Problématique de l'acquisition des connaissances dans des environnements informatiques fortement orientés connaissances : vers un outil auteur pour le projet AMBRE

Awa Diattara

▶ **To cite this version:**

Awa Diattara. Problématique de l'acquisition des connaissances dans des environnements informatiques fortement orientés connaissances : vers un outil auteur pour le projet AMBRE. Environnements Informatiques pour l'Apprentissage Humain. Université Grenoble Alpes, 2017. Français. NNT : 2017GREAM055 . tel-01814720

HAL Id: tel-01814720

https://theses.hal.science/tel-01814720

Submitted on 13 Jun 2018

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.

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Awa DIATTARA

Thèse dirigée par **Vanda Luengo**
Codirigée par **Nathalie GUIN**

préparée au sein du Laboratoire d'Informatique de Grenoble dans l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique

Problématique de l'acquisition des connaissances dans des environnements informatiques fortement orientés connaissances : vers un outil auteur pour le projet AMBRE

Thèse soutenue publiquement le **20 Octobre 2017**,
devant le jury composé de :

M. Serge GARLATTI
Professeur à IMT Atlantique, Rapporteur
Mme. Marie-Hélène ABEL
Professeur de l'université de Technologies de Compiègne, Rapporteur
M. Abdelhamid CHAACHOUA
Professeur de l'université Grenoble Alpes, Président
Mme. Vanda LUENGO
Professeur à l'université Pierre et Marie Curie, Directrice de thèse
Mme. Nathalie GUIN
Maître de conférences à l'université Claude Bernard Lyon1, Co-directrice de thèse

Remerciements

Au-delà des remerciements formels, je tiens à exprimer ma profonde gratitude à mes encadrantes : Nathalie Guin pour son suivi constant, ses conseils, sa disponibilité, sa générosité, les conditions de travail ; Vanda Luengo pour ses critiques constructives, sa disponibilité, ses conseils avisés lors des moments clés. Durant ces années de thèse, elles m'ont accompagnée, chouchoutée et soutenue. Je les remercie aussi pour la confiance accordée tout au long de cette thèse et de m'avoir fait découvrir le domaine des EIAH. Ce fut un réel plaisir de travailler avec elles. Et j'espère avoir acquis un peu de leur rigueur scientifique afin de devenir un jour, une enseignante chercheuse à la hauteur de leur mentor.

Je remercie également Amélie CORDIER pour sa disponibilité, pour son approche pédagogique et de m'avoir fait bénéficier de sa grande expérience et de ses conseils.

Mes remerciements aussi à Marie LEFEVRE pour ses remarques pertinentes qui ont permis d'améliorer mon travail.

J'adresse également mes remerciements à la région Auvergne Rhône Alpes qui a financé cette thèse.

Mes remerciements vont également à l'endroit de Madame Marie-Hélène ABEL et Monsieur Serge GARLATTI pour avoir rapporté sur cette thèse. Les jugements et remarques qu'ils ont fournis à l'égard de mon travail ont été pertinents et très enrichissants. J'adresse également mes remerciements à Monsieur Hamid CHAACHOUA pour m'avoir fait l'honneur d'examiner ce travail de recherche.

Je remercie l'ensemble des membres du LIRIS et du LIG, en particulier, les collègues des équipes TWEAK et MeTAH pour leur accueil chaleureux, leur sympathie, les échanges et conseils lors des réunions de présentation de travail, les moments agréables

passés pendant ces années de thèse.

Merci enfin à Isabelle, Brigitte et Catherine d'avoir rendu si facile le côté administratif de la recherche.

Ensuite, vient le tour de mes amis sans qui cette thèse aurait été interminable : Arame, Thierno, Lycka, Laure Esnault, Pascaline, Adamaz, Oumou salam, Nabou, Mamy, Adja, Assitan, Lucie, les amies de Mariama Bâ, les amis de MASS, les amis de Sanar, les amis de Microsoft, les amis de Montpellier, les amis de Lyon... et tous les autres que j'aurais oubliés de citer.

Mention spéciale à ma famille qui m'a toujours soutenue. Merci à mes parents qui m'ont permis d'arriver jusque-là. Mention spéciale à ma maman, qui a toujours cru en moi. C'est grâce à elle que j'ai pu tenir pendant toutes ces années d'étude. Mes remerciements également à ma belle-famille pour leur soutien.

Et pour finir, je remercie mon merveilleux mari, pour son amour inconditionnel, son soutien et ses encouragements. Mamour, merci de m'avoir suivi à Lyon, de m'avoir supporté et de me supporter encore, merci pour ta compréhension et ta patience.

Un grand merci à tous ceux qui de près ou de loin ont participé à la réussite de ce travail.

Résumé

Cette thèse aborde la problématique de l'acquisition des connaissances dans le cadre de la conception des Environnements Informatiques pour l'Apprentissage Humain (EIAH). Dans le contexte spécifique de ce travail, nous nous intéressons à des EIAH destinés à enseigner des méthodes de résolution de problèmes. De telles méthodes permettent, dans un domaine donné, de reconnaître la classe d'un problème et d'être capable de savoir quelle technique de résolution appliquer pour le résoudre. Le coût de conception de ces EIAH est cependant très élevé, en particulier du fait de l'élicitation des connaissances qui nécessite non seulement une expertise dans le domaine concerné mais également en programmation.

Afin de réduire le coût de conception de ces EIAH et permettre à des auteurs (enseignants, professionnels ou pédagogues plus experts) de pouvoir éliciter sans programmer les connaissances nécessaires à l'EIAH pour résoudre des problèmes, nous proposons un processus d'acquisition interactive de ces connaissances. Ce processus est mis en œuvre à travers la conception d'un outil auteur : AMBRE-KB. Pour ce faire, nous avons d'abord proposé des méta-modèles qui permettent de décrire la forme des connaissances à acquérir. Ces connaissances ne sont pas celles d'un expert dans un domaine, mais les connaissances telles qu'on voudrait qu'elles fonctionnent chez l'apprenant à l'issue de l'apprentissage. En s'appuyant sur ces méta-modèles, nous avons ensuite proposé un processus d'acquisition de ces connaissances qui assiste l'auteur lors de l'élicitation des connaissances nécessaires, en lui permettant de construire un modèle de connaissances spécifique à un domaine.

Deux expérimentations ont été menées afin de mettre à l'essai AMBRE-KB. Les résultats de la première expérimentation - réalisée sur trois domaines - ont conduit à faire évoluer les fonctionnalités et les interfaces de AMBRE-KB afin d'améliorer l'interactivité

du système, mais également permettre à l'auteur d'avoir un feedback sur le processus d'élicitation. Cette nouvelle version de AMBRE-KB a permis de mener une seconde expérimentation réalisée sur quatre domaines. Celle-ci a permis de mettre en évidence les points forts et les points faibles de AMBRE-KB :

— Pour un domaine donné, AMBRE-KB permet de générer les modèles de connaissances permettant au résolveur de résoudre les problèmes relevant de la méthode à enseigner

— Les interfaces proposées sont faciles à utiliser dans l'ensemble

— Les auteurs ont apprécié l'assistance fournie, en particulier le feedback

— Une amélioration de certaines interfaces et un feedback plus poussé faciliteraient davantage le processus d'élicitation.

Mots clés : EIAH – Outil auteur – Acquisition de connaissances – Enseignement de méthodes – Modèles de connaissances.

Abstract

The general issue we addressed in this thesis is the challenge of knowledge elicitation in Intelligent Tutoring Systems (ITS). In the context of this work, we are interested in ITS teaching problem solving methods. Teaching methods for solving problems consists in teaching students how to think about the problem before starting its resolution. In a given field, such a method is based on a categorization of problems. Knowing to recognize the class of a problem enables students to choose the resolution technique associated with this class. However, designing such ITS is tedious and costly, and specially requires expertise in the application domain and in programming.

In order to reduce the design cost of these ITS and to enable an author (for example a teacher) to be able to elicit knowledge needed without programming, we propose an interactive knowledge elicitation process. This process is implemented through the design of an authoring tool : AMBRE-KB. For that, we first propose meta-models for the knowledge to be acquired. This knowledge is not an expert knowledge, but knowledge such as we would want that it works at the end of the learning. Next, we propose a knowledge acquisition process based on these meta-models, which enable the author to be assisted in the elicitation process enabling him/her to build specific knowledge models for a given domain.

Two experiments were conducted to evaluate the knowledge acquisition process and its implementation in the AMBRE-KB tool. The results of the first experiment, conducted in three domains, led to modify the features and the interfaces of AMBRE-KB, so that to improve the interactivity of the tool, and to enable authors to have feedback about the knowledge elicitation. The second experiment based on the new version of AMBRE-KB, was conducted in four domains. The results of this experiment enable to highlight the strengths and the limits of AMBRE-KB :

— For a given domain, AMBRE-KB can produce the knowledge models needed by the solver to solve problems about the method to be taught

— On the whole, interfaces are easy to use

— The assistance provided is useful and particularly the feedback

— An improvement of the interfaces and of the interactivity of the system would facilitate even more the elicitation process.

Keywords :

ITS – Authoring tool – Knowledge acquisition – Teaching methods – Knowledge models

Table des matières

Introduction générale **21**

0.1 Introduction 22

0.2 Problématique et objectifs de la thèse 25

0.3 Contributions 27

0.4 Structure du document 27

I CONTEXTE ET ÉTAT DE L'ART **31**

1 Cadre des recherches : le projet AMBRE **32**

1.1 Principe des EIAH AMBRE 34

1.1.1 Des EIAH pour enseigner des méthodes 34

1.1.2 Un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC) 35

1.2 Comment appliquer une méthode de résolution de problèmes ? 37

1.2.1 Connaissances pour appliquer une méthode de résolution de problèmes 37

1.2.2 Processus de résolution de problèmes 41

1.3 Comment guider l'apprenant dans sa résolution de problèmes ? 42

1.3.1 Connaissances pour guider l'apprenant durant son apprentissage . 42

1.3.2 Processus de raisonnement 44

1.4 Conclusion 45

2 État de l'art **47**

2.1 De l'acquisition des connaissances à l'Ingénierie des Connaissances 49

2.2 Méthodes et techniques d'acquisition de connaissances en Ingénierie des Connaissances 50

2.2.1 Méthodes empiriques 50

2.2.2 Méthodes analytiques 52

2.3 IC et EIAH 58

2.4 Outils auteurs 59

2.4.1 Outils auteurs pour la création de parcours pédagogiques 59

2.4.2 Outils auteurs pour la création de tuteurs intelligents 60

2.5 Synthèse des principaux verrous 62

II CONTRIBUTIONS THÉORIQUES **66**

3 Des méta-modèles pour guider l'élicitation des connaissances **67**

3.1 Comment décrire les problèmes d'un domaine ? 69

3.1.1 Méta-modèle du vocabulaire permettant de décrire des problèmes 70

3.1.2 Méta-modèle permettant de décrire des problèmes 71

3.2 Comment définir les connaissances sur la méthode à enseigner ? 73

3.2.1 Méta-modèle des connaissances de classification 74

3.2.2 Méta-modèle des connaissances de reformulation 77

3.2.3 Méta-modèles des connaissances de résolution 82

3.3 Comment définir les connaissances permettant d'accompagner l'apprenant ? 86

3.3.1 Méta-modèle des connaissances d'aide 87

3.3.2 Méta-modèle des connaissances de diagnostic 89

3.3.3 Connaissances d'explications sur les erreurs 91

3.4 Conclusion 93

4 Un processus interactif d'acquisition des connaissances pour des EIAH destinés à enseigner des méthodes 95

4.1 Étapes du processus d'acquisition des connaissances 97

4.1.1 Définition du vocabulaire pour le domaine 98

4.1.2 Description des problèmes à résoudre 99

4.1.3 Élicitation des connaissances sur la méthode 100

4.1.4 Conception de l'interface de l'EIAH 104

4.1.5 Élicitation des connaissances destinées à accompagner l'apprenant 104

4.2 Feedback après l'élicitation des connaissances 104

4.3 Synthèse 106

III MISE EN ŒUVRE ET ÉVALUATIONS 109

5 AMBRE-KB : un outil auteur pour des EIAH destinés à enseigner des méthodes 110

5.1 Élicitation des connaissances avec AMBRE-KB 113

5.1.1 Définition du vocabulaire 115

5.1.2 Description des problèmes à résoudre 116

5.1.3 Élicitation des connaissances sur la méthode 118

5.2 Génération des modèles de connaissances 126

5.3 Tests des modèles de connaissances 130

5.4 Conclusion 135

6 Évaluations 137

6.1 Mise à l'essai de AMBRE-KB 139

6.1.1 Première expérimentation 139

6.1.2 Seconde expérimentation 143

6.2 Bilan des expérimentations 154

Introduction générale 156

6.3 Synthèse de nos travaux 157

6.4 Discussion 159

6.5 Perspectives 161

Annexes **163**

Bibliographie **206**

Table des figures

1 Exemple de résolution du problème P1 en utilisant une technique de résolution 24

1.1 Fonctionnement du résolveur de problèmes dans AMBRE [40] 38

1.2 Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif 39

2.1 KADS : cycle de vie d'un système expert [31] 53

3.1 Méta-modèle permettant de décrire le vocabulaire d'un domaine 71

3.2 Méta-modèle permettant de décrire un problème en s'appuyant sur le vocabulaire 72

3.3 Lien entre les connaissances de classification, de reformulation et de résolution 74

3.4 méta-modèle permettant de décrire une classe de problèmes 75

3.5 Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif 76

3.6 Lien entre connaissances de classification et connaissances de reformulation 78

3.7 Méta-modèle permettant de définir une fonction de calcul. 79

3.8 Méta-modèle permettant de décrire une règle 80

3.9 Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif, avec les règles de reformulation 81

3.10 Lien entre connaissances de reformulation et connaissances de résolution 83

3.11 Méta-modèle permettant de décrire une technique de résolution 84

3.12 Conjugaison du verbe Parler au présent de l'indicatif 85

3.13 Lien entre connaissances sur la méthode et celle destinées à accompagner l'apprenant 87

3.14 Exemple de fenêtre d'aide à deux niveaux dans l'étape d'adaptation dans l'EIAH AMBRE-add. le deuxième niveau active le coloriage facilitant la mise en correspondance du problème-type et du problème à résoudre . . 88

3.15 Aide à l'analogie à l'étape d'adaptation dnas AMBRE-add 89

3.16 Méta-modèle des connaissances d'aide 89

3.17 Aide à l'analogie à l'étape d'adaptation dans AMBRE-add 90

3.18 Exemple de diagnostic réalisé par le système 91

3.19 Méta-modèle permettant de décrire un message d'explications 92

3.20 Exemple de message d'explications généré à partir de la réponse de l'apprenant 93

4.1 Approche générale du processus d'élicitation 98

5.1 Approche générale de AMBRE-KB 112

5.2 Vue d'ensemble de AMBRE-KB 113

5.3 Une partie du vocabulaire pour le domaine des problèmes additifs 115

5.4 Description des objets du problème réunion_1 117

5.5 Définition d'une classe de problèmes en utilisant AMBRE-KB 119

5.6 Une partie de l'arbre de classification pour les problèmes additifs 120

5.7 Une partie de l'arbre de classification pour les problèmes additifs 122

5.8 représentation d'une partie des prémisses de la regle_reunion_operande dans AMBRE-KB 124

5.9 représentation de la partie conclusion de la regle_reunion_operande dans AMBRE-KB 125

5.10 représentation de la technique tech_réunion_opérande dans ambre-kb . 126

5.11 Exemple de feedback lorsque la résolution réussit. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème 130

5.12 Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a réussi 131

5.13 Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème 132

5.14 Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée parce que le système n'a pas trouvé de règle permettant de conclure sur la valeur d'un attribut 133

5.15 Exemple de feedback lorsque la résolution a échoué lors de l'application d'un technique de résolution. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème 134

5.16 Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a échoué lors de l'application d'une technique de résolution . 135

6.1 Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie 151

6.2 Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie 152

3 Définiiton du vocabulaire pour le pluriel des noms avec AMBRE-KB . . . 166

4 Définiton du problème pluriel_detail avec AMBRE-KB 167

5 Une partie de l'arbre de classification pour le pluriel des noms en "al", "ail", "ou", "eu, "au", "eau" 168

6 une aprtied e l'arbre de classification 172

7 Définition de la règle *regle_ail_exception* en utilisant AMBRE-KB 173

8 Définition de la technique associée à la classe *c_plu_ail_pas_ex* en utilisant AMBRE-KB 175

9 Arbre de classification pour les équations défini par le sujet 1 180

10 Arbre de classification pour les équations défini par le sujet 2 181

11 Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie 188

12 Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie 189

13 Une partie de l'arbre de classification pour la formation des adverbes en "ment" 192

Liste des tableaux

3.1 Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème 82

3.2 Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème 85

5.1 Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème 123

5.2 Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé 123

5.3 Exemple de technique permettant de résoudre les problèmes de la classe *réunion* 126

5.4 Modèle du problème *reunion_1* en Prolog généré par AMBRE-KB . . . 127

5.5 Une partie du modèle de connaissances représentant les connaissances de classification (cf. Figure 5.6) généré par AMBRE-KB en Prolog 128

5.6 Une partie du modèle de connaissances décrivant les règles généré par AMBRE-KB en Prolog 129

5.7 Une partie du modèle de connaissances décrivant les connaissances de résolution, généré par AMBRE-KB en Prolog 130

6.1 Critère d'évaluation pour la première expérimentation 140

6.2 Critère retenu pour évaluer le vocabulaire d'un domaine. 147

6.3 Critères retenus pour évaluer la flexibilité de AMBRE-KB. 147

6.4 Critères d'évaluation de l'ergonomie de l'interface permettant de définir le vocabulaire 148

6.5 Domaines testés 149

6.6 Résultats des expérimentations pour les équations par les sujets 1 et 2 . . 149

6.7 Vocabulaire défini par le sujet 1. 150

6.8 Vocabulaire défini par le sujet 2. 150

6.9 Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2 150

6.10 Résultats des expérimentations pour le pluriel des noms et la formation des adverbes en "ment" testés par le même sujet 153

6.11 Domaines testés 155

12 vocabulaire proposé pour le pluriel noms terminés par "al", "ail", "ou", "eu, "au", "eau" en français 165

13 Modèle descriptif du problème pluriel_bail généré par AMBRE-KB . . . 168

15 Critères retenus pour évaluer le vocabulaire d'un domaine 174

16 Exemple de modèles de connaissances (décrivant des techniques) pour le pluriel des noms générés par AMBRE-KB 176

17 Comparaison des résultats des expérimentations pour les équations par les sujets 1 et 2 177

18 Vocabulaire défini par le sujet 1. 178

19 Vocabulaire défini par le sujet 2. 179

20 Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé 182

22 Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2 185

23 vocabulaire pour le calcul d'aire proposé par le sujet 1 186

24 vocabulaire pour le calcul d'aire proposé par le sujet 2 187

25 Exemple de règles de reformulation définies pour le calcul d'aire en géométrie 190

26 Exemple de techniques de résolution définies pour le calcul d'aire en géométrie 190

27 vocabulaire pour la formation des adverbes français en "ment" 191

28 Critères retenus pour évaluer le vocabulaire d'un domaine 195

29 Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné 195

30 Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné 195

31 Critères retenus pour évaluer l'élicitation des connaissances de reformulation pour un domaine donné 196

32 Critères retenus pour évaluer l'élicitation des connaissances de résolution pour un domaine donné 196

33 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la définition du vocabulaire pour un domaine donné 196

34 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la Description des problèmes à résoudre pour un domaine donné 197

35 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de classification pour un domaine donné 197

36 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de reformulation pour un domaine donné 197

37 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de résolution pour un domaine donné 198

38 Critères retenus pour évaluer la flexibilité proposée dans AMBRE-KB . . 198

39 Critères retenus pour évaluer les fonctionnalités de guidage proposées dans AMBRE-KB 198

40 Critères retenus pour évaluer les fonctionnalités de contrôle de connaissances proposées dans AMBRE-KB 199

41 Critères retenus pour évaluer les fonctionnalités de feedback proposées dans AMBRE-KB 199

Introduction générale

Ce chapitre donne un aperçu général du contenu de ce manuscrit. Il présente tout d'abord le sujet abordé et dégage la problématique ainsi que les objectifs de cette thèse. Les propositions et les travaux réalisés sont présentés par la suite. Une section décrivant l'organisation du document clôt le chapitre.

Sommaire

0.1 Introduction 22
0.2 Problématique et objectifs de la thèse 25
0.3 Contributions 27
0.4 Structure du document 27

0.1 Introduction

Les travaux présentés dans ce mémoire s'inscrivent dans le cadre des recherches menées sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH). Ce domaine de recherche est dédié à la conception d'environnements informatiques destinés à favoriser l'apprentissage - la construction de connaissances chez un apprenant [10] - dans des contextes variés. Il peut en effet s'agir d'un apprentissage en classe avec un enseignant, ou à distance lors d'une activité en toute autonomie ou en complément d'un enseignement, dans l'optique de s'entraîner, de s'évaluer, de remédier à des difficultés, ou d'approfondir certains sujets.

Le champ de recherche des EIAH est un domaine pluridisciplinaire, à la croisée de l'informatique, de la psychologie cognitive, de la didactique des disciplines et des sciences de l'éducation. Nos recherches relèvent des apports de l'informatique à la question de l'ingénierie des EIAH, champ de recherche qui porte sur les « *questions liées aux concepts, méthodes, théories, techniques et technologies utiles à la conception des EIAH* » [79]. Au sein de ce champ, nous nous intéressons plus particulièrement à la manière de représenter, au niveau de l'environnement informatique, les connaissances qui permettent d'accompagner l'apprenant au cours de son apprentissage. Il s'agit d'une part, de représenter explicitement les connaissances enjeu de l'apprentissage, et d'autre part, de mettre en œuvre des mécanismes permettant au système informatique interagissant avec l'apprenant de s'appuyer sur ces connaissances afin de susciter chez lui un apprentissage.

Les EIAH sont extrêmement variés. Dans le cadre de cette thèse, nous nous intéressons à des situations d'apprentissage individuel, dans des domaines où il est possible d'établir une classification des problèmes et des outils de résolution correspondants. Dans ces situations, des apprenants de divers niveaux interagissent avec l'environnement informatique autour d'une tâche de résolution de problèmes, sans collaboration ni interaction entre eux. Dans le contexte spécifique de ce travail, nous nous focalisons sur des EIAH destinés à enseigner des méthodes de résolution de problèmes [65, 67, 71, 72]. De telles méthodes, souvent issues de recherches en didactique des disciplines, permettent - dans un domaine donné - de reconnaître la classe d'un problème et ainsi d'en déduire la technique de résolution appropriée.

Afin d'illustrer nos propos, prenons le cas concret de la conjugaison française, où les problèmes à résoudre consistent à conjuguer à toutes les personnes un verbe V à un temps T donné. L'utilisation d'une méthode pour résoudre ces problèmes va alors consister à déterminer la classe du problème, afin de pouvoir appliquer par la suite la bonne technique de résolution : comment déterminer le radical du verbe ainsi que les terminaisons adéquates pour les différentes personnes au temps demandé.

Exemple de problème :
P1 « *Conjuguer le verbe parler au présent de l'indicatif* ».

Classe du problème à résoudre :
La classification des problèmes permet d'identifier la classe de ce problème : *conjuguer un verbe du premier groupe régulier au présent de l'indicatif.*

Technique de résolution à appliquer :
Pour les problèmes de cette classe, la technique de résolution à appliquer est la suivante :

- Déterminer le radical du verbe : enlever la terminaison *er* de l'infinitif du verbe (parler) pour obtenir son radical (parl) ;
- Déterminer la correspondance entre personnes et terminaisons : au présent de l'indicatif, pour les verbes réguliers du premier groupe, les terminaisons des différentes personnes *je, tu, il/elle, nous, vous, ils/elles* sont respectivement *e, es, e, ons, ez, ent*.
- Construire la forme conjuguée aux différentes personnes en utilisant le radical et les terminaisons identifiés.

La **Figure 1** illustre ainsi la solution au problème *P1* après application de la technique de résolution.

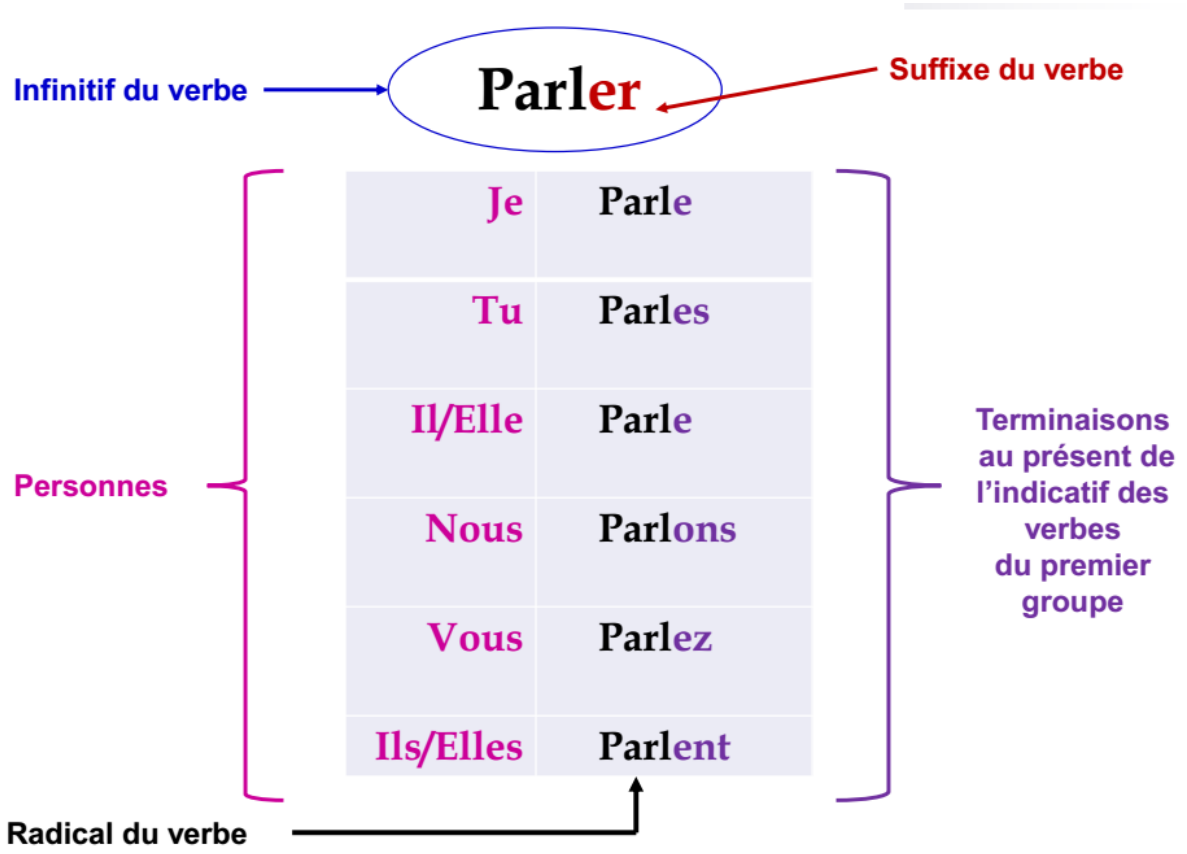

FIGURE 1 – Exemple de résolution du problème P1 en utilisant une technique de résolution

L'objectif des EIAH n'est pas bien sûr de remplacer l'enseignant, mais de lui fournir des outils destinés à accompagner son enseignement. Il peut s'agir d'environnements qu'il fait utiliser à ses apprenants soit en classe en leur compagnie, soit en totale autonomie chez eux, afin d'approfondir certains sujets, de s'entraîner, de s'évaluer, etc. Cependant, pour que ces environnements soient utiles aux enseignants, il faut qu'ils soient adaptés aux besoins et pratiques de chacun d'entre eux. Pour ce faire, il faut donc donner à l'enseignant les moyens de s'approprier l'environnement, soit en le configurant, soit en lui permettant de le concevoir lui-même.

Pour répondre à cette problématique, certains chercheurs comme Leroux [50] et Bruillard [16] ont proposé d'associer les enseignants à la conception des EIAH, afin de favoriser leur acceptabilité par leurs collègues. Toutefois, cela ne garantit pas pour autant de répondre au besoin individuel de chaque enseignant. D'autres chercheurs ont quant à eux proposé de donner la possibilité à chaque enseignant d'achever lui-même la

conception de son EIAH, grâce un module dédié et via lequel il peut paramétrer l'environnement destiné à ses apprenants [38]. Ce module lui permet ainsi de choisir ou de créer les activités que ses apprenants auront à effectuer : exemples AMBRE-enseignant [32] ou encore aplusix [15]. Cependant, comme le souligne [20] pour favoriser la réingénierie de tels outils, il faut également fournir à l'enseignant des outils lui permettant d'observer et d'analyser l'activité de ses élèves. D'autres encore se sont penchés sur les outils auteurs, afin de permettre à l'enseignant de concevoir lui-même les EIAH correspondant à ses besoins. Un outil auteur est un environnement informatique permettant de construire un EIAH, en partie ou dans sa globalité, avec un coût réduit voire nul en programmation

Dans le cadre spécifique de cette thèse, nous nous intéressons à des outils auteurs destinés à concevoir des EIAH dits centrés connaissances, c'est-à-dire des environnements informatiques dans lesquels l'ensemble des interactions entre l'apprenant et le système sont fondés sur une représentation explicite des connaissances [89]. Ces connaissances ne sont pas celles d'un expert dans un domaine, mais les connaissances telles qu'on voudrait qu'elles fonctionnent chez l'apprenant à l'issue de l'apprentissage [23]. Nous ciblons particulièrement des utilisateurs experts des connaissances de la méthode à enseigner (enseignant, professionnel, pédagogue plus expert, etc.), familiarisés avec l'outil informatique, avec ou sans compétences en programmation. Ces experts auront pour tâche d'expliciter l'ensemble des connaissances nécessaires à l'EIAH pour accompagner l'apprenant durant son apprentissage. L'enjeu est alors d'assister ces experts dans leur tâche en automatisant une partie du processus de création d'activités pédagogiques. Dans la suite de ce document, ces utilisateurs experts seront dénommés auteurs.

0.2 Problématique et objectifs de la thèse

Notre travail de recherche aborde la question de l'élicitation des connaissances dans le cadre de la conception des EIAH. Notre problématique de recherche concerne plus particulièrement l'élicitation de connaissances à travers la conception et le développement d'un outil auteur. La **formalisation des différents types de connaissances à éliciter** demeure une des premières questions auxquelles nous nous sommes confrontées. En effet, plusieurs types de connaissances interviennent lors de la conception d'un EIAH destiné à enseigner des méthodes. Outre les connaissances permettant la résolution des problèmes, il existe d'autres types de connaissances qui vont permettre de guider l'ap-

prenant durant son apprentissage, en lui fournissant aide, diagnostic de ses réponses et explications sur ses erreurs. Il s'agira donc dans un premier temps de proposer des méta-modèles de connaissances qui permettront, pour un domaine donné, d'exprimer les modèles de connaissances nécessaires à la conception d'un EIAH de ce type.

Une fois les différents méta-modèles de connaissances à acquérir formalisés, il faut ensuite réfléchir sur un **processus d'élicitation de ces connaissances**. Il s'agira alors de permettre à des non-programmeurs, en s'appuyant sur ces méta-modèles, de pouvoir éliciter les connaissances nécessaires à l'EIAH. Dans le cadre spécifique de cette thèse, nous nous focaliserons uniquement sur l'élicitation des connaissances permettant au système de résoudre des problèmes. Néanmoins, puisque les connaissances permettant de guider l'apprenant sont fortement dépendantes des tâches que ce dernier doit accomplir lors de sa résolution de problèmes, et donc de l'interface de l'EIAH, il faut gérer l'articulation entre la conception de l'interface de l'EIAH et l'élicitation de ces connaissances.

Une autre question très importante dans le cadre de ce travail est celle de l'**assistance** à fournir à l'auteur lors de l'élicitation des connaissances. En effet, pour permettre une acquisition aisée des connaissances, il est important d'assister l'auteur durant tout le processus d'élicitation en lui proposant des fonctionnalités d'aide, de guidage et de contrôle de ces connaissances. Enfin, une fois le processus d'élicitation terminé, une dernière question à laquelle il faut répondre est : **comment permettre à l'auteur d'avoir un feedback sur les connaissances élicitées ?** En effet, il est très important de donner à l'auteur les moyens de contrôler les connaissances élicitées, de savoir s'il a élicité toutes les connaissances nécessaires, et de pouvoir corriger et/ou compléter les connaissances manquantes le cas échéant.

Ces questions doivent être traitées dans l'objectif de proposer un processus d'acquisition des connaissances qui permettra à un auteur, informaticien ou non, de pouvoir expliciter les connaissances nécessaires à l'EIAH sans aucune programmation. Nous nous posons donc comme contrainte de proposer un processus interactif d'acquisition des connaissances, indépendant du domaine, qui puisse être implémenté dans un outil auteur, et dans lequel l'auteur est assisté tout du long. L'outil auteur à développer doit permettre à l'auteur d'éliciter les connaissances qui permettront au système de résoudre les problèmes pour un domaine donné. Il doit également permettre à l'auteur d'avoir un feedback sur le processus, de sorte à pouvoir corriger/compléter les connaissances au besoin.

0.3 Contributions

Afin de répondre au premier point de notre problématique, nous proposons des méta-modèles pour formaliser les différents types de connaissances à acquérir. Ces méta-modèles permettent, pour un domaine donné, de définir la forme des connaissances à acquérir pour la conception d'un EIAH destiné à enseigner une méthode de résolution de problèmes. La réponse au second point est un processus interactif d'acquisition de connaissances fondé sur ces méta-modèles et indépendant du domaine. Il est mis en ?uvre à travers la conception d'un outil auteur.

Pour un domaine donné, cet outil auteur permet d'acquérir les différents types de connaissances nécessaires à la résolution des problèmes, et génère les modèles de connaissances qui constituent les bases de connaissances de l'EIAH. Pour chaque type de connaissances, l'outil auteur propose également une assistance à l'auteur dans le processus d'élicitation, et lui permet d'obtenir un feedback une fois cette étape effectuée. Deux expérimentations pour évaluer le processus d'acquisition des connaissances et sa mise en ?uvre dans l'outil auteur ont été menées. La première expérimentation a permis de valider que pour un domaine donné, les modèles de connaissances générés par l'outil auteur permettent à l'EIAH de résoudre des problèmes. Les résultats de cette expérimentation ont également conduit à une extension de l'outil auteur afin d'améliorer l'interactivité du système, mais aussi permettre à l'auteur d'avoir un feedback sur les connaissances explicitées. Cette nouvelle version de l'outil auteur a permis de mener une seconde expérimentation qui s'est conclue avec des résultats satisfaisants concernant le processus d'élicitation, les interfaces proposées, ainsi que l'assistance fournie et en particulier le feedback

0.4 Structure du document

Après cette introduction générale, le reste de ce manuscrit s'articule autour de trois grandes parties.

La première partie est consacrée à la description du cadre applicatif de la thèse ainsi qu'à un état de l'art sur le sujet.

Chapitre 1 : ce chapitre présente le contexte applicatif de nos recherches : le projet AMBRE. Une première section décrit le principe des EIAH AMBRE en précisant ce qu'est l'enseignement de méthodes. Une autre section présente le Raisonnement à Partir de Cas (RàPC) avant de montrer comment il est utilisé dans le cadre du projet AMBRE pour faciliter l'apprentissage de méthodes de résolution de problèmes. Ce chapitre se termine par la présentation des différents types de connaissances nécessaires à la conception d'un EIAH AMBRE. Ces connaissances sont, dans un domaine donné, regroupées en deux grandes catégories : celles permettant de résoudre des problèmes, et celles destinées à accompagner l'apprenant en lui fournissant aide, diagnostic de ses réponses et explications sur ses erreurs.

Chapitre 2 : l'objectif de ce chapitre est de présenter un état de l'art sur les fondements théoriques de nos travaux de thèse. Après une présentation de quelques principales techniques d'acquisition de connaissances les plus répandues en Ingénierie des Connaissances (IC), nous montrons comment elles sont utilisées dans le domaine des EIAH par les outils auteurs les plus proches de notre problématique. Ce chapitre s'achève par une synthèse des principaux verrous ou manques actuels par rapport à notre problématique.

La deuxième partie de ce rapport expose les différentes contributions théoriques apportées dans cette thèse.

Chapitre 3 : ce chapitre présente les différents méta-modèles proposés pour guider l'élicitation des connaissances. Ces méta-modèles permettent de définir la forme des connaissances à acquérir et de contraindre le processus d'acquisition de connaissances proposé. Nous présentons dans un premier temps des méta-modèles pour guider l'élicitation des connaissances sur la méthode à enseigner, et dans un deuxième temps des méta-modèles des connaissances destinées à accompagner l'apprenant durant son apprentissage.

Chapitre 4 : ce chapitre décrit le processus d'acquisition des connaissances que nous avons proposé pour la conception d'EIAH destinés à enseigner une méthode de résolution dans un domaine donné. Il présente également les fonctionnalités d'assistance proposées afin d'aider l'auteur dans le processus d'élicitation de connaissances.

La troisième partie présente la mise en œuvre de nos contributions théoriques ainsi que leur évaluation.

Chapitre 5 : ce chapitre explique le fonctionnement général de l'outil auteur que nous proposons : AMBRE-KB. Nous présentons d'abord l'approche générale adoptée pour sa conception. Ensuite, nous illustrons l'utilisation de AMBRE-KB, en montrant comment éliciter, pour un domaine donné, les différentes connaissances nécessaires au système pour résoudre des problèmes.

Chapitre 6 : ce chapitre expose les différentes évaluations conduites pour valider nos contributions. Nous présentons notamment les expérimentations menées pour mettre à l'essai AMBRE-KB. L'objectif de ces expérimentations est de vérifier que les connaissances générées par AMBRE-KB permettent au système de résoudre les problèmes pour un domaine donné, mais aussi d'évaluer l'ergonomie des interfaces, l'assistance fournie avec AMBRE-KB, et le feedback proposé à l'auteur

Enfin, une **conclusion générale** fait le bilan des contributions, présente une discussion des résultats obtenus et propose des perspectives d'enrichissement et de continuité des travaux présentés dans ce manuscrit.

Première partie

CONTEXTE ET ÉTAT DE L'ART

Chapitre 1

Cadre des recherches : le projet AMBRE

Ce chapitre se consacre pour l'essentiel à la description du cadre applicatif de nos recherches : le projet AMBRE (Apprentissage de Méthodes Basé sur le Raisonnement à partir de l'Expérience) [43]. Ce projet de recherche pluridisciplinaire a pour objectif de concevoir des environnements d'apprentissage s'appuyant sur les étapes du cycle du raisonnement à partir de cas (RàPC) pour guider l'acquisition de méthodes de résolution de problèmes.

La première partie de ce chapitre se focalise sur le principe des EIAH AMBRE en précisant tout d'abord ce qu'est l'enseignement de méthodes. Elle se conclut par la présentation du RàPC et de son utilisation dans le cadre du projet AMBRE afin de faciliter l'apprentissage de méthodes de résolution de problèmes.

La seconde partie de ce chapitre est consacrée à la présentation des différents types connaissances nécessaires à la conception d'un EIAH AMBRE. Elle décrit d'abord les connaissances nécessaires pour résoudre des problèmes dans un domaine donné en appliquant une méthode de résolution, puis les connaissances destinées à accompagner l'apprenant durant son apprentissage en lui fournissant aide, diagnostic et explications sur ses erreurs.

Sommaire

1.1 Principe des EIAH AMBRE 34
1.1.1 Des EIAH pour enseigner des méthodes 34
1.1.2 Un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC) 35
1.2 Comment appliquer une méthode de résolution de problèmes ? 37
1.2.1 Connaissances pour appliquer une méthode de résolution de problèmes 37
1.2.2 Processus de résolution de problèmes 41
1.3 Comment guider l'apprenant dans sa résolution de problèmes ? 42
1.3.1 Connaissances pour guider l'apprenant durant son apprentissage 42
1.3.2 Processus de raisonnement 44
1.4 Conclusion 45

1.1 Principe des EIAH AMBRE

1.1.1 Des EIAH pour enseigner des méthodes

L'objectif du projet AMBRE est la conception d'environnements informatiques destinés à enseigner des méthodes pour résoudre des problèmes dans un domaine donné. Il se fonde sur des recherches en didactique des disciplines qui proposent d'enseigner explicitement des méthodes text [65, 71, 73].

Une méthode peut être définie comme « *la description d'un ensemble d'activités du sujet, portant sur l'analyse et le classement des problèmes à résoudre dans un domaine assez précis, l'utilisation des outils et des techniques disponibles, les stratégies et tactiques possibles, la gestion du temps des choix, des stratégies et de leur déroulement, la conscience de ces choix, les moyens de contrôle et de retour en arrière pour procéder à d'autres choix...* » [66]. Toutefois dans le cadre du projet AMBRE, nous restreignons la méthode en considérant qu'elle permet d'organiser une résolution de problèmes en se fondant sur le classement des problèmes et des outils de résolution. Notre objectif demeure que l'apprenant acquière cette méthode. Pour cela, il doit apprendre à construire progressivement des classes de problèmes, à reconnaître à quelle classe appartient un nouveau problème, à associer une technique de résolution à chaque classe de problèmes et à appliquer ces techniques de résolution.

Afin de permettre à un apprenant d'acquérir de telles méthodes, Rogalski [67] et Schöenfeld [70] proposent un enseignement de méthodes basé sur la catégorisation des problèmes et des techniques de résolution à l'aide d'un tuteur intelligent. Rogalski précise que le résolveur d'un tuteur « *donneur de leçons de méthodes* » doit fonctionner selon les méthodes qu'il veut enseigner, et non pas selon des méthodes expertes du domaine concerné. Suivant ces recommandations, les résolveurs utilisés dans le cadre du projet AMBRE fonctionnent selon les méthodes que nous souhaitons enseigner. Toutefois, toujours dans le cadre du projet AMBRE, cette méthode n'est pas présentée explicitement à l'apprenant. Cela s'explique par le fait que dans certains domaines, les termes employés pour définir les classes et les techniques de résolution ne sont pas connus des apprenants puisque, institutionnellement, ils ne sont pas utilisés.

Ainsi, afin d'aider l'apprenant à construire sa propre méthode, les EIAH AMBRE

proposent un processus d'apprentissage en deux étapes :

— Présenter à l'apprenant des problèmes-type résolus ;

— Inciter l'apprenant à mettre en œuvre un raisonnement par analogie pour résoudre de nouveaux problèmes.

En effet, comme le montrent de nombreuses recherches en Psychologie Cognitive [68], la présentation d'exemples et la résolution de problèmes peuvent aider l'apprenant à savoir comment et quand appliquer une technique de résolution. De plus, comme le soulignent d'autres recherches [25,27,36], le raisonnement par analogie permet d'acquérir des connaissances contextuelles et des connaissances abstraites. Par ailleurs, ce type de raisonnement n'est pas toujours spontané et l'acquisition de connaissances lors la résolution de problème par analogie n'est pas toujours automatique [36]. Ainsi, pour encourager l'apprenant à raisonner par analogie, les EIAH AMBRE le guident à travers le cycle du raisonnement à partir de cas.

1.1.2 Un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC)

Pour accompagner l'apprenant dans sa résolution des problèmes, les EIAH AMBRE proposent à l'apprenant un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC) [69]. Cependant, contrairement à d'autres systèmes [2, 37, 45, 55], dans AMBRE, le RàPC n'est pas utilisé par le système, mais proposé à l'apprenant comme une stratégie d'apprentissage. En effet, les EIAH AMBRE proposent d'aider l'apprenant en lui permettant de se référer à un cas proche pour résoudre un nouveau problème [62]. Cependant, contrairement aux autres systèmes, le cas est choisi par l'apprenant lui-même, et non par le système. Pour chaque problème, l'apprenant doit passer par les toutes étapes du RàPC, pour retrouver lui-même le cas le plus proche de son problème et en adapter la solution.

Une séance de travail avec un EIAH AMBRE se déroule en deux phases. La première consiste à la présentation par le système à l'apprenant, de scénarios de résolution de problèmes-types. Après plusieurs séances, l'apprenant aura rencontré un problème-type correspondant à chaque classe de problèmes représentée au niveau de la méthode. La seconde phase est alors consacrée à la résolution par l'apprenant de plusieurs problèmes. L'EIAH guide l'apprenant à travers cinq étapes inspirées du RàPC, le cycle

AMBRE :

- **Lecture de l'énoncé**. L'apprenant commence d'abord par lire l'énoncé afin d'identifier des éléments pertinents qui lui permettront de reformuler le problème.
- **Reformulation du problème**. Cette tâche consiste à identifier les éléments importants de l'énoncé (traits structuraux du problème). L'apprenant est guidé par le système pour construire une nouvelle formulation du problème posé, à partir de l'énoncé du problème. La reformulation obtenue est débarrassée de l'essentiel des traits de surface du problème initial, et devient une référence pour la suite de la résolution.
- **Choix d'un problème-type**. Cette étape consiste pour l'apprenant à comparer le problème à résoudre à ceux déjà rencontrés, en observant les similarités entre la reformulation du problème à résoudre et les reformulations des problèmes déjà rencontrés. L'apprenant doit choisir le problème prototype qui lui semble le plus proche de son problème à résoudre, en se fondant sur les reformulations (étape analogue à la remémoration du RàPC). Il identifie ainsi de manière implicite la classe du problème à résoudre.
- **Adaptation de la solution du problème-type au problème à résoudre**. L'apprenant doit adapter la solution du problème-type au problème à résoudre. L'objectif de cette étape est de l'aider à identifier, par généralisation, une technique de résolution adaptée aux problèmes de cette classe.
- **Classement du problème**. Enfin, cette dernière étape consiste pour l'apprenant à associer le problème qu'il vient de résoudre à un problème-type, de manière à constituer des groupes de problèmes correspondant aux classes, un groupe étant représenté par un problème-type (étape analogue à la mémorisation du RàPC).

