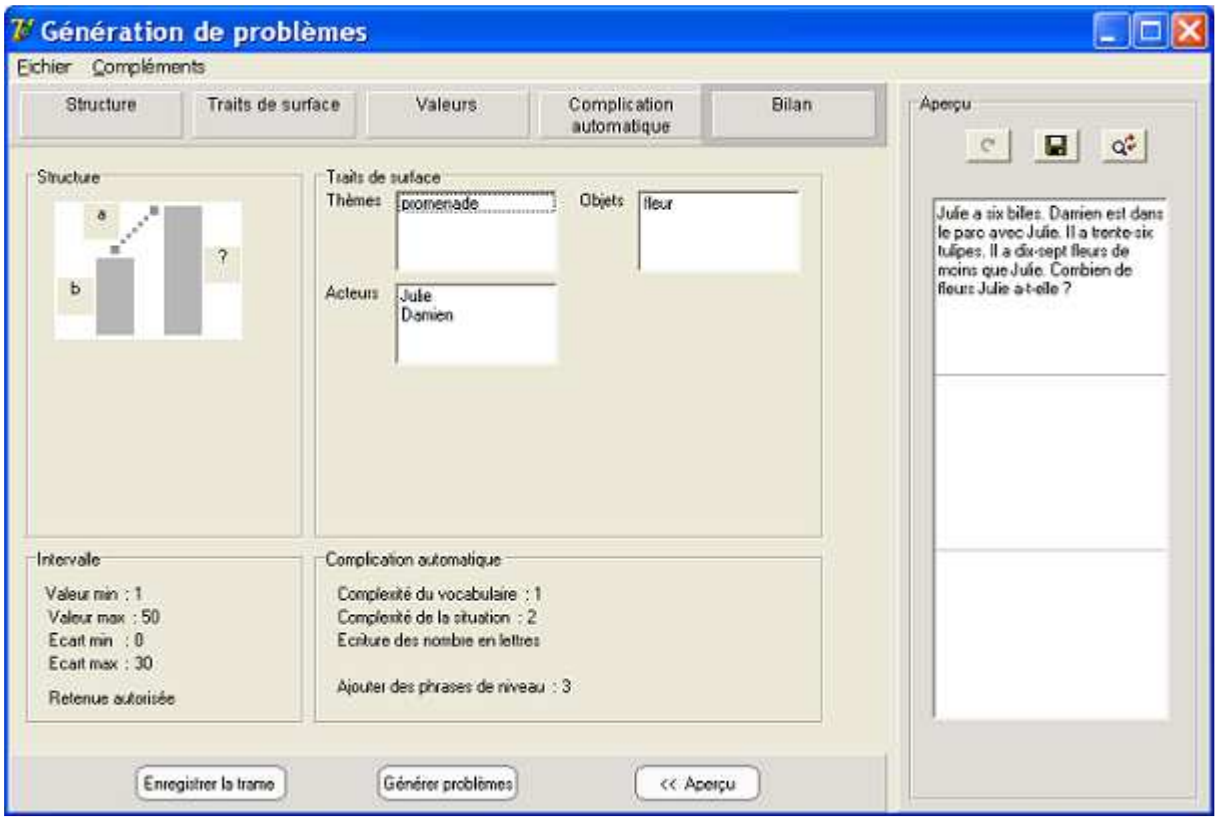


Figure 8 : Interface de l'onglet bilan

Pour finir, nous allons voir la fenêtre d'enregistrement des problèmes, qui est affichée lorsque l'utilisateur appuie sur le bouton 'Générer problèmes' et choisit le nombre de problèmes à créer.

Fenêtre d'enregistrement Dans cet exemple, nous n'avons cette fois-ci précisé aucune contrainte sur les exercices à générer. Et nous avons demandé au système de nous en générer quatre. Rappelons que par défaut, aucune complication n'est ajoutée à l'énoncé. On obtient l'interface présentée en figure 9. L'enseignant peut éventuellement dans cette fenêtre remplacer un exercice qui ne lui conviendrait pas par un autre.