Une phase de test et de révision est incluse à chaque étape du cycle AMBRE. En effet, à chaque étape, le système peut évaluer les réponses de l'apprenant et l'aider à les réviser. Afin de pouvoir mettre en œuvre tout cela, le système a besoin d'un certain nombre de connaissances que nous présentons dans la section suivante.

1.2 Comment appliquer une méthode de résolution de problèmes ?

1.2.1 Connaissances pour appliquer une méthode de résolution de problèmes

Au sein du projet AMBRE, enseigner une méthode de résolution consiste à identifier de quelle classe relève le problème afin de pouvoir choisir une technique de résolution adaptée aux problèmes de cette classe. Pour que l'EIAH puisse accompagner l'apprenant dans sa résolution des problèmes, il doit être capable d'appliquer une telle méthode pour résoudre des problèmes du domaine. Pour ce faire les EIAH AMBRE utilisent d'une part, des modèles de connaissances portant sur la méthode à enseigner, et d'autre part un moteur de raisonnement capable d'exploiter ces modèles de connaissances. Nous obtenons ainsi, pour chaque domaine, un résolveur de problèmes qui applique la méthode explicitée par les modèles de connaissances. Ces connaissances sur la méthode à enseigner sont de trois types : des connaissances de classification, des connaissances de reformulation et des connaissances de résolution [42].

Dans AMBRE, les problèmes à résoudre ne sont pas donnés au résolveur en langage naturel, mais en Prolog, sous la forme d'un modèle que nous appelons modèle descriptif [40]. Pour résoudre un problème dans un domaine donné, le système utilise les connaissances de classification et les connaissances de reformulation afin de (i) déterminer de quelle classe relève le problème, et (ii) construire un nouveau modèle du problème appelé modèle opérationnel. La résolution proprement dite consiste alors à appliquer au modèle opérationnel la technique de résolution associée à la classe pour trouver la solution (cf. **Figure 1.1**).

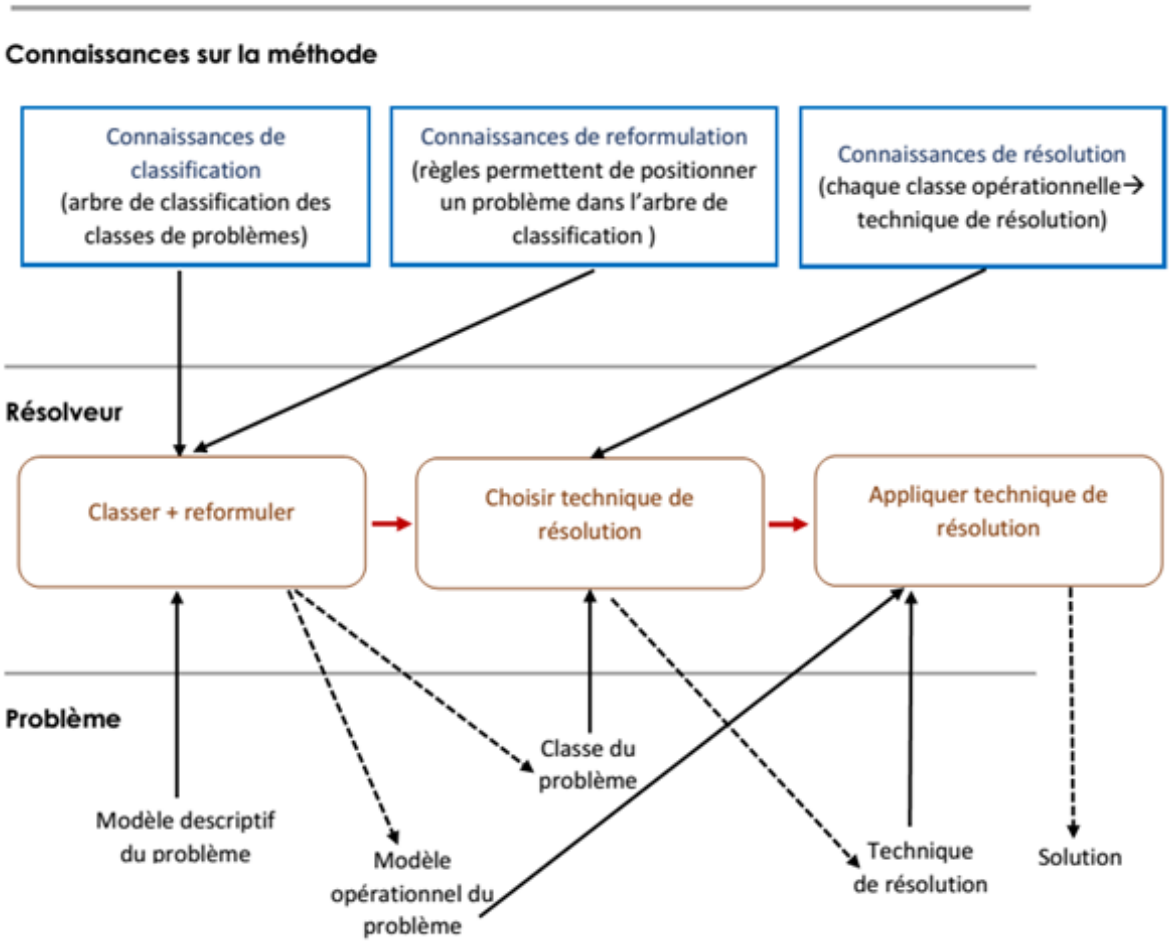

FIGURE 1.1 – Fonctionnement du résolveur de problèmes dans AMBRE [40]

1.2.1.1 Connaissances de classification

Dans chaque domaine d'application, les connaissances de classification sont représentées par une hiérarchie des classes de problèmes à résoudre. La classification représentée par cette hiérarchie est évidemment dépendante du domaine choisi. Toutefois, la structure de représentation ainsi que son utilisation restent les mêmes quel que soit le domaine.

Dans une telle hiérarchie, une classe C2 est sous-classe d'une classe C1 si tout problème de C2 est aussi un problème de C1. Une *classe racine* est définie comme étant la classe la plus générale, alors que les feuilles sont les classes les plus spécifiques.

Chaque classe possède un *attribut discriminant* portant une valeur différente dans chacune de ses sous-classes. D'autres attributs appelés *attributs du problème* peuvent également être définis. Ces attributs ne sont pas discriminants pour la classification, mais demeurent utiles pour la résolution. Les valeurs de ces attributs dépendent ainsi du problème à résoudre. Les classes qui sont suffisamment spécifiques pour qu'on puisse leur associer une technique de résolution sont appelées *classes opérationnelles*. La **Figure 1.2** par exemple représente une partie du graphe de classification pour la conjugaison française. *C_Conjugaison* représente la classe racine et a pour attribut discriminant *temps_simples*. Les valeurs discriminantes de cet attribut sont : présent, et futur. Cette classe possède donc deux sous-classes : une classe non opérationnelle *C_Présent* ainsi qu'une classe opérationnelle *C_Futur*. L'attribut discriminant de la classe C_Présent est *groupe*. Les valeurs discriminantes de cet attributs sont : *1er groupe* et *2ème groupe*. La classe C_Présent possède donc deux sous-classes : la classe *C_Présent_1er_Gr* qui est non opérationnelle et la classe *C_Présent_2ème_Gr* qui est opérationnelle.

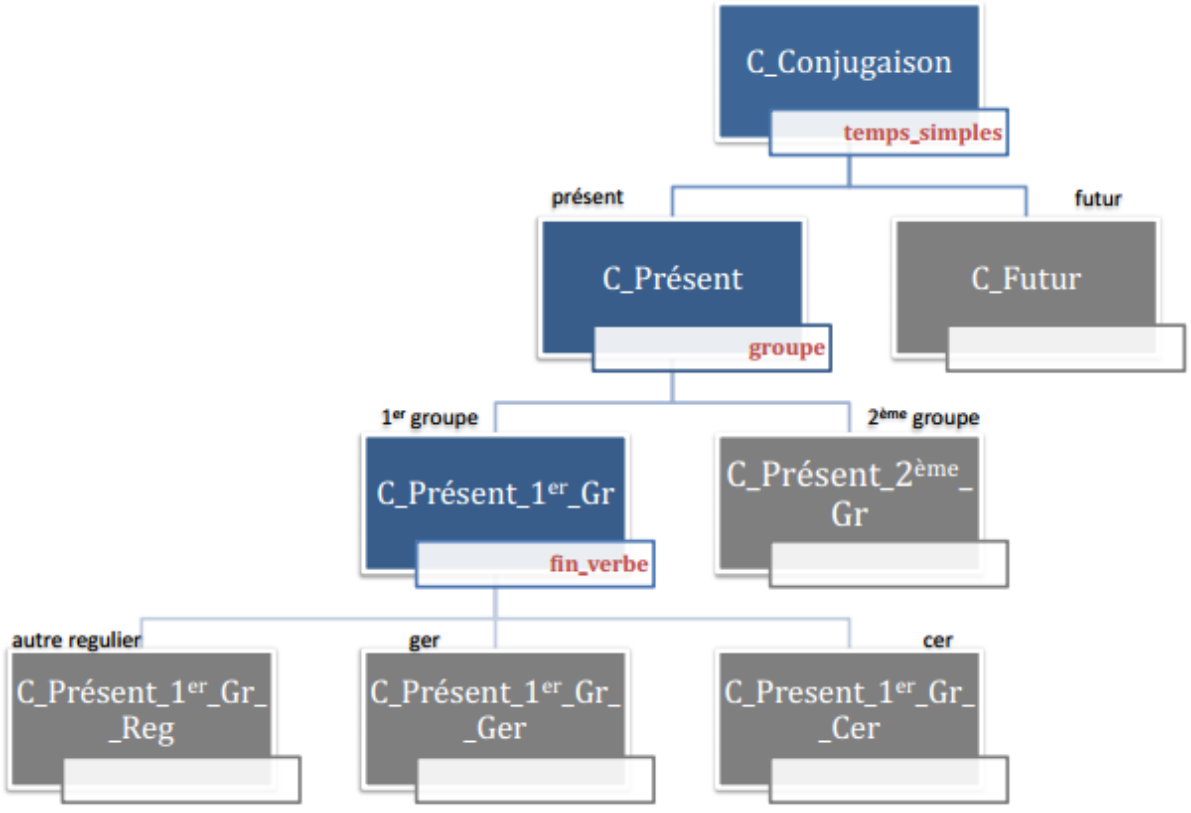

FIGURE 1.2 – Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif

Pour que le système puisse utiliser cette classification afin d'identifier la classe d'un problème, un ensemble de règles doivent être définies. Par exemple, si nous reprenons le problème P1 sur la conjugaison française (cf. Introduction générale) : « *Conjuguer le verbe parler au présent de l'indicatif* », le résolveur doit disposer de règles lui permettant, à partir de l'énoncé du problème, de calculer les valeurs des attributs discriminants afin de situer ce problème au niveau de la hiérarchie des classes. l'ensemble de toutes les règles définies pour un domaine constituent les connaissances de reformulation.

1.2.1.2 Connaissances de reformulation

Pour chaque attribut (discriminant ou non) défini dans la classification, une(des) règle(s) permettant de conclure sur sa/ses valeur(s) doit (doivent) être définie(s). Une règle est définie par son *nom*, un *ensemble de prémisses*, et un *ensemble de conclusions*. Les prémisses portent sur des éléments d'énoncé ou sur des attributs du problème qui pourront eux-mêmes nécessiter l'application de règles pour calculer leurs valeurs. Ces prémisses peuvent également faire appel à des fonctions de calcul Prolog. Les conclusions quant à elles permettent de conclure sur la valeur des attributs. On y retrouve également des appels à des fonctions de calcul.

La règle suivante permet par exemple de situer le problème *P1* au niveau de la classification (cf. **Figure 1.2**).

Règle 1 :
Si dans un problème on a :

- un verbe *V*
- la terminaison du verbe *V* est égale à *er*
- le verbe *V* est différent du verbe *aller*

Alors la valeur de l'attribut discriminant *groupe* est égale à *1er groupe*.

Cette règle stipule que si dans un problème il existe un verbe V, si la terminaison du verbe est *er* et si ce verbe est différent du verbe aller, alors la valeur de l'attribut groupe est égal à *1er groupe*.

Les connaissances de reformulation permettent de passer d'un modèle descriptif du problème à un modèle que nous appelons opérationnel - constitué des attributs ainsi que

de leurs valeurs - plus adapté à l'application d'une technique de résolution. L'ensemble des techniques de résolution constituent ce que nous nommons les connaissances de résolution.

1.2.1.3 Connaissances de résolution

Les techniques de résolution sont fortement liées aux classes de problèmes. En effet, pour connaître quelle technique employer, il faut savoir quelle technique est adaptée pour une classe de problèmes. Cependant c'est une expertise difficile à acquérir. Des recherches en Psychologie Cognitive ont montré que les experts se distinguent des novices par leur capacité, en situation de résolution de problèmes, à reconnaître la classe du problème afin d'y appliquer la technique de résolution la plus adaptée. Chi et al. [19] ont montré que les novices sont souvent en difficultés parce qu'ils n'ont pas les mêmes critères de classification que les experts.

Dans les EIAH AMBRE, une technique de résolution doit être associée à chaque classe opérationnelle définie dans l'arbre de classification. Une technique de résolution est définie par son nom, un ensemble de données portant sur le problème, un ensemble d'éléments de solution, et une fonction de calcul permettant de déterminer les éléments de solution à partir des données. Ces techniques sont spécifiques au domaine, en conjugaison par exemple, il s'agit d'une part du processus permettant d'obtenir la partie fixe pour l'ensemble des personnes et d'autre part de la liste des terminaisons pour les six personnes.

1.2.2 Processus de résolution de problèmes

Pour chaque domaine, une fois que les trois modèles de connaissances sont définis : connaissances de classification, connaissances de reformulation et connaissances de résolution, le résolveur de AMBRE met en œuvre une phase d'opérationnalisation pour résoudre des problèmes :

— **Classement du problème**. Pour classer un problème, le résolveur utilise l'arbre de classification - comme un guide - et les règles de reformulation. Il part de la racine, et affine le classement petit à petit en passant à une classe plus spécifique.

Pour descendre dans une sous-classe, il utilise les règles afin de déterminer la valeur de l'attribut discriminant de la classe et éventuellement les valeurs des attributs du problème. Il essaie ainsi d'aller le plus loin possible vers les feuilles, au moins jusqu'à une classe opérationnelle. Il obtient ainsi la classe la plus spécifique selon laquelle les connaissances permettent de modéliser le problème.

— **Construction du modèle opérationnel**. Ce modèle est construit au cours du classement. Vide au départ, il est enrichi à chaque fois que l'on descend dans une sous-classe. À la fin de l'opérationnalisation, on obtient un couple formé par la classe à laquelle appartient le problème, et le nouveau modèle du problème contenant assez d'informations permettant de le résoudre.

— **Construction de la solution du problème**. Pour trouver la solution au problème, le résolveur applique au modèle opérationnel la technique de résolution associée à la classe du problème (déterminée dans les étapes précédentes).

Le résolveur de problème et le processus de raisonnement dans son ensemble sont indépendants du domaine. En d'autres termes, leur fonctionnement reste le même quel que soit le domaine. Seul le contenu des trois bases de connaissances (connaissances de classification, connaissances de reformulation, connaissances de résolution) change suivant le domaine concerné. Cependant, le résolveur ne permet pas à lui seul de fournir de l'aide à l'apprenant, ni de diagnostiquer ses réponses ou de lui fournir des explications pertinentes sur ses erreurs. Pour assurer ces fonctions, l'EIAH a besoin de connaissances complémentaires qui lui permettent de mieux accompagner l'apprenant.

1.3 Comment guider l'apprenant dans sa résolution de problèmes ?

1.3.1 Connaissances pour guider l'apprenant durant son apprentissage

Afin de favoriser l'apprentissage, les EIAH AMBRE assistent l'apprenant dans la résolution de nouveaux problèmes en permettant d'une part de répondre à une demande d'aide de l'apprenant, et d'autre part de diagnostiquer ses réponses à chaque étape

de résolution et l'aider à réviser une réponse erronée en lui proposant des explications pertinentes. Ces connaissances sont donc fortement dépendantes des connaissances précédemment définies sur la méthode à enseigner et des étapes de résolution. Elles sont de trois types : les *connaissances d'aide*, les *connaissances de diagnostic* et les *connaissances d'explications* sur les erreurs commises.

1.3.1.1 Connaissances d'aide

Afin d'assister l'apprenant dans sa tâche de résolution, l'EIAH doit proposer à l'apprenant des aides de différentes natures en fonction de l'étape de résolution en cours et des erreurs commises. Dans AMBRE-add [63], un EIAH développé pour l'apprentissage de la résolution des problèmes additifs au primaire, des messages d'aide sont proposés à l'apprenant. Ces messages d'aide permettent de rappeler à l'apprenant ce qu'il doit faire en fonction de l'étape de résolution à laquelle il se trouve. Ils sont souvent structurés en deux niveaux. Lorsque le premier ne suffit pas à l'apprenant pour comprendre, ce dernier peut utiliser le second.

1.3.1.2 Connaissances de diagnostic

Les connaissances de diagnostic permettent de comparer la réponse de l'apprenant à celle fournie par le résolveur de AMBRE, afin de pouvoir déterminer si une réponse est correcte ou non. En effet, le résolveur donne une réponse correcte. Cependant, il peut arriver que l'apprenant donne une réponse différente de celle du résolveur quand bien même elle reste correcte. Il faut donc des connaissances pour gérer ces équivalences.

1.3.1.3 Connaissances d'explications sur les erreurs

Les connaissances d'explications font suite au diagnostic. Elles permettent de donner à l'apprenant des explications pertinentes en cas d'erreur. Il s'agit de définir un ensemble de réponses fausses attendues, et d'y associer des messages explicatifs. L'ensemble des connaissances que nous venons de décrire (connaissances sur la méthode à enseigner, connaissances d'aide, de diagnostic et d'explications, connaissances de communication) sont associés à un processus spécifique afin de permettre les fonctionnalités

recherchées, c'est-à-dire le diagnostic des réponses de l'apprenant, et la construction de messages d'aide et d'explications sur les erreurs.

1.3.2 Processus de raisonnement

Grâce aux connaissances précédemment décrites, le système peut répondre à trois types de demande : une demande de l'interface de l'EIAH, une demande d'aide de l'apprenant ou une demande de diagnostic (de l'apprenant ou de l'interface). Nous décrivons dans la section suivante les processus permettant de répondre à ces demandes.

1.3.2.1 Productions d'informations pour l'interface

Le processus de production d'informations destinées à l'interface utilise le résolveur, sa connaissance des problèmes ainsi que les connaissances de configuration de l'interface figurant dans les connaissances de communication. En début de séance par exemple, l'interface fait appel au résolveur pour présenter à l'apprenant des problèmes-types résolus. De même pour demander à l'apprenant de choisir un problème-type, l'interface a besoin de connaître les problèmes-types que l'apprenant a déjà vus. Elle doit aussi savoir quels problèmes ont déjà été résolus par l'apprenant, et à quels problème-type l'apprenant les a associés. L'interface peut également faire appel au résolveur pour constituer des éléments d'interface.

Dans AMBRE-add par exemple, pour construire les listes déroulantes utilisées par l'apprenant lors de la construction de la phrase de réponse, le système utilise les données du problème pour lui proposer les éléments corrects et incorrects : le sujet de la phrase (les prénoms de l'énoncé et d'autres noms), les objets (ceux de l'énoncé, par exemple les billes et ceux figurant dans d'autres problèmes tels que le problème-type), le verbe de la phrase (a, avait, a gagné, a perdu, a mangé, etc.).

1.3.2.2 Production d'aide

Lorsque l'apprenant demande de l'aide, l'interface de l'EIAH indique au système quel est l'avancement exact de l'apprenant dans le cycle AMBRE. Ce qui permet au

système de construire ainsi un message d'aide en utilisant les connaissances sur le cycle AMBRE et les connaissances liées à la langue naturelle. De même, afin de pouvoir mettre en évidence les éléments analogues du problème-type et du problème à résoudre, le système utilise les connaissances sur les liens entre ces éléments ainsi que les connaissances du résolveur sur les deux problèmes.

1.3.2.3 Diagnostic des réponses et production d'explications

Le diagnostic peut être effectué automatiquement par le système ou demandé par l'apprenant à la fin de chaque étape ou sous-étape de résolution. L'EIAH procède en deux temps lors d'un diagnostic. Dans un premier temps, le système utilise ses connaissances de diagnostic pour comparer la réponse de l'apprenant à celle du résolveur. Puis dans le second, il génère en cas d'erreur un message ou des informations que l'interface mettra en œuvre visuellement. Pour ce faire, il utilise les connaissances d'explications sur les erreurs en s'appuyant sur le résolveur ainsi que l'ensemble des connaissances permettant de communiquer avec l'apprenant.

1.4 Conclusion

Ce chapitre a présenté le cadre applicatif de nos recherches : le projet AMBRE. Après avoir décrit le principe des EIAH AMBRE en montrant comment le RàPC a été utilisé dans le cadre de ce projet pour soutenir la démarche d'apprentissage proposée à l'apprenant, nous avons présenté les différentes connaissances qui entrent en jeu lors de la conception de tels EIAH. Ce chapitre se termine par la présentation du processus de raisonnement associé aux différentes connaissances et permettant d'assurer les fonctionnalités attendues que sont la résolution des problèmes, l'aide, le diagnostic des réponses, et les explications sur les erreurs de l'apprenant.

Une des particularités du projet AMBRE est que les EIAH développés proposent une démarche d'apprentissage inspirée du raisonnement à partir de cas, alors que les résolveurs de problèmes se fondent sur une classification des problèmes du domaine. Face à la difficulté d'assister l'apprenant dans sa résolution de problèmes, la notion de problème prototype représentant une classe est utilisée, afin de faire le lien entre les

connaissances du système et le raisonnement de l'apprenant.

Un des problèmes majeurs de l'ingénierie des EIAH demeure leur coût de conception élevé. En particulier dans les EIAH centrés connaissances, ce phénomène s'explique en grande partie par la difficulté d'une élicitation des connaissances ayant recours à des langages de programmation : c'est le cas de Prolog dans le cadre des EIAH AMBRE. Cette nécessité de compétences en programmation de la part des concepteurs restreint ainsi l'ingénierie des EIAH à une communauté spécifique.

Notre problématique est alors de réduire ce coût de conception, tout en étendant la communauté des auteurs d'EIAH, pour une utilisation à plus grande échelle. La conception d'un outil auteur peut permettre à un auteur d'éliciter, sans recourir à de la programmation, les connaissances qui permettront à l'EIAH de résoudre les problèmes d'un domaine qu'il aura choisi. Nous ciblons des utilisateurs experts des connaissances de la méthode à enseigner (enseignant, professionnel ou pédagogue plus expert), et familiarisé avec l'outil informatique, avec ou sans compétences en programmation. Ainsi, l'outil auteur à développer doit permettre d'acquérir auprès de tels auteurs les différentes connaissances entrant en jeu lors de la conception de tels environnements. La mise en œuvre dudit outil auteur passe par la :

— Formalisation des différentes connaissances qui entrent en jeu lors de la conception de tels environnements ;

— Proposition, à travers la conception d'un outil auteur, des interfaces permettant d'accompagner l'auteur dans l'élicitation des connaissances (en lui proposant une assistance durant le processus).

Avant de répondre à notre problématique, dressons d'abord un état de l'art concernant les fondements théoriques des travaux abordés dans cette thèse.

Chapitre 2

État de l'art

Dans ce chapitre, nous nous intéressons à la problématique de l'acquisition des connaissances, plus particulièrement dans le cadre de la conception d'EIAH. Nous présentons dans un premier temps un petit historique sur l'évolution de l'acquisition des connaissances vers l'Ingénierie des Connaissances (IC) avant de décrire les méthodes et techniques d'acquisition de connaissances les plus répandues en IC. Nous montrons ensuite comment ces méthodes et techniques sont utilisées dans le domaine des EIAH et plus spécifiquement lors de la conception d'outils auteurs. Ce chapitre se termine par la synthèse des principaux verrous et manques actuels par rapport à notre problématique.

Sommaire

2.1 De l'acquisition des connaissances à l'Ingénierie des Connaissances 49
2.2 Méthodes et techniques d'acquisition de connaissances en Ingénierie des Connaissances 50
2.2.1 Méthodes empiriques 50
2.2.2 Méthodes analytiques 52
2.3 IC et EIAH 58
2.4 Outils auteurs 59
2.4.1 Outils auteurs pour la création de parcours pédagogiques . . 59
2.4.2 Outils auteurs pour la création de tuteurs intelligents 60
2.5 Synthèse des principaux verrous 62

2.1 De l'acquisition des connaissances à l'Ingénierie des Connaissances

L'acquisition des connaissances pour la conception de systèmes à base de connaissances (SBC) est l'un des principaux obstacles à leur développement. Les travaux dans ce domaine se sont clairement divisés en deux groupes ayant des objectifs relativement différents : *le transfert d'expertise* [6, 17] et l'*apprentissage automatique*. Le transfert d'expertise vise à mettre en place des moyens pour donner à un système informatique des connaissances issues d'un expert, de documents ou de manuels. L'apprentissage automatique, quant à lui, vise à doter les systèmes de capacités d'acquisition de nouvelles connaissances par eux-mêmes. Dans le cadre de ce travail, nous nous intéresserons essentiellement au transfert d'expertise dans le sens donné ci-dessus. Présentons d'abord un petit historique sur l'évolution de l'acquisition des connaissances vers l'Ingénierie des Connaissances (IC).

L'acquisition des connaissances est apparue comme un domaine de recherche à la fin des années 80 [8]. Pendant longtemps, les recherches dans ce domaine se sont focalisées sur la modélisation psychologique ou empirique des connaissances d'un expert dans le but de les coder dans un système expert [17]. Il s'agissait essentiellement d'une activité de transfert de connaissances entre des sources d'expertise (experts humains, documents, etc.) et un outil informatique, de façon à disposer d'un système à base de connaissances pouvant être consulté comme un expert.

Par la suite elle a migré vers l'organisation de modèles structurés vers des modèles conceptuels de différents types de connaissances nécessaires à la résolution des problèmes : on tente à partir d'une formulation linguistique du problème d'élaborer une représentation plus ou moins formelle du problème. En d'autres termes, il s'agissait de définir quelles connaissances sont pertinentes, pour ensuite les exploiter en formulant un modèle du problème. Les recherches en acquisition des connaissances ayant évolué ces dernières années, elles s'intéressent aujourd'hui à la mise en place de systèmes informatiques s'intégrant dans des tâches humaines faisant appel à des connaissances spécialisées [7]. Ces motivations ont fait évoluer les attentes de l'acquisition des connaissances vers l'Ingénierie des Connaissances où l'apport de méthodes et techniques reproductibles est devenu central.

2.2 Méthodes et techniques d'acquisition de connaissances en Ingénierie des Connaissances

Dès les premiers travaux sur l'acquisition des connaissances autour des années 1980, l'IC s'est positionnée comme un domaine actif proche de l'Intelligence Artificielle, mais aussi à la croisée de plusieurs disciplines concernées par la mise en ouvre, le repérage, l'analyse de connaissances comme la sémantique, l'ergonomie, la linguistique ou encore la sociologie.

Ce champ de recherche vise la définition de solutions techniques, méthodologiques et organisationnelles pour identifier des connaissances, les modéliser et les restituer au sein d'applications informatiques [7]. Plus formellement, l'IC se définit comme étant l'étude des concepts, des méthodes et des techniques permettant de modéliser et/ou d'acquérir les connaissances pour des systèmes réalisant ou aidant à réaliser des tâches se formalisant a priori peu ou pas [18]. Sa problématique et ses résultats sont marqués par l'évolution des systèmes qu'elle cherche à produire, qui sont des logiciels pouvant aider un utilisateur à mettre en œuvre des connaissances. Son usage couvre une large variété d'applications dont notamment : acquisition de connaissances auprès d'experts humains, acquisition de connaissances à partir d'un corpus de textes, recherche d'informations dans des bases de données ou sur le Web, mémoire d'entreprise, etc. Nous nous intéresserons dans cette thèse à l'acquisition de connaissances auprès d'experts humains.

Lors de la mise en œuvre des premiers systèmes experts, les méthodes utilisées pour effectuer le transfert de la connaissance depuis le monde réel vers le système artificiel étaient totalement empiriques. Elles se sont par la suite précisées et structurées (méthodes analytiques), voire même automatisées pour certaines. Nous détaillons ci-dessous quelques méthodes et techniques d'acquisition de connaissances les plus répandues en Ingénierie des Connaissances.

2.2.1 Méthodes empiriques

Les premiers développements de systèmes experts se sont faits sans aucune méthode précise quant à l'acquisition du savoir-faire des experts. Le recueil de la connaissance et

sa traduction étaient alors effectués par un ou plusieurs ingénieurs de la connaissance. Cette phase de transfert d'expertise est souvent considérée comme le « *goulot d'étranglement* » du développement d'un système expert. En effet, elle peut durer des mois, voire des années, et compte-tenu de la complexité des connaissances de l'expert qui parvient difficilement à expliciter ses processus mentaux, la connaissance extraite risque souvent d'être inexacte, incomplète, voire inconsistante [28] .

De nombreuses techniques de verbalisation et d'aide aux interviews ont été développées dans le but d'aider à mener correctement le processus d'acquisition de connaissances et ainsi faciliter le développement de bases de connaissances [6, 13, 29, 85] ; Ces techniques, souvent inspirées des travaux en psychologie cognitive, sont entre autres :

— Les **interviews**, généralement enregistrées et souvent non dirigées en début d'extraction.

— L'**analyse de protocoles verbaux**, où l'expert résout un problème en réfléchissant à haute voix.

— L'**observation directe**, où l'expert est filmé en train d'exercer son activité dans son environnement habituel. Les actions de l'expert, sa démarche générale, les cas et types de problèmes traités sont ainsi notés. Cette technique est un bon moyen d'accéder aux connaissances de contrôle qui échappent aux verbalisations.

— La **simulation du comportement** du futur système par l'expert.

— Les **commentaires sur des protocoles** antérieurs (reprise de cas).

— Le **tri** par l'expert des problèmes suivant leur degré de difficulté, le type de résolution attachée, etc.

— La **consultation de documents** : le cogniticien lit des documents conseillés par l'expert (documents généraux sur le domaine, documents particuliers rédigés par l'expert, etc.).

— L' **utilisation d'un questionnaire**, une fois qu'une partie de l'expertise a été identifiée.

— Le **brainwriting** pour regrouper et comparer les idées de plusieurs spécialistes : chacun enrichit les idées des autres en les commentant par écrit (une feuille par idée par exemple).

— Etc.

Dans chacune de ces techniques, le plus difficile était de minimiser les déformations de l'expertise. En effet, le décalage entre le langage dans lequel l'expert exprime ses connaissances et le niveau d'abstraction des formalismes classiques de représentation des connaissances apparaît très vite comme une difficulté, un facteur limiteur. Progressivement, les chercheurs se sont rendu compte que le problème n'était pas simplement un problème de transcription, mais fondamentalement un problème de modélisation. L'acquisition des connaissances se fixa alors pour nouveau but la construction de modèles [21, 26, 86]. À partir de ce moment, de nombreux travaux se sont engagés à définir des méthodes et outils pour assister le cogniticien chargé de mettre au point une base de connaissances : les méthodes analytiques.

2.2.2 Méthodes analytiques

Plus structurées que les méthodes empiriques, les méthodes analytiques nécessitent l'intervention d'un cogniticien. Un des points forts de ces méthodes est qu'elles contribuent à construire un modèle de connaissances intermédiaire entre le monde réel et le monde symbolique, permettant ainsi une meilleure appréhension des connaissances acquises. Nous allons présenter brièvement deux de ces méthodes, pertinentes par rapport à notre problématique : KADS et KOD. Ces méthodes ont pour intérêt d'être assez générales pour être utilisées dans plusieurs domaines, de prendre en compte tout le processus d'acquisition des connaissances, et de proposer chacune un modèle de la connaissance experte.

2.2.2.1 KADS

KADS (Knowledge and Analysis Design Support) [75, 87] est une méthodologie orientée vers la conception de systèmes à base de connaissances. C'est une méthode dirigée par les modèles qui propose :

— Un **cycle de vie** comprenant plusieurs phases : analyse, conception, implémentation, installation, utilisation, maintenance et enrichissement des connaissances (cf. **Figure 2.1**).

— Un **ensemble d'outils conceptuels** pour la modélisation des connaissances, in-

dépendamment du langage d'implémentation.

— Et un **support logiciel** avec l'outil SHELLEY [14] guidant la démarche du cogniticien (module de conseil pour l'utilisation de la méthodologie, modules permettant la structuration et l'analyse des interviews, la conceptualisation du domaine en entités et relations, la proposition de plusieurs modèles d'interprétation, etc.).

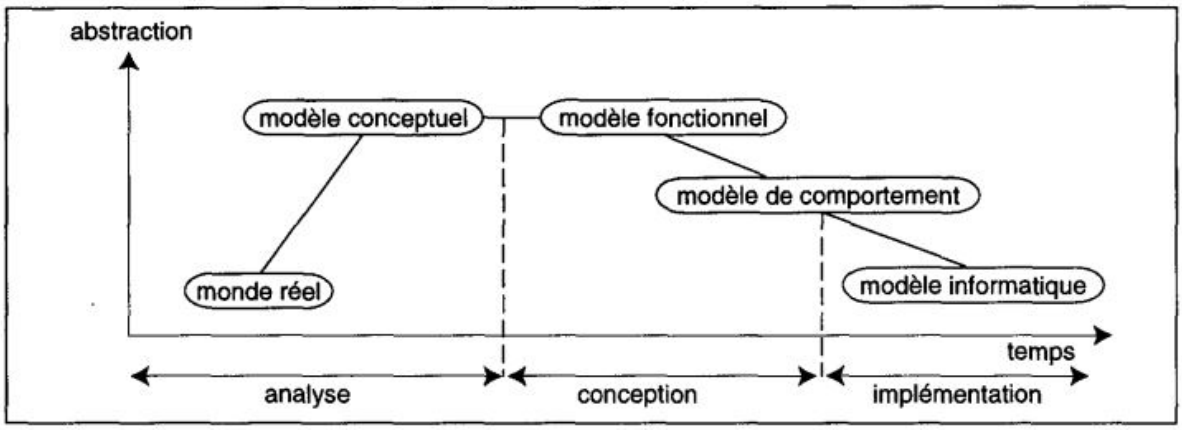

FIGURE 2.1 – KADS : cycle de vie d'un système expert [31]

Dans KADS, le processus de développement d'un SBC consiste à construire successivement plusieurs modèles : le **modèle d'expertise**, le **modèle de coopération** entre le futur système et ses utilisateurs, le **modèle externe** qui intègre les contraintes qui vont influer sur la conception et la réalisation du système final, le **modèle de conception fonctionnelle**, le **modèle logique** puis le modèle physique du système. Une bibliothèque de modèles de tâches génériques appelés modèles d'interprétation permet une acquisition structurée des connaissances.

La définition du modèle d'expertise dans KADS s'appuie sur une structure à quatre niveaux :

— Le **niveau domaine** qui décrit les connaissances statiques du domaine (indépendamment de son utilisation) en termes d'objets, de concepts, d'attributs et de relations entre ces divers éléments.

— Le **niveau inférence** permettant de décrire les inférences qui peuvent être faites sur les connaissances du niveau précédent. Il contient un ensemble de fonctions permettant de faire des inférences lors de la résolution de problèmes. Ces fonctions sont appelées sources de connaissances et leurs arguments méta-classes. Une méta-

classe indique comment un ensemble spécifique de concepts du domaine peut être utilisé.

— Le **niveau tâches** décrit comment appliquer les inférences possibles (les étapes du raisonnement) en fonction des buts. Il s'agira d'une part de préciser comment la structure d'inférence sera parcourue, et d'autre part d'associer les éléments de contrôle nécessaires.

— Le **niveau stratégie** contient les connaissances conduisant à l'élaboration du plan d'action. Il décrit le plan de résolution du problème posé ainsi que les connaissances nécessaires au contrôle de celui-ci. Le plan de résolution précise l'ordonnancement des buts à atteindre pour résoudre le problème et formalise le comportement de résolution du problème : par exemple comment résoudre le problème, comment réaliser le contrôle de l'exécution des buts, comment réagir en cas d'impasse.

Pour faciliter l'identification de ces différents niveaux, KADS offre :

— Des *techniques d'analyse de protocoles* pour identifier les éléments du niveau domaine.

— Un *ensemble de primitives d'inférence* pour l'identification des connaissances du niveau inférence.

— Une *bibliothèque de modèles d'interprétation* qui sont des modèles génériques pour la résolution d'un type de problème particulier et qui servent de cadre pour l'élaboration du modèle des tâches et de guide pour les séances de recueil des connaissances.

— Le niveau stratégie quant à lui ne bénéficie d'aucun outil, son identification étant laissée à l'initiative du cogniticien.

La méthodologie KADS a connu plusieurs évolutions. La version la plus récente est commonKADS [74, 90], développée par Cap gemini et qui propose de construire le modèle conceptuel autour de cinq modèles : le modèle d'organisation, le modèle des tâches, le modèle des agents, le modèle des connaissances (qui précise les types, les structures et les rôles de la connaissance) et le modèle de communication.

Conclusion

KADS permet de traiter tout le processus de construction d'un SBC, du recueil au développement du système. Pour faciliter l'analyse des connaissances, KADS propose une structure à quatre niveaux permettant la formalisation des connaissances, ainsi qu'une bibliothèque de modèles d'interprétation. L'utilisation des modèles génériques permet de faciliter la construction d'un modèle d'expertise : le cogniticien travaille par affinements successifs des éléments des structures génériques. Le recueil des connaissances est alors guidé par les modèles.

L'intérêt majeur de KADS est qu'elle offre un support pour guider l'acquisition des connaissances. L'autre intérêt réside dans la généricité et la réutilisabilité des modèles qui en résultent. Cependant, un des inconvénients de KADS est qu'il transforme l'activité d'analyse en une activité de choix de modèles qui nécessite une bonne expérience de manipulation des modèles d'interprétation. De plus, il est nécessaire de disposer d'une large bibliothèque de modèles et/ou de composantes génériques. En outre, l'adaptation d'un modèle existant à une application spécifique est généralement une tâche difficile. Nous constatons également qu'à la fin de la construction des différents modèles, KADS ne propose aucun lien pour passer des différents niveaux de connaissances vers les langages de représentation disponibles actuellement. Dans la section qui suit, nous présentons KOD, qui elle, spécifie les langages correspondant à chaque type de connaissances obtenu par cette méthode

2.2.2.2 KOD

KOD (Knowledge Oriented Design) [85] est une méthodologie d'acquisition de connaissances mise au point par Claude Vogel. Contrairement à KADS, le cycle de vie de KOD ne couvre que l'aspect "base de connaissances". L'atelier proposé par KOD comme support logiciel est la K-STATION. C'est une approche ascendante qui sépare le transfert d'expertise en deux étapes séquentielles :

— L' **analyse cognitive** : elle consiste à comprendre et à modéliser le savoir de l'expert. Cette étape s'appuie sur l'utilisation de techniques issues des travaux en linguistique et en anthropologie pour le découpage et l'analyse du discours de l'expert.

— La **réalisation informatique** : elle consiste à développer le SBC proprement dit, en utilisant les résultats obtenus lors de l'analyse cognitive.

La méthode KOD s'intéresse donc au recueil des connaissances d'un expert, à leur modélisation à partir de sources d'informations (discours de l'expert, documentations, observation, etc.), puis à la spécification de la base de connaissances du futur système, selon une analyse en plusieurs étapes des données recueillies. Ses fondements linguistiques la rendent bien adaptée pour l'acquisition de connaissances exprimées en langage naturel. Ainsi, elle propose un cadre méthodologique pour guider la collecte des termes et les organiser à partir d'une analyse terminologique. Grâce à ses fondements anthropologiques, le cogniticien dispose d'un cadre méthodologique facilitant l'analyse sémantique de la terminologie utilisée, pour produire un modèle cognitif. Enfin, KOD guide le travail du cogniticien depuis l'extraction de la connaissance jusqu'à l'élaboration du modèle informatique.

Pour mener l'analyse cognitive, KOD propose une décomposition en trois étapes donnant lieu à trois niveaux de modèles :

- Le **modèle pratique** : il permet de définir des dictionnaires de connaissances à partir d'une analyse textuelle des entretiens. Cette étape consiste à analyser le discours de l'expert en identifiant les termes pertinents et en les étiquetant selon une typologie clairement définie :
  - Les *taxèmes*, qui décrivent les objets du monde physique.
  - Les *actèmes*, qui décrivent le changement d'état causé par un sujet sur un destinataire.
  - Les *schémèmes* : des schémas d'interprétation (déduction, induction) qui se manifestent par des contraintes et des inférences. Ils permettent d'établir de nouvelles propositions à partir de celles déjà acquises.
- Le **modèle cognitif** : il permet de structurer et valider les connaissances acquises. Il se compose de trois structures principales :
  - Une *taxinomie*, qui organise les taxèmes selon la relation *est-un*.
  - Les *actinomies*, qui articulent la structuration des schémas mentaux de l'expert permettant de répondre à des situations déjà connues.
  - Et des *schémas d'interprétation*, constitués par des ensembles de règles d'inférence associées correspondant aux stéréotypes de raisonnement de l'expert, qui se déclenchent selon les observations du domaine.
- Le **modèle informatique** : il permet la mise en oeuvre des connaissances dans un système informatique. Il est constitué d'objets, de méthodes et de règles. La

taxonomie devient une hiérarchie d'objets, les actinomies donnent les méthodes, et les schémas donnent des règles. Vogel considère que les différents modèles de programmation actuels se réfèrent aux paradigmes suivants :

- La programmation par objets, qui permet d'exprimer les données et les taxonomies ;
- La programmation par acteurs, qui permet d'exprimer les traitements et les actionomies ;
- La programmation par la logique, qui permet d'exprimer les contraintes et les règles.

Conclusion

L'un des avantages de la méthode KOD est qu'elle laisse la liberté aux experts d'exprimer leurs perceptions ainsi que leurs tâches sans les contraindre [11] Cette approche était largement utilisée pour l'élaboration des premières générations des systèmes experts. Aujourd'hui, elle reste particulièrement adaptée à des domaines où il n'existe pas de modèles de connaissances prédéfinis, et où il manque de la formalisation. L'exploration d'un nouveau domaine en est un exemple. La K-STATION, qui se présente comme une plate-forme logicielle, offre un ensemble d'outils pour aider à la mise en œuvre de la méthode KOD. Certains de ces outils, de type hypertexte, permettent de gérer les fichiers de retranscription, de conceptualiser les informations qu'ils contiennent à l'aide des types de structures constituées.

Cependant, comme la plupart des approches, KOD présente un certain nombre d'inconvénients [30] : le coût élevé en termes de temps et d'expertise dans le processus d'élicitation ainsi que dans le processus de validation, aussi bien pour les experts du domaine que pour les ingénieurs de la connaissance. Les modèles issus de cette approche souffrent aussi d'un manque d'abstraction, de généricité et sont donc difficilement réutilisables [33].

Comme KADS, KOD détaille également les différentes phases du projet de construction d'un SBC et propose un modèle de connaissances à l'expert. En revanche, elle ne propose rien qui permet et de distinguer les différents niveaux de connaissances de l'expert, ce qui peut nuire à l'explicabilité du système ainsi construit. On peut également lui reprocher sa lourdeur, le manque de méthode dans la conduite des entretiens, le coût et la lenteur de leur structuration systématique ainsi que la formalisation presque automatique de connaissances dont beaucoup ne sont pas utiles à la conception du système.

2.3 IC et EIAH

Ingénierie des Connaissances (IC) et EIAH sont deux domaines distincts. Un système à base de connaissances et un EIAH sont deux objets de nature différente. En effet, des problèmes spécifiques sont soulevés lors de la conception d'un EIAH : perception et interprétation des actions de l'apprenant, contrôle de l'interaction, prise en compte des caractéristiques individuelles des apprenants et de l'évolution de leurs connaissances, évaluation des apprentissages, etc. [10].

Cependant, comme le souligne [79], ces deux domaines sont suffisamment voisins par la nature de certaines problématiques qu'ils abordent : représentation et modélisation des connaissances en IC, versus modélisation des connaissances du domaine d'apprentissage, des connaissances de l'apprenant, etc. en EIAH ; résolution de problème en IC, versus résolution d'exercices, diagnostic des réponses de l'apprenant, etc. en EIAH. Alors que l'IC fournit sur ces aspects des outils (conceptuels, techniques, méthodologiques) utiles à la construction d'EIAH, le domaine des EIAH quant à lui présente des spécificités dont la prise en compte a notablement contribué à l'évolution de l'IC. On peut citer par exemple les travaux de Clancey [22] consacrés à la recherche d'une utilisation pédagogique de MYCIN. Les travaux en EIAH sur la notion de connaissances ou l'étude des interactions épistémiques sont également pertinents pour l'IC. En effet, l'avènement des TIC a amené de nouveaux points de convergence entre les deux domaines. On peut citer par exemple la construction et l'utilisation d'ontologies [24, 54, 78], la modélisation de documents pédagogiques en XML, les hypermédias adaptatifs, etc. Enfin, l'IC et le domaine des EIAH se rejoignent très largement sur les travaux sur l'apprentissage et le travail collaboratif (CSCL : Computer Supported Collaborative Learning, CSCW : Computer Supported Collaborative Work), deux problématiques que certains proposent même de confondre [9]. Dans le domaine des EIAH, les outils auteurs ont été proposés pour aider les concepteurs d'EIAH, entre autres, dans l'élicitation des connaissances nécessaires à l'EIAH, sans aucune ou peu de connaissances en programmation. Nous présentons dans la section suivante les méthodes et techniques d'acquisition de connaissances issues de l'IC et utilisées lors de la conception d'outils auteurs en EIAH.

2.4 Outils auteurs

Les outils auteurs sont des environnements informatiques permettant de construire tout ou partie des EIAH avec un coût réduit ou nul en programmation. De manière générale, les outils auteurs ont pour objectif de minimiser, d'une part le coût de conception des EIAH, et d'autre part la nécessité d'une grande expertise chez l'auteur [3]. Ils permettent par exemple d'éliciter les connaissances nécessaires à l'EIAH, de construire graphiquement l'interface de l'EIAH en utilisant des Toolkits et UI-Builder, de créer les exercices à résoudre, de proposer de l'aide à l'apprenant, de diagnostiquer ses réponses, etc. [57,59]. Les recherches sur les outils auteurs ont débuté il y a une quinzaine d'années. Ce champ permet d'espérer résoudre le problème de la difficulté et du coût de conception des EIAH. En effet, comme le soutient Woolf [89], une plus grande offre d'outils auteurs permettrait de supporter le développement plus rapide d' EIAH, les rendant ainsi plus variés.

À l'heure actuelle, plusieurs outils auteurs - plus de deux douzaines - ont été proposés dans la littérature. Ils sont souvent divisés en deux grandes catégories [57,60] :

— Ceux s'attachant à construire des parcours pédagogiques à partir de ressources existantes ;

— Ceux visant la construction d'EIAH de type tuteurs intelligents, fondés sur une représentation des connaissances permettant la résolution de problèmes et la construction de rétroactions pertinentes pour l'apprenant.

2.4.1 Outils auteurs pour la création de parcours pédagogiques

Dans la catégorie d'outils auteurs destinés à créer des parcours pédagogiques, nous retrouvons les travaux autour des scénarios d'apprentissage [34,39] et de la modélisation pédagogique [48]. Les travaux sur les outils auteurs permettant de créer des hypermédias adaptatifs [34] tels que les interbook, métalinks, et ceux permettant d'expliciter des stratégies d'enseignement [1] font également partie de cette catégorie.

Énormément d'outils ont été créés dans cette catégorie. Toutefois, de manière générale, la limite notoire de ces outils est le manque de connaissances sur le domaine et sur l'apprenant [88]. Ces outils ne répondent donc pas à notre problématique, plutôt centrée

sur la représentation des connaissances. C'est la raison pour laquelle nous ne présentons pas individuellement ces outils. Nous nous focalisons plutôt sur les outils destinés à créer des tuteurs intelligents, qui sont les plus proches de notre problématique.

2.4.2 Outils auteurs pour la création de tuteurs intelligents

Les outils auteurs développés dans cette catégorie visent la construction d'EIAH de type *tuteurs intelligents*, fondés sur une représentation des connaissances. Dans cette catégorie, certains outils se sont focalisés sur l'acquisition de connaissances procédurales. On y retrouve des systèmes opérationnels tels que Knomic [82], Disciple [80], Demonstr8 [12], RIDES [56], XAIDA [60], SIMQUEST [81], etc.

Knomic par exemple, est un système orienté learning-by-observation qui permet d'acquérir des connaissances procédurales dans un environnement de simulation. Pour ce faire, l'expert effectue ses tâches dans l'environnement simulé en annotant les points où il/elle a changé un but qu'il n'a pas réussi ou a abandonné. Le système génère ensuite les traces d'observations correspondantes. Knomic observe ces traces et utilise un algorithme de généralisation pour apprendre les conditions des actions, buts et opérateurs. Demonstr8 quant à lui est un outil auteur destiné à créer des model-tracing tuteurs pour le domaine de l'arithmétique. Afin de réduire le coût de conception de ces tuteurs, cet outil utilise la programmation par démonstration. Il propose un outil graphique pour concevoir l'interface de l'EIAH. Le système définit automatiquement chaque élément de l'interface comme un élément de mémoire de travail (WME), et les WME sont définis manuellement. Le système génère ensuite des règles de production en observant la résolution des problèmes. Cependant, ces outils se limitent à l'acquisition de connaissances procédurales. Ils ne permettent donc pas d'acquérir toutes les connaissances nécessaires aux EIAH de type AMBRE. En effet, dans AMBRE, en plus des connaissances de reformulation (sous forme de règles), les systèmes à base de connaissances sous-jacents utilisent des connaissances de classification de problèmes (sous forme d'arbre), des connaissances sur l'analogie entre les problèmes, des connaissances de génération de problèmes, des connaissances de communication avec l'apprenant, etc.

D'autres outils auteurs se sont concentrés sur la création de tuteurs à base de contraintes. Dans ASPIRE [52,53] par exemple, les auteurs utilisent des ontologies pour recueillir les connaissances expertes : (1) construction d'une ontologie du domaine, (2)

acquisition de contraintes syntaxiques directement à partir de l'ontologie, (3) mise en place d'une boîte de dialogue avec l'auteur afin de déduire des contraintes syntaxiques, (4) et utilisation d'un algorithme d'apprentissage pour déduire les contraintes sémantiques. Cependant, ces outils sont limités à des domaines où l'ensemble des connaissances peuvent être représentées sous forme de contraintes. Une contrainte est représentée par le motif <Cr,Cs>, tel que Cr représente le critère de pertinence, et Cs le critère de satisfaction. Plus spécifiquement, Cr spécifie l'ensemble des situations pour lesquelles la contrainte est pertinente, et Cs l'ensemble des situations pour lesquelles la contrainte est satisfaite. La sémantique représentée par une contrainte est que si Cr est satisfait dans une situation, alors Cs devrait aussi l'être, sinon il y a erreur.

Les outils auteurs développés autour de CTAT (Cognitive Tutor Authoring Tools) permettent la création de deux types de tuteurs : les *cognitives tutors* [46] et les *example-tracing tutors* [4, 5, 46, 47]. Les premiers sont fondés sur un modèle cognitif à base de règles de production, et portent en général sur des tâches de résolution. La résolution est conduite pas à pas, et le comportement de l'élève est analysé à chaque étape, puis corrigé s'il s'écarte de la procédure prévue. Cependant, les cognitives tutors se limitent aux domaines ou la tâche de résolution de problèmes s'effectue pas à pas et ou l'ensemble des connaissances du domaine peut s'exprimer sous forme de règles de production. Ils ne permettent donc pas l'apprentissage de connaissances déclaratives comme les concepts, ni l'apprentissage à partir d'exemples [89]. De plus, la création de outils nécessite des compétences en programmation. Les example-tracing tutors [4] ont quant à eux l'avantage d'être développés rapidement et sans programmation. Pour construire un tuteur de ce type, l'enseignant commence d'abord par définir l'interface de l'apprenant à l'aide d'outils graphiques. Il définit ensuite, à l'aide d'un graphe de résolution, les stratégies de résolution des apprenants et leurs misconceptions. Cependant, les example-tracing tutors présentent les mêmes inconvénients que les cognitive tutors concernant la résolution des problèmes qui est conduite pas à pas, et par conséquent ne répondent pas au principe des EIAH AMBRE qui demandent à l'apprenant de prendre du recul avant d'entamer la résolution.

2.5 Synthèse des principaux verrous

Nous avons vu dans ce chapitre que la littérature en IC abonde de méthodologies d'acquisition de connaissances. La méthodologie commonKADS a particulièrement retenu notre attention en ce sens qu'elle est basée sur une démarche d'acquisition de connaissances dirigée par les modèles. Cependant, ses composants génériques permettant de résoudre des problèmes très généraux ne sont pas forcément adaptés à ce que nous cherchons à enseigner en EIAH. En effet, dans le domaine des EIAH, la démarche d'élicitation se particularise par le fait que :

— Il ne s'agit pas de modéliser les connaissances de l'expert, mais les connaissances telles qu'on voudrait qu'elles fonctionnent chez l'élève après l'apprentissage ;

— Il existe d'autres types de connaissances à modéliser, telles que les connaissances de diagnostic des compétences de l'apprenant, ou les connaissances permettant de lui fournir des rétroactions ou de l'assistance. Or ces connaissances ne sont pas des connaissances de résolution de problèmes proprement dites : par exemple, lorsque l'on parle de diagnostic des connaissances de l'apprenant en EIAH, il ne s'agit pas de la même tâche qu'un diagnostic de panne que l'on pourrait modéliser avec la méthode CommonKads.

C'est pourquoi, compte tenu des problématiques spécifiques au champ des EIAH, nous ne retiendrons de cette méthode que sa démarche.

La méthodologie KOD, quant à elle, présente l'avantage d'être bien adaptée à l'acquisition de connaissances exprimées en langage naturel. Cependant, les modèles issus de cette approche souffrent d'un manque d'abstraction, de généricité, et sont donc difficilement réutilisables. De plus, KOD ne propose rien permettant de distinguer les différents niveaux de connaissances de l'expert, ce qui peut nuire à l'explicabilité du système. Un système destiné à enseigner une stratégie de résolution (comme ceux produits dans AMBRE) doit avoir accès à une représentation explicite des connaissances et du processus mis en jeu. Cela permet de faciliter la production d'explications au sein de l'EIAH ; celles-ci sont d'autant plus faciles à donner que les connaissances sont explicitées. Par conséquent, nous ne retiendrons de cette méthode que son approche ascendante.

À l'heure actuelle, plus de deux douzaines d'outils auteurs ont été proposés dans le domaine des EIAH. Ces outils diffèrent pour la plupart par le type de domaine ou de tâches prises en compte, le degré par lequel ils facilitent la création d'EIAH, et les

possibilités offertes pour représenter ou non les connaissances ou les compétences à enseigner [58]. Chacun de ces outils a ses forces et faiblesses. C'est pourquoi nous proposons de discuter des limites des approches existantes en considérant deux critères d'analyse liés à notre problématique de recherche : l'approche de résolution de problème adoptée, et la possibilité de représenter toutes les connaissances dont on a besoin (cf. Chapitre 1).

Approche de résolution de problèmes

Les outils développés autour de CTAT sont les plus aboutis actuellement en terme de création d'EIAH centrés connaissances. Cependant, ils se limitent à des domaines ou la tâche de résolution de problèmes s'effectue pas à pas. Ils ne répondent donc pas à notre problématique d'outils auteurs pour l'enseignement de méthodes, qui demande à l'apprenant de prendre du recul et de raisonner globalement sur le problème à résoudre.

Possibilité de représenter les connaissances dont a besoin

La principale limite des outils auteurs destinés à l'acquisition de connaissances procédurales est qu'ils ne permettent pas de représenter toutes les connaissances nécessaires à la conception des EIAH auxquels nous nous intéressons (arbre de classification, règles, techniques de résolution). En effet, ces outils se limitent à des domaines où l'ensemble des connaissances peuvent être représentées sous forme de connaissances procédurales. La limite des outils auteurs basés sur les contraintes repose sur le fait que les connaissances du domaine sont représentées par un ensemble de contraintes. En effet, les connaissances nécessaires à la conception d'EIAH destinés à l'enseignement de méthodes ne peuvent pas être représentées sous forme de contrainte.

Notre analyse de l'existant ne nous a pas permis de trouver d'approche permettant de répondre complètement à notre problématique : comment permettre à un auteur d'éliciter les connaissances nécessaires à conception d'un EIAH destiné à enseigner des méthodes. L'approche que nous proposons consiste donc à :

— Proposer des méta-modèles de connaissances pour formaliser les connaissances qui entrent en jeu lors de la conception des EIAH destinés à enseigner des méthodes. Pour cela, nous adopterons une approche ascendante pour la construction des méta-modèles ;

— Proposer un processus d'élicitation des connaissances en nous fondant sur ces méta-modèles (approche dirigée par les méta-modèles). Il s'agira en particulier de proposer des interfaces permettant d'accompagner l'auteur dans l'élicitation des connaissances.

Deuxième partie

CONTRIBUTIONS THÉORIQUES

Chapitre 3

Des méta-modèles pour guider l'élicitation des connaissances

Dans ce chapitre, nous décrivons les méta-modèles que nous avons proposés pour formaliser les connaissances nécessaires à la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes. Ces méta-modèles sont indépendants du domaine. Ils permettent d'une part de définir la forme des connaissances à acquérir, et d'autre part de contraindre le processus d'acquisition de ces connaissances décrit dans le chapitre 4.

Une première partie présente les méta-modèles permettant de décrire les problèmes à résoudre, alors que la suivante se consacre aux méta-modèles des connaissances portant sur la méthode à enseigner. Ce chapitre se termine par la description des méta-modèles des connaissances permettant d'accompagner l'apprenant durant son apprentissage.

Sommaire

3.1 Comment décrire les problèmes d'un domaine ? 69
3.1.1 Méta-modèle du vocabulaire permettant de décrire des problèmes 70
3.1.2 Méta-modèle permettant de décrire des problèmes 71
3.2 Comment définir les connaissances sur la méthode à enseigner ? 73
3.2.1 Méta-modèle des connaissances de classification 74
3.2.2 Méta-modèle des connaissances de reformulation 77
3.2.3 Méta-modèles des connaissances de résolution 82
3.3 Comment définir les connaissances permettant d'accompagner l'apprenant ? 86
3.3.1 Méta-modèle des connaissances d'aide 87
3.3.2 Méta-modèle des connaissances de diagnostic 89
3.3.3 Connaissances d'explications sur les erreurs 91
3.4 Conclusion 93

Les travaux menés dans le cadre du projet AMBRE et présentés dans le chapitre 1 (cf. chapitre 1) de ce rapport permettent la conception d'EIAH destinés à l'enseignement de méthodes. Ces EIAH proposent à l'apprenant un processus d'apprentissage inspiré du raisonnement à partir de cas : le cycle AMBRE [43]. Ils reposent sur des systèmes à base de connaissances leur permettant de résoudre des problèmes en appliquant la méthode à enseigner, de diagnostiquer les réponses de l'apprenant, et de lui fournir de l'aide et des explications pertinentes sur ses erreurs. La conception de tels EIAH est cependant très coûteuse, en particulier du fait que les connaissances doivent être explicitées en Prolog, un langage informatique de représentation des connaissances utilisé par ces systèmes.

Afin de remédier à ce problème, nous souhaitons permettre à des auteurs (enseignant, professionnel ou pédagogue plus expert), de pouvoir expliciter eux-mêmes les connaissances nécessaires à l'EIAH, sans recourir à de la programmation. Il devient dès lors nécessaire d'identifier et de formaliser les différents types de connaissances entrant en jeu dans la conception d'un EIAH AMBRE. Pour ce faire, nous avons proposé des méta-modèles décrits par des schémas XML. Ces méta-modèles constituent ainsi une réponse au problème de formalisation des connaissances posé dans l'introduction générale de ce manuscrit. Pour la construction de nos méta-modèles, nous avons adopté une méthode ascendante [77] en comparant les bases de connaissances des différents EIAH AMBRE existants.

3.1 Comment décrire les problèmes d'un domaine ?

Dans AMBRE, les problèmes à résoudre ne sont pas donnés au système en langue naturelle, mais plutôt en Prolog sous la forme d'un modèle que nous appelons modèle descriptif du problème [40]. Ce modèle décrit une situation concrète qui est celle représentée dans l'énoncé. Il contient des traits de surface du problème considéré : par exemple dans les problèmes additifs, ce sont les éléments concrets tels que billes, les jetons, les bonbons, etc. Ils forment l'aspect contextuel du problème appelé aussi encodage linguistique.

Pour permettre au résolveur de AMBRE de résoudre des problèmes, il faut d'abord pouvoir décrire ces problèmes, c'est-à-dire les éléments de l'énoncé du problème. De plus, la description des règles et des techniques de résolution (cf. Chapitre 1) nécessite souvent

aussi de pouvoir se référer aux éléments de l'énoncé. Une question importante à laquelle il fallait répondre était alors :
Comment décrire les éléments descriptifs de l'énoncé des problèmes ?
Notre réponse à cette question a été de proposer que pour chaque domaine, la description des problèmes soit fondée sur un vocabulaire permettant de décrire les objets du domaine présents dans les énoncés. Ce vocabulaire va servir d'élément d'ancrage autour duquel s'organise la description des problèmes à résoudre, les règles permettant de les situer au niveau de la classification des problèmes, et les techniques intervenant dans leur résolution. De ce fait nous avons proposé un méta-modèle permettant de décrire un vocabulaire pour un domaine donné.

3.1.1 Méta-modèle du vocabulaire permettant de décrire des problèmes

Pour créer le vocabulaire d'un domaine, il faut donc créer les types d'objets qui vont le constituer. Chacun d'entre eux doit posséder un *identifiant* et une *liste de caractéristiques*. Chaque caractéristique a un *nom* et un *type* qui peut être simple (chaîne de caractères, booléen, nombre, etc.) ou complexe (un objet représentant une relation avec un autre objet). Par exemple, dans les problèmes additifs que nous présentons dans les sections suivantes, pour décrire les listes d'éléments possédés par chaque personne, un objet *liste_éléments_possédes* avec les caractéristiques suivantes peut être créé : *identifiant* de type chaîne de caractères, *taille_des_éléments_possédés* de type nombre entier, et le *type_des_éléments_possédés* de type chaîne de caractères.

La **Figure 3.1** présente le méta-modèle proposé pour définir le vocabulaire pour un domaine donné. Pour chaque domaine choisi (cf *nom_du_domaine*), il faut alors définir le vocabulaire, c'est-à-dire créer les types d'objets nécessaires pour décrire les problèmes de ce domaine. Créer un type d'objet revient à définir :

— Son **identifiant**

— La liste de ses **caractéristiques**. Pour chaque caractéristique, il faut préciser le nom (cf nom_caractéristique) ainsi que le type (cf type_caractéristique) qui peut être : type_simple ou type_objet.

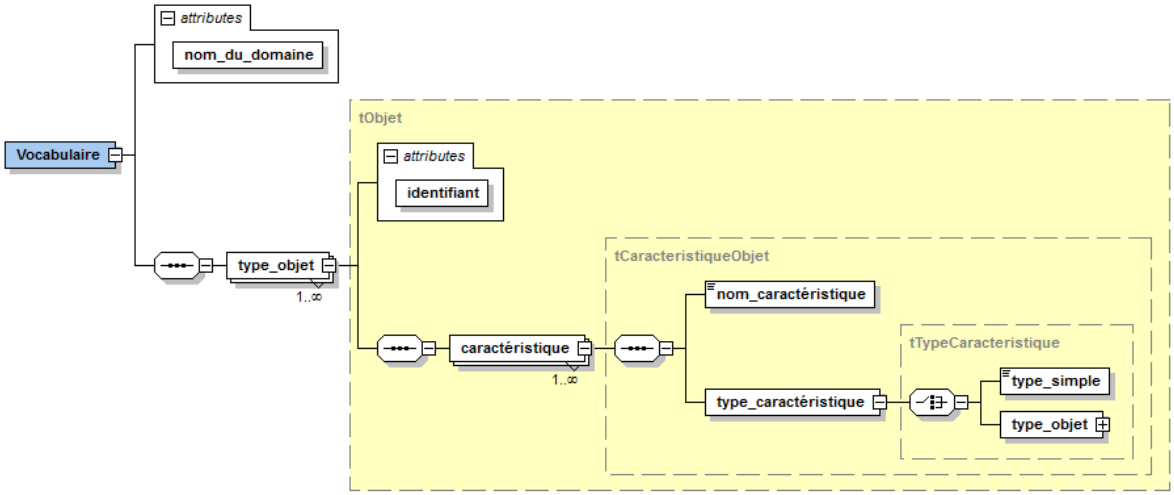

FIGURE 3.1 – Méta-modèle permettant de décrire le vocabulaire d'un domaine

Une fois défini, le vocabulaire pourra alors être utilisé dans la description des différents problèmes à résoudre par le système et par l'apprenant.

3.1.2 Méta-modèle permettant de décrire des problèmes

La description des problèmes à résoudre s'appuie sur le vocabulaire défini pour le domaine choisi. Ainsi, pour chaque problème, il faut choisir, dans l'ensemble des types d'objets du vocabulaire défini pour ce domaine, ceux qui sont nécessaires pour décrire l'énoncé, puis les instancier, nous les appelons objets du problèmes. Il faut ensuite spécifier la question à résoudre pour ce problème en précisant la caractéristique sur laquelle porte l'inconnue. La **Figure 3.4** représente le méta-modèle proposé pour décrire un problème en s'appuyant sur un vocabulaire.

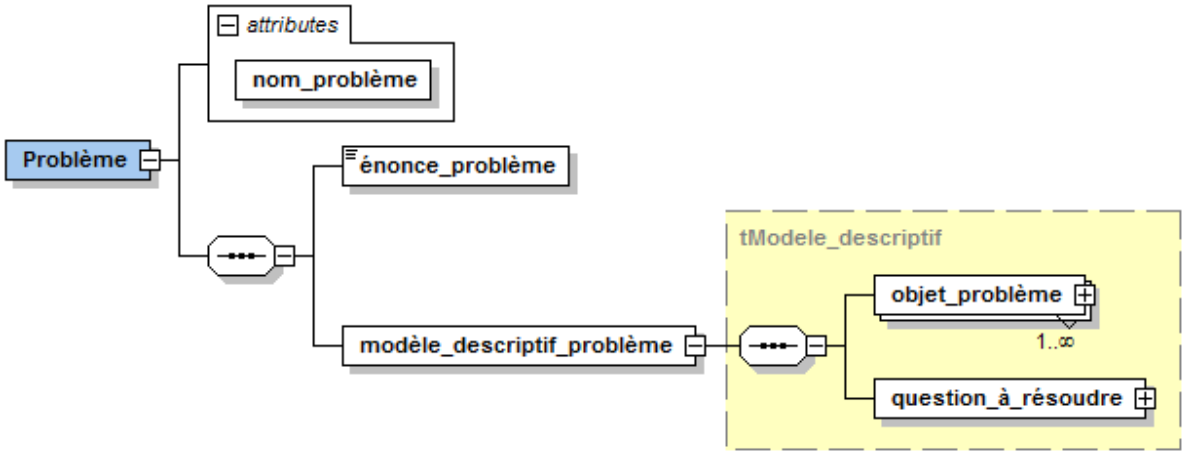

FIGURE 3.2 – Méta-modèle permettant de décrire un problème en s'appuyant sur le vocabulaire

Prenons le cas des problèmes additifs étudiés à l'école élémentaire. Ces problèmes décrivent une situation concrète comme celle représentée dans l'exemple du problème *P2* suivant :
« *Paul a 31 billes rouges. En ajoutant les billes rouges de Pierre, Paul et Pierre en ont 44 à eux deux. Trouver le nombre de billes rouges de Pierre* ».
Pour pouvoir décrire ce genre de problèmes, le vocabulaire à définir doit contenir des objets permettant de décrire les personnes impliquées, les ensembles formés, etc. En utilisant le méta-modèle pour la définition du vocabulaire (cf. **Figure 3.1**), nous pouvons, entre autres, définir les types d'objets suivants pour décrire les problèmes additifs :

— **Personne** : pour désigner les personnes impliquées dans les problèmes. Une personne a un identifiant et possède une liste d'éléments.

— **Liste_d'éléments** : pour désigner les listes d'éléments possédés par chaque personne. Chaque liste est caractérisée par sa taille, et le type des éléments qui la constituent.

— **Liste_de_personnes** : pour désigner la liste des personnes impliquées, avec des listes d'éléments de même type. Cette liste de personnes possède une liste d'éléments caractérisée par sa taille et le type des éléments possédés par les personnes de la liste.

Ainsi, en utilisant le méta-modèle de description des problèmes, le problème *P2* pourrait par exemple être décrit comme suit :

Énoncé du problème P2 :
Paul a 31 billes rouges. En ajoutant les billes rouges de Pierre, Paul et Pierre en ont 44 à eux deux. Trouvez le nombre de billes rouges de Pierre.

Modèle descriptif du problème P2 :

- Paul possède une liste d'éléments E1 de taille 31 et de type "billes rouges".
- Pierre possède une liste d'éléments E2 de taille X et de type "billes rouges".
- Nous disposons d'une liste de personnes P comportant deux personnes : Paul et Pierre. La liste P possède une liste d'éléments E de taille 44 et de type "billes rouges".
- La question à résoudre consiste à calculer la taille X de la liste d'éléments E2 possédé par Pierre.

Dans AMBRE, lorsque le résolveur doit résoudre un problème, il effectue une phase d'opérationnalisation afin d'identifier la classe du problème et d'appliquer la technique de résolution associée. Pour cela, il utilise les connaissances sur la méthode à enseigner.

3.2 Comment définir les connaissances sur la méthode à enseigner ?

Dans notre contexte, une méthode permet d'organiser la résolution de problèmes en se basant sur une classification des problèmes et des outils de résolution. Résoudre un problème en utilisant une méthode consiste alors à identifier la classe dont relève ce problème afin de choisir une technique de résolution adaptée. Dans AMBRE, les connaissances sur la méthode, c'est-à-dire intervenant dans la résolution de problèmes, sont de trois types : des connaissances de classification, des connaissances de reformulation et des connaissances de résolution. Pour un domaine donné, l'association de ces trois types de connaissances au processus de raisonnement conduit à l'obtention d'un résolveur qui fonctionne selon la méthode modélisée.

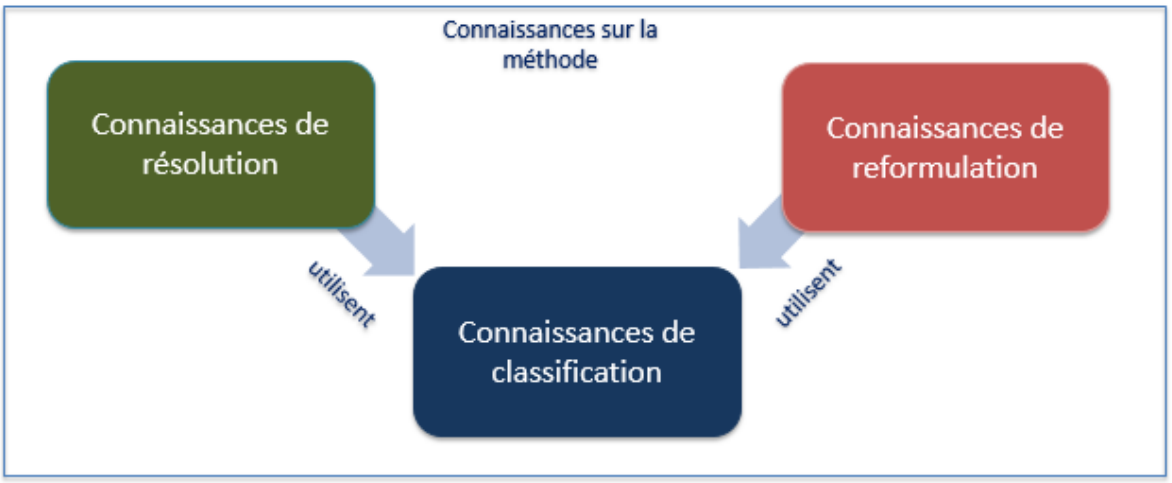

FIGURE 3.3 – Lien entre les connaissances de classification, de reformulation et de résolution

Dans les travaux précédents de l'équipe, une étude exploratoire a été menée sur la formalisation des connaissances sur la méthode (B. K. W. Chang et A. K. W. Chang en collaboration avec N. Guin et M. Lefevre, 2013). Afin de compléter cette étude, nous avons proposé des méta-modèles permettant de décrire les connaissances sur la méthode à enseigner : classification, reformulation, résolution.

3.2.1 Méta-modèle des connaissances de classification

Comme J. Pitrat [44], nous jugeons intéressant de changer la représentation d'un problème afin de se ramener à un problème pour lequel nous disposons d'outils efficaces pour le résoudre. La classification des problèmes du domaine est une connaissance de niveau méta qui permet de classer un problème donné et ainsi choisir une meilleure représentation et une meilleure technique de résolution. Elle permet également de donner une idée sur la difficulté du problème à résoudre. C'est cette classification des problèmes que nous nommons connaissances de classification.

Pour définir ces connaissances, une hiérarchie des classes de problèmes à résoudre doit être définie pour chaque domaine d'application. Cette hiérarchie se présente sous la forme d'un arbre de classification où une classe C2 est une sous-classe d'une classe C1 si tout problème de C2 est aussi un problème de C1. Dans un tel arbre, une classe

racine est définie comme étant la classe la plus générale, les feuilles étant les classes les plus spécifiques. Les classes suffisamment spécifiques qu'on puisse leur associer une technique de résolution sont des classes dites opérationnelles. La **Figure 3.4** présente le méta-modèle proposé pour représenter une classe de problème. Créer une classe revient ainsi à définir :

— Son **nom**.

— Son **statut** : opérationnelle ou non opérationnelle.

— La **valeur de l'attribut qui discrimine cette classe par rapport à son parent**, sauf pour la classe racine qui n'a pas de parent.

— Un **attribut discriminant** permettant de discriminer les sous-classes de cette classe, sauf pour les feuilles qui n'ont pas de sous-classes.

— D'autres **attributs du problème**, s'il y en a besoin pour la résolution des problèmes de cette classe.

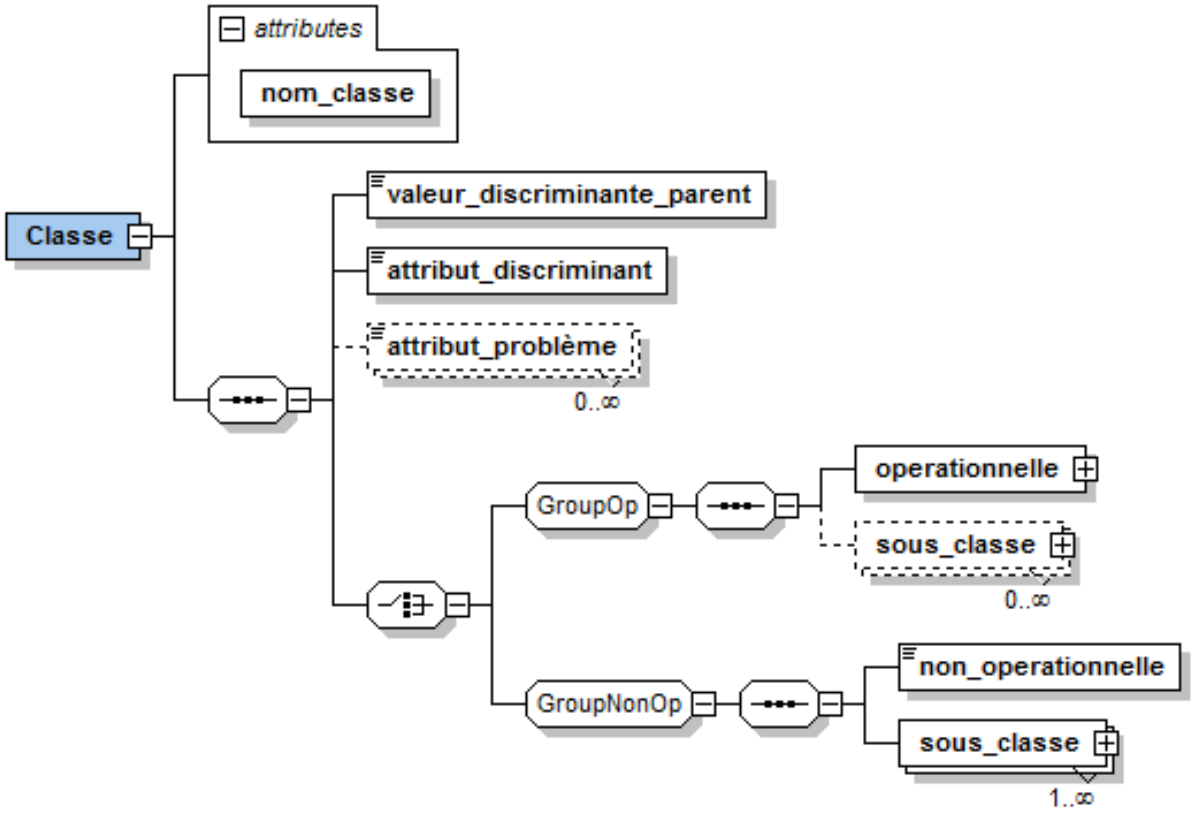

FIGURE 3.4 – méta-modèle permettant de décrire une classe de problèmes

Reprenons notre exemple sur la conjugaison française (cf. Introduction générale) : Problème *P1* : « *Conjuguer le verbe Parler au présent de l'indicatif* ».

Pour résoudre un problème, le système a besoin de déterminer d'abord la classe à laquelle appartient ce problème, afin de choisir par la suite la technique de résolution la plus adaptée. La **Figure 3.5** représente une partie de l'arbre de classification pour la conjugaison française. *C_Conjugaison* représente la classe racine et a pour attribut discriminant *temps_simples*. Les valeurs discriminantes de cet attribut sont : *présent*, et *futur*. Cette classe possède donc deux sous-classes : une classe non opérationnelles *C_Présent* ainsi qu'une classe opérationnelle *C_Futur*. L'attribut discriminant de la classe *C_Présent* est groupe. Les valeurs discriminantes de cet attribut sont : *1er groupe* et *2ème groupe*. La classe *C_Présent* possède donc deux sous-classes : la classe *C_Présent_1er_Gr* qui est non opérationnelle et la classe *C_Présent_2ème_Gr* qui est opérationnelle.

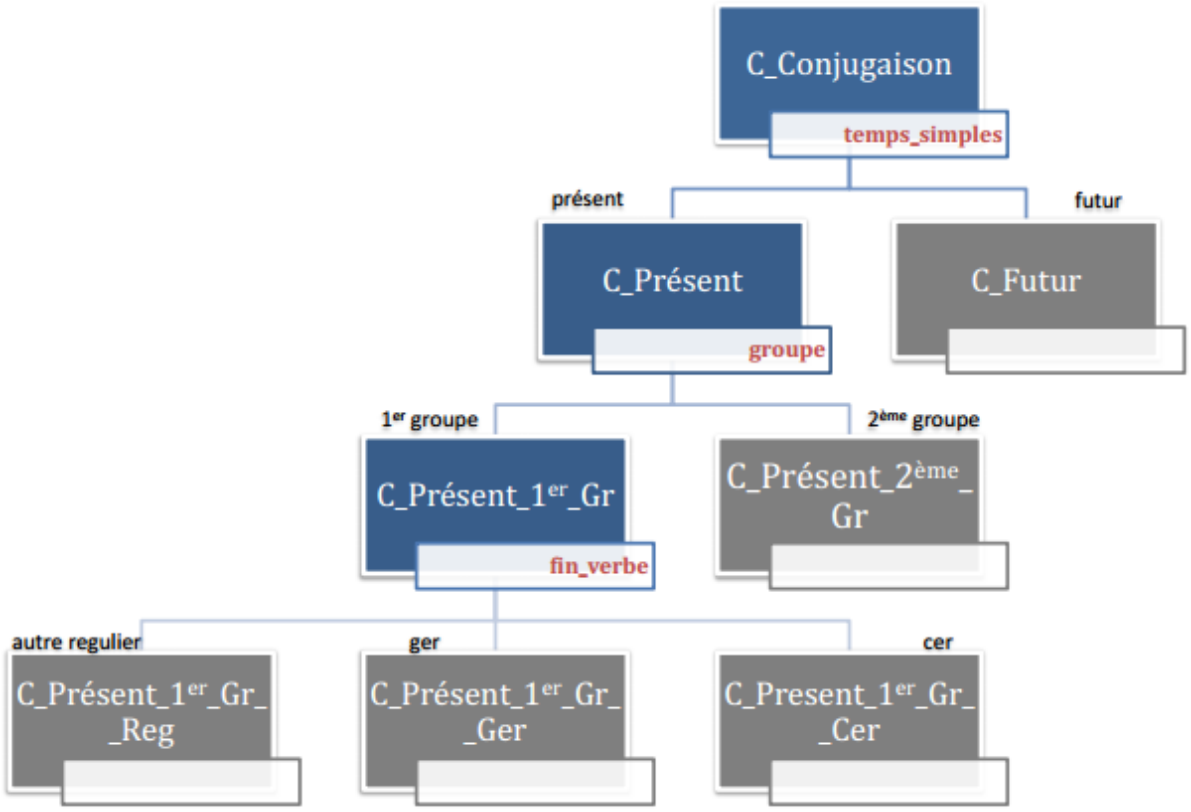

FIGURE 3.5 – Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif

Pour classer un problème dans AMBRE, le résolveur utilise l'arbre de classification comme guide. Partant de la racine de l'arbre, il affine progressivement le classement du problème pour passer à une classe plus spécifique. En d'autres termes, il parcourt l'arbre en profondeur jusqu'aux feuilles ou au moins jusqu'à une classe opérationnelle. Lorsqu'une classe opérationnelle est atteinte, le système essaie quand même de descendre vers une sous-classe pour avoir un classement le plus précis possible. Cette descente se termine soit lorsqu'une feuille est atteinte, soit parce que la valeur de l'attribut discriminant n'est pas celle qui permet de descendre vers les cas particuliers. On obtient alors la classe la plus spécifique selon laquelle les connaissances permettent de modéliser le problème.

Descendre dans une sous-classe revient alors à déterminer la valeur de l'attribut discriminant. Cependant, les attributs intervenants dans la classification sont souvent des attributs relevant de la théorie du domaine. Ces attributs ne sont la plupart du temps pas présents dans les modèles descriptifs des problèmes. Il est donc nécessaire de recourir à des connaissances permettant de déterminer la valeur de ces attributs. Ces connaissances constituent ce que nous nommons : connaissances de reformulation.

3.2.2 Méta-modèle des connaissances de reformulation

Pour chaque attribut (discriminant ou non) de l'arbre de classification, l'auteur doit définir des règles si-alors. Celles-ci permettent, à partir de l'énoncé du problème, de calculer la/les valeur(s) des attributs afin de situer ce problème dans l'arbre de classification. L'ensemble de ces règles constitue les connaissances de reformulation (cf. **Figure 3.6**).

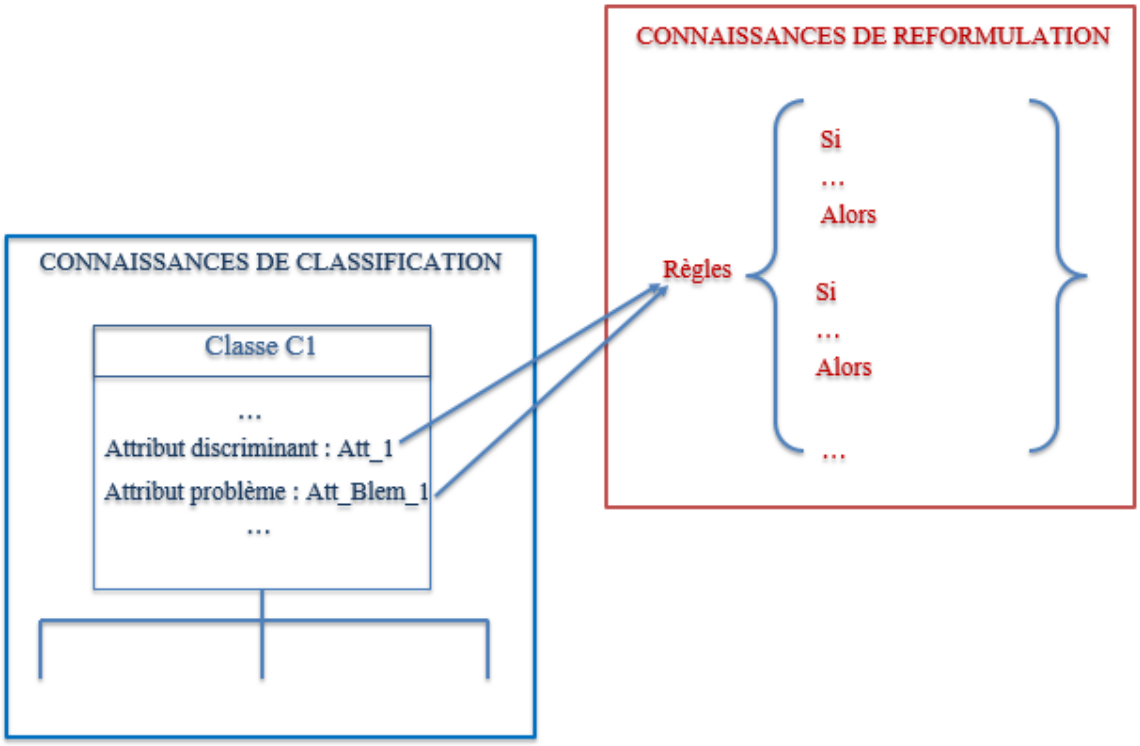

FIGURE 3.6 – Lien entre connaissances de classification et connaissances de reformulation

Nous avons vu dans le chapitre 1 que chaque règle est définie par :

— Son **nom** ;

— Un **ensemble de prémisses** portant sur des éléments de l'énoncé, c'est-à-dire les différents objets du problème ou des contraintes sur ces objets. Les prémisses peuvent également porter sur des attributs du problème qui pourront eux-mêmes nécessiter l'application de règles pour calculer leurs valeurs. On peut également y trouver des appels à des fonctions de calcul. Les connaissances représentées dans les prémisses sont des traitées avec l'opérateur logique ET.

— Un **ensemble de conclusions** permettant de calculer ou de modifier les valeurs des attributs du problème. On peut y retrouver également des appels à des fonctions de calcul.

Pour décrire les prémisses d'une règle, nous avons utilisé le méta-modèle du vocabulaire défini plus haut (cf. section 1.1) pour décrire les éléments de l'énoncé. Un verrou à lever dans le cadre de ce travail était également de pouvoir définir des fonctions de

calcul que l'on retrouve parfois dans les conclusions des règles, mais également lors de la définition des techniques de résolution. De ce fait, nous avons proposé un méta-modèle permettant de décrire des fonctions de calcul. Pour créer une fonction, il faut définir :

— Son **nom**.

— Sa **description en langue naturelle**. Il peut également spécifier dans cette partie les valeurs possibles de la fonction.

— Son **nombre de paramètres** ainsi que leur type.

— Son **type de retour**.

Les **Figures 3.7** et **3.8** représentent respectivement le méta-modèle permettant de décrire une fonction de calcul, et celui permettant de décrire une règle.

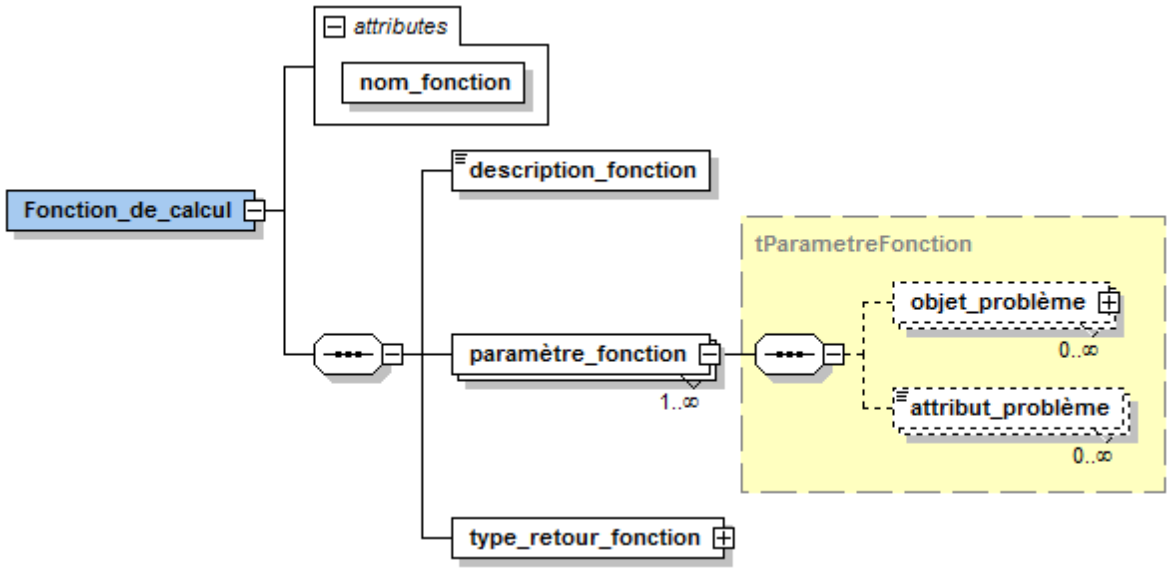

FIGURE 3.7 – Méta-modèle permettant de définir une fonction de calcul.

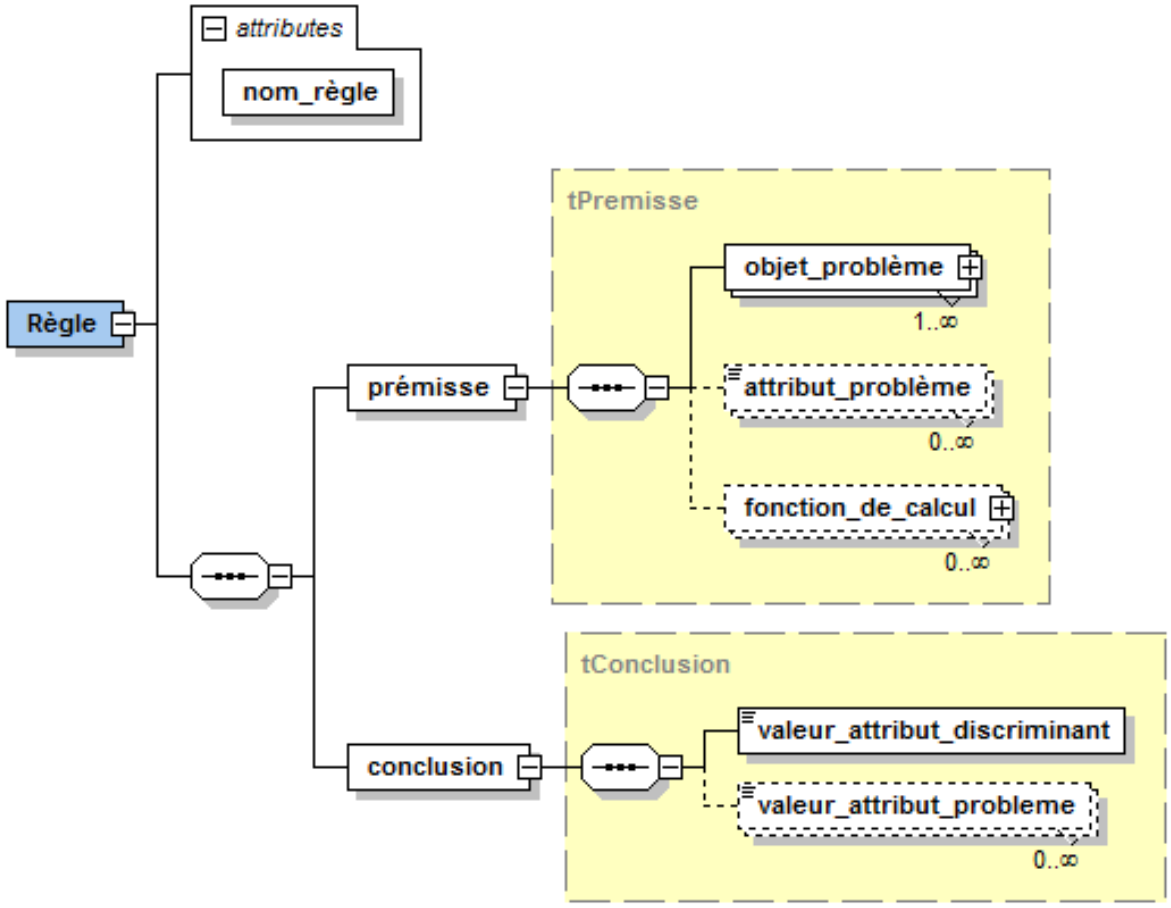

FIGURE 3.8 – Méta-modèle permettant de décrire une règle

Reprenons notre exemple P1, la conjugaison française : « *Conjuguer le verbe Parler au présent de l'indicatif* ».