On remarque que les exercices générés sont divers, et utilisent aléatoirement les connaissances de thèmes du domaine (connaissances encore pauvres dans notre prototype). Sur quatre exercices, nous avons obtenu au moins un exercice de chaque classe (changement, comparaison, réunion). Les énoncés sont de plus syntaxiquement corrects, et utilisent des traits de surface différents.

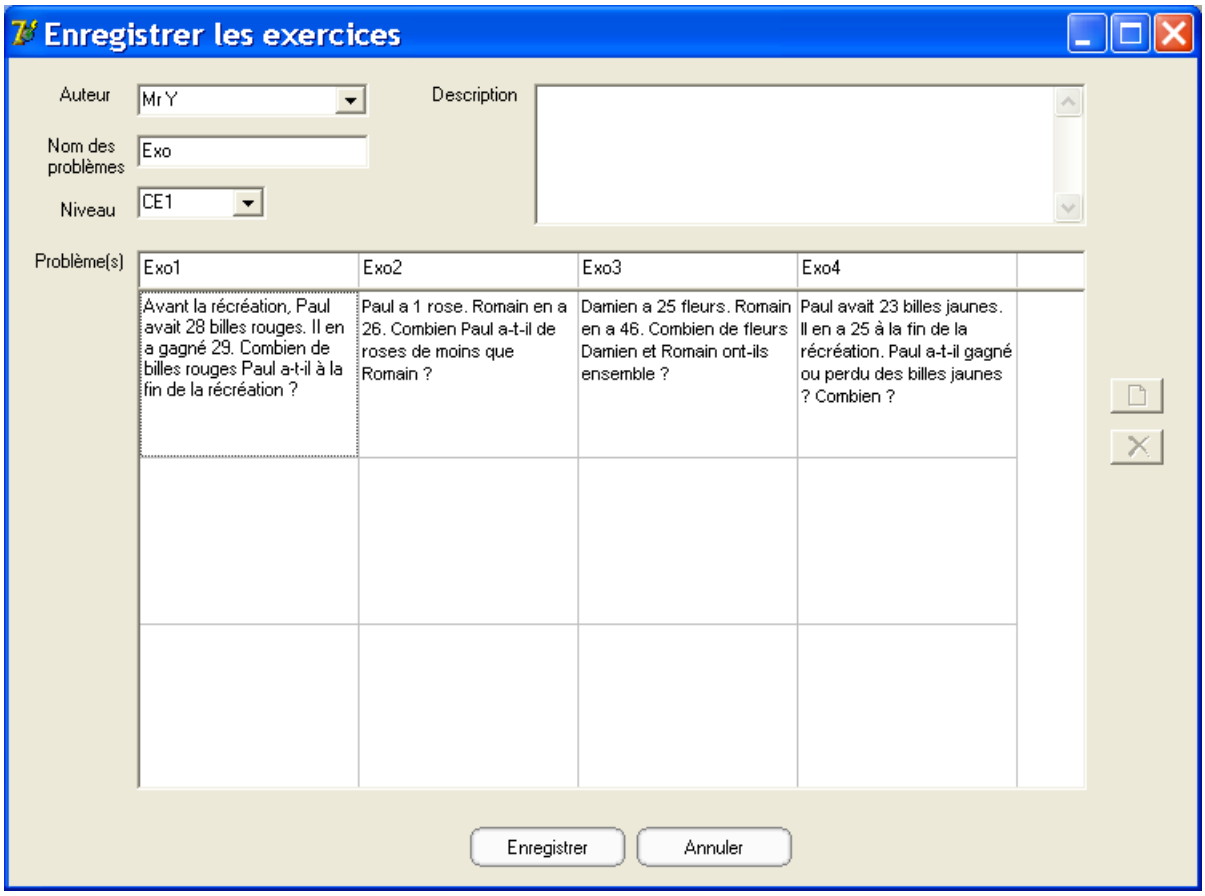


Figure 9 : Interface de la fenêtre d'enregistrement