Afin de déterminer la classe de ce problème, le système utilise l'arbre de classification (cf. **Figure 3.5**) ainsi que les règles de reformulation définies par l'auteur. Celui-ci doit donc définir des règles permettant de conclure sur la valeur des attributs discriminants : *temps_simples*, *groupe*, *fin_verbe*. Soient les règles *regle_temps*, *regle_groupe*, et *regle_fin_verbe* permettant de conclure respectivement sur ces attributs (cf. **Figure 3.9**)
.

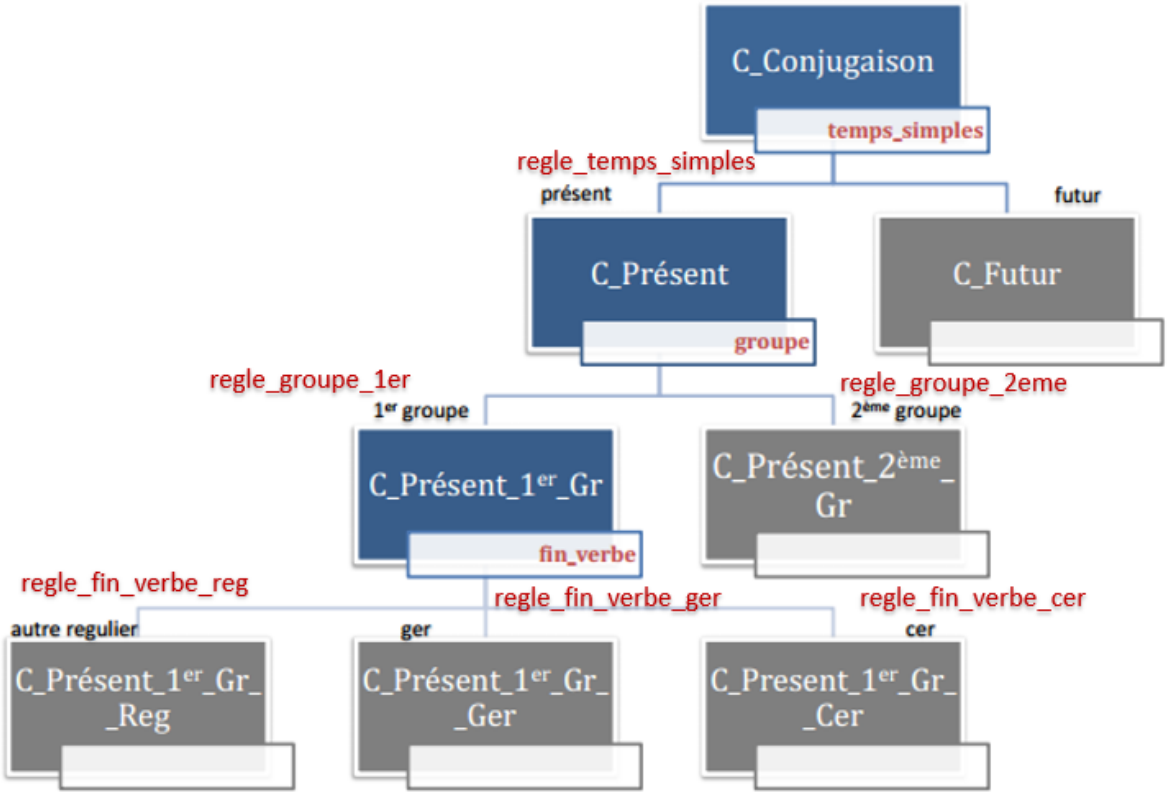

FIGURE 3.9 – Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif, avec les règles de reformulation

/***règle permettant de conclure sur la valeur de l'attribut temps_simples***/ Si dans un problème, il existe : - un verbe V - un temps T Alors la valeur de l'attribut *temps_simples* est égale à *T*.
/***règle permettant de conclure sur la valeur de l'attribut groupe***/ Si dans un problème, il existe : - un verbe *V* - la terminaison du verbe V est égale à *er* - le verbe V est différent du verbe *aller* Alors la valeur de l'attribut *groupe* est égale à *1er groupe*.
/***règle permettant de conclure sur la valeur de l'attribut fin_verbe***/ Si dans un problème, il existe : - un verbe V - V est un verbe du premier groupe - le verbe V est différent du verbe aller Alors la valeur de l'attribut *fin_verbe* est égale à *autre régulier*.

Tableau 3.1 – Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème

En utilisant ces connaissances, le système parcourt respectivement les classes : *C_Conjugaison*, *C_Présent*, *C_Présent_1er_Gr*, pour arriver à la classe opérationnelle *C_present_1er_Gr_Reg*. Puisque cette classe est une feuille, le système en conclut que ce problème appartient à cette classe.

Les connaissances de reformulation permettent de passer du modèle descriptif d'un problème à un modèle opérationnel constitué d'un ensemble d'attributs avec leurs valeurs et plus adapté à l'utilisation d'une technique de résolution. L'ensemble des techniques de résolution constituent les connaissances de résolution.

3.2.3 Méta-modèles des connaissances de résolution

Pour que la classification soit utile dans la résolution des problèmes, une technique de résolution doit être associée à chaque classe opérationnelle de l'arbre de classification. (Cf. **Figure 3.10**).

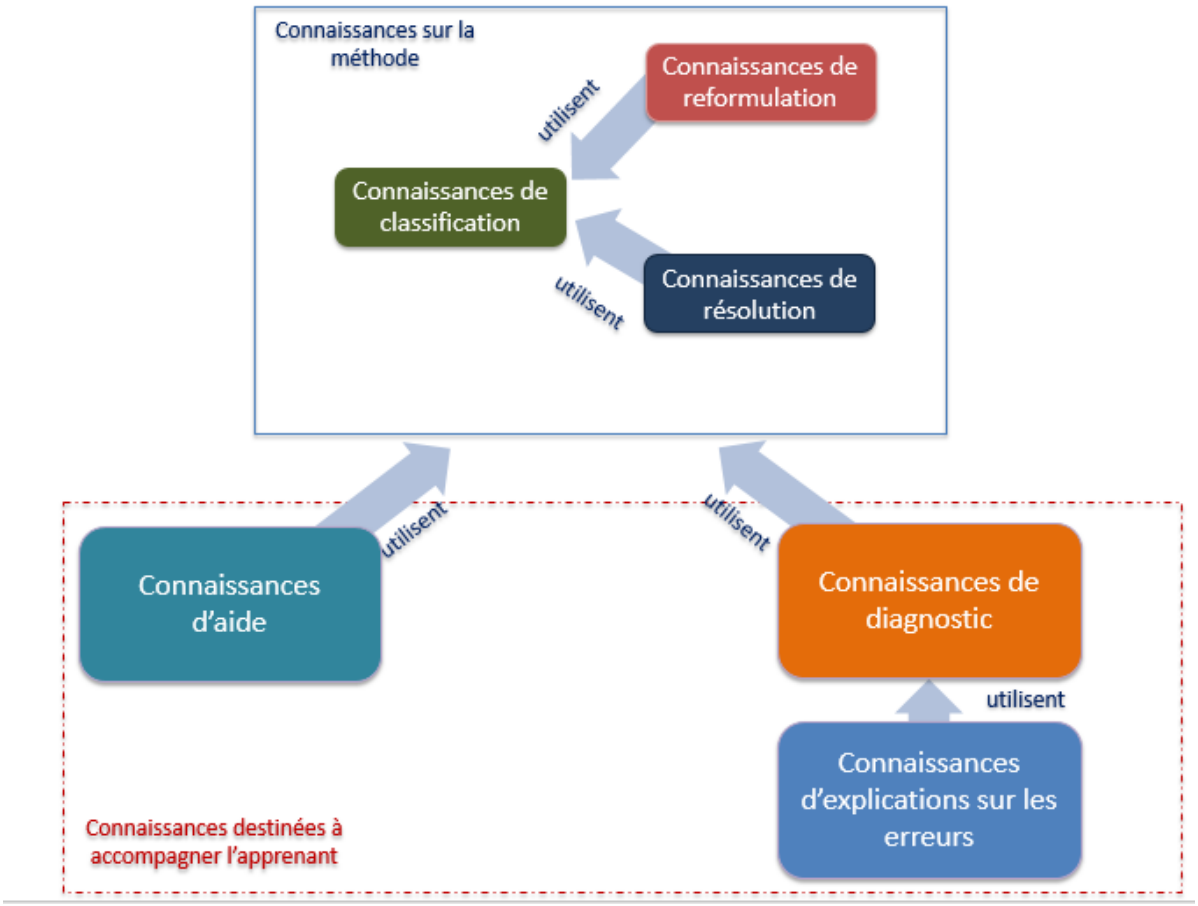

FIGURE 3.10 – Lien entre connaissances de reformulation et connaissances de résolution

La **Figure 3.11** représente le méta-modèle proposé pour décrire une technique. Chaque technique de résolution est définie par :

— Son **nom** ;

— Un **plan de résolution**. Il s'agit d'une description de type algorithmique des actions (étapes de résolution) à réaliser pour résoudre un problème. L'analyse de ce plan doit permettre de trouver la solution du problème. En général, c'est le type de plan qu'on demande aux élèves de fournir dans une copie pour justifier le résultat d'un problème. Dans notre cas, la description des actions portent sur les objets et/ou attributs du problème.

— Une **fonction de calcul** permettant de déterminer les éléments de solution à partir des données issues de l'énoncé.

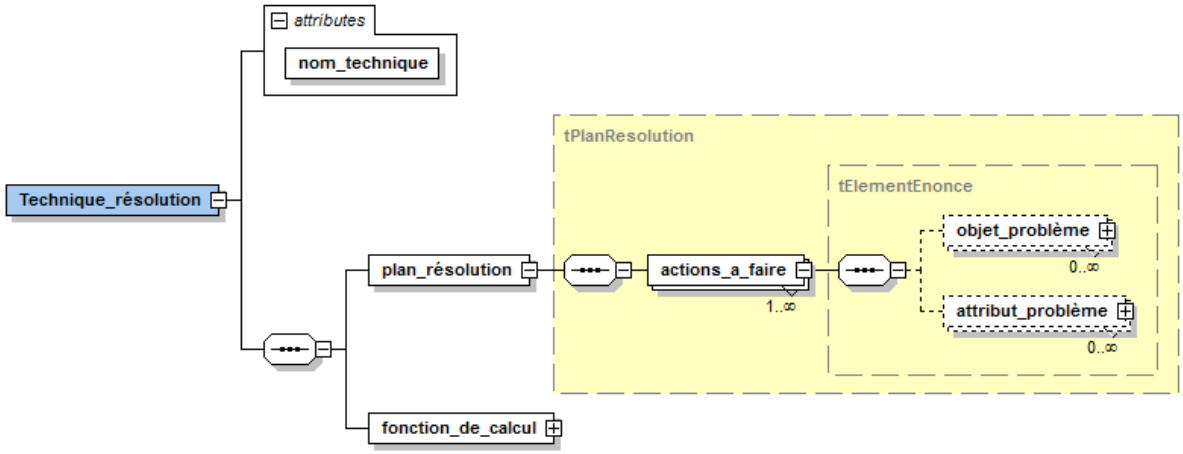

FIGURE 3.11 – Méta-modèle permettant de décrire une technique de résolution

Afin de guider l'auteur dans la formalisation des connaissances de résolution, une question cruciale soulevée est la suivante :
Comment permettre à un auteur de définir des fonctions de calcul intervenant dans l'explicitation des techniques de résolution ?
Notre réponse à cette question a été de proposer le méta-modèle illustré par la **Figure 3.7**.

Les techniques de résolution sont spécifiques au domaine. Pour la conjugaison française, il s'agit d'une part du processus permettant de déterminer le radical du verbe pour l'ensemble des personnes, et d'autre part la liste des terminaisons pour les six personnes.

Reprenons notre exemple sur la conjugaison française : « *Conjuguer le verbe parler au présent de l'indicatif* ».
Après la détermination de la classe du problème C_Présent_1erGroupe_Régulier (section 3.2), Il s'agit alors d'appliquer la technique associée à cette classe pour trouver la solution.

Technique associée à la classe C_Présent_1erGroupe_Régulier :

— Déterminer le radical du verbe : enlever le suffixe ou la terminaison du verbe (er dans notre cas) de l'infinitif du verbe (parler) pour obtenir son radical (parl) ;

— Déterminer la correspondance entre personnes et terminaisons en fonction du temps : au présent de l'indicatif les terminaisons des différentes personnes je, tu, il ou elle, nous, vous, ils ou elles sont respectivement *e, es, e, ons, ez, ent*.

Le **Tableau 3.2.3** et la **Figure 3.12** illustrent ainsi la solution à notre problème après application de cette technique de résolution.

Soit infinitif	= Parler
Soit radical	= Infinitif - "er"
	= "parl"
Soit liste_radical	= [radical,radical,radical,radical,radical,radical]
	= [parl,parl,parl,parl,parl,parl]
Soit liste_terminaisons	= [e,es,e,ons,ez,ent]
Soit liste_personnes	= [je,tu,il/elle,nous, vous,ils/elles]
Soit forme_conjuguée	= fusion des liste : [liste_radical, liste_terminaison]
Soit solution	= Concaténation des liste : [liste_personnes, forme_conjuguée]

Tableau 3.2 – Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème

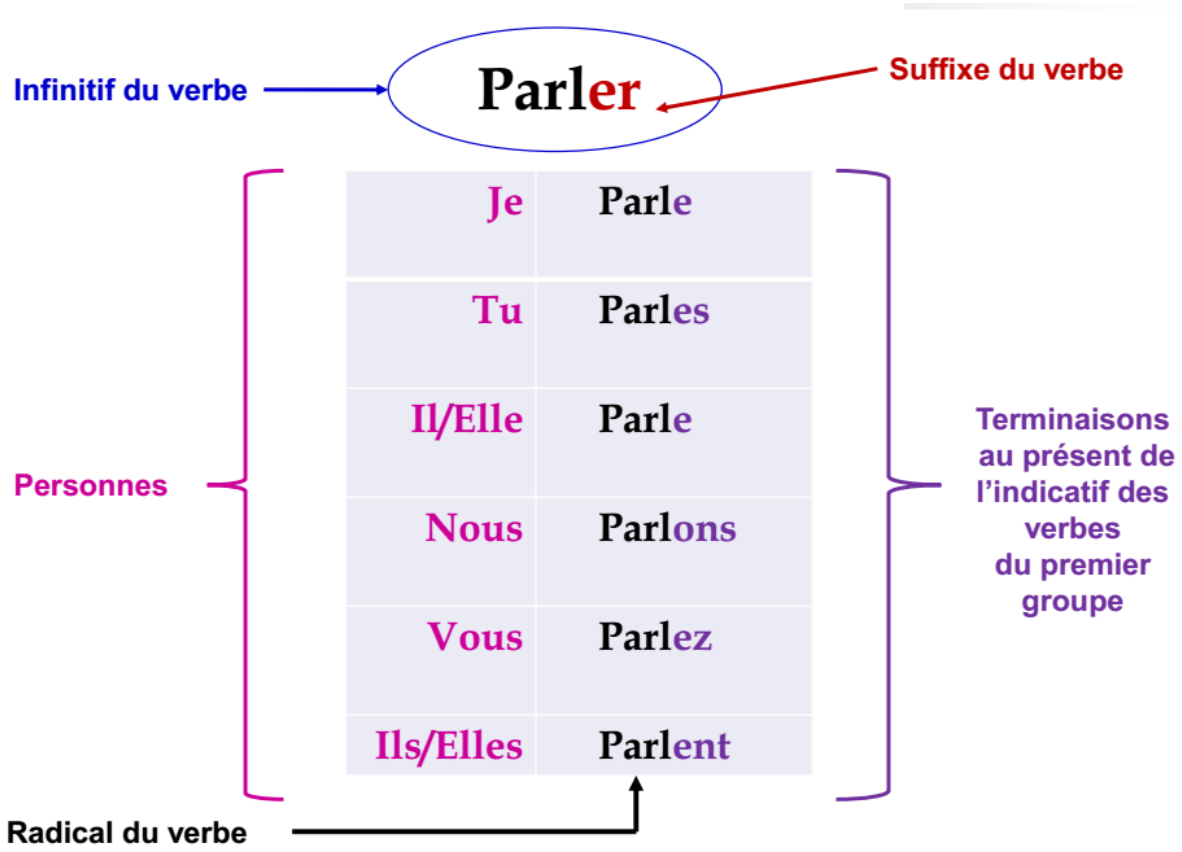

FIGURE 3.12 – Conjugaison du verbe Parler au présent de l'indicatif

Les connaissances sur la méthode que nous venons de présenter (connaissances de

classification, connaissances de reformulation et connaissances de résolution) sont les différentes connaissances qui interviennent lors de la résolution d'un problème par le résolveur de AMBRE. Elles ne permettent cependant pas de fournir de l'aide à l'apprenant, ni de diagnostiquer ses réponses, ni de lui fournir des explications sur ses erreurs. Pour assurer ces fonctionnalités, le système a besoin de connaissances complémentaires que nous présentons dans la section 3.3.

3.3 Comment définir les connaissances permettant d'accompagner l'apprenant ?

Pour que le système puisse assister l'apprenant dans sa résolution de problème et ainsi favoriser l'apprentissage, l'auteur doit définir des connaissances qui permettent au système d'une part de répondre à une demande d'aide de l'apprenant, et d'autre part de diagnostiquer ses réponses pour chaque étape de la résolution et de l'aider à réviser une réponse erronée en lui proposant des explications pertinentes. Dans AMBRE, ces connaissances sont de trois types : des connaissances d'aide, des connaissances de diagnostic et des connaissances d'explications sur les erreurs (cf. **Figure 3.13**).

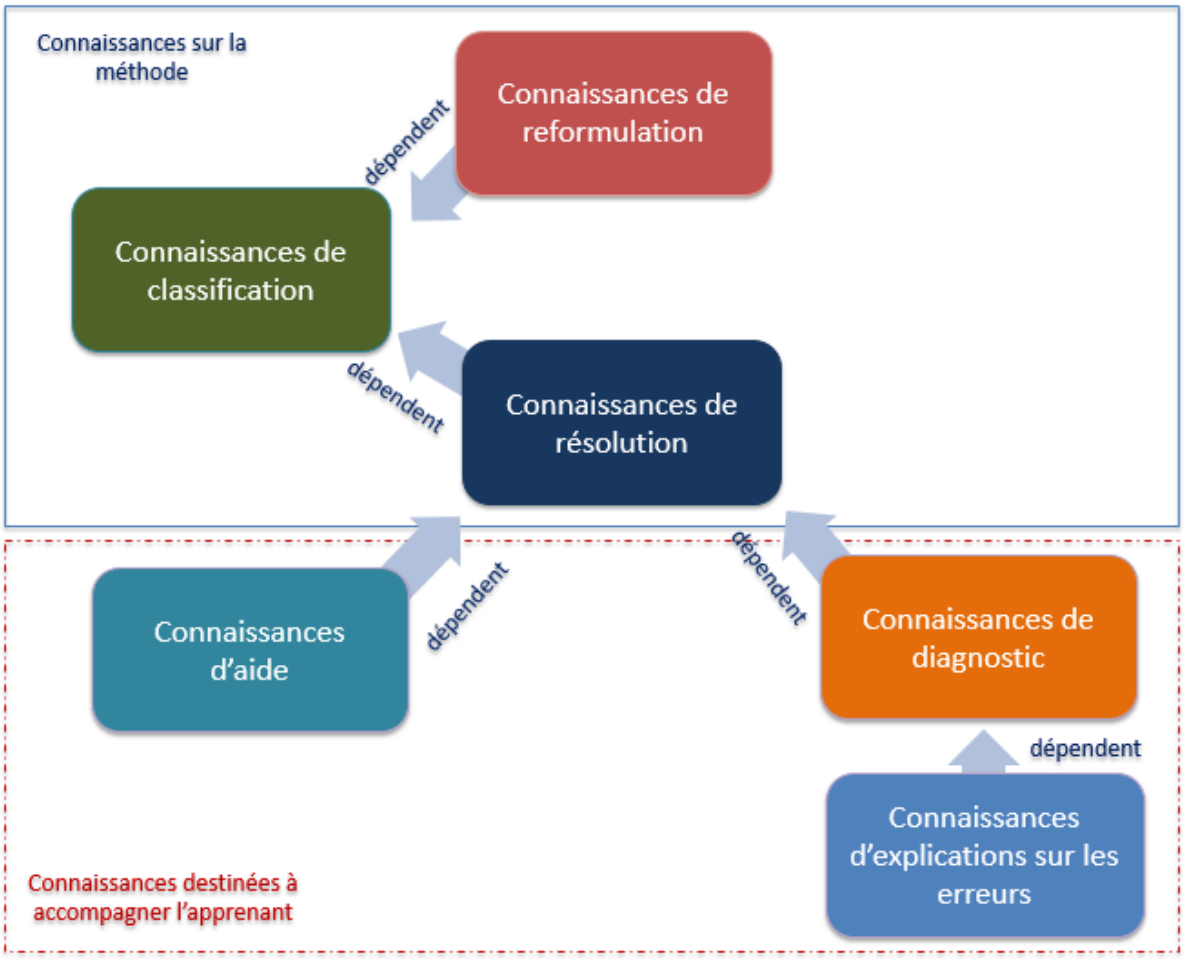

FIGURE 3.13 – Lien entre connaissances sur la méthode et celle destinées à accompagner l'apprenant

Afin de permettre à un auteur de pouvoir expliciter ces connaissances, nous avons proposé des méta-modèles pour chacun de ces types de connaissances.

3.3.1 Méta-modèle des connaissances d'aide

Pour que l'EIAH puisse accompagner l'apprenant de manière efficace durant son apprentissage, il doit pouvoir lui proposer des aides de différentes natures en fonction de l'étape de résolution en cours. Dans AMBRE, l'apprenant peut demander de l'aide à tout moment de la résolution. Lorsque le système reçoit une demande d'aide de la part de l'apprenant, il lui propose deux types d'aide en fonction de l'état d'avancement en

utilisant soit :

— Des **messages** : ces messages d'aide sont souvent des phrases permettant de rappeler à l'apprenant ce qu'il doit faire. Ils sont généralement de deux niveaux : lorsque le premier ne suffit pas à l'apprenant pour comprendre, ce dernier peut passer au second. Par exemple dans l'EIAH AMBRE-add, pour demander de l'aide l'apprenant peut utiliser l'icône en forme de bouée de sauvetage (cf. **Figure 3.14**).

— Les **éléments de l'interface** : dans certains cas des éléments de l'interface sont utilisés afin de rappeler à l'apprenant ce qu'il doit faire. Pour cela, l'EIAH peut renvoyer l'apprenant vers des pages de rappel ou vers des outils favorisant la comparaison entre problèmes-type et problèmes à résoudre à travers la coloration des éléments de l'énoncé et de la reformulation ayant la même fonction dans les deux problèmes (cf. **Figure 3.15**).

FIGURE 3.14 – Exemple de fenêtre d'aide à deux niveaux dans l'étape d'adaptation dans l'EIAH AMBRE-add. le deuxième niveau active le coloriage facilitant la mise en correspondance du problème-type et du problème à résoudre

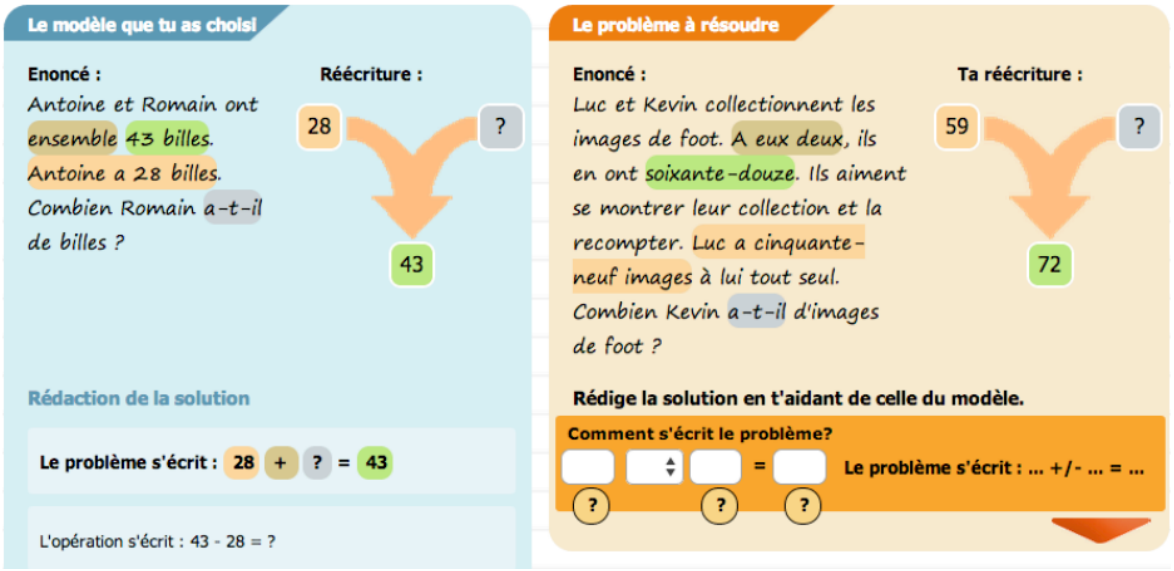

FIGURE 3.15 – Aide à l'analogie à l'étape d'adaptation dnas AMBRE-add

La **Figure 3.16** détaille le méta-modèle que nous avons proposé pour décrire ces connaissances d'aide.

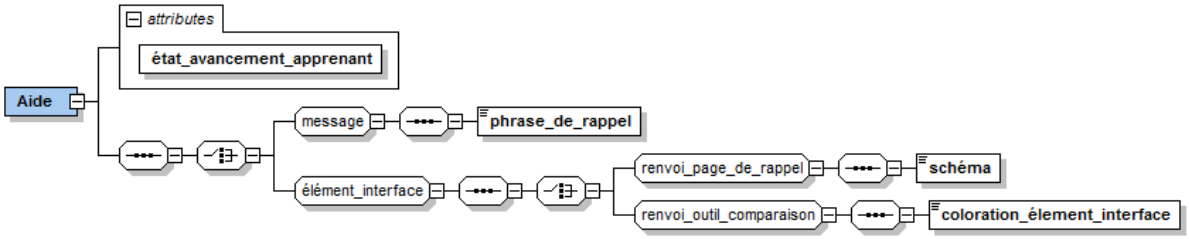

FIGURE 3.16 – Méta-modèle des connaissances d'aide

3.3.2 Méta-modèle des connaissances de diagnostic

Les connaissances de diagnostic permettent la comparaison entre les réponses de l'apprenant et celles fournies par le système afin de pouvoir déterminer si une réponse est correcte ou non. Dans AMBRE-add par exemple, le diagnostic peut être effectué automatiquement par le système à chaque étape ou à la fin d'une sous-étape du cycle

AMBRE, suivant un paramétrage défini par l'enseignant. Il peut également être demandé par l'apprenant à tout moment en cliquant sur l'icône représentant un feu tricolore (cf. **Figure 3.15**) en haut à droite).

Pour définir les connaissances de diagnostic, des points de diagnostic doivent être définis à chaque étape de résolution (cf. **Figure 3.17**). Définir un point de diagnostic revient à préciser l'attribut du problème concerné et à définir un ensemble de fonctions permettant de conclure sur le résultat de la comparaison entre la réponse de l'apprenant et celle du résolveur. Trois cas sont possibles : la réponse de l'apprenant peut être conforme à celle du résolveur, elle peut correspondre à une réponse fausse prévue/non prévue ou à une réponse correcte équivalente à la réponse du résolveur. Dans le cas où la réponse de l'apprenant est fausse, un message d'erreur à afficher à lui doit être prévu.

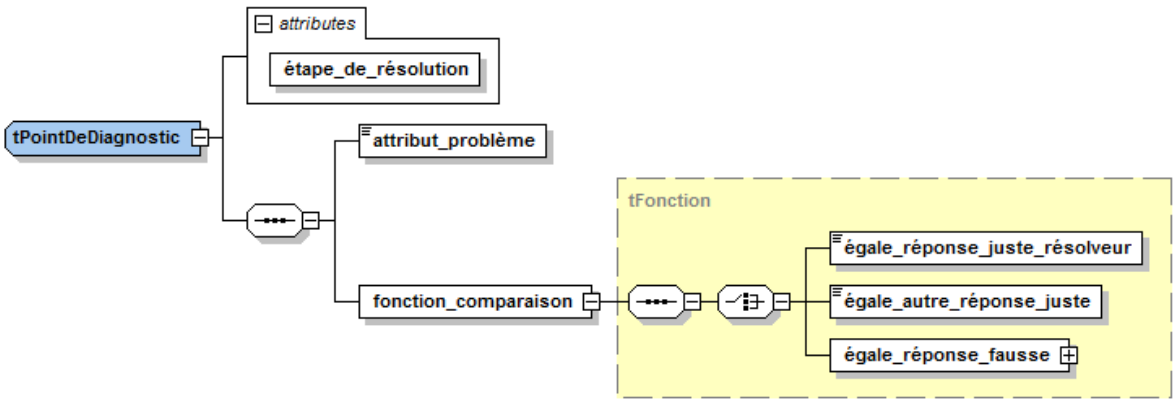

FIGURE 3.17 – Aide à l'analogie à l'étape d'adaptation dans AMBRE-add

La **Figure 3.18** est tirée de l'EIAH AMBRE-add. Dans cet exemple, le diagnostic est réalisé automatiquement par le système et le résultat matérialisé à travers la coloration de certains éléments de l'interface (couleur verte lorsque la réponse est correcte, et rouge lorsqu'elle est fausse).

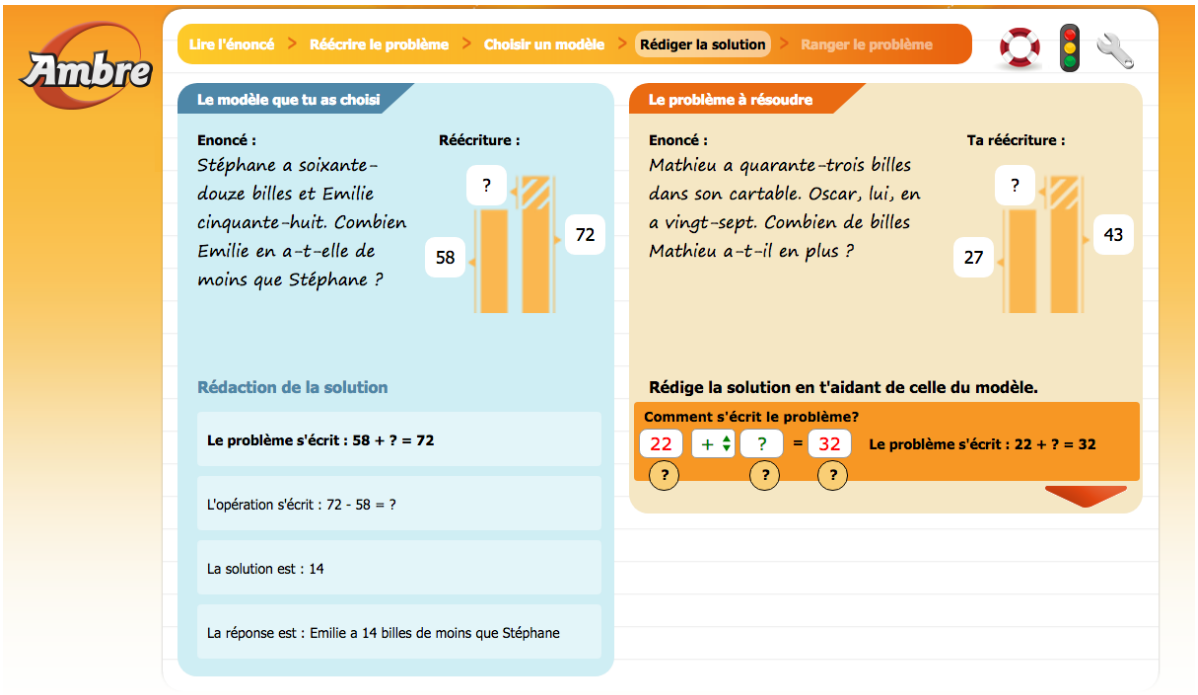

FIGURE 3.18 – Exemple de diagnostic réalisé par le système

3.3.3 Connaissances d'explications sur les erreurs

Les connaissances d'explications font suite au diagnostic. Elles permettent de donner à l'apprenant des explications pertinentes en cas d'erreur. Il s'agit alors de définir un ensemble de réponses fausses attendues et d'y associer des messages explicatifs. La **Figure 3.19** représente le méta-modèle proposé pour décrire ces messages d'explications.

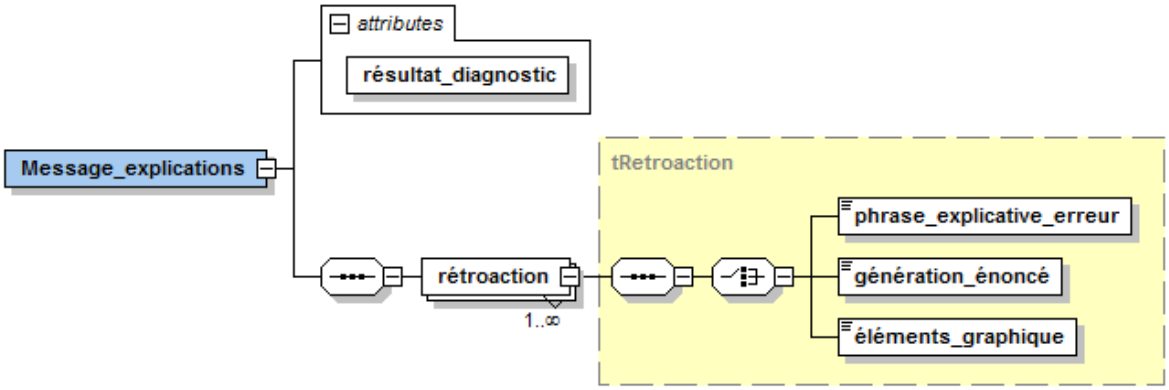

FIGURE 3.19 – Méta-modèle permettant de décrire un message d'explications

Ces messages représentent les différentes rétroactions à envoyer à l'apprenant. Elles peuvent se présenter sous plusieurs formes :

— Une **phrase explicative de l'erreur** : par exemple lors de la rédaction de la phrase de réponse dans AMBRE-add, il est fréquent que les apprenants écrivent une phrase qui est dans l'énoncé. Dans ce cas, le système explique « Ce que tu as écrit est vrai, mais ce n'est pas la réponse à la question ».

— Un **énoncé généré par le système** correspondant à la réponse de l'apprenant. Cette génération d'énoncé peut souvent lui permettre de comprendre en quoi sa réponse est erronée. Cette fonctionnalité de génération d'énoncé est par exemple utilisée dans AMBRE-add pour expliquer à l'apprenant pourquoi sa reformulation à l'aide d'un schéma n'est pas correcte : le système génère un énoncé de problème correspondant au schéma donné par l'apprenant (cf. Figure 14).

— Des **éléments graphiques** : le recours aux éléments graphiques peut se faire lorsque le système ne sait pas interpréter la réponse de l'apprenant ou lorsque ce format est plus facile à comprendre que des explications textuelles complexes. Par exemple, le système colorie en vert des éléments de la réponse qui sont corrects et en rouge ceux qui sont erronés (cf. **Figure 3.20**). Lorsque la réponse de l'apprenant est incorrecte, comme sur la Figure 12 par exemple, l'apprenant peut cliquer sur le point d'interrogation à côté de chaque champ encadré en rouge pour afficher le message d'erreur.

FIGURE 3.20 – Exemple de message d'explications généré à partir de la réponse de l'apprenant

3.4 Conclusion

Dans ce chapitre, nous avons présenté les différentes connaissances identifiées comme étant nécessaire à expliciter lors de la conception d'un EIAH AMBRE. Ces connaissances peuvent être classées en deux grandes catégories : les connaissances sur la méthode à enseigner permettant de résoudre des problèmes, et celles destinées à accompagner l'apprenant durant son apprentissage en lui fournissant aide, diagnostic de ses réponses et explications pertinentes sur ses erreurs. Pour formaliser ces différentes connaissances, nous avons proposé des méta-modèles décrits par des schémas XML. Ces méta-modèles permettent de décrire la forme des connaissances à acquérir, cherchent être indépendants du domaine.

L'objectif de ce travail était surtout d'identifier les types de connaissances à acquérir et de les formaliser. Même si les méta-modèles proposés ont été proposés en s'appuyant sur les connaissances des EIAH AMBRE, leur utilisation reste ouverte à d'autres types d'applications. Par exemple, le méta-modèle portant sur la description des problèmes

peut être utilisé dans des domaines nécessitant de modéliser des problèmes.

Dans [84] par exemple les auteurs présentent un outil auteur web permettant d'assister l'enseignant dans la création d'exercices portant sur les équations algébriques. Cet outil repose sur un résolveur de problèmes qui utilise les informations du problème et accompagne l'apprenant durant sa résolution en lui fournissant un feedback. Les problèmes sont décrits en termes de variables, d'unités de mesures, de formules. Les méta-modèles du vocabulaire et des problèmes que nous avons proposés peuvent, par exemple, être utilisés dans le cadre de cet outil auteur web, afin de mieux assister l'auteur dans la définition des connaissances. Le méta-modèle permettant de décrire les fonctions peut être utilisé pour cet outil afin de faciliter la description des formules. De même, les méta-modèles permettant d'accompagner l'apprenant peuvent être utiles ici pour donner des explications pertinentes aux apprenants sur leurs erreurs.

Dans Andes [76, 83], un EIAH destiné à l'enseignement de la Physique, l'auteur doit définir formellement le problème, de sorte que le système puisse le résoudre. Utiliser un méta-modèle permettant de décrire les problèmes peut être très utile pour faciliter l'élicitation. Les méta-modèles des connaissances sur la méthode peuvent quant à eux être utilisés dans le domaine de l'aide à prise de décision, et plus particulièrement pour la conception d'outils de prévision. Par exemple, le méta-modèle des connaissances de classification peut être utilisé pour organiser les différentes informations, le méta-modèle des connaissances pour décrire les règles, et les techniques pour représenter les différentes prévisions en fonction des cas.

Nous nous sommes appuyées sur ces méta-modèles pour mettre en place un processus d'acquisition des connaissances décrit dans le chapitre 4.

Chapitre 4

Un processus interactif d'acquisition des connaissances pour des EIAH destinés à enseigner des méthodes

Ce chapitre présente le processus interactif que nous avons proposé pour acquérir les connaissances nécessaires à conception d'un EIAH AMBRE. Il se consacre pour l'essentiel à l'élicitation des connaissances portant sur la méthode à enseigner, ainsi que l'assistance fournie à l'auteur durant le processus. Cependant, même si l'élicitation des connaissances destinées à guider l'apprenant n'est pas présentée ici, nous montrons comment elle est prise en compte dans le processus. Ce chapitre se termine par le feedback proposé à l'auteur à la fin de l'élicitation.

Sommaire

4.1 Étapes du processus d'acquisition des connaissances 97
4.1.1 Définition du vocabulaire pour le domaine 98
4.1.2 Description des problèmes à résoudre 99
4.1.3 Élicitation des connaissances sur la méthode 100
4.1.4 Conception de l'interface de l'EIAH 104
4.1.5 Élicitation des connaissances destinées à accompagner l'apprenant 104
4.2 Feedback après l'élicitation des connaissances 104
4.3 Synthèse 106

L'acquisition des connaissances pour la conception d'EIAH est l'un des principaux obstacles à leur développement. Cette difficulté est liée au fait qu'elle repose en général sur des experts humains mais aussi sur un processus d'explicitation de leurs connaissances. Outre le fait qu'il est difficile d'amener l'expert à expliciter ses connaissances, cette pratique s'avère également longue et ardue. En effet, elle nécessite de recommencer l'explicitation des connaissances pour chaque nouveau domaine. La mise en place d'un processus interactif pourrait favoriser une acquisition plus aisée de ces connaissances. C'est dans ce sens que nous proposons un processus pour permettre à un auteur d'éliciter sans programmer les connaissances nécessaires à la conception d'un EIAH AMBRE.

4.1 Étapes du processus d'acquisition des connaissances

La **Figure 4.1** résume l'approche générale que nous avons adoptée pour la mise en place d'un processus d'élicitation des connaissances nécessaires à la conception d'un EIAH AMBRE. Ce processus permet à un auteur, expert des connaissances qu'il souhaite enseigner, de pouvoir expliciter les connaissances nécessaires à l'EIAH sans aucune programmation. Ce processus s'appuie sur les méta-modèles indépendants du domaine et proposés dans le Chapitre 3 : un méta-modèle pour décrire le vocabulaire, un méta-modèle pour décrire les problèmes, des méta-modèles pour les connaissances portant sur la méthode à enseigner, et des méta-modèles pour les connaissances destinées à accompagner l'apprenant. Pour rappel, ces méta-modèles permettent de décrire les différents types des connaissances que l'auteur doit définir pour construire des modèles de connaissances spécifiques à un domaine. Ces méta-modèles contraignent ainsi les différentes étapes du processus d'acquisition des connaissances.

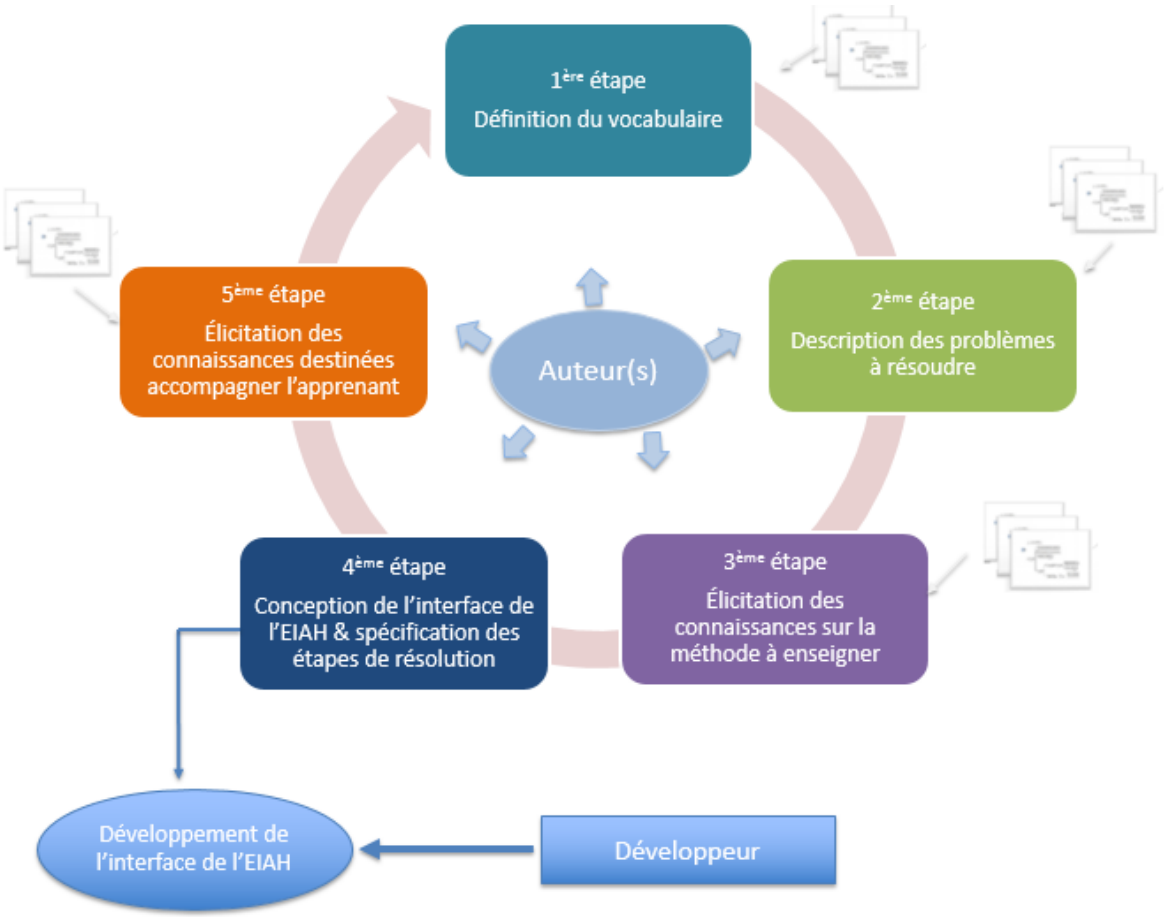

FIGURE 4.1 – Approche générale du processus d'élicitation

4.1.1 Définition du vocabulaire pour le domaine

L'objectif ici est de permettre à l'auteur de pouvoir définir les connaissances nécessaires à la conception d'un EIAH destiné à enseigner des méthodes dans un domaine qui l'intéresse. Les EIAH de ce type reposent sur une classification des problèmes à résoudre et des techniques de résolution. Pour ce faire, l'auteur doit d'abord commencer par bien cibler le domaine d'application en respectant ce critère d'existence d'une classification. Une fois le domaine d'application choisi, et avant d'entamer l'élicitation des connaissances, il est important de bien réfléchir au type de problèmes à faire résoudre à l'apprenant (et donc au système). Ces éléments sont en effet nécessaires pour que l'auteur puisse définir le vocabulaire du domaine choisi, qui joue un rôle central dans le processus d'élicitation des connaissances que nous proposons.

Pour définir le vocabulaire, l'auteur dispose d'une interface lui permettant de créer, modifier ou supprimer des types d'objets. Pour chaque type d'objet créé, il doit pouvoir ajouter des caractéristiques (nom et type des caractéristiques), les modifier et les supprimer au besoin.

Pour faciliter la définition du vocabulaire, le système doit proposer une grande flexibilité à l'auteur (sans trop le contraindre) lors de l'élicitation des connaissances. Nous entendons par flexibilité la capacité du système à offrir des modes d'interaction multiples pour répondre aux besoins, préférences et expériences de l'utilisateur. Ainsi, le système doit, dans la mesure du possible, laisser le choix à l'auteur quant à l'ordre de définition des connaissances. Par exemple, après avoir réfléchi sur le champ d'application de la méthode à enseigner et donc de l'ensemble des problèmes qui seront proposé à l'apprenant, l'auteur doit pouvoir choisir de définir l'ensemble des objets dont il aura besoin pour décrire tous les problèmes. Il doit également pouvoir choisir de définir les types d'objets au fur et à mesure qu'il en a besoin pour décrire chaque problème.

Une fois le vocabulaire défini, l'auteur doit l'utiliser d'une part pour décrire les différents problèmes à résoudre, et d'autre part pour éliciter les connaissances sur la méthode.

4.1.2 Description des problèmes à résoudre

Dans cette étape, l'auteur utilise le vocabulaire précédemment défini pour décrire les problèmes à résoudre. Pour ce faire, le système doit lui proposer une interface basée sur le méta-modèle permettant de décrire des problèmes (cf. Chapitre 3). Ainsi, pour chaque problème, il doit pouvoir :

— Définir l'**énoncé du problème** qui va être présenté à l'apprenant lors de la résolution d'un problème. Pour cela, il doit disposer d'un champ texte à remplir. L'énoncé décrit dans ce champ sera présenté tel quel à l'apprenant.
— Définir les **objets du problème**. L'auteur doit pouvoir choisir, par les types d'objets précédemment définis dans son vocabulaire, ceux qui sont nécessaires pour décrire son problème, puis les instancier en mettant des valeurs aux identifiants, ainsi qu'aux caractéristiques, tout en respectant le type de ces dernières.
— Spécifier la **question à résoudre** pour ce problème. Le système doit également

permettre à l'auteur de préciser la caractéristique qui constitue l'inconnue à calculer pour le problème concerné.

Pour faciliter la description des problèmes, proposer des fonctionnalités de guidage peut être utile à l'auteur. Nous entendons par guidage l'ensemble des moyens proposés à l'auteur afin de l'orienter et de le conseiller durant tout le processus d'élicitation. Par exemple, proposer des fonctionnalités d'auto-complétions peut être très utile parce qu'elles permettent à l'auteur, au fur et à mesure de la définition des connaissances, de savoir ce qui a déjà été défini et de le réutiliser au besoin. Par exemple, connaître les objets déjà définis pour pouvoir les réutiliser permet une élicitation plus facile des connaissances, avec moins de risque d'erreurs.

Le fait que les modèles de connaissances soient fondés sur des méta-modèles permet aussi de pouvoir proposer des fonctionnalités de contrôle de ces connaissances. Celles-ci vont permettre d'une part de vérifier que toutes les connaissances nécessaires ont été explicitées, et d'autre part que ces connaissances sont bien conformes aux différents méta-modèles. Par exemple, le système doit permettre à l'auteur vérifier qu'il a défini toutes les connaissances nécessaires pour décrire un problème. Il s'agit de mettre en place une fonctionnalité permettant de vérifier d'une part que pour chaque objet, les types des caractéristiques ont bien été respectés. Par exemple lorsqu'une caractéristique est de type « nombre entier », le système va contraindre le champ de sorte que l'auteur ne puisse saisir que des nombres. D'autre part, il s'agira de vérifier aussi que pour chaque problème, l'auteur a bien indiqué la caractéristique sur laquelle porte l'inconnue à calculer.

4.1.3 Élicitation des connaissances sur la méthode

Pour les connaissances sur la méthode, l'auteur doit définir trois types de connaissances en s'appuyant sur les méta-modèles correspondants (cf. chapitre 3) : les connaissances de classification, les connaissances de reformulation et les connaissances de résolution.

4.1.3.1 Élicitation des connaissances de classification

Pour définir les connaissances de classification, il revient à l'auteur de construire un arbre de classification des problèmes à résoudre pour le domaine choisi. Pour cela, il doit disposer d'une interface lui permettant de construire cet arbre au fur et à mesure. Pour la construction de cet arbre, l'auteur doit avoir deux possibilités : définir toutes les classes de problèmes d'abord, puis les organiser ensuite en une hiérarchie ou alors définir les classes une par une en les hiérarchisant au fur et à mesure. Pour chaque classe créée, il doit pouvoir définir un nom, un attribut discriminant permettant de discriminer ses sous-classes. Des attributs du problème peuvent également être définis s'ils ont un sens pour les problèmes de cette classe. L'auteur doit également pouvoir indiquer le statut de chaque classe : les classes suffisamment spécifiques pour qu'une technique de résolution puisse être appliquée doivent être opérationnelles et les autres non opérationnelles.

L'auteur doit, à tout moment de la construction de son arbre, pouvoir vérifier si les connaissances qu'ils a définies jusque-là sont conformes au méta-modèle des connaissances de classification (cf. Chapitre 3). Pour cela, le système doit pouvoir vérifier, par exemple, que l'auteur a défini des noms pour toutes les classes, qu'il a défini des attributs discriminants pour toutes les classes non opérationnelles, que toutes les classes non opérationnelles ont au moins une sous-classe. En effet, afin de classer un problème lors de la résolution, le résolveur essaie d'aller le plus loin possible dans la classification : au moins jusqu'à une classe opérationnelle, du fait que les classes non opérationnelles ne possèdent pas de technique de résolution. C'est pourquoi d'ailleurs les feuilles de l'arbre doivent être des classes opérationnelles pour que la classification soit utile pour résoudre des problèmes.

Le système doit également assister l'auteur durant la construction de l'arbre en lui proposant des fonctionnalités de guidage. Par exemple : colorer toutes les classes que l'auteur aura définies comme étant opérationnelles. Cela permet de lui rappeler qu'il doit associer des techniques de résolution à ces classes, mais également d'avoir un aperçu des classes qui sont opérationnelles et de celles qui ne le sont pas. Lors de la définition des attributs discriminants, le système doit proposer des fonctionnalités d'auto-complétion pour permettre à l'auteur d'avoir un aperçu des attributs qu'il a déjà définis et d'en utiliser un s'il le désire. Lorsque l'auteur définit le même attribut pour deux sous-classes, le système doit lui proposer soit de définir cet attribut comme un attribut du problème au niveau de la classe parent la plus proche, soit de renommer l'un des deux attributs.

En choisissant de définir l'attribut comme un attribut du problème dans la classe parent, cela lui permettra par exemple de définir les règles associées une seule fois et de pouvoir les utiliser pour les deux classes.

Une fois que l'auteur a défini une partie de l'arbre de classification, il peut entamer la définition des règles. Celles-ci vont permettre de calculer les valeurs des attributs discriminants, et ainsi de positionner les problèmes dans la classification. L'ensemble des règles constitue les connaissances de reformulation.

4.1.3.2 Élicitation des connaissances de reformulation

Pour chaque attribut (discriminant ou non) défini dans l'arbre de classification, l'auteur doit associer des règles permettant de conclure sur sa/ses valeur(s). L'auteur doit donc disposer d'une interface permettant de les définir. Pour chaque règle, il doit pouvoir définir un nom, un ensemble de prémisses portant sur des éléments de l'énoncé du problème et un ensemble de conclusions permettant de conclure sur la/les valeur(s) des attributs concernés.

Comme pour les connaissances de classification, le système doit offrir à l'auteur de la flexibilité lors de la définition des connaissances de reformulation. Par exemple, l'auteur doit pouvoir définir les règles au fur et à mesure de la construction de son arbre de classification. Ainsi pour chaque classe définie, une fois les attributs définis, l'auteur doit pouvoir spécifier le(s) règles(s) qui permettront de déterminer sa/ses valeur(s). L'auteur doit également avoir le choix de définir son arbre entièrement, et ensuite procéder à la définition des règles. Le système doit permettre également de modifier une règle, la supprimer ou la dupliquer à partir d'une autre en complétant/modifiant certaines valeurs.

Des fonctionnalités de contrôle des connaissances de reformulation doivent également être proposées. Celles-ci permettront à l'auteur au fur et à mesure de vérifier si les connaissances qu'il a explicitées sont conformes au méta-modèle des connaissances de reformulation (cf. chapitre 3). Il s'agit notamment de vérifier qu'il a associé au moins une règle à tous les attributs (discriminants ou non) de l'arbre de classification, de vérifier que chaque règle comporte une partie conclusion donnant des informations permettant de conclure sur la/les valeurs(s) des attributs correspondants. Ces connaissances de re-

formulation sont très importantes parce qu'elles vont permettre au système de passer d'un modèle initial du problème, à un nouveau modèle plus adapté à l'application d'une technique de résolution. L'ensemble des techniques de résolution constituent les connaissances de résolution.

4.1.3.3 Élicitation des connaissances de résolution

Pour définir les connaissances de résolution, l'auteur doit associer à chaque classe opérationnelle de l'arbre de classification, une technique de résolution. Pour ce faire, il doit disposer d'une interface lui permettant pour chaque classe, de définir sa technique en décrivant le plan de résolution des problèmes de cette classe. Il s'agir pour l'auteur de décrire l'ensemble des actions que le système doit réaliser pour résoudre un problème donné. Dans certains domaines, il peut également être amené à définir une fonction de calcul permettant de déterminer les éléments de solution à partir des données fournies dans l'énoncé des problèmes. Pour l'assister dans cette élicitation, le système doit proposer des fonctionnalités d'auto-complétion sur les objets des problèmes, ainsi qu'un contrôle sur les connaissances élicitées.

De manière similaire aux règles, l'auteur doit pouvoir choisir de définir les techniques de résolution au fur et à mesure de la définition des classes de l'arbre. Dans ce cas, il peut définir une classe opérationnelle et sa technique de résolution de manière simultanée. Il doit également pouvoir définir l'ensemble des classes, puis l'ensemble des techniques et enfin associer les techniques aux classes correspondantes. Le système doit également permettre de modifier une technique, la supprimer ou la dupliquer en complétant/modifiant certaines valeurs.

Une fois ces trois types de connaissances définies, le système est apte à résoudre les problèmes du domaine. Toutefois, afin de permettre au système de pouvoir fournir de l'aide à l'apprenant, de diagnostiquer ses réponses et de lui fournir des explications pertinentes sur ses erreurs, l'auteur doit définir des connaissances complémentaires : les connaissances destinées à guider l'apprenant (cf. section 4.5).

4.1.4 Conception de l'interface de l'EIAH

Avant d'entamer l'élicitation des connaissances destinées à guider l'apprenant (étape 5), l'auteur doit concevoir l'interface de l'EIAH, et particulièrement les différentes tâches à effectuer par l'apprenant pour résoudre les problèmes. En effet, les connaissances destinées à guider l'apprenant sont fortement dépendantes des connaissances portant sur la méthode (étape 4.1.3), mais aussi des différentes étapes de résolution proposées dans l ?EIAH. Le développement de cette interface doit bien sûr être faite par un informaticien.

4.1.5 Élicitation des connaissances destinées à accompagner l'apprenant

En s'appuyant sur les connaissances sur la méthode (étape 3), les différentes tâches à effectuer (étape 4.1.4), ainsi que les méta-modèles destinés à accompagner l'apprenant, l'auteur définit les connaissances nécessaires permettant de fournir à l'apprenant de l'aide, un diagnostic de ses réponses et des explications pertinentes sur ses erreurs.

Une fois le processus d'élicitation terminé, le système doit permettre à l'auteur d'avoir un feedback sur les connaissances élicitées.

4.2 Feedback après l'élicitation des connaissances

Permettre à l'auteur d'avoir un feedback sur le processus d'élicitation des connaissances est très important. Le feedback présenté ici est différent de la fonctionnalité de contrôle des connaissances présentée dans les sections précédentes. En effet, dans AMBRE-KB, pour chaque type de connaissances à éliciter (vocabulaires du domaine, problèmes à résoudre, connaissances de classification, connaissances de reformulation, connaissances de résolution), l'auteur dispose d'une fonctionnalité lui permettant de vérifier qu'il a explicité toutes les connaissances nécessaires, et le cas échéant d'avoir un message lui permettant de comprendre son erreur et de la corriger. Par exemple, lorsque l'auteur définit son arbre de classification, il peut utiliser cette fonctionnalité pour véri-

fier qu'il a associé des noms à toutes les classes, des attributs discriminants à toutes les classes non opérationnelles, que les feuilles de l'arbre sont toutes des classes opérationnelles, etc. Ces fonctionnalités de contrôle de connaissances sont proposées au moment de l'élicitation des connaissances, et permettent de réduire les risques d'erreurs.

La fonctionnalité de feedback quant à elle est proposée après l'élicitation de toutes les connaissances nécessaires à l'EIAH pour résoudre les problèmes du domaine. Cette fonctionnalité permet à l'auteur de tester la résolution des problèmes qu'il a définis. Pour cela, il doit d'abord, en utilisant AMBRE-KB, générer les modèles correspondants aux connaissances qu'il a explicité, et qui seront utilisés par le système pour résoudre les problèmes du domaine. Ensuite, pour chaque problème défini, il dispose d'une fonctionnalité lui permettant de tester la résolution. Ainsi, pour chaque problème, le système doit envoyer un certain nombre d'informations concernant sa résolution. Deux cas sont possibles : soit le système parvient à résoudre le problème, soit le système ne parvient pas à résoudre le problème. Dans les deux cas, il doit envoyer un certains nombre d'informations concernant la résolution du problème.

Lorsque le système parvient à résoudre un problème, il doit, par exemple, afficher la solution du celui-ci. Dans ce cas, cela voudrait dire que l'auteur a bien défini toutes les connaissances nécessaires pour résoudre ce problème. Il peut alors consulter des informations concernant les différentes étapes de résolution telles que la classe du problème, la liste des classes parcourues, les attributs calculés, et pour chacun d'eux la liste des règles essayées, la règle exécutée et la technique de résolution appliquée. Les règles essayées correspondent aux règles qui une partie des prémisses sont satisfaites, mais l'ensemble des prémisses ne sont pas respectées pour permettre à la règle d'être déclenchée.

Il peut arriver aussi que le système n'arrive pas à résoudre un problème soit parce qu'il n'a pas assez de connaissances, soit parce qu'il y a des erreurs dans les connaissances explicitées. Par exemple, la résolution d'un problème peut échouer lorsque le système n'a pas pu trouver de règles pour conclure sur la valeur d'un attribut, ou lorsque les règles essayées ne lui permettent pas de conclure sur la valeur de cet attribut. Dans ce cas, le système doit afficher un message pour indiquer que la classification a été bloquée en renseignant la classe à laquelle elle est bloquée. Il doit également afficher la liste des règles essayées. La résolution peut également échouer lors de l'application d'une technique de résolution. Dans ce cas, le système doit afficher un message pour le spécifier à l'auteur. Pour cela, il doit afficher la classe du problème pour permettre à l'auteur de vérifier la

technique associée à cette classe et de la corriger ou la compléter.

4.3 Synthèse

Dans ce chapitre, nous avons présenté le processus interactif d'élicitation des connaissances que nous avons proposé. Ce processus s'appuie sur des méta-modèles indépendants du domaine permettant d'une part de décrire la forme des connaissances à acquérir, et d'autre part de contraindre le processus d'élicitation associé.

L'avantage de cette approche est de permettre à des auteurs de définir ensuite des modèles, qui du fait de leur conformité aux méta-modèles seront exploitables par l'EIAH. Par exemple, dans AMBRE, un moteur de raisonnement (résolveur) fondé sur ces méta-modèles a été mis en place. Celui-ci est capable d'exploiter ainsi les modèles de connaissances qui sont conformes à ces méta-modèles. Pour résoudre un problème, le résolveur de AMBRE utilise le modèle des connaissances de classification (comme un guide) et le modèle des connaissances de reformulation pour trouver la classe du problème et construire un nouveau modèle du problème à résoudre, plus adapté à l'application d'une technique de résolution. Il utilise ensuite le modèle des connaissances de résolution pour appliquer la technique de résolution a associée à cette classe afin de trouver la solution au problème donné. De la même manière, le système utilise les modèles des problèmes pour fournir des informations à l'interface. Par exemple, dans AMBRE-add [63], pour construire les listes déroulantes utilisées par l'apprenant lors de la construction de la phrase de réponse, le système utilise les données du problème : le sujet de la phrase (les prénoms de l'énoncé et des autres noms), les objets (ceux de l'énoncé, par exemple billes, fleurs, autocollants, etc.), le verbe de la phrase, etc. De même, pour fournir des fonctionnalités d'aide, de diagnostic et d'explications sur les erreurs à l'apprenant - toujours dans AMBRE-add - le système utilise dans un premier temps, les modèles des connaissances de diagnostic pour comparer les réponses de l'élève à celles du résolveur. Ensuite, il utilise les modèles des connaissances d'explications pour construire un message d'explications ou des informations que l'interface mettra en œuvre visuellement.

Notre approche est donc cohérente avec celle de T. Nodenot [61], qui préconise que pour soulager les chercheurs en EIAH non informaticiens du poids des développements, il est urgent de leur demander de développer et réutiliser des modèles plutôt que du

code. Selon lui, pour faciliter le dialogue entre ces modèles et les modèles que savent produire les informaticiens, et dans le but de rendre les modèles exploitables par les informaticiens, il est nécessaire d'expliciter les méta-modèles qui permettent d'interpréter les divers modèles proposés. D'un point de vue Ingénierie des Connaissances (IC), nous nous inscrivons dans une démarche d'acquisition guidée par les modèles, puisque les méta-modèles guident le processus d'Ingénierie des Connaissances.

De plus, le fait que la résolution d'un problème soit séparée en différents processus (modélisation, classement, application d'une technique) permet de mieux distinguer les démarches et les types de difficultés propres à chaque processus. Un autre avantage de ce découpage est de permettre une réutilisabilité plus aisée des connaissances contenues dans la classification, et surtout de les faire évoluer plus facilement et avec moins de risques d'erreurs.

Notre processus est mis en œuvre à travers la conception d'un outil auteur d'un outil auteur que nous présentons dans le chapitre suivant.

Troisième partie

MISE EN ŒUVRE ET ÉVALUATIONS

Chapitre 5

AMBRE-KB : un outil auteur pour des EIAH destinés à enseigner des méthodes

Afin de mettre en œuvre les différentes contributions théoriques présentées dans les chapitres précédents, nous avons développé l'outil auteur AMBRE-KB (AMBRE-Knowledge Builder). En particulier, AMBRE-KB met en œuvre le processus d'acquisition des connaissances portant sur la méthode à enseigner (connaissances de classification, connaissances de reformulation et connaissances de résolution) et présenté dans le chapitre 4.

Dans la première section de ce chapitre, nous décrivons l'approche adoptée pour la conception de AMBRE-KB. La deuxième section illustre l'utilisation de AMBRE-KB en montrant, pour un domaine donné, comment éliciter les connaissances qui permettent au système de résoudre des problèmes. Nous présentons également, pour chacun des types de connaissances à acquérir, les fonctionnalités d'assistance proposées à l'auteur pour l'accompagner durant ce processus d'élicitation. Enfin, nous terminons ce chapitre en montrant comment l'auteur peut tester les modèles de connaissances générés par AMBRE-KB.

Sommaire

5.1 Élicitation des connaissances avec AMBRE-KB 113
5.1.1 Définition du vocabulaire 115
5.1.2 Description des problèmes à résoudre 116
5.1.3 Élicitation des connaissances sur la méthode 118
5.2 Génération des modèles de connaissances 126
5.3 Tests des modèles de connaissances 130
5.4 Conclusion 135

L'objectif de AMBRE-KB est de permettre à des auteurs (enseignant, professionnel, pédagogue plus expert) - experts de la méthode qu'ils souhaitent enseigner dans un domaine donné - de pouvoir éliciter, sans programmer, les connaissances nécessaires à la conception d'un EIAH AMBRE pour ce domaine. Pour ce faire, nous avons proposé un processus interactif d'acquisition des connaissances (cf. chapitre 4) mis en œuvre via les interfaces de AMBRE-KB. Ces interfaces sont contraintes par les méta-modèles présentés au chapitre 3, et qui décrivent la forme des connaissances que l'auteur doit définir pour construire des modèles de connaissances spécifiques au domaine qu'il aura choisi.

Pour chaque domaine, l'auteur doit définir les différents modèles de connaissances à l'aide de AMBRE-KB ; le système génère ensuite une version Prolog de ces modèles de connaissances qui constituent les bases de connaissances de l'EIAH (cf. **Figure 5.1**).

FIGURE 5.1 – Approche générale de AMBRE-KB

5.1 Élicitation des connaissances avec AMBRE-KB

Afin de permettre à un auteur de décrire les différents types de connaissances décrits au chapitre 4, nous avons proposé une interface dont le menu horizontal est organisé en onglets. Chaque onglet dispose d'un menu vertical à gauche et d'un espace de travail à droite (cf. **Figure 5.2**).

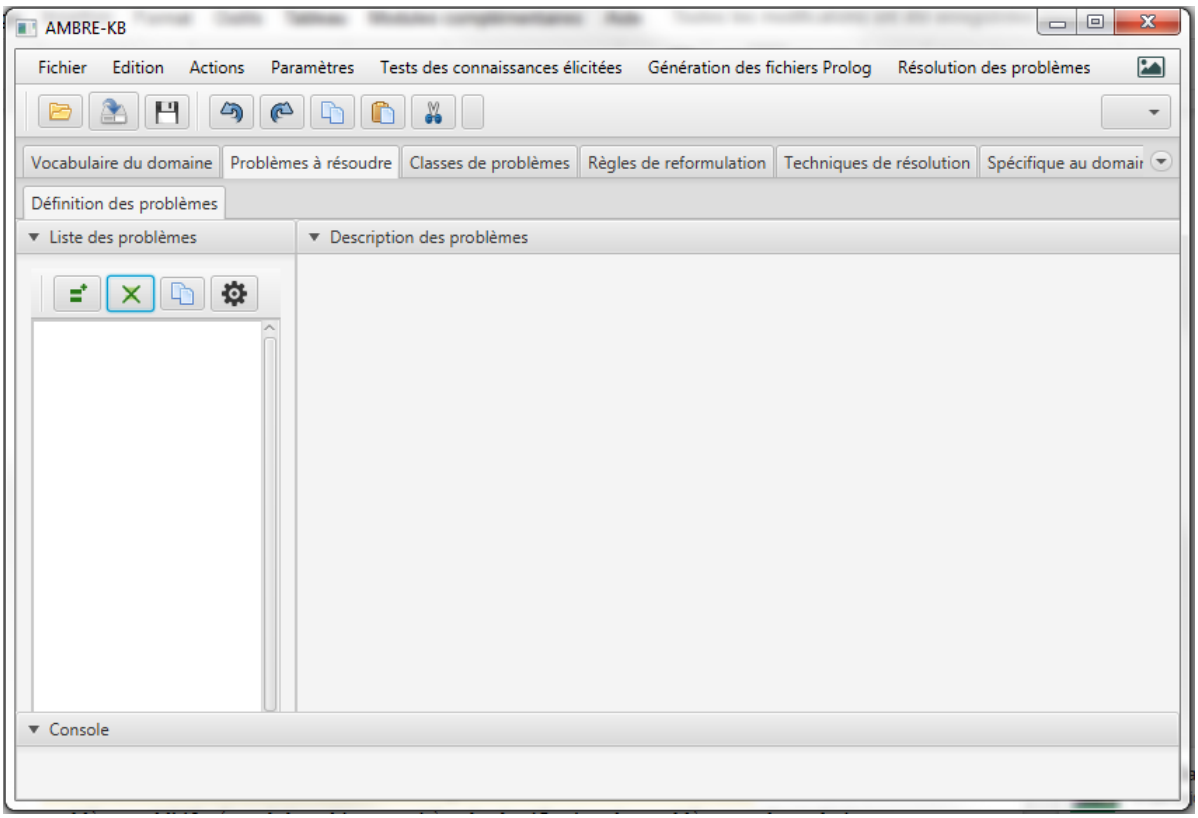

FIGURE 5.2 – Vue d'ensemble de AMBRE-KB

Pour expliquer le fonctionnement de cette interface, nous allons prendre un exemple complet avec le domaine des problèmes additifs, un des domaines les plus largement étudiés en didactique des mathématiques, psychologie cognitive, et linguistique [35]. Un EIAH AMBRE a déjà été conçu pour ce domaine, mais nous avons choisi de tester AMBRE-KB dans ce domaine pour voir comment les bases de connaissances auraient pu être définies avec notre outil auteur. Le domaine des problèmes additifs répond donc au critère de classification des problèmes et des techniques de résolution. Riley, Greeno

et Heller [64] ont proposé une classification des problèmes additifs qui distingue trois catégories de problèmes :

— Les problèmes de type « **réunion** » qui portent sur des situations statiques. Par exemple : « *Paul et pierre ont ensemble 10 billes* » ou encore « *Paul a 6 billes, Pierre en 4* ». La question consiste ici à trouver le total possédé par les deux ou un état partiel.

— Les problèmes de type « **changement** » qui décrivent une transformation appliquée à un état initial pour aboutir à un état final. Exemples : « *Sophie avait 3 fleurs pendant la récréation. On lui en a offert 4.* » ou encore « *Pierre a maintenant 7 autocollants* ». L'inconnue à chercher dans ce genre de problème peut concerner l'état initial, la transformation (qui peut être additive ou soustractive) ou l'état final.

— Les problèmes de types « **comparaison** » qui comparent des quantités statiques à l'aide de formules du type «*de plus que*», «*de moins que*». Exemples : « *Marie a 3 billes. Anna a 8 billes* ». Ou encore « *Hélène a 9 autocollants de plus que Laure* ». L'inconnue peut porter ici sur la grande partie, la petite partie, ou la différence.

En outre, le domaine des problèmes additifs est intéressant pour nous, en ce sens que les problèmes décrivent une situation concrète ; la plupart du temps, les élèves sont en difficulté pour résoudre ce genre de problèmes, parce qu'ils n'arrivent pas à représenter correctement la situation décrite dans l'énoncé [64].

Prenons par exemple le problème de type réunion suivant, que nous appelons *réunion_1*. Ce problème décrit un jeu de billes : « *Julien et Sophie ont des billes. Julien en a 3. Ensemble ils en ont 12. Combien Sophie a-t-elle de billes* » ?

Pour un problème de ce type par exemple, nous nous attendons à une résolution en plusieurs étapes :

— Décrire le problème à l'aide d'une « opération à trou » : $3 + ? = 12$

— Écrire comment s'effectue le calcul : 12 - 3 = ?

— Effectuer le calcul : 9

— Écrire la réponse à la question : *Sophie a 9 billes.*

Afin de pouvoir décrire ce genre de problème dans AMBRE-KB, l'auteur doit d'abord définir un vocabulaire pour le domaine des problèmes additifs. Pour cela, il doit

cliquer sur l'onglet « *Vocabulaire du domaine* » (cf. **Figure 5.2**).

5.1.1 Définition du vocabulaire

En cliquant sur l'onglet « *Vocabulaire du domaine* », une interface apparaît permettant ainsi à l'auteur de définir les *types d'objets*. Pour chaque type d'objet créé, le système ajoute automatiquement une caractéristique nommée *identifiant* permettant de le distinguer des autres types d'objets. Le type de cette caractéristique est une chaîne de caractères, sa valeur devra être renseignée par l'auteur à chaque fois qu'il sera utilisé pour définir des problèmes, règles ou techniques de résolution. L'auteur peut également associer d'autres caractéristiques à chaque type d'objet. Pour chacune d'elle, l'auteur doit définir un *nom* et un *type* qui peut être simple (chaîne de caractères, booléen, nombre, type énuméré, etc.) ou complexe (type objet). Il doit également spécifier si cette caractéristique est obligatoire ou pas. Définir une caractéristique comme étant obligatoire signifie que sa valeur devra être renseignée pour tous les problèmes dans lesquels le type d'objet aura été utilisé. Cette fonctionnalité est également importante, parce qu'une fois qu'une caractéristique est définie comme étant obligatoire, le système peut, à la demande de l'auteur, faire des contrôles lors de la définition des problèmes pour vérifier que sa valeur a été bien renseignée, et dans le cas contraire un message d'erreur est envoyé à l'auteur pour le lui rappeler.

La **Figure 5.3** montre une partie du vocabulaire proposé pour le domaine des problèmes additifs. Nous avons la liste des différents objets (à gauche), et pour chaque objet la liste de ses caractéristiques (au centre). Les types d'objets *personne* et *liste de personnes* par exemple, permettent de définir les personnes impliquées dans le jeu, et pour chacune d'elles, le type d'éléments possédés ainsi que leur nombre. Nous avons au centre la liste des caractéristique du type d'objet personne qui est sélectionné.

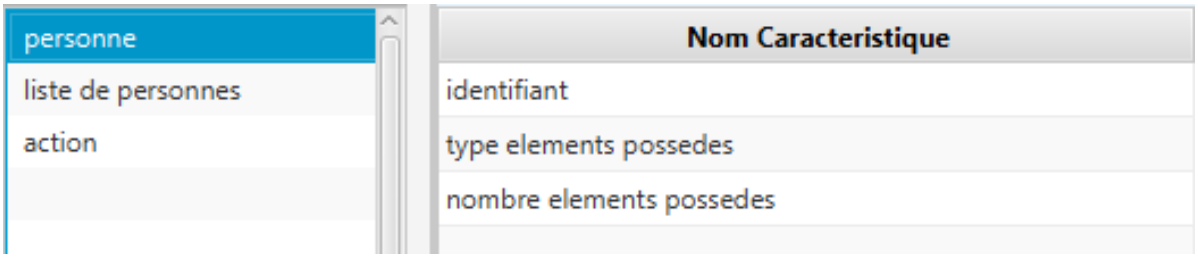

FIGURE 5.3 – Une partie du vocabulaire pour le domaine des problèmes additifs

Pour définir le vocabulaire d'un domaine, l'auteur a deux possibilités. Il peut choisir de définir d'abord l'ensemble des types objets dont il aura besoin pour décrire les problèmes à résoudre. Il peut également décider de définir les types d'objets au fur et à mesure qu'il en a besoin pour décrire les problèmes.

Le système permet à l'auteur à tout moment de vérifier si son vocabulaire est conforme au méta-modèle du vocabulaire (cf. chapitre 3). Il vérifie par exemple qu'il n'y a pas de duplication au niveau du nom des objets, et que pour chaque caractéristique définie, l'auteur a bien associé un type.

Une fois que l'auteur a défini au moins une partie du vocabulaire pour le domaine qu'il a choisi, il peut commencer à décrire des problèmes à résoudre. Pour cela, il doit cliquer sur l'onglet « *Problèmes à résoudre* » (cf. **Figure 5.2**).

5.1.2 Description des problèmes à résoudre

En cliquant sur l'onglet dédié à la description des problèmes, l'auteur dispose d'une interface permettant de définir de nouveaux problèmes, de les modifier/supprimer, d'en créer de nouveaux à partir de ceux déjà définis (dupliquer des problèmes). Pour chaque problème, l'auteur dispose d'un menu lui permettant de choisir les types d'objets lui permettant de décrire l'énoncé de son problème. Ensuite, il doit nommer chaque objet ainsi que chacune de ses caractéristiques. Le problème *réunion_1*, précédemment présenté, peut être défini par exemple comme suit :

- Nous disposons d'une liste E de deux personnes : *Julien* et *Sophie*
- Le nombre d'éléments possédés par *Julien* est *5*, et le type des éléments possédés est : *billes*
- Le nombre d'éléments possédés par *Sophie* est *X*, et le type des éléments possédés est *billes*
- Le nombre d'éléments possédés par l'ensemble des deux personnes est 12
- La question à résoudre consiste à calculer le nombre d'éléments *X* possédés par *Sophie*.

E représente ici l'identifiant de l'objet « *liste de personnes* », « *Sophie* » et « *Julien* » représentent les identifiants des objets *personnes* (éléments de la liste de personnes).

La **Figure 5.4** représente la description des objets du problème *réunion_1* en utilisant AMBRE-KB. Nous disposons d'un ensemble de deux personnes : Sophie et Julien. Julien possède 5 éléments de type billes. Sophie possède X éléments de types billes. Le nombre total d'éléments possédés par Sophie et Julien est 12. Nous cherchons ici le nombre d'éléments possédés par Sophie.

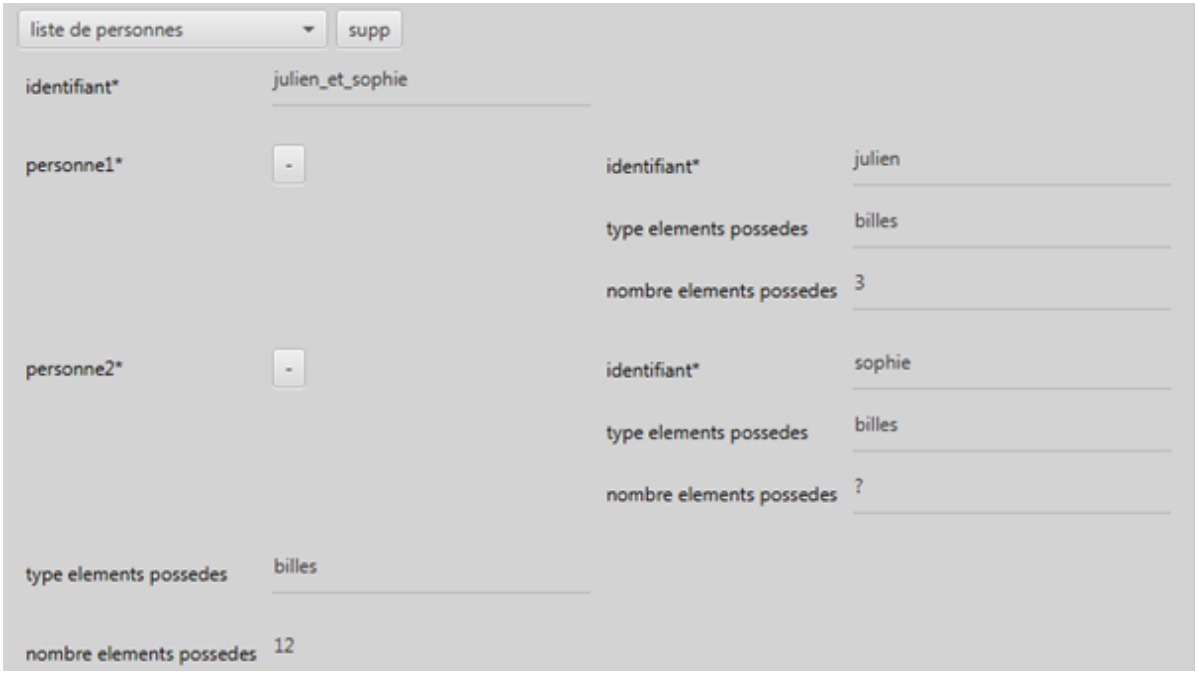

FIGURE 5.4 – Description des objets du problème réunion_1

Comme pour le vocabulaire, le système permet à l'auteur, à tout moment de la description des problèmes, de vérifier si les connaissances définies sont bien conformes au méta-modèle des problèmes (cf. chapitre 3). Par exemple, pour définir les caractéristiques des objets choisis, le système contraint les champs à remplir de sorte à permettre à l'auteur de respecter le type attendu. Pour définir par exemple, la caractéristique *nombre elements possedes* par une personne qui est de type *nombre entier*, l'auteur ne peut définir que des nombres entiers ; sauf si cette caractéristique est l'inconnue que l'on cherche et dans ce cas il met un point d'interrogation pour avertir le système que l'inconnue à calculer porte sur cette caractéristique. Le système vérifie également que les valeurs de toutes les caractéristiques définies comme étant obligatoires sont renseignées. Des fonctionnalités d'auto-complétion sont également proposées à l'auteur au fur et à mesure de la définition des objets du problème, de sorte qu'il puisse les réutiliser en cas de besoin. Une fois la description des problèmes terminée, l'auteur peut commencer à définir les

règles et les techniques de résolution. Les règles permettent d'identifier, à partir de l'énoncé des problèmes, des éléments permettant au système de les situer dans l'arbre de classification, et les techniques permettent au système de calculer les solutions des problèmes. L'ensemble formé par l'arbre de classification, les règles et les techniques constitue les connaissances sur la méthode à enseigner.

5.1.3 Élicitation des connaissances sur la méthode

Pour définir les connaissances sur la méthode, l'auteur doit définir trois types de connaissances : les connaissances de classification, les connaissances de reformulation et les connaissances de résolution. Nous présentons l'élicitation de ces connaissances dans les sections qui suivent.

5.1.3.1 Élicitation des connaissances de classification

Pour définir les connaissances de classification, l'auteur dispose d'un onglet spécifique « *Connaissances de classification* » (cf. **Figure 5.2**). En cliquant sur cet onglet, le système lui propose une interface permettant de définir un arbre de classification des problèmes. Cette interface dispose d'un menu permettant d'ajouter de nouvelles classes, de modifier ou de supprimer des classes. Pour chaque classe (cf. **Figure 5.5**) l'auteur doit :

— Définir le nom en cliquant sur (1),

— Définir l'attribut discriminant en cliquant sur (2)

— Définir statut de la classe. Par défaut toutes les classes sont non opérationnelles. Pour changer le statut d'une classe, l'auteur doit cliquer sur (3). Le système colore en jaune toutes les classes que l'auteur aura définies comme étant opérationnelles.

— Ajouter des sous-classes en cliquant sur le bouton "+" (4)

Pour chaque classe, l'auteur peut également définir des attributs du problème. Pour ce faire, pour chaque attribut, il doit définir le nom en cliquant sur (5), puis sur (6) pour le valider. Le système ajoute alors l'attribut et efface le champ (5) pour qu'il puisse rajouter d'autres attributs.

FIGURE 5.5 – Définition d'une classe de problèmes en utilisant AMBRE-KB

Pour définir un arbre de classification, le système offre plusieurs modes d'interaction, afin de répondre aux besoins et préférences de l'auteur. Par exemple, l'auteur peut choisir de construire son arbre de la racine vers les feuilles ou vice-versa. Il peut aussi choisir de définir toutes les classes et de les relier ensuite entre elles. Le système offre également la possibilité à l'auteur de définir une classe à partir d'autres classes : classe mère, classe fille, classe sœur. La **Figure 5.6** représente une partie de l'arbre de classification pour les problèmes additifs. *C_PROBLEME* représente la classe de l'arbre. Elle a pour attribut discriminant : *type du problème*. *Reunion*, *changement* et *comparaison* représentent les valeurs discriminantes de cet attribut. Les classes colorées en jaunes représentent des classes *opérationnelles* auxquelles il faudra associer une technique de résolution, les autres classes sont *non opérationnelles*.

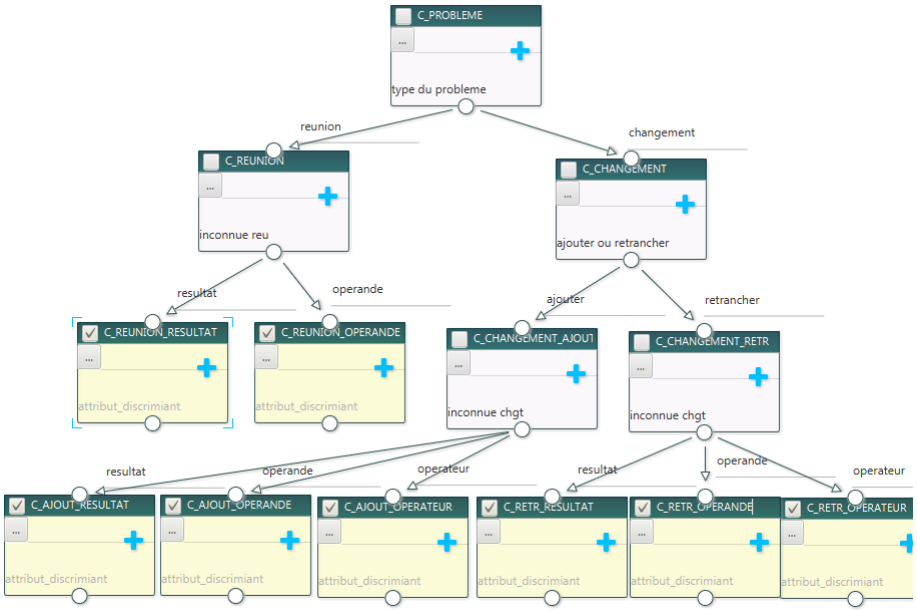

FIGURE 5.6 – Une partie de l'arbre de classification pour les problèmes additifs

L'auteur est également guidé pour la construction de l'arbre de classification. Par exemple, le fait de colorer les classes opérationnelles permet de rappeler à l'auteur qu'il doit associer des techniques de résolution à ces classes, mais également d'avoir un aperçu des classes qui sont opérationnelles et de celles qui ne le sont pas. De même, lorsque l'auteur définit une sous-classe, le système lui propose par exemple un nom pour cette classe en fonction de la valeur de l'attribut discriminant correspondant. Cette fonctionnalité permet à l'auteur d'aller plus vite dans la construction de l'arbre de classification, mais également de pouvoir repérer facilement la classe parent d'une classe en fonction des attributs discriminants.

Lorsque l'auteur définit un même attribut pour des sous-classes (par exemple, l'attribut *inconnue chgt* les classes *C_CHANGEMENT_AJOUT* et *C_CHANGEMENT_RET* cf. Figure 5.6), le système lui propose de définir cet attribut au niveau de leur parent commun le plus proche (plus proche ancêtre commun). Cette fonctionnalité va, par exemple, permettre à l'auteur de définir une seule fois les règles permettant de conclure sur la valeur de cet attribut, et de pouvoir par la suite les utiliser pour les deux classes. Dans notre exemple, les règles permettant de conclure sur la valeur de l'attribut *inconnue chgt* seront définies une seule fois.

Le système permet à l'auteur, à tout instant, de vérifier que son arbre de classi-

fication est conforme au méta-modèle correspondant (cf. Chapitre 3). Le système vérifie, par exemple, que toutes les classes terminales de l'arbre sont opérationnelles, que toutes les classes non opérationnelles ont au moins une sous-classe, etc. Lorsque l'auteur veut définir un nouvel attribut discriminant par exemple, le système lui propose des fonctionnalités d'auto-complétion pour lui donner la possibilité de vérifier les attributs discriminants déjà définis et d'en réutiliser un s'il le désire.

Une fois qu'une partie de l'arbre est définie, l'auteur peut (s'il le désire) commencer à définir les règles qui permettent au résolveur de situer les problèmes dans l'arbre de classification. Pour ce faire, l'auteur doit aller sur l'onglet « *Règles de reformulation*» (cf. **Figure 5.2**)..

5.1.3.2 Élicitation des connaissances de reformulation

Afin de faciliter l'élicitation des règles de reformulation, le système charge automatiquement dans l'interface des règles tous les attributs définis dans l'arbre de classification. Ainsi, en cliquant sur l'onglet « *Règle de reformulation* » l'auteur peut directement associer à chaque attribut les règles permettant de conclure sur sa(ses) valeur(s).

Comme pour les connaissances de classification, le système offre plusieurs modes d'interaction pour définir les règles. Par exemple, l'auteur peut choisir de définir toutes les règles au fur et à mesure de la définition de l'arbre de classification. En d'autres termes, il s'agit pour chaque classe, lors de la définition d'un attribut discriminant ou d'un attribut du problème, de définir la(s) règle(s) qui permettra(tront) de déterminer la(les) valeur(s) correspondante(s). L'auteur peut également choisir de définir entièrement l'arbre de classification, et ensuite procéder à la définition des règles. Pour l'aider dans la définition des règles, le système offre également à l'auteur de définir une règle à partir d'une autre déjà définie, en la complétant ou en la modifiant.

Le système vérifie que les connaissances explicitées sont conformes au méta-modèle correspondant. Pour cela, il contrôle que, pour chaque règle, l'auteur a défini une partie prémisse et une partie conclusion qui permet de conclure sur la valeur de l'attribut. Pour chacun des attributs, le système vérifie également qu'au moins une règle a été associée afin de permettre de déterminer sa valeur. Pour guider l'auteur, le système lui propose au fur et à mesure de la définition des règles des fonctionnalités d'auto-complétion : par

exemple dans les parties prémisses, pouvoir visualiser les objets qu'il a déjà définis et les réutiliser au besoin.

Reprenons notre exemple sur les problèmes additifs.
Les règles *regle_reunion_1*, *regle_changement_1* permettent de conclure sur la valeur de l'attribut discriminant *type du problème*. Les règles *regle_reunion_operateur* et *regle_reunion_operande* permettent de conclure sur la valeur de l'attribut *inconnue reu* (cf. **Figure 5.7**).

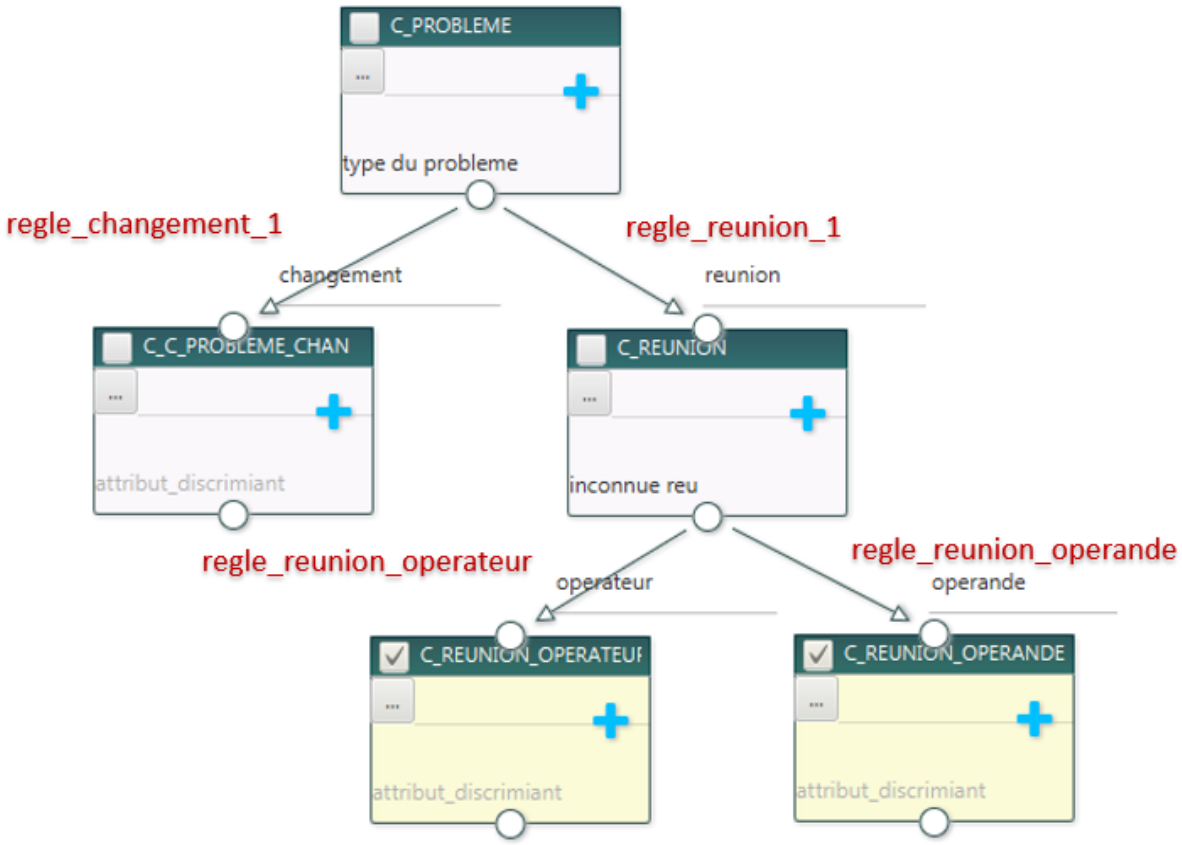

FIGURE 5.7 – Une partie de l'arbre de classification pour les problèmes additifs

La règle *regle_réunion_1* par exemple peut être représentée comme suit :

Si dans un problème on a :
- une liste de personnes L
- une personne P1 appartenant à L
- le type des éléments possédés par P1 est T1
- le nombre d'éléments possédés par P1 est N1
- une personne P2 appartenant à L
- le type d'éléments possédés par P2 est T2
- le nombre d'éléments possédés par P2 est N2
- le type d'éléments possédés par L est T
- le nombre d'éléments possédés par L est N
- T1, T2 et T sont de même catégorie
- deux parmi N1, N2 et N3 sont connus

Alors la valeur de l'attribut discriminant *type du problème* est égale à *réunion*.

Tableau 5.1 – Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème

Soient *nombre_total_possede* et nombre *partiel_possede* deux attributs du problème définis au niveau de la classe *c_reunion*, la règle *regle_reunion_operande* permettant de conclure sur la valeur de l'attribut *inconnue_reu* peut être représentée comme suit :

Si dans un problème on a :
- une liste de personnes L
- une personne P1 appartenant à L
- le type des éléments possédés par P1 est T1
- le nombre d'éléments possédés par P1 est N1
- une personne P2 appartenant à L
- le type d'éléments possédés par P2 est T2
- le nombre d'éléments possédés par P2 est N2
- le type d'éléments possédés par L est T
- le nombre d'éléments possédés par L est N
- T1, T2 et T sont de même catégorie
- l'inconnue à calculer porte sur N2

Alors la valeur de l'attribut discriminant *inconnue réu* est égale à *opérande*,
la valeur de l'attribut du problème nombre total possédé est égale à N,
et la valeur de l'attribut du problème nombre partiel possédé est égale à N1.

Tableau 5.2 – Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé

Les **Figures 5.8** et **5.9** représentent respectivement les prémisses et conclusion de la règle regle_reunion_operande définie en utilisant AMBRE-KB.

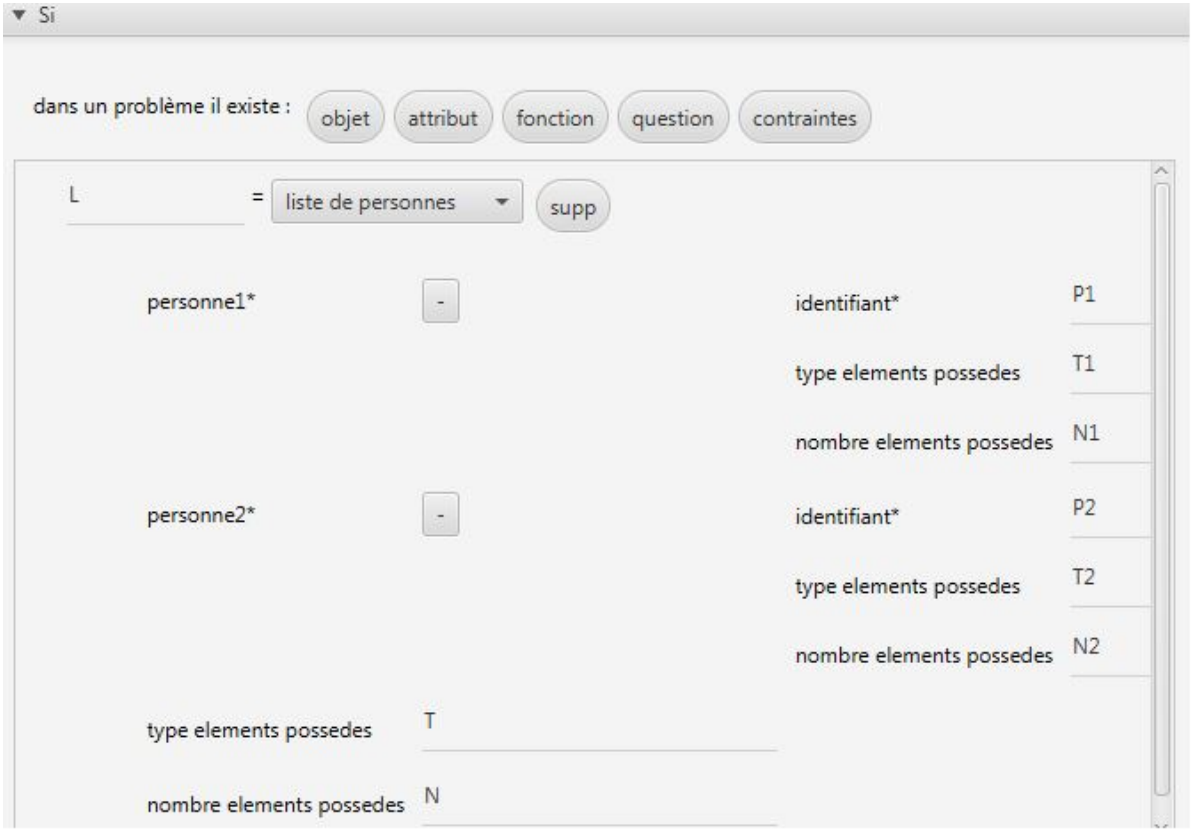

FIGURE 5.8 – représentation d'une partie des prémisses de la regle_reunion_operande dans AMBRE-KB

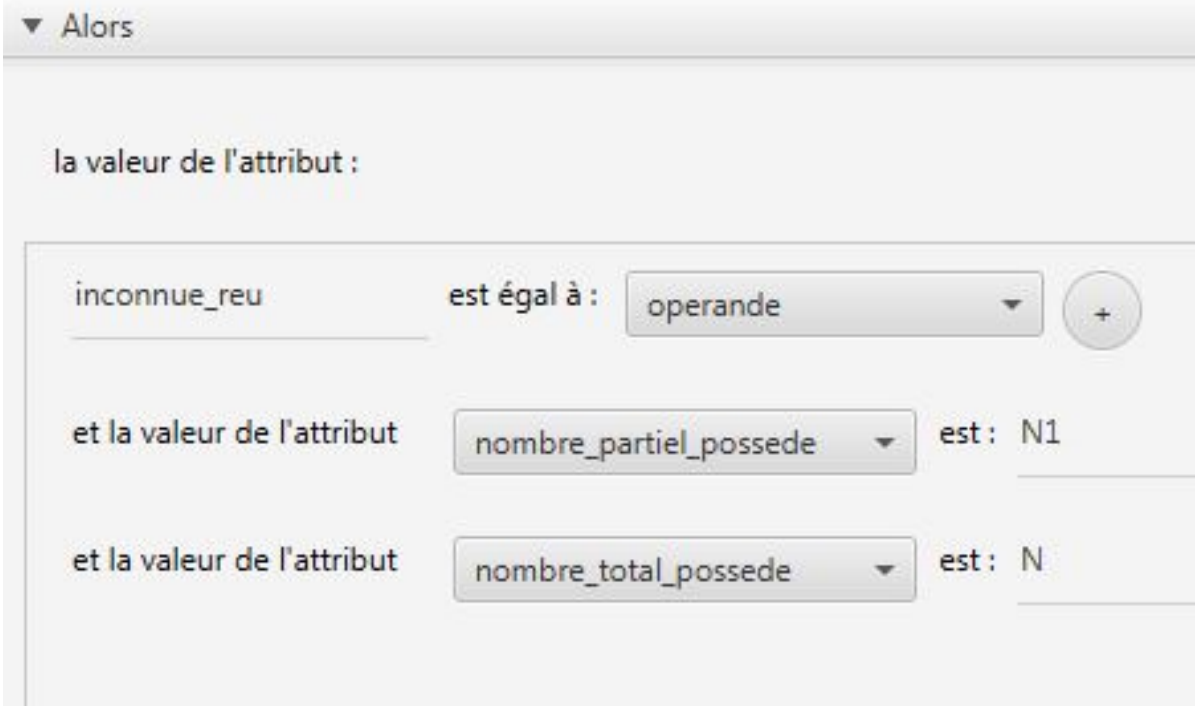

FIGURE 5.9 – représentation de la partie conclusion de la regle_reunion_operande dans AMBRE-KB

D'autre part, pour associer des techniques de résolution aux classes opérationnelles de l'arbre de classification, l'auteur doit cliquer sur l'onglet « *Techniques de résolution* » (cf. **Figure 5.2**).

5.1.3.3 Élicitation des connaissances de résolution

Lorsque l'auteur a cliqué sur l'onglet « *Technique de résolution* » une nouvelle interface s'ouvre, lui permettant ainsi de décrire les techniques de résolution. Pour faciliter l'élicitation de ces connaissances, le système charge automatiquement dans l'interface toutes les classes que l'auteur aura définies comme étant opérationnelles au niveau de l'arbre de classification. Ainsi, l'auteur peut associer directement une technique de résolution à chacune de ces classes. Pour ce faire, il peut choisir de définir les techniques au fur et à mesure de la définition des classes opérationnelles dans l'arbre de classification. Il peut également choisir de définir toutes les classes dans un premier temps et ensuite leur associer des techniques de résolution.

La technique *tech_reunion_opérande* permet par exemple de résoudre les problèmes de la classe réunion opérande.

Technique tech_réunion_opérande :

Soit P un problème,
Soit N la valeur de l'attribut *nombre total possédé* dans P,
Soit N1 la valeur de l'attribut *nombre partiel possédé* dans P,
Soit N2 le résultat de la soustraction entre N et N1,
Alors la **solution du problème** P est égale à N2.

Tableau 5.3 – Exemple de technique permettant de résoudre les problèmes de la classe *réunion*

La **Figure 5.10** montre une représentation de cette technique dans AMBRE-KB.

FIGURE 5.10 – représentation de la technique tech_réunion_opérande dans ambre-kb

5.2 Génération des modèles de connaissances

Une fois le processus d'élicitation terminé, en cliquant sur le menu « *générer les modèles de connaissances* » dans AMBRE-KB, l'auteur peut générer les modèles de

connaissances correspondants aux différentes connaissances explicitées.

Les **Figures 5.8**, **5.9**, **5.10**, et **5.11** représentent respectivement des extraits : du modèle descriptif du problème réunion1, du modèle des connaissances de classification, du modèle des connaissances de reformulation, et du modèle des connaissances de résolution.

```
fait(probleme(reunion_1)). /* soit le problème reunion_1*/
fait(enonce_naturel(reunion_1,Julien et Sophie ont des billes. Julien en a 3.
Ensemble ils en ont 12. Combien Sophie a-t-elle de billes ')).
/* énoncé de reunion_1*/
fait(est_un(julien_et_sophie,liste_de_personnes)).
/* julien_et_sophie une liste de personnes*/
fait(personne1(julien_et_sophie,julien)). /* julien une personne
appartenant à la liste de personnes*/
fait(est_un(julien,personne)). /*julien est une personne*/
fait(type_elements_possedes(julien,billes)). /* type éléments possédés
par julien est billes*/
fait(nombre_elements_possedes(julien,3)). /* nombre éléments possédés
par julien est 3 billes*/
fait(personne2(julien_et_sophie,sophie)). /* sophie une personne*/
/* appartenant à la liste de personnes*/
fait(est_un(sophie,personne)). /* sophie est une personne*/
fait(type_elements_possedes(sophie,billes)).
/* type éléments possédés
/*par sophie est billes*/
fait(nombre_elements_possedes(sophie,X)).
/* nombre éléments possédés*/
/*par sophie est X billes*/
fait(type_elements_possedes(julien_et_sophie,billes)). /* type éléments
possédés par julien et sophie est billes*/
fait(nombre_elements_possedes(julien_et_sophie,12)). /* nombre éléments
possédés par julien et sophie est 12 billes*/
fait(calculer(reunion_1,X)).
/* résoudre le problème reunion_1 consiste
/*à calculer la valeur de x*/
```

Tableau 5.4 – Modèle du problème *reunion_1* en Prolog généré par AMBRE-KB

```
/********** La Classe racine C_PROBLEME *********/
racine_arbre(c_probleme).
statut(c_probleme,non_operationnelle).
test_fils(c_probleme,P,type_du_probleme(P,_)).
valeur_fils(c_probleme,P,c_reunion) :-eval_prem(type_du_probleme(P,[reunion])).
valeur_fils(c_probleme,P,c_changement) :-eval_prem(type_du_probleme(P,[changement])).
attribut(c_probleme,P,type_du_probleme(P,_),type_du_probleme,[regle_changement_1,regle_reu
/********* La Classe C_REUNION **********/
pere(c_reunion,c_probleme).
statut(c_reunion,non_operationnelle).
test_fils(c_reunion,P,inconnue_reu(P,_)).
valeur_fils(c_reunion,P,c_reunion_resultat) :-eval_prem(inconnue_reu(P,[resultat])).
valeur_fils(c_reunion,P,c_reunion_operande) :-eval_prem(inconnue_reu(P,[operande])).
attribut(c_reunion,P,inconnue_reu(P,_),inconnue_reu,[regle_reunion_operateur,regle_reunion_op
attribut(c_reunion,P,nombre_total_possede(P,_),nombre_total_possede,[]).
attribut(c_reunion,P,nombre_partiel_possede(P,_),nombre_partiel_possede,[]).
/******** La Classe C_REUNION_OPERANDE*********/
pere(c_reunion_operande,c_reunion).
statut(c_reunion_operande,operationnelle).
...
```

Tableau 5.5 – Une partie du modèle de connaissances représentant les connaissances de classification (cf. Figure 5.6) généré par AMBRE-KB en Prolog

```
/* représentation de la règle regle_reunion_1 en Prolog*/
regle(regle_reunion_1) :-
        si([probleme(P),
                est_un(E,liste_de_personnes),
                personne1(E,P1),
                est_un(P1,personne),
                type_elements_possedes(P1,T1),
                nombre_elements_possedes(P1,X1),
                personne2(E,P2),
                est_un(P2,personne),
                type_elements_possedes(P2,T2),
                nombre_elements_possedes(P2,X2),
                type_elements_possedes(E,T3),
                nombre_elements_possedes(E,X3),
                eval_prolog(sont_meme_categorie(T1,T2)),
                eval_prolog(sont_meme_categorie(T1,T3)),
                eval_prolog(deux_sont_connus(X1,X2,X3))]),
        alors([ajouter(type_du_probleme(P,[reunion]))]).
```

```
/* représentation de la règle regle_reunion_operande en Prolog*/
regle(reunion_reunion_operande) :-
        si([probleme(P),
                est_un(E,liste_de_personnes),
                personne1(E,P1),
                est_un(P1,personne),
                type_elements_possedes(P1,T1),
                nombre_elements_possedes(P1,X1),
                personne2(E,P2),
                est_un(P2,personne),
                type_elements_possedes(P2,T2),
                nombre_elements_possedes(P2,X2),
                type_elements_possedes(E,T3),
                nombre_elements_possedes(E,X3),
                calculer(P,X2)]]),
        alors([ajouter(inconnue_reu(P,[operande])),
                ajouter(nombre_total_possede(P,X3)),
                ajouter(nombre_partiel_possede(P,X1))]).
```

Tableau 5.6 – Une partie du modèle de connaissances décrivant les règles généré par AMBRE-KB en Prolog

```
/* technique à appliquer pour résoudre les problèmes de la classe c_reunion_operande*/
appliquer_resolution(P,c_reunion_operande) :-
eval_prem(nombre_total_possede(P,N)),
eval_prem(nombre_partiel_possede(P,N1)),
soustraction(N,N1,N2),
ajouter_bf(solution(P,[N2])).
```

Tableau 5.7 – Une partie du modèle de connaissances décrivant les connaissances de résolution, généré par AMBRE-KB en Prolog

5.3 Tests des modèles de connaissances

Une fois le processus d'élicitation terminé et les modèles de connaissances générés, AMBRE-KB offre à l'auteur la possibilité d'avoir un feedback sur les connaissances explicitées, en testant la résolution de problèmes qu'il a définis. Pour cela, l'auteur doit choisir un problème et lancer la résolution. Le résolveur effectue alors une phase d'opérationnalisation qui lui permet de résoudre le problème sélectionné. Lorsque la résolution du problème a réussi, le système affiche à l'auteur un message lui permettant de voir que tout s'est bien passé, ainsi que la solution du problème. Il lui propose aussi de consulter la trace de la résolution (les règles appliquées, les attributs calculés ainsi que leurs valeurs, les règles essayées, et celles exécutées,la technique de résolution appliquée).

Les **Figures 5.11** et **5.12** montrent des exemples de feedback renvoyés par AMBRE-KB en testant la résolution du problème *reunion_1*.

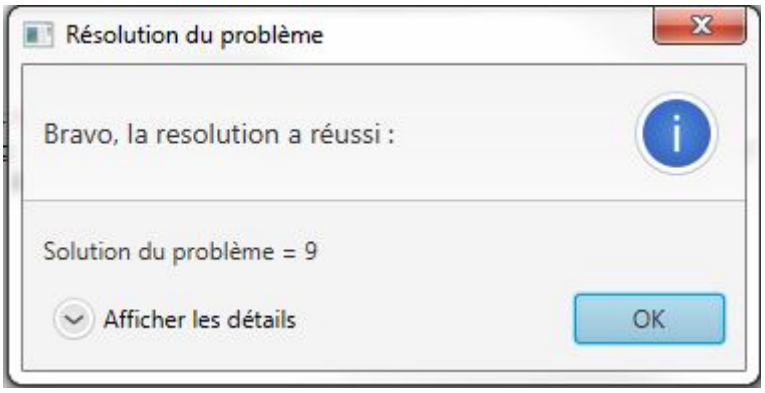

FIGURE 5.11 – Exemple de feedback lorsque la résolution réussit. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème

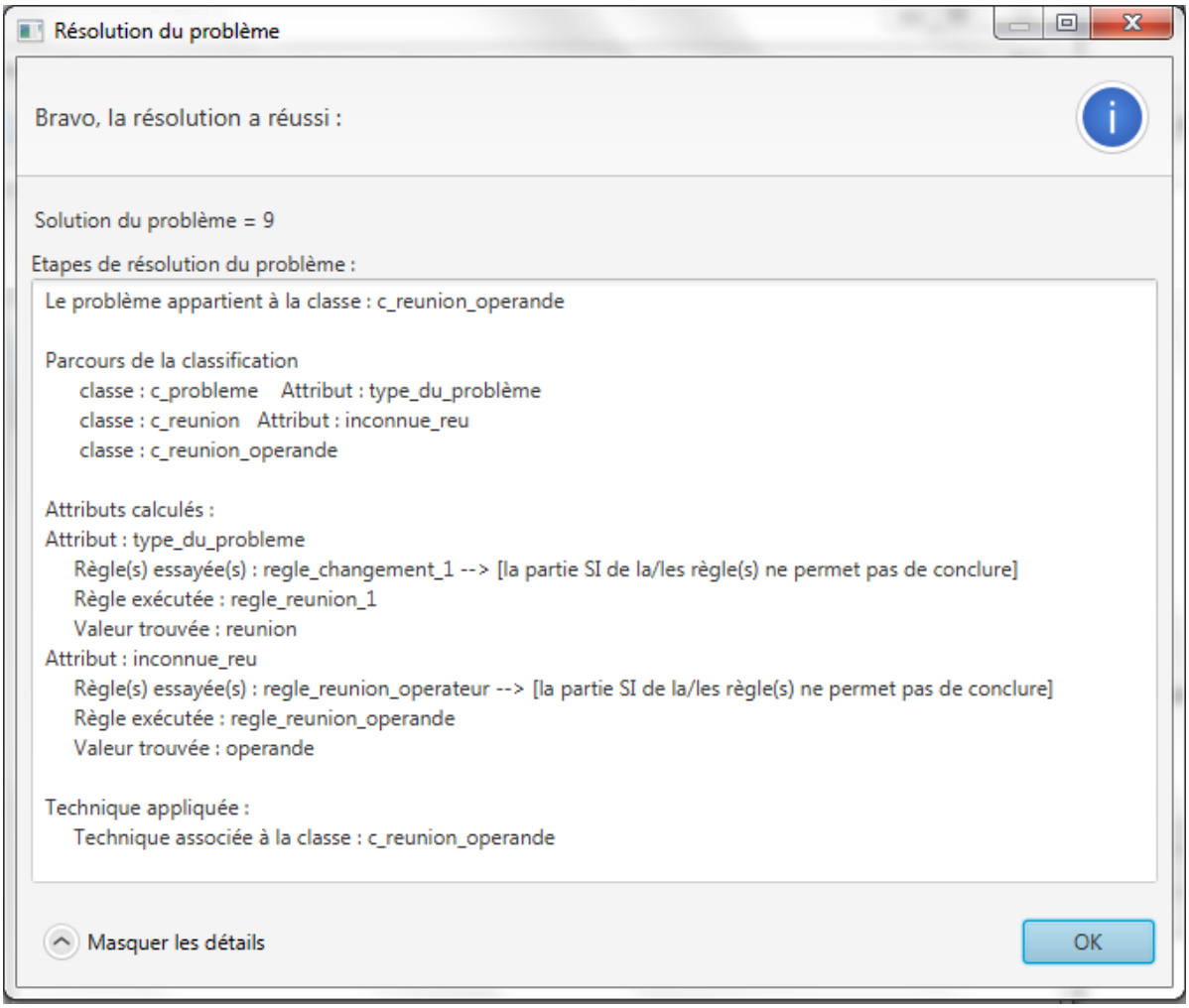

FIGURE 5.12 – Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a réussi

De même, lorsque le résolveur ne parvient pas à résoudre un problème, il affiche un message informant l'auteur que la résolution a échoué. Par exemple, la résolution d'un problème peut échouer lorsque le système n'arrive pas à trouver de règles permettant de conclure sur la valeur d'un attribut discriminant. Dans ce cas, le système envoie un message d'erreur pour indiquer à l'auteur que la classification est bloquée au niveau de la classe de cet attribut.

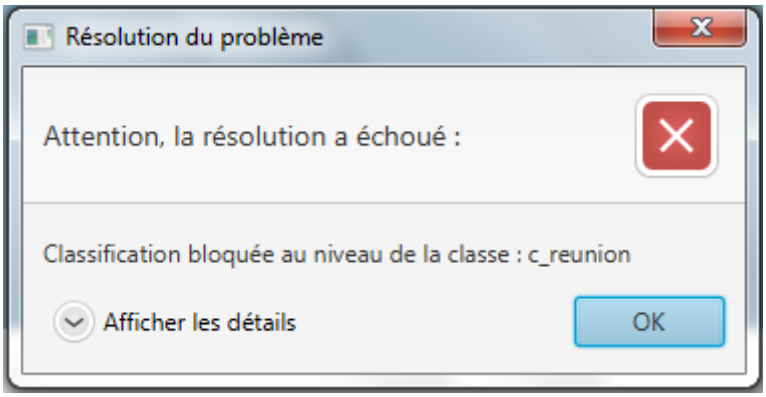

FIGURE 5.13 – Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème

En cliquant sur "*afficher les détails*" (cf. **Figure 5.13**), le système affiche :

— les classes rencontrées, avec pour chacune d'elle la valeur de l'attribut discriminant

— les attributs calculés. Pour chaque attribut, le système affiche les règles essayées (s'il en existe) et pourquoi elles ne se sont pas déclenchées. Les règles essayées représentent les règles dont l'ensemble des prémisses ne sont pas satisfaites, elles ne permettent donc pas au système de conclure. Si des règles ont été exécutées, il les affiche également, ainsi que la valeur trouvée.

— l'attribut auquel la classification est bloquée

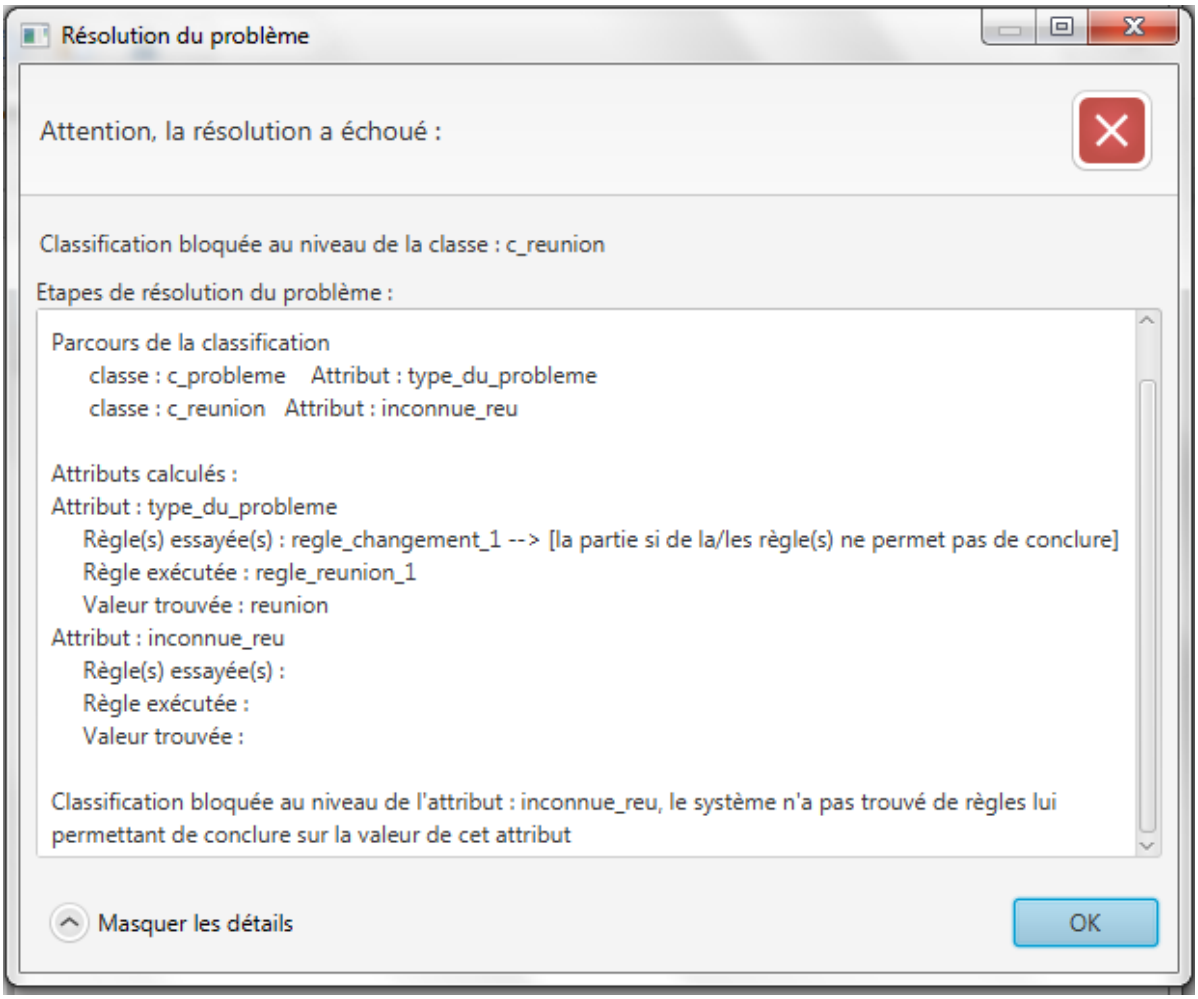

FIGURE 5.14 – Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée parce que le système n'a pas trouvé de règle permettant de conclure sur la valeur d'un attribut

La résolution d'un problème peut également échouer lorsque le résolveur identifie la classe du problème, mais que la technique de résolution a associé est incomplète ou comporte des erreurs. Dans ce cas, le système envoie un message d'erreur permettant à l'auteur de comprendre que la résolution a échoué à cause d'un problème rencontré lors de l'application d'une technique de résolution.

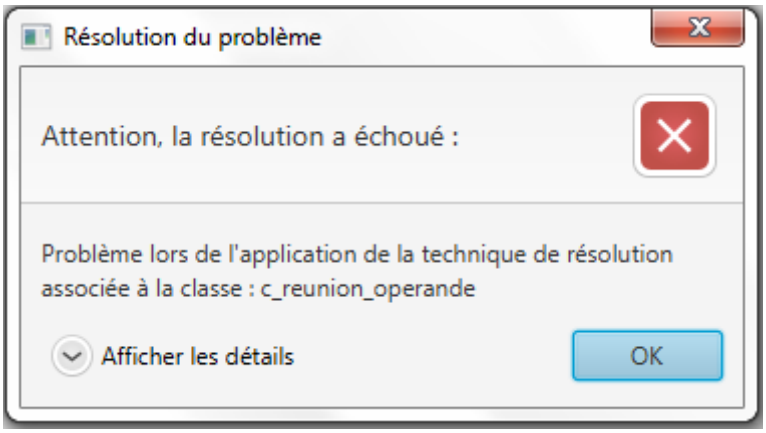

FIGURE 5.15 – Exemple de feedback lorsque la résolution a échoué lors de l'application d'un technique de résolution. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème

En cliquant sur "*afficher les détails*" (cf. **Figure 5.15**), le système affiche les différentes étapes de résolution : la classe du problème, les classes rencontrées, les attributs calculés et pour chacun d'eux, les règles essayées, celles exécutées et les valeurs calculées. Ces informations permettent à l'auteur de savoir qu'il doit corriger ou compléter la technique de résolution associée à la classe pour pouvoir corriger son erreur. La **Figure 5.16** montre un exemple de feedback plus détaillé.

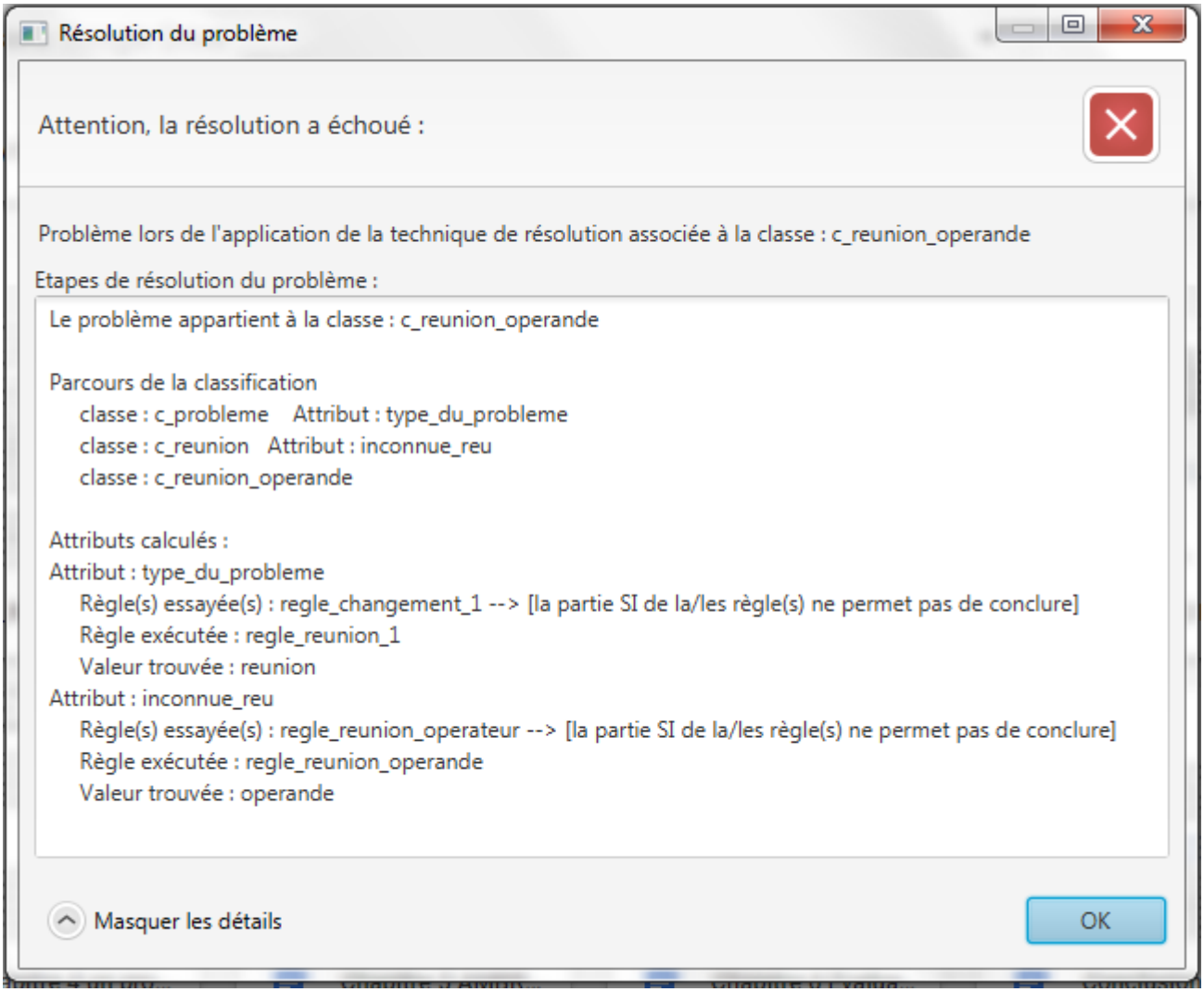

FIGURE 5.16 – Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a échoué lors de l'application d'une technique de résolution

5.4 Conclusion

Nous avons présenté dans ce chapitre notre outil auteur, AMBRE-KB, dont l'objectif est d'assister un auteur dans la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes. En particulier, AMBRE-KB permet d'acquérir les connaissances nécessaires au système pour résoudre des problèmes dans un domaine donné (connaissances de classification, connaissances de reformulation et connaissances de résolution). L'élicitation des connaissances permettant d'accompagner l'apprenant

(connaissances d'aide, de diagnostic et d'explications sur les erreurs) n'est pas encore prise en compte.

AMBRE-KB repose sur un processus interactif d'élicitation des connaissances. Ce processus est basé sur des méta-modèles décrivant la forme des différentes connaissances à acquérir, et permettant de contraindre les interfaces dédiées à éliciter les connaissances. AMBRE-KB assiste également l'auteur durant le processus en lui offrant une grande flexibilité, notamment concernant l'ordre de définition des connaissance, en le guidant, et en lui permettant de contrôler au fur et à mesure que les connaissances explicitées sont conformes aux différents méta-modèles. À la fin du processus d'élicitation, le système offre à l'auteur la possibilité d'avoir un feedback en lui permettant de tester la résolution des problèmes qu'il a définis, et de corriger ses erreurs au besoin.

Nous avons utilisé le domaine des problèmes additifs pour illustrer le fonctionnement de AMBRE-KB. Un EIAH AMBRE existe déjà pour ce domaine, mais nous avons choisi de tester AMBRE-KB à nouveau dans ce domaine pour voir comment les bases de connaissances auraient pu être créées en utilisant notre outil auteur. Les modèles de connaissances générés par AMBRE-KB ont permis de résoudre les problèmes du domaine, mais des tests supplémentaires doivent être réalisés dans de nouveaux domaines pour évaluer l'outil.

Nous présentons dans le chapitre 6 les différentes expérimentations menées pour évaluer la mise en œuvre de l'outil auteur AMBRE-KB

Chapitre 6

Évaluations

Cette thèse a abouti à plusieurs propositions visant à la mise en place d'un outil auteur destiné à assister l'auteur dans l'élicitation des connaissances nécessaires à la conception d'EIAH dédiés à enseigner des méthodes de résolution de problèmes dans des domaines spécifiques. Ces propositions ont été mises en œuvre de manière opérationnelle à travers l'outil auteur AMBRE-KB. Ce chapitre présente les évaluations conduites pour évaluer le processus d'acquisition de connaissances ainsi que sa mise en œuvre dans l'outil auteur AMBRE-KB.

Sommaire

6.1 Mise à l'essai de AMBRE-KB 139

6.1.1 Première expérimentation 139

6.1.2 Seconde expérimentation 143

6.2 Bilan des expérimentations 154

Les travaux présentés dans cette thèse portent sur l'acquisition des connaissances nécessaires à la conception d'EIAH destinés à enseigner, pour un domaine donné, une méthode de résolution de problèmes. Les questions que nous nous sommes posées sont les suivantes :

1. **Comment permettre à un auteur (enseignant, professionnel ou pédagogue plus expert), pour un domaine donné, d'éliciter, sans programmer, les connaissances nécessaires à l'EIAH pour résoudre les problèmes de ce domaine relevant de la méthode à enseigner ?**
2. **Comment assister l'auteur durant ce processus d'élicitation ?**

Pour répondre à ces questions, nous avons d'abord proposé des méta-modèles pour décrire la forme des connaissances à acquérir. Nous avons ensuite proposé un processus interactif d'acquisition interactif des connaissances, fondé sur ces méta-modèles. Enfin, nous avons mis en œuvre ce processus en implémentant l'outil auteur AMBRE-KB. Même si l'élicitation des connaissances destinées à accompagner l'apprenant durant la résolution (connaissances d'aide, de diagnostic et d ?explications sur les erreurs) n'est pas encore prise en compte par AMBRE-KB, cet outil permet d'assister l'auteur durant l'élicitation des connaissances portant sur la méthode à enseigner, en le guidant tout au long du processus, en lui offrant une grande flexibilité notamment sur l'ordre de définition des connaissances, et en lui permettant de contrôler les connaissances au fur et à mesure qu'il les explicite. En effet, pour un domaine donné, celles-ci (les connaissances portant sur la méthode) permettent à elles seules de pouvoir résoudre les problèmes relevant de cette méthode.

Afin de mettre à l'essai AMBRE-KB, nous avons donc mené deux expérimentations portant sur l'élicitation des connaissances de la méthode à enseigner. Ce chapitre décrit de manière détaillée comment ces expérimentations ont été conduites, ainsi que les résultats obtenus. Il se termine par un bilan de ces expérimentations.

6.1 Mise à l'essai de AMBRE-KB

6.1.1 Première expérimentation

Nous avons mené une première expérimentation afin de mettre à l'essai AMBRE-KB. Cette expérimentation était divisée en deux étapes :

— **1ère étape**. Elle consistait à faire utiliser AMBRE-KB par des auteurs pour éliciter les connaissances à enseigner pour le domaine qu'ils ont choisi. Ces auteurs avaient pour tâche d'une part de définir de définir un vocabulaire pour leur domaine et d'utiliser ce vocabulaire pour décrire les problèmes qu'ils souhaitent faire résoudre par l'apprenant et le système. Ils devaient aussi, d'autre part définir les connaissances portant sur la méthode à enseigner (connaissances de classification, connaissances de reformulation et connaissances de résolution). Cette étape se terminait lorsqu'ils avaient fini d'explicité leurs connaissances.

— **2ème étape**. La deuxième étape était alors réalisée sans l'auteur. Il s'agissait de générer les modèles de connaissances correspondant aux connaissances explicitées par l'auteur en utilisant AMBRE-KB, puis de tester ces modèles indépendamment de AMBRE-KB, en utilisant le résolveur de AMBRE. Nous avons testé, pour chaque domaine, la résolution des problèmes définis par l'auteur, en utilisant le résolveur et les modèles de connaissances générés. Notre hypothèse était la suivante : « si le résolveur de AMBRE parvient à résoudre les problèmes des domaines à partir des modèles de connaissances générés par AMBRE-KB, alors AMBRE-KB est capable d'acquérir les connaissances nécessaires à un EIAH AMBRE pour résoudre les problèmes d'un domaine ».

Notre objectif pour cette expérimentation était double : nous souhaitions d'une part tester AMBRE-KB avec un domaine déjà existant (un EIAH AMBRE existant) afin de voir comment les bases de connaissances peuvent être créées à partir de AMBRE-KB. Le domaine des problèmes additifs a été ainsi testé. D'autre part, nous voulions tester le fonctionnement de AMBRE-KB sur de nouveaux domaines. Deux domaines ont ainsi été testés : le *calcul d'aire* et la *pâtisserie*.

Le **Tableau 6.1** résume le critère d'évaluation retenu pour cette expérimentation.

Point à analyser	Pour un domaine donné, les modèles de connaissances générés par AMBRE-KB permettent-ils au résolveur de résoudre des problèmes ?
Méthodologie envisagée	Pour chaque domaine, tester la résolution des problèmes en utilisant les modèles de connaissances générés par AMBRE-KB, et en lançant le résolveur en dehors de AMBRE-KB.
Condition de validation	Si le résolveur parvient à résoudre les problèmes d'un domaine, alors AMBRE-KB est capable d'acquérir les connaissances nécessaires à l'EIAH pour résoudre les problèmes de ce domaine.

Tableau 6.1 – Critère d'évaluation pour la première expérimentation

6.1.1.1 Protocole expérimental

Cette expérimentation s'est déroulée en deux étapes avec 3 sujets, dans trois domaines : deux nouveaux domaines (géométrie et pâtisserie) et un domaine (les problèmes additifs) dans lequel un EIAH AMBRE existe déjà (EIAH AMBRE-add). Les deux sujets qui ont testé sur les deux nouveaux domaines (géométrie et pâtisserie) ont d'abord commencé par choisir leur domaine. Pour cela, ils avaient pour contrainte de respecter le critère de classification des problèmes et des techniques de résolution imposé par l'architecture des EIAH AMBRE. Une fois qu'ils ont choisi leur domaine, ils ont commencé par réfléchir sur le type d'exercices à faire résoudre par le système et l'apprenant. Cette étape leur a permis de proposer le vocabulaire pour le domaine. Ensuite, ils ont dessiné sur papier une partie de l'arbre de classification de leur domaine. Ils ont également explicité sur papier quelques problèmes à résoudre, quelques règles, quelques techniques de résolution. En s'inspirant des problèmes à résoudre définis dans l'EIAH AMBRE-add, le troisième sujet a également proposé un vocabulaire pour le domaine. L'expérimentation s'est ensuite déroulée en deux étapes pour les trois sujets :

Étape 1 : Dans cette étape, les auteurs utilisent AMBRE-KB pour définir le vocabulaire de leur domaine, les problèmes à résoudre ainsi que les connaissances de classification, de reformulation et de résolution.

Étape 2 : Une fois que les auteurs ont terminé d'éliciter les connaissances pour leur domaine, à l'aide de AMBRE-KB, nous générons les modèles de connaissances corres-

pondants. Ensuite, en lançant le résolveur indépendamment de AMBRE, nous testons la résolution des différents problèmes définis.

6.1.1.2 Résultats

A l'issue de cette expérimentation, nous avons obtenu les résultats suivants :

— **Connaissances de classification** : pour tous les domaines les auteurs ont pu définir toutes les connaissances nécessaires

— **Autres connaissances** (vocabulaire, connaissances de reformulation, connaissances de résolution) : les interfaces proposées par AMBRE-KB n'ont pas permis de représenter toutes les connaissances nécessaires

— **Pas de génération de modèles de connaissances** : les interfaces proposées n'ayant pas permis d'expliciter toutes les connaissances, il n'y a pas eu de génération de fichiers, et par conséquent la résolution des problèmes n'a pas pu être testée.

En outre, les observations sur l'utilisation du système par les auteurs suggèraient plus d'interactivité de la part du système. En effet, durant le processus d'élicitation des connaissances, AMBRE-KB ne proposait rien permettant aux auteurs de vérifier s'ils avaient bien explicité les connaissances ou pas. De plus, à la fin de l'expérimentation, les utilisateurs ont manifesté le besoin d'avoir eux-mêmes un feedback sur le processus d'élicitation, c'est-à-dire de pouvoir vérifier si les connaissances ont été bien explicitées, et le cas échéant d'avoir un retour qui leur permettrait de comprendre l'origine de leur erreur afin de pouvoir la corriger. Ils ont posé les questions suivantes : comment est-ce que le système résout les problèmes ? comment savoir si la résolution des problèmes s'est bien passée ? et, le cas échéant, quelle est la cause de l'échec de la résolution et comment la corriger ?

En résumé, cette expérimentation ne nous a pas permis de valider notre hypothèse de départ. En effet, les interfaces proposées n'ayant pas permis aux auteurs d'expliciter toutes les connaissances nécessaires, les modèles de connaissances nécessaires à la résolution des problèmes n'ont pas pu être générés par le système. De plus, le système ne proposant aucun mécanisme d'interaction, ni de feedback, les auteurs n'avaient aucun moyen pour vérifier s'ils avaient bien défini les connaissances, si les connaissances définies étaient incomplètes ou s'ils avaient commis des erreurs lors de l'élicitation de celles-ci.

6.1.1.3 Recommandations et modifications

Les résultats de la première expérimentation ont conduit à modifier AMBRE-KB afin d'intégrer des fonctionnalités permettant d'assurer l'interactivité du système, mais aussi de permettre à l'auteur d'avoir un feedback sur les connaissances explicitées, ainsi que sur la résolution des problèmes par le résolveur.

Pour améliorer l'interactivité de AMBRE-KB, une nouvelle fonctionnalité a été mise en place pour chaque type de connaissances. Ainsi, après avoir explicité chaque type de connaissances (connaissances de classification, ou connaissances de reformulation, ou connaissances de résolution), l'auteur peut tester si les connaissances élicitées sont bien conformes au méta-modèle. Par exemple, pour les connaissances de classification, AMBRE-KB vérifie que les feuilles de l'arbre sont toutes des classes opérationnelles. Il vérifie également que chaque classe non opérationnelle possède au moins une sous classe, etc.

De plus, comme les interfaces proposées ne permettaient pas d'expliciter toutes les connaissances (notamment de reformulation et de résolution, et le vocabulaire), il était nécessaire de faire évoluer les méta-modèles correspondants, et donc les interfaces qui sont contraintes par ceux-ci. Par exemple, pour le vocabulaire, les auteurs souhaitaient pouvoir énumérer toutes les valeurs possibles pour une caractéristique, mais l'interface proposée par AMBRE-KB ne le permettait pas. Nous avons donc fait évoluer les interfaces pour intégrer les nouvelles fonctionnalités.

Une nouvelle fonctionnalité a également été développée pour permettre à l'auteur d'avoir un feedback sur les connaissances explicitées ainsi que sur la résolution des problèmes par le résolveur. Elle permet à l'auteur de tester lui-même la résolution des problèmes qu'il a définis à partir de AMBRE-KB. Cette fonctionnalité consiste à fournir un ensemble d'informations après la résolution du problème. Lorsque le résolveur réussit à résoudre un problème, ces informations peuvent être, par exemple, la classe du problème, les règles essayées, celles exécutées, les valeurs des attributs déterminés, la solution du problème, etc. Lorsque la résolution d'un problème échoue, le résolveur doit envoyer un ensemble d'informations lui permettant de comprendre son erreur et de pouvoir la corriger : par exemple, la classe à laquelle le résolveur s'est arrêté, les règles essayées, celles exécutées, etc.

Pour faciliter le processus d'élicitation, les auteurs ont également manifesté le besoin d'avoir des fonctionnalités d'auto-complétion. Cela leur permettrait en effet, au fur et à mesure du processus d'élicitation, d'avoir une idée des connaissances déjà explicitées et de pouvoir les réutiliser au besoin. Par exemple, ces fonctionnalités peuvent être très utiles lors de la description des problèmes par exemple : connaître les objets déjà définis pour pouvoir les réutiliser. Dans les problèmes additifs par exemple, où l'on a des listes de personnes, cela peut être utile lorsqu'on commence d'abord par définir les types d'objet *personne*, de pouvoir les réutiliser lorsqu'on définit les *listes de personnes*. La nouvelle version de AMBRE-KB, incluant ces différentes améliorations, a permis à nouveau de mettre à l'essai l'outil, dans une seconde expérimentation.

6.1.2 Seconde expérimentation

À partir de la version modifiée de AMBRE-KB, une seconde expérimentation a été menée.

6.1.2.1 Protocole expérimental

Pour des raisons pratiques, l'auteur est filmé pendant toute la durée de l'expérimentation. Le matériel utilisé se compose d'un ordinateur (avec souris et clavier) sur lequel est installé AMBRE-KB, une caméra, un magnétophone, un chronomètre et un questionnaire.

6.1.2.2 Déroulement

L'expérimentation s'est déroulée en quatre étapes.

Étape 1 : La première phase a consisté à présenter à l'auteur :

- Le principe de AMBRE ainsi que les méta-modèles de connaissances : la forme et le rôle des différents types de connaissances à acquérir.
- Le fonctionnement général de AMBRE-KB.

À la fin de cette étape, l'auteur est invité à commencer à réfléchir sur un domaine sur lequel il va tester l'outil.

Étape 2 : Lors de cette étape, l'auteur commence par choisir le domaine sur lequel il va tester AMBRE-KB. Il réfléchit ensuite aux types de problèmes qu'il souhaite faire résoudre à l'apprenant. Cette étape va en effet lui permettre de proposer un vocabulaire pour son domaine.

Une fois qu'il a proposé un vocabulaire pour son domaine, il est important que l'auteur réfléchisse également sur les règles et techniques. Cette étape est tout aussi importante que la précédente, parce qu'elle va nous permettre d'anticiper la phase de test des fichiers générés. En effet, le résolveur de AMBRE étant codé en Prolog, tous les modèles de connaissances doivent être générés en Prolog, de même que les toutes les fonctions utilisées par le résolveur pour appliquer une technique de résolution. Pour définir une technique pour un domaine, généralement, l'auteur a besoin de définir des fonctions spécifiques, comme par exemple : calculer la somme de deux nombres (les problèmes additifs par exemple), la racine carrée d'un nombre (les équations par exemple), etc. Pour que le résolveur puisse tester la résolution d'un problème il doit disposer de toutes les fonction (en Prolog) définies par l'auteur. Dans AMBRE-KB, l'auteur va définir la spécification de la fonction (description de la fonction ainsi que de ses paramètres) et nous nous chargeons de développer les fonctions correspondantes. À la fin de cette phase, l'auteur est prêt à utiliser AMBRE-KB pour éliciter les connaissances de son domaine.

Étape 3 : Lors de cette étape, l'auteur utilise en toute autonomie AMBRE-KB pour expliciter les différentes connaissances qu'il avait explicitées sur papier. Nous nous positionnons ainsi en retrait afin d'observer l'interaction de l'auteur avec le logiciel, tout en notant sur une fiche les temps passés sur chaque type de connaissance, les difficultés rencontrées au cours de l'utilisation de AMBRE-KB, les gestes et comportements non verbaux de l'auteur ainsi que ses remarques et questions. Lorsque les questions posées par l'auteur portent sur le fonctionnement de AMBRE-KB, nous l'envoyons dans un premier temps sur l'aide proposée. Toutefois, si l'aide n'est pas suffisante, nous lui donnons les indications nécessaires pour qu'il puisse poursuivre l'élicitation.

Lorsque nous remarquons que l'auteur n'avance plus dans le processus d'élicitation, nous considérons qu'il est bloqué ou éprouve des difficultés à comprendre le fonction-

nement de l'outil. Dans ce cas, nous lui posons une question pour savoir ce qu'il veut faire exactement. Si la réponse à cette question correspond à une fonctionnalité prise en compte par l'outil, nous le guidons en lui montrant comment faire. Si ce qu'il veut faire n'est pas pris en compte par l'outil, nous considérons que cette tâche ne peut être achevée, et dans ce cas il passe à l'élicitation des autres types de connaissances. Au terme de l'explicitation des connaissances, l'auteur peut générer les modèles de connaissances correspondants en utilisant AMBRE-KB. Ces différents modèles constituent les bases de connaissances de l'EIAH dont le résolveur a besoin pour résoudre les problèmes.

Étape 4 : Dans cette étape, l'auteur teste la résolution des problèmes dans AMBRE-KB. Deux cas sont possibles :

— Soit le résolveur parvient à résoudre un problème, et il affichera ainsi la classe du problème, les règles exécutées, la valeur des différents attributs et la solution du problème.

— Soit le résolveur ne parvient pas à résoudre un problème, et il affichera ainsi un message permettant à l'auteur de comprendre son erreur pour pouvoir ensuite la corriger.

Une fois que l'auteur a fini de tester la résolution des problèmes, il remplit un questionnaire portant sur le fonctionnement de AMBRE-KB : l'élicitation des différents types de connaissances, les interfaces proposées, l'assistance fournie par le système ainsi que le feedback, etc., ce qui conclut la séance. Les questionnaires (cf. Annexes) présentés aux auteurs contenaient 31 questions réparties en quatre grandes catégories :

— **Profil de l'auteur et domaine testé (2 questions)**.

— **Elicitation des différentes connaissances (17 questions)** : définition du vocabulaire, description des problèmes, élicitation des connaissances de classification, élicitation des connaissances de reformulation, élicitation des connaissances de résolution. Les auteurs disposaient de 10 questions, avec 3 degrés de compréhension pour donner leur impression : « *j'ai compris dès le début* », « *j'ai compris vers la fin* », et « *je n'ai pas compris* » ; 2 questions à réponses directes (*oui* ou *non*) avec la possibilité de commenter leurs réponses lorsqu'elles sont négatives ; 5 questions ouvertes leur permettant de s'exprimer sur les interfaces proposées et de proposer des suggestions d'amélioration.

— **Principes du processus interactif d'acquisition de connaissances** mis en œuvre dans AMBRE-KB. Les auteurs disposaient de 7 questions pour juger l'as-

sistance fournie par AMBRE-KB : 4 questions à choix multiples et 3 questions leur permettant de s'exprimer sur l'utilité et l'efficacité de l'assistance et de proposer des suggestions.

— **Évaluation générale de AMBRE-KB**. Les auteurs disposaient de 5 questions pour évaluer le fonctionnement général de AMBRE-KB : 2 questions à choix multiples et 3 questions ouvertes leur permettant de s'exprimer et de faire des suggestions.

6.1.2.3 Critères d'évaluation

Est-ce que AMBRE-KB permet d'éliciter les connaissances nécessaires à l'EIAH pour résoudre des problèmes ?

L'objectif de cette partie est de tester si AMBRE-KB permet d'éliciter toutes les connaissances nécessaires à l'EIAH pour résoudre des problèmes. Pour ce faire les facettes suivantes ont été identifiées : *définition du vocabulaire du domaine choisi, description des problèmes à résoudre, élicitation des connaissances de classification, élicitation des connaissances de reformulation et élicitation des connaissances de résolution.* Pour chacune d'elles nous avons considéré les éléments suivants :

— **Point à analyser** : est-ce-que AMBRE-KB permet à l'auteur d'expliciter toutes les connaissances nécessaires à chaque facette pour le domaine qu'il aura choisi ?

— **Méthodologie envisagée** : élicitation des connaissances avec AMBRE-KB par l'auteur, puis test de la résolution des problèmes avec le résolveur.

— **Condition de validation** : notre hypothèse est qu'AMBRE-KB permet d'éliciter toutes les connaissances nécessaires pour chaque facette à évaluer.

Le **Tableau 6.2** montre un exemple des critères d'évaluation(cf. Annexes) du vocabulaire pour un domaine donné.

Facette à évaluer	Définition du vocabulaire pour le domaine choisi.
Point à analyser	Est-ce que AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi ?
Méthodologie envisagée	Définition du vocabulaire de son domaine par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire.
Condition de validation	AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi.

Tableau 6.2 – Critère retenu pour évaluer le vocabulaire d'un domaine.

Afin d'évaluer les principes du processus interactif d'acquisition des connaissances mise en œuvre dans AMBRE-KB, nous avons identifié trois facettes à évaluer : flexibilité, guidage et contrôle des connaissances pendant l'élicitation des connaissances, puis feedback à la fin du processus d'élicitation. Pour chacune de ces facettes, nous avons considéré les éléments suivants :

— **Point à analyser** : est-ce que l'aide fournie est utile pour l'auteur ?

— **Méthodologie envisagée** : élicitation des connaissances avec AMBRE-KB par l'auteur, puis réponse aux questions concernant l'aide dans le questionnaire.

— **Condition de validation** : l'aide est utile et a aidé l'auteur.

Le **Tableau 6.3** montre un exemple des critères d'évaluation de la flexibilité offerte par AMBRE-KB lors de la définition des connaissances.

Facette à évaluer	Flexibilité offerte par AMBRE-KB.
Point à analyser	Est-ce que la flexibilité fournie à l'auteur est utile durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire.
Condition de validation	La flexibilité proposée dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation.

Tableau 6.3 – Critères retenus pour évaluer la flexibilité de AMBRE-KB.

1.2.2.2. Est-ce que les interfaces proposées dans AMBRE-KB sont assez intuitives ?

L'objectif dans cette partie est de tester la satisfaction de l'auteur : est-ce-que les interfaces proposées sont assez intuitives ? Nous optons pour une évaluation empirique qui consiste à observer l'auteur lors de l'utilisation du logiciel, et à recueillir ses impressions à travers un questionnaire.

À l'image de l'évaluation décrite en section 1.2.2, différentes facettes ont également été identifiées pour cette évaluation : *définition du vocabulaire du domaine choisi*, *description des problèmes à résoudre*, *élicitation des connaissances de classification*, *élicitation des connaissances de reformulation* et *élicitation des connaissances de résolution*. Pour chacune d'elles, nous avons ainsi considéré les éléments suivants :

— **Point à analyser** : ergonomie des interfaces proposées.

— **Méthodologie envisagée** : élicitation des connaissances avec AMBRE-KB par l'auteur, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)

— **Condition de validation** : l'auteur a pu définir le type de connaissances sans aide extérieure.

Le **Tableau 6.4** montre par exemple les critères d'évaluation de la fonctionnalité de définition du vocabulaire pour un domaine donné.

Facette à évaluer	Définition du vocabulaire pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour la définition du vocabulaire
Méthodologie envisagée	Définition du vocabulaire par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir le vocabulaire sans aide extérieure

Tableau 6.4 – Critères d'évaluation de l'ergonomie de l'interface permettant de définir le vocabulaire

6.1.2.4 Résultats

Quatre domaines ont été testés, mais 5 sujets ont participé à l'expérimentation. L'un d'entre eux a testé deux domaines (*Pluriel des noms* et *Formation des adverbes en "ment"*) et les deux autres domaines ont été testés chacun par deux sujets (cf. **Tableau 6.5**).

	Domaines			
	Équation	Calcul d'aire	Pluriel des noms	Adverbes en "ment"
Nombres de sujets	2	2	1	1

Tableau 6.5 – Domaines testés

Nous présentons d'abord les résultats des domaines testés par deux sujets à la fois, nous terminons par les autres domaines testés par le même sujet.

Le **Tableau 6.6** décrit le nombre de connaissances produites par des deux sujets pour la résolution des équations.

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	4	3
Classes	6	7
Règles	5	5
Techniques	4	4
Problèmes créés	20	12
% Problèmes résolus au total sans le feedback	75%	41%
% Problèmes résolus au total avec le feedback	100%	100%

Tableau 6.6 – Résultats des expérimentations pour les équations par les sujets 1 et 2

Pour la résolution des équations, les deux sujets ont défini des connaissances différentes : ils ont proposé des arbres de classification différents, ainsi que des vocabulaires différents. Le fait que les vocabulaires soient différents explique le fait que les problèmes à résoudre, les règles et les techniques de résolution ne sont pas définies de la même façon.

Les **Tableaux 18** et **6.8** par exemple montrent le vocabulaire défini respectivement par le sujet 1 et le sujet 2. Le système a pu résoudre 75% des problèmes définis par le sujet 1 sans résolveur, puis 100% en utilisant le feedback. Pour le sujet 2, le système a pu résoudre 41% sans utiliser le feedback puis 100% en utilisant le résolveur.

Types objets	**Caractéristiques objets**
Équation	Nom_inconnu –>chaine de caractères Valeur_inconnu–>nombre entier Liste_termes–>type énuméré(deux_termes,trois_termes)
Deux_termes	Premier_terme–>terme_inconnu Deuxième_terme –>terme_constante
Terme_inconnu	coefficient–>nombre entier
Terme_constante	valeur_constante–>nombre entier

Tableau 6.7 – Vocabulaire défini par le sujet 1.

Types objets	**Caractéristiques objets**
Équation	Terme_1–>type énuméré(terme_inconnu,terme_constante) Terme_2–>type énuméré(terme_inconnu,terme_constante) Terme_3–>type énuméré(terme_inconnu,terme_constante)
Terme_inconnu	Coefficient–>nombre entier Nom_variable–>chaine de caractères Degré–>nombre entier Valeur–>nombre entier
Terme_constante	valeur_constante –> nombre entier

Tableau 6.8 – Vocabulaire défini par le sujet 2.

Le **Tableau 22** décrit le nombre de connaissances produites par deux sujets pour le calcul d'aire en géométrie.

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	5	8
Classes	12	12
Règles	11	11
Techniques	7	7
Problèmes créés	28	18
% Problèmes résolus au total sans le feedback	57%	61%
% Problèmes résolus au total avec le feedback	85%	89%

Tableau 6.9 – Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2

Pour le calcul d'aire en géométrie, même si les sujets ont défini des arbres de classification avec le même nombre de classes, les classifications définies sont différentes.

Les **Figures 6.1** et **6.2** représentent une partie des arbres de classifications définis par les deux auteurs :

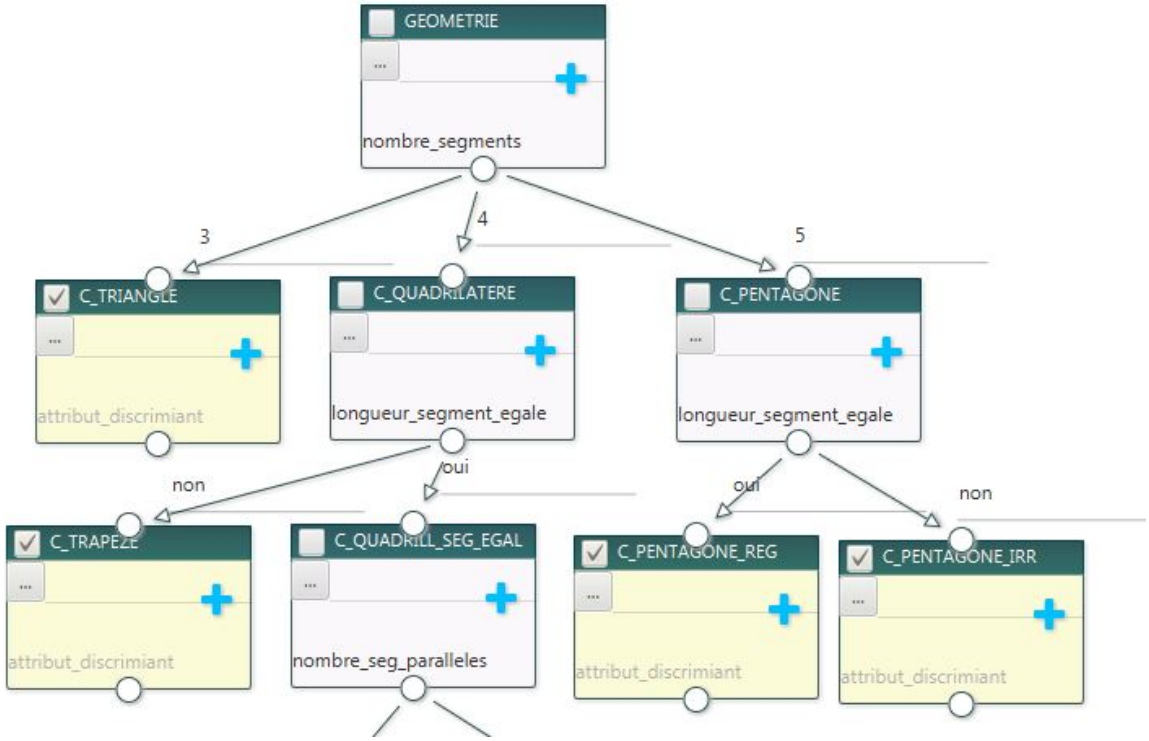

FIGURE 6.1 – Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie

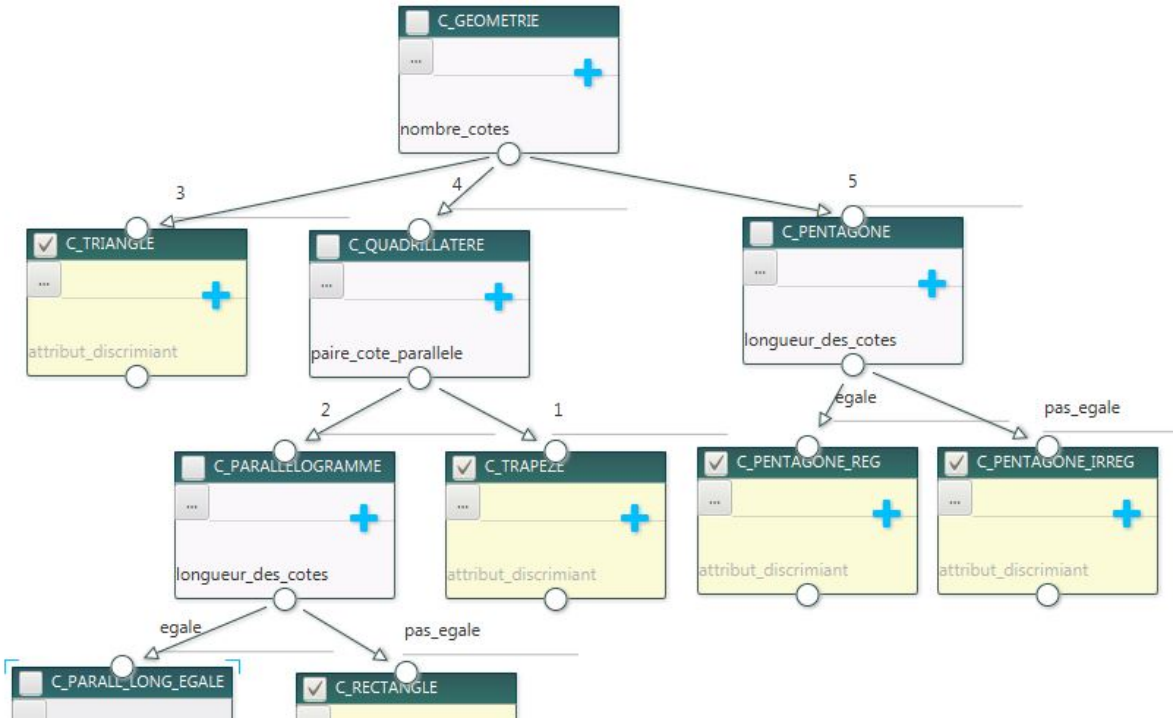

FIGURE 6.2 – Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie

Les sujets ont également défini des vocabulaires différents, et donc des règles et techniques de résolution différentes. Pour le sujet 1, le système a pu résoudre 57% des problèmes définis sans utiliser le feedback, et 85% en utilisant le feedback. Les 15% de problèmes qui ont échoué appartenaient à la même classe. L'auteur n'a pas pu tester leur résolution parce qu'il a défini au dernier moment une nouvelle fonction dont il avait besoin pour décrire la technique associée à cette classe. Comme le résolveur ne disposait pas de la version Prolog de cette fonction, la résolution de ces problèmes a achoué. Pour rappel, lors de la première étape de l'expérimentation, l'auteur doit donner la description de toutes les fonctions dont il aura besoin, de sorte que l'on puisse définir une version Prolog utilisable par le résolveur lors de la résolution des problèmes. Pour le sujet 2, le système a pu résoudre 61% des problèmes sans le feedback, et 89% avec le feedback. 11% de problèmes ont échoué parce que le système n'arrivait pas à trouver de règles lui permettant de conclure sur la valeur d'un attribut discriminant, mais comme l'auteur n'a pas corrigé l'erreur, la résolution de ces problèmes a échoué.

Le tableau **6.10** résume les connaissances décrites pour le pluriel des noms et la formation des adverbes en "ment" testés par le même sujet.

	Pluriel	Adverbes
vocabulaire (types d'objets)	1	1
Classes	13	17
Règles	9	16
Techniques	8	11
Problèmes créés	40	25
% Problèmes résolus au total sans le feedback	66%	87%
% Problèmes résolus au total avec le feedback	90%	100%

Tableau 6.10 – Résultats des expérimentations pour le pluriel des noms et la formation des adverbes en "ment" testés par le même sujet

Le pluriel des noms et la formation des adverbes ont été testés par le même sujet. Il a d'abord commencé par la formation des adverbes et a terminé par le pluriel des noms. Pour le premier domaine, le système a pu résoudre 52% des problèmes sans utiliser le feedback, puis 80% en utilisant le résolveur. Pour les 20% autres problèmes, le sujet a affirmé qu'il avait compris les messages d'erreur mais il a volontairement décidé de ne pas les corriger et de s'en tenir aux problèmes résolus.

Pour le pluriel des noms, le système a pu résoudre 87% des problèmes sans utiliser le feedback, mais une fois qu'il a utilisé le feedback, il a pu résoudre le reste des problèmes.

6.1.2.5 Analyse des résultats

Les 5 auteurs ont affirmé avoir pu définir toutes les connaissances nécessaires à l'EIAH pour résoudre les problèmes du domaine qu'ils avaient choisi. Ils ont également affirmé que les interfaces leur convenaient dans l'ensemble et étaient faciles à utiliser. Toutefois, l'un d'entre eux a quand même affirmé avoir rencontré quelques difficultés au début de l'élicitation des connaissances de reformulation car, selon lui, les interfaces permettant de définir les règles ne sont pas très intuitives au début, mais une fois le principe compris il devient facile de définir les connaissances de reformulation.

Concernant la définition du vocabulaire, deux des auteurs ont observé une différence entre le vocabulaire qu'ils avaient défini sur papier au début de l'expérimentation et celui qu'ils ont finalement explicité en utilisant l'outil. Cela s'explique par le fait que AMBRE-KB ne leur permettait pas de représenter les connaissances telles qu'ils les avaient explicitées. Néanmoins, avec une petite modification de la version papier,

ils ont pu parvenir à définir un vocabulaire pour leur domaine avec les fonctionnalités déjà implémentées dans AMBRE-KB. Un des auteurs a affirmé que la spécification de la question à résoudre lors de la description d'un problème n'était pas très intuitive et qu'il préférerait avoir un menu explicite pour la définir.

Quatre des auteurs ont déclaré avoir utilisé l'aide proposée. Et tous ont jugé que l'assistance proposée les a aidés durant le processus d'élicitation. Ils ont particulièrement apprécié le feedback proposé par l'outil et ont affirmé son aide pour mieux comprendre le fonctionnement de l'outil. Néanmoins, l'un d'entre eux a déclaré que cette assistance pourrait être enrichie en augmentant l'interactivité du système avec des pop-ups plus fréquentes durant le processus.

6.2 Bilan des expérimentations

Nous avons mené deux expérimentations avec 8 sujets pour mettre à l'essai AMBRE-KB. La première expérimentation a été réalisé avec 3 sujets et a porté un domaine déjà existant (EIAH AMBRE existant au moment de l'expérimentation) et deux nouveaux domaines : le calcul d'aire en géométrie et la pâtisserie. L'objectif de cette expérimentation était de mettre à l'essai AMBRE-KB afin de tester si AMBRE-KB était capable d'acquérir les connaissances nécessaires à un EIAH AMBRE pour résoudre des problèmes. Les résultats de cette expérimentation n'ont pas permis de valider notre hypothèse de départ, mais elle s'est conclue avec des remarques intéressantes qui ont permis d'étendre l'outil AMBRE-KB pour améliorer l'interactivité du système, et intégrer des fonctionnalités permettant à l'auteur d'avoir un feedback sur le processus d'élicitation et la résolution des problèmes.

Avec la nouvelle version de AMBRE-KB, l'outil a été mis à l'essai dans une seconde expérimentation. Cette-ci a été réalisée par 5 sujets et quatre domaines ont été testés : la résolution des équations du second degré (testé par deux sujets), le calcul d'aire en géométrie (testé par deux sujets), le pluriel des noms et la formation des adverbes en "ment", testés par le même sujet. De cette expérimentation, nous retenons les conclusions suivantes :

— Pour un domaine donné, AMBRE-KB permet de générer les modèles de connaissances qui permettent au résolveur de résoudre les problèmes.

— Les interfaces proposées sont dans l'ensemble faciles à utiliser.

— Les auteurs ont apprécié l'assistance fournie, en particulier le feedback sur le processus d'élicitation.

— Pour tous les domaines, au moins 40% des problèmes ont pu être résolus par sans utiliser le feedback, et au moins 80% en utilisant le feedback (cf. **Tableau 6.11**).

	Équation		**Calcul d'aire**		**Pluriel**	**Adverbes**
	Sujet 1	Sujet 2	Sujet 1	Sujet 2	Sujet	Sujet
sans le feedback	75%	41%	57%	61%	66%	87%
avec le feedback	100%	100%	85%	89%	90%	100%

Tableau 6.11 – Domaines testés

Parmi les sujets de cette deuxième expérimentation, deux d'entre eux sont des informaticiens, et trois des non-informaticiens. Les premiers ont rencontré moins de difficulté pour comprendre les principes mis en œuvre dans l'outil. Cependant, tous les sujets ont pu, sans programmer, éliciter les connaissances nécessaires à l'EIAH pour résoudre les problèmes dans le domaine qu'ils avaient choisi. Nous pouvons donc en conclure que AMBRE-KB est apte à acquérir les connaissances nécessaires à un EIAH AMBRE pour résoudre les problèmes portant sur la méthode à enseigner. L'analyse des questionnaires nous permet également d'affirmer que les interfaces de AMBRE-KB sont dans l'ensemble assez intuitives et facile à appréhender. Celle permettant de définir des connaissances de reformulation pourrait cependant être améliorées, de même que celle destinée à la description des problèmes.

Le fait que les auteurs aient pu éliciter toutes les connaissances dont ils avaient besoin pour la méthode qu'ils souhaitaient enseigner permet également de conclure que les méta-modèles sous-jacents à AMBRE-KB sont suffisants, même s'il est possible qu'en utilisant AMBRE-KB pour d'autres domaines, l'on constate qu'il est nécessaire de les enrichir.

Conclusion générale

Dans cette dernière partie du manuscrit, nous présentons d'abord une synthèse de nos contributions. Nous discutons ensuite de ces contributions ainsi que des limites soulevées par ce travail. Nous terminons enfin en présentant les différentes perspectives de recherche qui pourraient être intéressantes pour la poursuite de ce travail.

Sommaire

6.3 Synthèse de nos travaux 157
6.4 Discussion 159
6.5 Perspectives 161

6.3 Synthèse de nos travaux

Dans cette thèse, nous nous sommes intéressés à la problématique de l'acquisition des connaissances dans le cadre de la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes dans des domaines spécifiques. L'objectif de notre travail a été de réduire le coût de conception de ces EIAH, dû en partie à l'élicitation des connaissances, mais également de permettre à des auteurs de pouvoir éliciter, sans programmer, les connaissances permettant à l'EIAH de résoudre des problèmes. Notre problématique s'est donc décomposée en deux points :

— **Comment permettre à un auteur d'éliciter, sans programmer, les connaissances permettant au système de résoudre des problèmes dans un domaine donné ?**

— **Comment accompagner l'auteur dans le processus d'élicitation ?**

Pour répondre à ces questions, notre première contribution a consisté à formaliser les différents types de connaissances qui interviennent dans la résolution d'un problème, dans le cadre d'un EIAH destiné à enseigner des méthodes. Pour cela, nous avons proposé des méta-modèles de connaissances. Ceux-ci sont décrits par des schémas XML et permettent de définir les connaissances à acquérir. Une fois les différents types de connaissances formalisés, nous avons ensuite proposé un processus interactif d'acquisition de ces connaissances. Ce processus, indépendamment du domaine, est fondé sur les méta-modèles et permet à un auteur de construire les modèles de connaissances spécifiques pour le domaine qu'il aura choisi.

Nous avons par la suite mis en œuvre ce processus en implémentant l'outil auteur AMBRE-KB. Cet outil permet non seulement d'acquérir les connaissances nécessaires à la résolution des problèmes dans un domaine, mais également d'assister l'auteur dans ce processus, en lui offrant une grande flexibilité lors de la définition des connaissances, en le guidant et en lui permettant de tester les connaissances durant le processus d'élicitation s'il le souhaite. Il offre également la possibilité à l'auteur d'avoir un feedback sur le processus d'élicitation en lui permettant de tester la résolution des problèmes. Dans le cas où la résolution d'un problème a réussi, le système lui affiche les différentes informations utilisées pour le résoudre. Ces informations sont notamment la classe du problème, les attributs calculés, les règles essayées, celles exécutées, les valeurs des différents attributs discriminants impliqués, la technique de résolution de résolution appliquée, ainsi que la solution du problème. Lorsque la résolution échoue, le système envoie un message

d'erreur afin de permettre à l'auteur de comprendre l'origine de son erreur et de pouvoir par la suite la corriger.

Enfin, nous avons évalué le processus d'acquisition des connaissances et sa mise en 'uvre dans AMBRE-KB en menant deux expérimentations. Ces expérimentations ont consisté dans un premier temps à vérifier que les modèles de connaissances générés par AMBRE-KB permettent à l'EIAH de résoudre des problèmes pour un domaine donné. Les résultats de cette expérimentation ont permis d'améliorer l'outil auteur, de sorte à pouvoir effectuer une deuxième expérimentation. Cette dernière a permis de mettre en évidence les points forts et les points faibles de AMBRE-KB. Ainsi, de cette expérience, nous retenons les conclusions suivantes :

— AMBRE-KB permet d'acquérir les connaissances nécessaires au résolveur pour résoudre les problèmes portant sur la méthode à enseigner.

— Les interfaces proposées sont assez intuitives et facile à utiliser dans l'ensemble, même si celles dédiées à l'élicitation des règles et à la description des problèmes peuvent être améliorées.

— L'assistance fournie durant le processus est utile. Le feedback proposé est particulièrement apprécié et contribue à faciliter la compréhension du fonctionnement de l'outil.

Les propositions effectuées dans le cadre de cette thèse et leur mise œuvre opérationnelle répondent donc à notre problématique générale. Nous restons convaincus de l'utilité du processus d'acquisition de connaissances proposé dans la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes. De plus, l'approche proposée pour la description des problèmes, et consistant à proposer un vocabulaire, qui est ensuite utilisé pour décrire des problèmes peut être réutilisé dans tout domaine où on a besoin de représenter des problèmes. En effet, nous partons du principe que dans un problème, on retrouve généralement les éléments descriptifs de l'énoncé, ainsi que la question à résoudre. Ce qui pourrait différencier les problèmes dans des domaines différents, ce sont les différents types de connaissances qu'on peut retrouver au niveau des éléments descriptifs de l'énoncé. Dans AMBRE, par exemple, dans les éléments descriptifs de l'énoncé, on retrouve les traits de surface du problème ; dans [84], les problèmes sont décrits en termes de variables, d'unités de mesures, de formules. Il s'agira donc, selon les domaines, de faire évoluer le méta-modèle du vocabulaire que nous avons proposé pour prendre en compte les types de connaissances nécessaires pour décrire les

problèmes de ce domaine.Nous soutenons également que le fait que les connaissances puissent être explicitées sans programmer, motivera plus d'auteurs (enseignants, professionnels, pédagogues plus experts) à davantage s'impliquer dans la conception de ce type de logiciels.

6.4 Discussion

Positionnement en Ingénierie des Connaissances
D'un point de vue Ingénierie des Connaissances, nous nous positionnons dans une démarche d'ingénierie dirigée par les modèles comme la méthodologie commonKADS. Cette méthodologie propose de construire un modèle conceptuel autour de cinq modèles génériques qui décrivent comment résoudre des problèmes très généraux tels que diagnostic et prédiction du comportement, évaluation, conception, configuration, planification, affectation et ordonnancement. Dans le domaine des EIAH, ces types de tâches très génériques ne sont pas forcément adaptés à ce que l'on cherche à enseigner. En effet, dans le domaine des EIAH, pour modéliser les connaissances du domaine, il ne s'agit pas de modéliser les connaissances de l'expert, mais les connaissances telle qu'elles devraient fonctionner chez l'apprenant après l'apprentissage. D'autre part, en EIAH, il faut modéliser d'autres types de connaissances, comme les connaissances de diagnostic des compétences de l'apprenant, ou les connaissances permettant de fournir des rétroactions ou de l'assistance. La démarche que nous proposons même si elle est similaire à celle de commonKADS s'applique à des problématiques plus spécifiques au champ des EIAH.

Positionnement par rapport aux outils auteurs existants en EIAH
Les Model Tracing Tutors développés à l'Université Carnegie Mellon de Pittsburgh sont les plus proches de notre problématique. En effet, même si les modèles sous-jacents sont très différents, l'approche centrée connaissances du projet AMBRE s'inscrit dans le même courant. En effet, les EIAH développés à Pittsburgh se fondent sur le modèle ACT-R qui considère que la tâche de résolution de problèmes s'effectue pas à pas, un élément de connaissance étant mobilisé à chaque pas de résolution, et que l'ensemble des connaissances du domaine peut s'exprimer sous forme de règles de production. Ce modèle assez restrictif limite les domaines sur lequel on peut concevoir des Model Tracing Tutors. L'enseignement de méthodes propose un modèle cognitif dans lequel l'apprenant raisonne globalement sur le problème à résoudre. C'est également un modèle restrictif

puisqu'il ne s'applique qu'aux domaines dans lesquels on peut établir une classification des problèmes et des techniques de résolution, mais il pourrait compléter l'offre de CTAT. Une des perspectives de cette thèse peut d'ailleurs consister à intégrer AMBRE-KB dans la plateforme CTAT. Comme le souligne Nathalie Guin [41], le succès des tuteurs intelligents et des outils auteurs CTAT permettant de créer les model tracing tutors renforce notre conviction qu'une approche centrée connaissances est souhaitable en EIAH.

Évaluation et validation des méta-modèles

Les méta-modèles de connaissances proposés n'ont pas été validés de manière formelle. Les méta-modèles portant sur la méthode à enseigner (méta-modèles des connaissances de classification, connaissances de reformulation et connaissances de résolution) ont été évalués en créant, pour chaque domaine testé, des modèles de connaissances fondés sur ces méta-modèles et utilisables par le résolveur pour tester la résolution des problèmes du domaine. Le fait que le résolveur puisse résoudre des problèmes en utilisant ces modèles assure donc leur cohérence. Nous ne prétendons pas que AMBRE-KB permet de représenter n'importe quel type de domaine, mais il permet d'acquérir les connaissances nécessaires à la conception d'EIAH AMBRE.

Positionnement par rapport à la méthodologie utilisée

Notre travail a impliqué des choix de formalisation et de méthode qui relèvent de prises de décisions de notre part. Nous souhaitions permettre à des auteurs de pouvoir concevoir eux-mêmes des EIAH dans les domaines qui les intéressent sans aucune compétence en programmation. L'approche que nous avons adoptée est de leur proposer des interfaces fondées sur les méta-modèles des connaissances à expliciter pour les assister durant tout le processus d'élicitation. D'autres approches ont préféré utiliser des techniques d'IA comme l'apprentissage automatique pour découvrir des connaissances. Même si les techniques d'IA peuvent s'avérer efficaces, notre objectif était plus d'impliquer les enseignants dans la conception de ces EIAH. Il serait donc nécessaire que les enseignants puissent comprendre de tels processus d'apprentissage automatique, de sorte qu'ils puissent valider les connaissances à acquérir par l'apprenant. De plus, cela ne faciliterait pas la confiance et l'appropriation des EIAH par les enseignants, qui pour la plupart ont encore une réaction méfiante vis-à-vis de ces outils. Les techniques d'apprentissage automatiques peuvent néanmoins s'avérer efficaces si elles sont intégrées à un processus interactif d'élicitation des connaissances, comme nous l'avons vu dans la

thèse de Sébastien Lallé [49].

6.5 Perspectives

Ce travail s'est focalisé sur l'acquisition des connaissances permettant la résolution des problèmes pour un domaine donné. L'une de nos perspectives à court terme est d'intégrer dans AMBRE-KB l'élicitation des connaissances destinées à accompagner l'apprenant, en lui fournissant aide, diagnostic et explications sur ses erreurs. Nous avons regroupé ces connaissances en trois grandes catégories : les connaissances d'aide, les connaissances de diagnostic, et les connaissances d'explications sur les erreurs. Pour chacune de ces connaissances, nous avons déjà proposé des méta-modèles qui permettent de définir la forme des connaissances à acquérir. Cependant, ces méta-modèles n'ont pas été testés dans le cadre de ce travail, puisque nous nous sommes focalisés sur les connaissances de la méthode dans le cadre du développement de AMBRE-KB. Néanmoins, l'élicitation desdites connaissances est déjà prise en compte dans le processus d'acquisition proposé. Il reste donc à valider ces méta-modèles, et à intégrer dans AMBRE-KB les fonctionnalités qui permettront d'éliciter ces connaissances. Il s'agira donc dans un premier temps de proposer des interfaces qui permettront à l'auteur de pouvoir éliciter ces connaissances, et dans un second temps, comme pour les connaissances de la méthode, de proposer à l'auteur des fonctionnalités d'assistance pour faciliter le processus d'élicitation, mais également des moyens de tester ces connaissances, une fois explicitées.

Les connaissances destinées à guider l'apprenant sont fortement dépendantes des connaissances sur la méthode à enseigner, mais également des étapes de résolution. Dans le processus d'acquisition de connaissances que nous avons proposé, pour gérer cette articulation entre la conception de l'interface de l'EIAH et l'élicitation de ces connaissances, nous avons proposé à l'auteur de concevoir l'interface et en particulier les différentes tâches de résolution des problèmes, le développement de cette interface étant ensuite réalisé par un informaticien. Une perspective pourrait être d'intégrer dans AMBRE-KB la possibilité pour l'auteur de construire lui-même cette interface en utilisant des composants graphiques via le drag and drop. L'approche utilisée dans CTAT avec les example-tracing-tutors [4], qui permet à un enseignant de concevoir l'interface par du drag and drop, pourrait être intéressante dans le cadre de ce travail.

Dans AMBRE, pour le moment, les connaissances représentées dans les règles sont traitées comme des conjonctions (ET). Il serait également intéressant de faire évoluer le résolveur de sorte qu'il puisse, par exemple, raisonner sur des connaissances représentant des disjonctions (OU) au niveau des prémisses des règles. Dans la même lancée, au niveau de l'élicitation des connaissances, il serait intéressant aussi de permettre à l'auteur de pouvoir définir des contraintes, des connaissances de comparaison (par exemple de plus que, de moins que), des connaissances contraires, etc.

En outre, dans la version actuelle de AMBRE-KB, avant d'entamer l'élicitation des connaissances, l'auteur doit définir la spécification de toutes les fonctions dont il a besoin pour décrire des règles et des techniques de résolution ; celles-ci sont ensuite traduites en Prolog pour permettre au résolveur de pouvoir les utiliser pour la résolution des problèmes ; cela peut constituer un frein à l'auteur car il a aura toujours besoin d'un expert Prolog pour pouvoir tester la résolution des problèmes qu'il a définis. Il a sera certes très difficile d'anticiper sur cela puisque c'est l'auteur qui spécifie les fonctions dont il a besoin pour la méthode qu'il souhaite enseigner. Néanmoins, intégrer dans AMBRE-KB un ensemble de fonctions utilitaires (par exemple des fonctions mathématiques permettant de calculer la somme, multiplication, division de nombres, etc, ou encore des fonctions permettant de concaténer/supprimer des chaînes de caractères) pourrait constituer une première étape pour faciliter le processus d'élicitation pour certains domaines.

Pour poursuivre le travail lié à l'assistance apportée à l'auteur, une piste à explorer est celle de la généralisation de donnaissances à partir d'exemples. On pourrait ainsi envisager une acquisition interactive des connaissances, l'auteur définissant des exemples à la place de connaissances abstraites, le système lui proposant ensuite une généralisation des connaissances qu'il pourrait accepter ou modifier. Par exemple, pour la définition des connaissances de classification, cela faciliterait une partie du travail de l'auteur en lui permettant de construire la hiérarchie à partir d'un ensemble de classes opérationnelles. Dans ce cas, l'auteur définirait uniquement des cas (des problèmes avec leur technique de résolution), préciserait les attributs discriminants et ceux qui instancient le problème par rapport à la classe, et le système construirait ensuite la hiérarchie de classes. Ce procédé est un premier exemple de généralisation de connaissances qui pourrait être mis en place. Un autre exemple peut consister à s'inspirer de l'approche utilisée dans SimStudent [51] qui repose sur la généralisation de connaissances abstraites à partir d'exemples.

Annexes

Connaissances décrites pour les Domaines testés

Nous avons décrit dans le chapitre 6 les différents domaines testés pour évaluer AMBRE-KB. Nous détaillons ici les connaissances décrites par les sujets.

Pluriel des noms terminés par "al","ail","ou", "eu", "eau" en français

Profil du sujet : Infographiste (non programmeur).

Méthode à enseigner : l'expert à choisi d'enseigner la formation du pluriel des noms terminés par "al", "ail", "ou", "eu, "au", "eau" en français. Le **Tableau 3.2.3** résume la méthode à enseigner.

Le sujet a défini :

- 1 type d'objet pour le vocabulaire
- 13 classes, dont 8 sont opérationnelles
- 9 règles
- 8 techniques de résolution
- 24 problèmes à résoudre
- 11 fonctions intervenant dans la définition des règles et des techniques de résolution
    - **Concatener_chaine** : permettant de concaténer des chaines de caractères

— **Pluriel_ail_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **Pluriel_al_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **Pluriel_ou_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **Pluriel_eu_au_eau_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **liste_exception_nom_ail** : renvoie la liste des mots terminés en « ail » qui sont des exceptions

— **liste_exception_nom_al** : renvoie la liste des mots terminés en « al » qui sont des exceptions

— **liste_exception_nom_ou** : renvoie la liste des mots terminés en « ou» qui sont des exceptions

— **liste_exception_nom_eu** : renvoie la liste des mots terminés en « eu » qui sont des exceptions

— **liste_exception_nom_au** : renvoie la liste des mots terminés en « au » qui sont des exceptions

— **liste_exception_nom_eau** : renvoie la liste des mots terminés en « ail » qui sont des exceptions

Vocabulaire proposé pour le domaine

Types objets	**Caractéristiques objets**
nom	identifiant–>chaine de caractères fin_nom–>chaine de caractères pluriel_nom–>chaine de caractères

Tableau 12 – vocabulaire proposé pour le pluriel noms terminés par "al", "ail", "ou", "eu, "au", "eau" en français

Définition du vocabulaire en utilisant AMBRE-KB

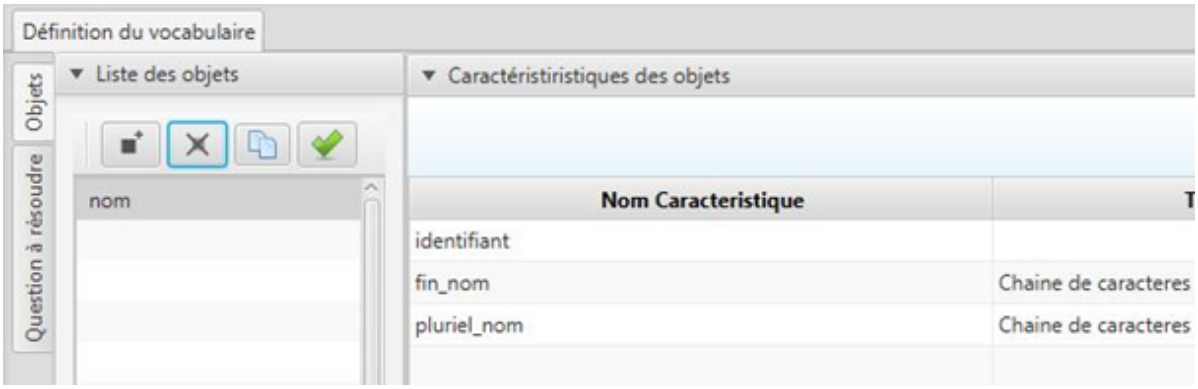

FIGURE 3 – Définiiton du vocabulaire pour le pluriel des noms avec AMBRE-KB

Problèmes à résoudre

Exemples de problèmes définis par le sujet

pluriel_detail : "Donner le pluriel du nom : (détail).
pluriel_bail : "Donner le pluriel du nom : (bail).
pluriel_travail : "Donner le pluriel du nom : (travail).
pluriel_bal : "Donner le pluriel du nom : (bal).
pluriel_chacal : "Donner le pluriel du nom : (chacal).
pluriel_jeu : "Donner le pluriel du nom : (jeu).
pluriel_bureau : "Donner le pluriel du nom : (bureau).
pluriel_joyau : "Donner le pluriel du nom : (joyau).
pluriel_pou : "Donner le pluriel du nom : (pou).

Exemple de définition d'un problème en utilisant AMBRE-KB

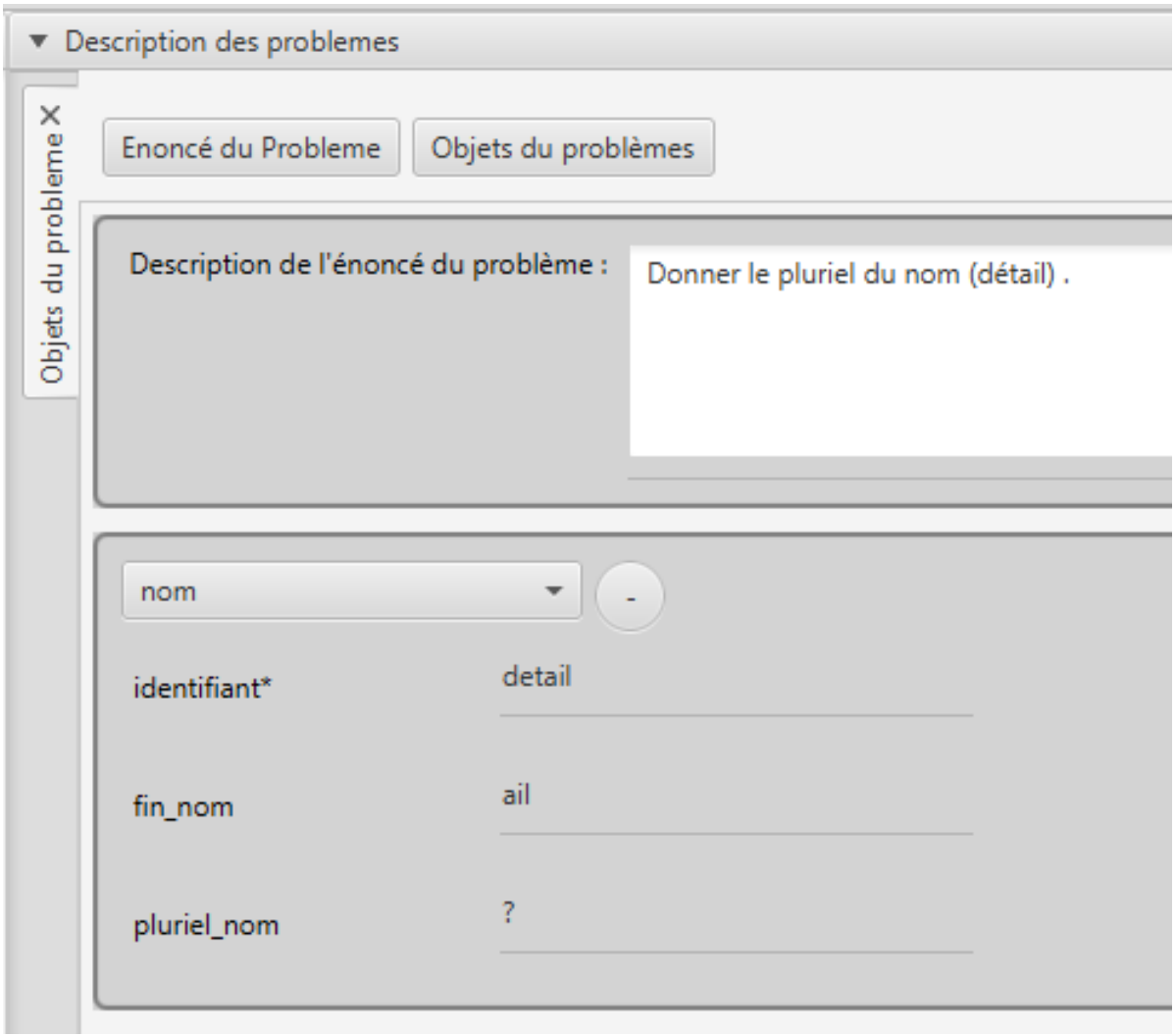

FIGURE 4 – Définiton du problème pluriel_detail avec AMBRE-KB

Exemple de modèles générés de connaissances par AMBRE-KB

```
/**Modèle descriptif représentant le problème pluriel_detail**/
fait(probleme(pluriel_bail)). /**soit le problème pluriel_bail**/
fait(enonce_naturel(pluriel_bail,'Donner le pluriel du nom (bail).')).
/**soit l'énoncé du problème pluriel_bail**/
fait(est_un(bail,nom)).**soit bail un nom**/
fait(fin_nom(bail,ail)). /**soit ail la fin du nom bail**/
fait(pluriel_nom(bail,Question)). /**soit Question le pluriel du nom bail**/
fait(inconnu(pluriel_bail,Question)). /**la question a résoudre consiste
/**à trouver la valeur de Question**/
```

Tableau 13 – Modèle descriptif du problème pluriel_bail généré par AMBRE-KB

Arbre de classification

.0.0.1 Définition de l'arbre de classification avec AMBRE-KB

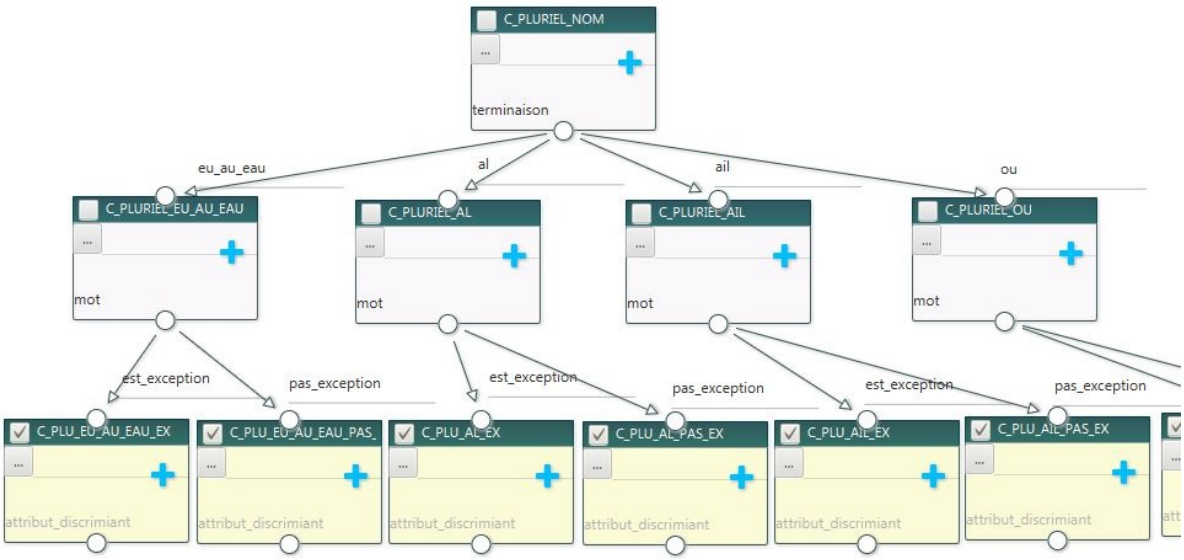

FIGURE 5 – Une partie de l'arbre de classification pour le pluriel des noms en "al", "ail", "ou", "eu, "au", "eau"

Modèle décrivant les connaissances de classification, généré par AMBRE-KB

```
racine_arbre(c_pluriel_nom). /*classe recine*/
statut(c_pluriel_nom,non_operationnelle). /* statut de la classe*/
test_fils(c_pluriel_nom,P,terminaison(P,_)).
attribut(c_pluriel_nom,P,terminaison(P,_),terminaison,[regle_terminaison]).
attribut(c_pluriel_nom,P,mot(P,_),mot,
        [regle_terminaison,regle_ail_exception,
        regle_ail_pas_exception,regle_al_exception,regle_al_pas_exception,
        regle_eu_au_eau_exception,regle_eu_au_eau_pas_exception,
regle_ou_exception,regle_ou_pas_exception]).
valeur_fils(c_pluriel_nom,P,c_pluriel_eu_au_eau) :-
        eval_prem(terminaison(P,[eu_au_eau])).
valeur_fils(c_pluriel_nom,P,c_pluriel_al) :-
        eval_prem(terminaison(P,[al])).
valeur_fils(c_pluriel_nom,P,c_pluriel_ail) :-
        eval_prem(terminaison(P,[ail])).
valeur_fils(c_pluriel_nom,P,c_pluriel_ou) :-
        eval_prem(terminaison(P,[ou])).
infos(c_pluriel_nom) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_pluriel_eu_au_eau ***/
pere(c_pluriel_eu_au_eau,c_pluriel_nom).
statut(c_pluriel_eu_au_eau,non_operationnelle).
        test_fils(c_pluriel_eu_au_eau,P,mot(P,_)).
valeur_fils(c_pluriel_eu_au_eau,P,c_c_plu_eu_au_eau_ex) :-
eval_prem(mot(P,[est_exception])).
valeur_fils(c_pluriel_eu_au_eau,P,c_c_plu_eu_au_eau_pas_ex) :-
        eval_prem(mot(P,[pas_exception])).
infos(c_pluriel_eu_au_eau) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_pluriel_al ***/
pere(c_pluriel_al,c_pluriel_nom).
statut(c_pluriel_al,non_operationnelle).
test_fils(c_pluriel_al,P,mot(P,_)).
valeur_fils(c_pluriel_al,P,c_c_plu_al_ex) :-
        eval_prem(mot(P,[est_exception])).
```

```
valeur_fils(c_pluriel_al,P,c_c_plu_al_pas_ex) :-
        eval_prem(mot(P,[pas_exception])).
infos(c_pluriel_al) :-
        writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_AL_EX ***/
pere(c_c_plu_al_ex,c_pluriel_al).
statut(c_c_plu_al_ex,operationnelle).
test_fils(c_c_plu_al_ex,P,(P,_)).
infos(c_c_plu_al_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_AL_PAS_EX ***/
pere(c_c_plu_al_pas_ex,c_pluriel_al).
statut(c_c_plu_al_pas_ex,operationnelle).
test_fils(c_c_plu_al_pas_ex,P,(P,_)).
/*** La Classe c_pluriel_ail ***/
pere(c_pluriel_ail,c_pluriel_nom).
statut(c_pluriel_ail,non_operationnelle).
test_fils(c_pluriel_ail,P,mot(P,_)).
attribut(c_pluriel_ail,P,mot(P,_),mot,
        [regle_ail_pas_exception,regle_ail_exception]).
valeur_fils(c_pluriel_ail,P,c_plu_ail_pas_ex) :-
        eval_prem(mot(P,[pas_exception])).
valeur_fils(c_pluriel_ail,P,c_plu_ail_ex) :-
        eval_prem(mot(P,[est_exception])).
/*** La Classe C_C_PLU_EU_AU_EAU_EX ***/
pere(c_c_plu_eu_au_eau_ex,c_pluriel_eu_au_eau).
statut(c_c_plu_eu_au_eau_ex,operationnelle).
test_fils(c_c_plu_eu_au_eau_ex,P,(P,_)).
/*** La Classe C_C_PLU_EU_AU_EAU_PAS_EX ***/
pere(c_c_plu_eu_au_eau_pas_ex,c_pluriel_eu_au_eau).
statut(c_c_plu_eu_au_eau_pas_ex,operationnelle).
test_fils(c_c_plu_eu_au_eau_pas_ex,P,(P,_)).
infos(c_c_plu_eu_au_eau_pas_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_pluriel_ou ***/
```

```
pere(c_pluriel_ou,c_pluriel_nom).
statut(c_pluriel_ou,non_operationnelle).
test_fils(c_pluriel_ou,P,mot(P,_)).
valeur_fils(c_pluriel_ou,P,c_c_plu_ou_ex) :-
        eval_prem(mot(P,[est_exception])).
valeur_fils(c_pluriel_ou,P,c_c_plu_pas_ex_) :-
        eval_prem(mot(P,[pas_exception])).
infos(c_pluriel_ou) :-
        writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_OU_EX ***/
pere(c_c_plu_ou_ex,c_pluriel_ou).
statut(c_c_plu_ou_ex,operationnelle).
test_fils(c_c_plu_ou_ex,P,(P,_)).
infos(c_c_plu_ou_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_plu_ail_pas_ex ***/
pere(c_plu_ail_pas_ex,c_pluriel_ail).
statut(c_plu_ail_pas_ex,operationnelle).
test_fils(c_plu_ail_pas_ex,P,(P,_)).
infos(c_plu_ail_pas_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_plu_ail_ex ***/
pere(c_plu_ail_ex,c_pluriel_ail).
statut(c_plu_ail_ex,operationnelle).
test_fils(c_plu_ail_ex,P,(P,_)).
infos(c_plu_ail_ex) :-writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_PAS_EX ***/
pere(c_c_plu_pas_ex_,c_pluriel_ou).
statut(c_c_plu_pas_ex_,operationnelle).
test_fils(c_c_plu_pas_ex_,P,(P,_)).
infos(c_c_plu_pas_ex_) :-
        writeln('aucune information sur le probleme').
/*** regle de propagation des attributs de mere en fille ***/
attribut(Classe,P,Att,Pred,Liste) :-
        pere(Classe,Pere),
```

attribut(Pere,P,Att,Pred,Liste).

Règles de reformulation

Les règles suivantes permettent par exemple de conclure sur la valeur des attributs : terminaison et mot (cf. **Figure 6**).

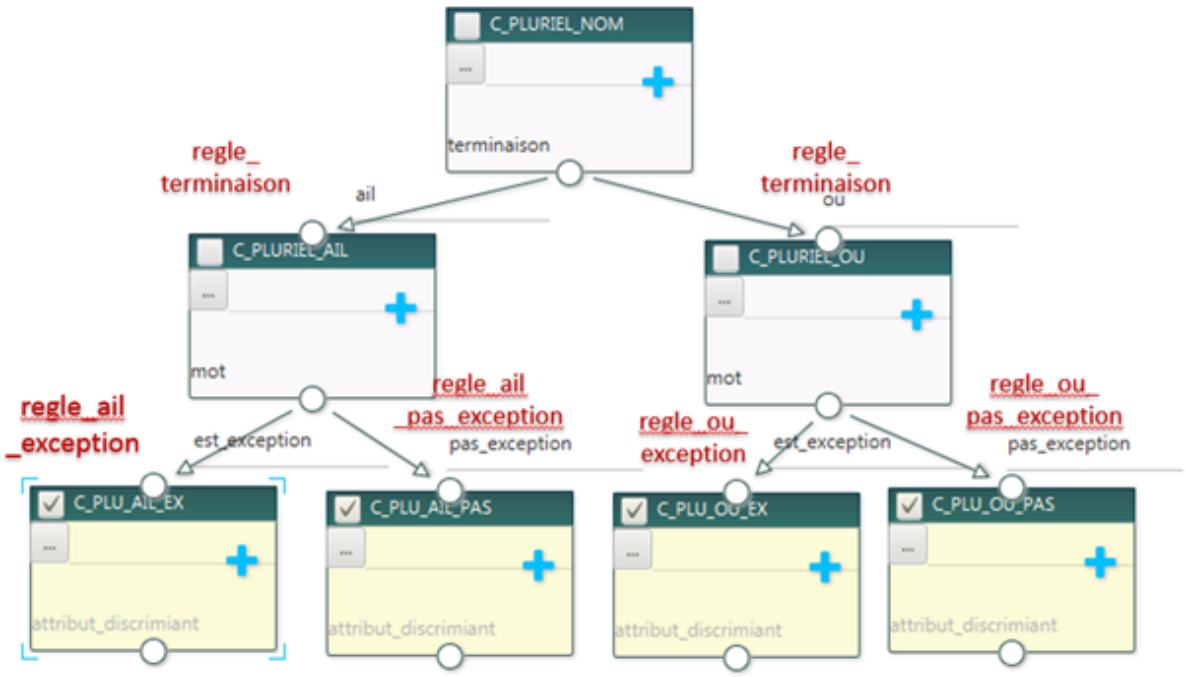

FIGURE 6 – une aprtied e l'arbre de classification

Exemple de règles permettant de conclure sur la valeur de l'attribut *terminaison*

regle_terminaison
Si dans un problème, il existe :
- un nom M
- la fin du nom est F
Alors la valeur de l'attribut *terminaison* est égale à *F*.

Exemples de règles permettant de conclure sur la valeur de l'attribut *mot*

regle_ail_exception
Si dans un problème, il existe :
- un nom M
- une liste L des exception des mots terminés par ail
- M appartient à la liste L
Alors la valeur de l'attribut *mot* est égale à *est_exception*.

regle_ail_pas_exception
Si dans un problème, il existe :
- un nom M
- une liste L des exception des mots terminés par ail
- M n'appartient pas à la liste L
Alors la valeur de l'attribut *mot* est égale à *pas_exception*.

Exemple de définition d'une règle en utilisant AMBRE-KB

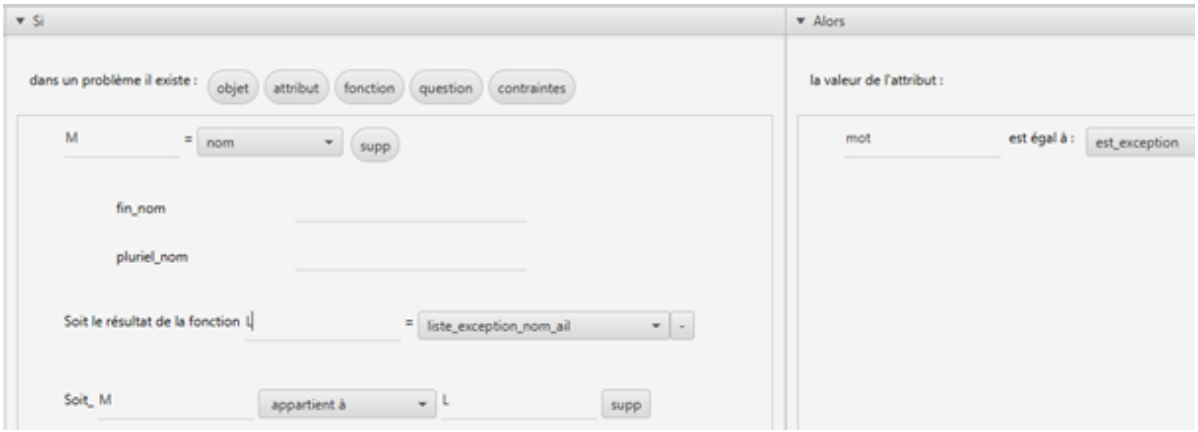

FIGURE 7 – Définition de la règle *regle_ail_exception* en utilisant AMBRE-KB

Exemple de modèles des connaissances de reformulation, généré par AMBRE-KB

/*** règle permettant de conclure sur la valeur de l'attribut terminaison ***/ regle(regle_terminaison) :-/*soit regle_terminaison nom de la règle*/ **si**([probleme(P),/*si dans un problème P*/ est_un(M,nom),/* M est un nom*/ fin_nom(M,F)]),/* T la fin du nom*/ alors([ajouter(terminaison(P,[F]))]). /*la valeur de l'attribut terminaison égale ou*/
/*** règle permettant de conclure sur la valeur de l'attribut mot ***/ regle(regle_ail_exception) :- /*soit regle_ail_exception nom de la règle*/ **si**([probleme(P), est_un(M,nom), eval_prolog(liste$_e$*xception*$_n$*om*$_a$*il*(*L*)), eval_prolog(member(M,L))]), **alors**([ajouter(mot(P,[est_exception]))]). /*la valeur de l'attribut mot égale est_exception*/
regle(regle_ail_pas_exception) :- **si**([probleme(P), est_un(M,nom), eval_prolog(liste$_e$*xception*$_n$*om*$_a$*il*(*L*)), /* L est la liste des exception en ail*/ eval_prolog(not member(M,L))]),/* M appartient à L*/ **alors**([ajouter(mot(P,[pas_exception]))]). /*la valeur de l'attribut mot égale pas_exception*/

Tableau 15 – Critères retenus pour évaluer le vocabulaire d'un domaine

Techniques de résolution

Quelques techniques définies par l'expert

Technique associée à la classe *c_plu_ail_pas_ex*

- soit un nom M
- soit Pluriel le nom obtenu en ajoutant 's' à M
- le pluriel du mot M est égal à Pluriel

Technique associée à la classe *c_plu_ail_ex*

- soit un nom M

- soit pluriel le résultat de la fonction permettant de former le pluriel des mot en ail qui sont des exception - le pluriel du mot M est égal à Pluriel

.0.0.2 Exemple de définition d'une technique de résolution en utlisant AMBRE-KB

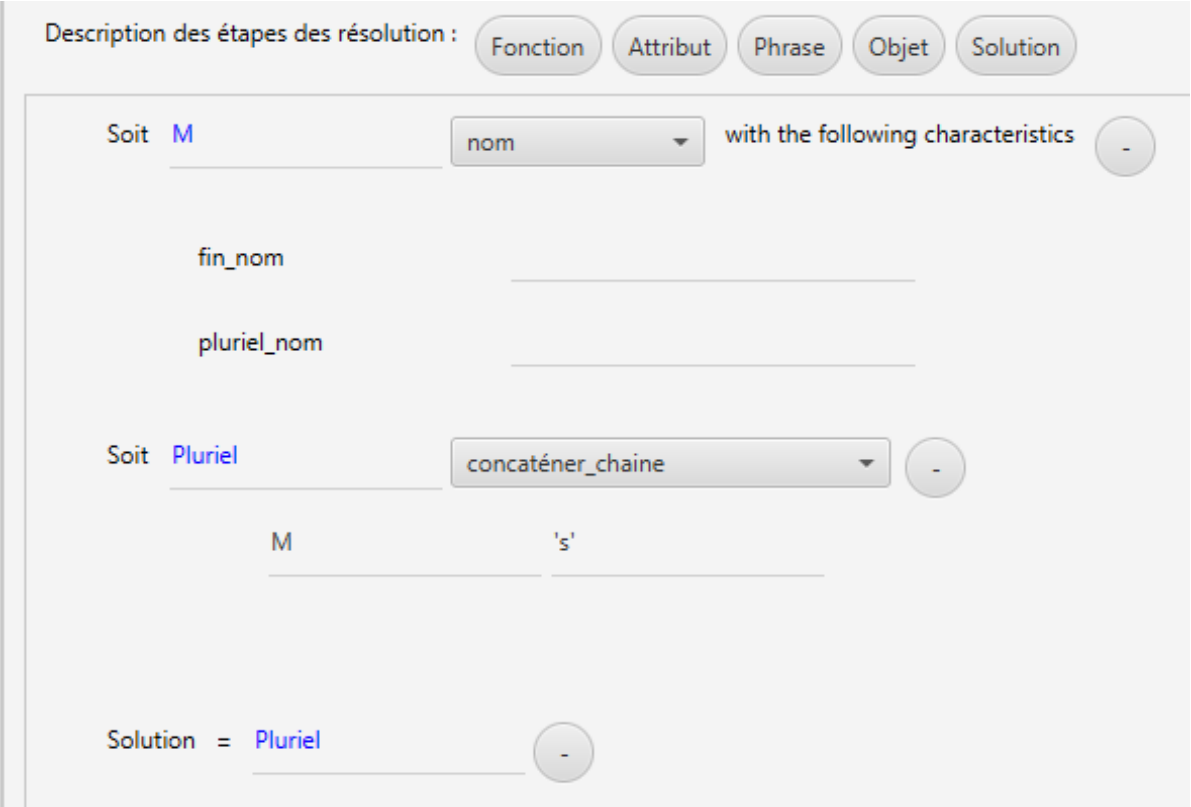

FIGURE 8 – Définition de la technique associée à la classe *c_plu_ail_pas_ex* en utilisant AMBRE-KB

Exemple de modèle décrivant des techniques de résolution, généré par AMBRE-KB

Technique associée à la classe *c_plu_ail_pas_ex* appliquer_resolution(P,c_plu_ail_pas_ex) :- eval_prem(est$_u n(M, nom)$), concatener_chaine(M,Chaine,Pluriel), ajouter_bf(solution(P,Pluriel)), write('').
Technique associée à la classe *c_plu_ail_ex* appliquer_resolution(P,c_plu_ail_ex) :- eval_prem(est_un(M,nom)), pluriel_ail_exception(M,Pluriel), ajouter_bf(solution(P,Pluriel)), write('').

Tableau 16 – Exemple de modèles de connaissances (décrivant des techniques) pour le pluriel des noms générés par AMBRE-KB

Résolution des équations à une seule inconnue

Ce domaine a été testé par deux sujets.

Profil du sujet 1 : Mathématicien (non programmeur).

Profil du sujet 1 : Informaticien (programmeur).

Méthode à enseigner : les sujets ont choisi d'enseigner la résolution des équations à une seule inconnue, du premier et second degré.
Les équations à résoudre ici sont toujours données sous les formes suivantes :
« *Coefficient_a*Inconnu +coefficient_b*Inconnu + Constante = 0* »
« *Coefficient_a*Inconnu + Constante = 0*»

Le **Tableau 6.6** décrit le nombre de connaissances produites par des deux sujets pour la résolution des équations.

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	4	3
Classes	6	7
Règles	5	5
Techniques	4	4
Problèmes	20	12
% Problèmes résolus sans feedback	75%	41%
% Problèmes résolus avec feedback	100%	100%

Tableau 17 – Comparaison des résultats des expérimentations pour les équations par les sujets 1 et 2

9 fonctions en plus ont été définies par les sujets. Celles-ci interviennent dans la définition des règles et des techniques de résolution pour le domaine :

— **Multiplication** : permettant de calculer le produit de deux nombres

— **Division** : qui renvoie le quotient de la division de deux nombres

— **Racine_carre** : permettant de calculer la racine carré d'un nombre

— **Determinant** : qui calcule le déterminant de trois valeurs (coefficient_b * coefficient_b - 4*coefficient_a*constante)

— **Determinant_nul_solution** : qui renvoie la solution lorsque le déterminant est nul (- coefficient_b/2*coefficient_a)

— **Determinant_positive_solution_1** : permettant de calculer une des solutions lorsque le déterminant est positif : (- coefficient_b ' racine_carré (determinant) /2*coefficient_a)

— **Determinant_positive_solution_2** : permettant de calculer une des solutions lorsque le déterminant est positif : (- coefficient_b + racine_carré (determinant) /2*coefficient_a)

— **Liste_nombres** : permettant de créer une liste à partir de deux nombres (par exemple, lorsque le déterminant est positif, renvoyer les deux solutions sous la forme d'une liste).

— **Phrase_reponse** : permet de construire une phrase (utilisée lorsque le déterminant est négatif, pour dire qu'il n'y a pas de solution).

Vocabulaires proposés pour le domaine

Vocabulaire défini par le sujet 1

Types objets	**Caractéristiques objets**
Équation	Nom_inconnu –>chaine de caractères Valeur_inconnu–>nombre entier Liste_termes–>type énuméré(deux_termes,trois_termes)
Deux_termes	Premier_terme–>terme_inconnu Deuxième_terme –>terme_constante
Terme_inconnu	coefficient–>nombre entier
Terme_constante	valeur_constante–>nombre entier

Tableau 18 – Vocabulaire défini par le sujet 1.

Vocabulaire défini par le sujet 2

Types objets	**Caractéristiques objets**
Équation	Terme_1–>type énuméré(terme_inconnu,terme_constante) Terme_2–>type énuméré(terme_inconnu,terme_constante) Terme_3–>type énuméré(terme_inconnu,terme_constante)
Terme_inconnu	Coefficient–>nombre entier Nom_variable–>chaine de caractères Degré–>nombre entier Valeur–>nombre entier
Terme_constante	valeur_constante –> nombre entier

Tableau 19 – Vocabulaire défini par le sujet 2.

La question à résoudre pour les différents problèmes consistait à trouver la valeur de valeur_variable.

Problèmes à résoudre

Exemples de problèmes définis par les sujets :

degre_1_1 : « Résoudre l'équation suivante : 2x-5=0. »
degre_1_2 : « Résoudre l'équation suivante : 7y+9=0. »
degre_2_nul : « Résoudre l'équation suivante : y2+2*Y+1=0. »
degre_2_positif : « Résoudre l'équation suivante : z2+6z+7=0=0. »
degre_2_negatif : « Résoudre l'équation suivante : 3*x2+3*x+3=0. »

Arbre de classification

Arbre de classification défini par le sujet 1.

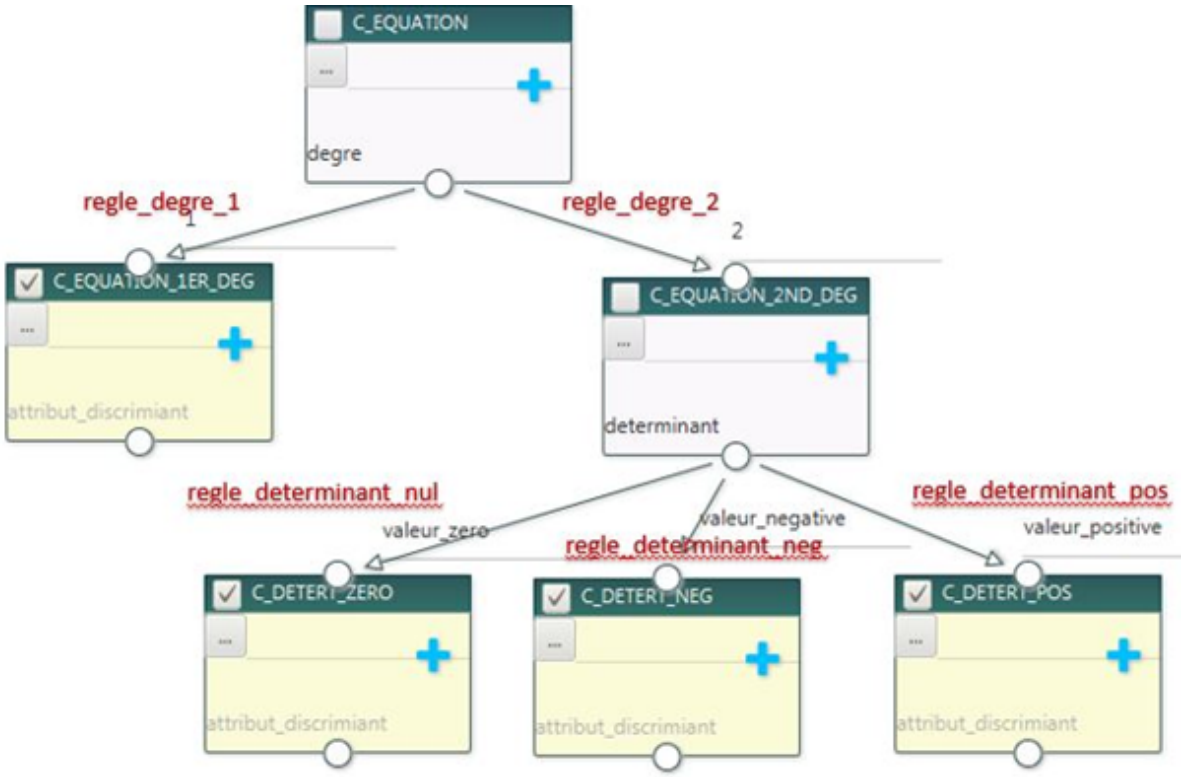

FIGURE 9 – Arbre de classification pour les équations défini par le sujet 1

Arbre de classification défini par le sujet 2.

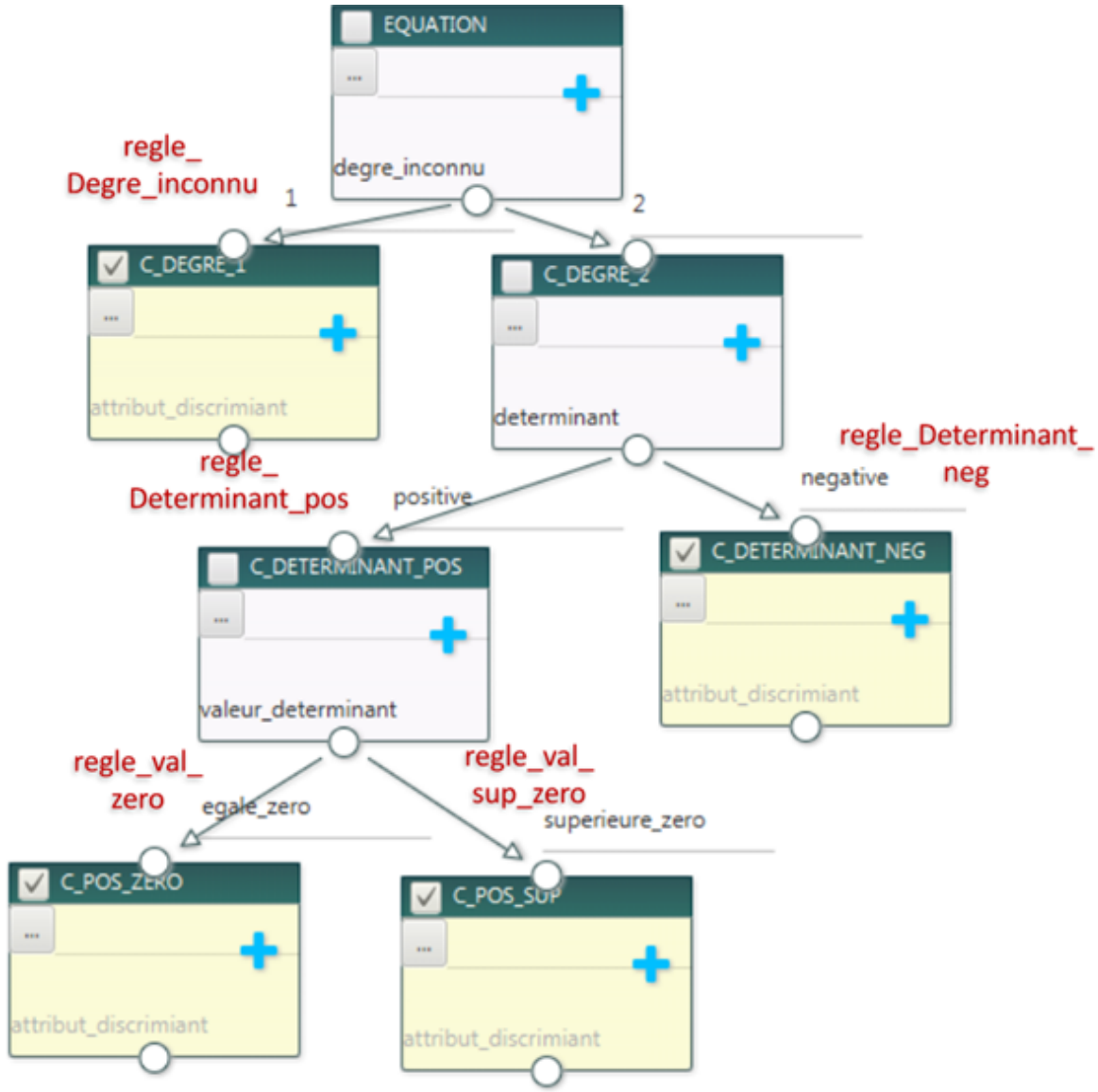

FIGURE 10 – Arbre de classification pour les équations défini par le sujet 2

Règles de reformulation

Exemples de règles définies par l'auteur

Exemples de règles de reformulation définies par les sujets.

regle_degre : **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le degré D, **Alors** la valeur de l'attribut degré est égal à D.
regle_determiannt_pos **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le coefficient est A, un deuxième terme T2 de type terme_inconnu dont le coefficient est B, un troisième terme de type T3 terme_constante dont la valeur de la constante est C, F le résultat de la fonction qui calcule le déterminant (B2-4*A*C) F>0, **alors** la valeur de l'attribut déterminant est égale à valeur_negative.
regle_determiannt_nul : **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le coefficient est A, un deuxième terme T2 de type terme_inconnu dont le coefficient est B, un troisième terme de type T3 terme_constante dont la valeur de la constante est C, F le résultat de la fonction qui calcule le déterminant (B2-4*A*C), F=0, **Alors** la valeur de l'attribut déterminant est égale à valeur_zero.
regle_determiannt_neg **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le coefficient est A, un deuxième terme T2 de type terme_inconnu dont le coefficient est B, un troisième terme de type T3 terme_constante dont la valeur de la constante est C, F le résultat de la fonction qui calcule le déterminant (B2-4*A*C), F<0, **Alors** la valeur de l'attribut déterminant est égale à valeur_negative.

Tableau 20 – Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé

Techniques de résolution

Exemple de techniques de résolution définies par les sujets.

```
/*Technique pour les résoudre les équations du premier degré*/
appliquer_resolution(P,c_equation_1er_deg) :-
        eval_prem(est_un(T1,premier_terme)),
        eval_prem(term1(T1,Term1)),
```

```
        eval_prem(est_un(Term1,terme_inconnu)),
        eval_prem(coefficient(Term1,A)),
        eval_prem(nom_variable(Term1,X)),
        eval_prem(degre_variable(Term1,D)),
        eval_prem(est_un(T3,troisieme_terme)),
        eval_prem(term3(T3,Term3)),
        eval_prem(est_un(Term3,terme_constante)),
        eval_prem(valeur_constante(Term3,C)),
        division(- C,A,Div),
        ajouter_bf(solution(P,[Div])),
        write('').
/*Technique pour les résoudre les équations du second
degré dont le déterminant est nul*/
appliquer_resolution(P,c_2nd_deter_zero) :-
        eval_prem(est_un(T1,premier_terme)),
        eval_prem(term1(T1,Term1)),
        eval_prem(est_un(Term1,terme_inconnu)),
        eval_prem(coefficient(Term1,A)),
        eval_prem(nom_variable(Term1,X)),
        eval_prem(est_un(T2,deuxieme_terme)),
        eval_prem(term2(T2,Term2)),
        eval_prem(est_un(Term2,terme_inconnu)),
        eval_prem(coefficient(Term2,B)),
        eval_prem(nom_variable(Term2,X)),
        eval_prem(est_un(T3,troisieme_terme)),
        eval_prem(term3(T3,Term3)),
        eval_prem(est_un(Term3,terme_constante)),
        eval_prem(valeur_constante(Term3,C)),
        discriminant_nul_solution(A,B,Sol),
        ajouter_bf(solution(P,[Sol])),
        write('').
```

```
/*Technique pour les équations du 1er degré
        Soit E une équation
        Soit nom_inconnu de E = X
        Soit valeur_inconnu de E = Val
```

Soit la liste L des termes de E = deux_termes
Soit premier_terme de L = T1
Soit T1 est un terme_inconnu
Soit coefficient de T1 = C1
Soit deuxieme_terme de L= T2
Soit T2 est un terme_constante
Soit valeur_constante de T2 = C2
Val = - C2/C1
Solution = valeur_inconnu.

/*Technique pour les résoudre les équations du second degré dont le déterminant est négatif*/
Soit E une équation
Soit nom_inconnu de E = X
Soit Valeur_inconnu de E = Val
Soit la liste L des termes de E = trois_termes
Soit premier_terme de L = T1
Soit T1 est un terme_inconnu
Soit coefficient de T1 = C1
Soit troisieme_terme de L= T3
Soit T3 est un terme_constante
Soit valeur$_c onstante de T3 = C3$
Soit le résultat de la fonction déterminant = 0
Val = - C2/(2*C1)
Solution = Val

Calcul d'aire en géométrie

Ce domaine a été testé par deux sujets.

Profil du sujet 1 : Statisticien (non programmeur).

Profil du sujet 2 : Informaticien (non programmeur).

Méthode à enseigner : les sujets ont choisi d'enseigner le calcul d'aire des figures géométriques ayant 3, 4 ou 4 cotés.

Le **Tableau 22** résume les connaissances décrites par les deux sujets :

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	5	8
Classes	12	12
Règles	11	11
Techniques	7	7
Problèmes	28	18
% Problèmes résolus sans feedback	57%	61%
% Problèmes résolus avec feedback	85%	89%

Tableau 22 – Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2

16 fonctions en plus, intervenant dans la définition des règles et des techniques de résolution ont été définies âr les sujets :

- **aire_triangle** :permettant de calculer l'aire d'un triangle
- **aire_rectangle** :permettant de calculer l'aire d'un rectangle
- **aire_carre** :permettant de calculer l'aire d'un carre
- **aire_trapeze** :permettant de calculer l'aire d'un trapèze
- **aire_losange** :permettant de calculer l'aire d'un losange
- **aire_pentagone** :permettant de calculer l'aire d'un pentagone régulier
- **aire_tpentagone_irregulier** :permettant de calculer l'aire d'un pentagone irrégulier
- **demi_perimetre_triangle** : permettant de calculer le demi périmètre d'un triangle

— **addition** : permettant de calculer la somme de deux nombres
— **soustraction** : permettant de calculer la soustraction de deux nombres
— **division** : permettant de calculer la division d'un nombre
— **racine_carre** : permettant de calculer la racine carré d'un nombre
— **tangente** : permettant de calculer la tangente
— **valeur_apotheme** : permettant de calculer l'apothème
— **cosinus** : permettant de calculer le cosinus
— **sinus** : permettant de calculer le sinus

Vocabulaires proposés pour le domaine

Vocabulaire proposé par le sujet 1.

Types objets	**Caractéristiques objets**
figure	identifiant aire_figure –> nombre segment_1 –> segment (obligatoire) segment_2 –> segment (obligatoire) segment_3 –> segment (obligatoire) segment_4 –> segment segment_5 –> segment angle_droit –> booléen
angle	identifiant segment_1 –> segment segment_2 –> segment valeur_angle –> nombre entier
segment	identifiant longueur_segment –> nombre entier
segments_paralleles	identifiant segment_1 –> segment segment_2 –> segment
segments_opposes	identifiant segment_1 –> segment segment_2 –> segment

Tableau 23 – vocabulaire pour le calcul d'aire proposé par le sujet 1

Vocabulaire proposé par le sujet 2.

Types objets	**Caractéristiques objets**
figure	identifiant aire_figure –> nombre liste_cote_figure –> type énuméré trois_cotes, quatre_cotes,cinq_cotes possede_angle_droit –> booléen
cote	identifiant longueur_cote –> nombre entier
trois$_c$*otes*	identifiant segment_1 –> segment segment_2 –> segment segment_3 –> segment
quatre$_c$*otes*	identifiant segment_1 –> segment segment_2 –> segment segment_3 –> segment segment_4 –> segment
cinq$_c$*otes*	identifiant segment_1 –> segment segment_2 –> segment segment_3 –> segment segment_4 –> segment segment_5 –> segment
segment	identifiant longueur_segment –> nombre entier
cote_paralleles	identifiant cote_1 –> cote cote_2 –> cote
cote_opposes	identifiant cote_1 –> cote cote_2 –> cote
cote_perpendiculaires	identifiant cote_1 –> cote cote_2 –> cote

Tableau 24 – vocabulaire pour le calcul d'aire proposé par le sujet 2

Problèmes à résoudre

Exemples de problèmes définis par les sujets :

Pentagone_régulier_1 : « Trouver le périmètre et l'aire d'un polygone ABCDE dont chaque côté mesure 3cm ».

Carre_1 : « On veut calculer l'aire et le périmètre d'un polygone ABCD dont chaque côté mesure 5cm et dont l'angle abc est droit ».

trapèze_1 : « Quelle est le périmètre et l'aire d'un champ qui a la forme d'un quadrilatère dont les dimensions des côtés opposés sont : AB =4 CD = 7».

Triangle_1 : « on cherche à calculer le périmètre et l'aire d'un polygone ABC dont l'angle ABC est droit, et les dimensions sont : AB=3 AC=5 ».

Parallélogramme_1 : « Calculer le périmètre et l'aire d'un polygone MNOP ».

Arbre de classification

Une partie de l'arbre de classification défini par le sujet 1.

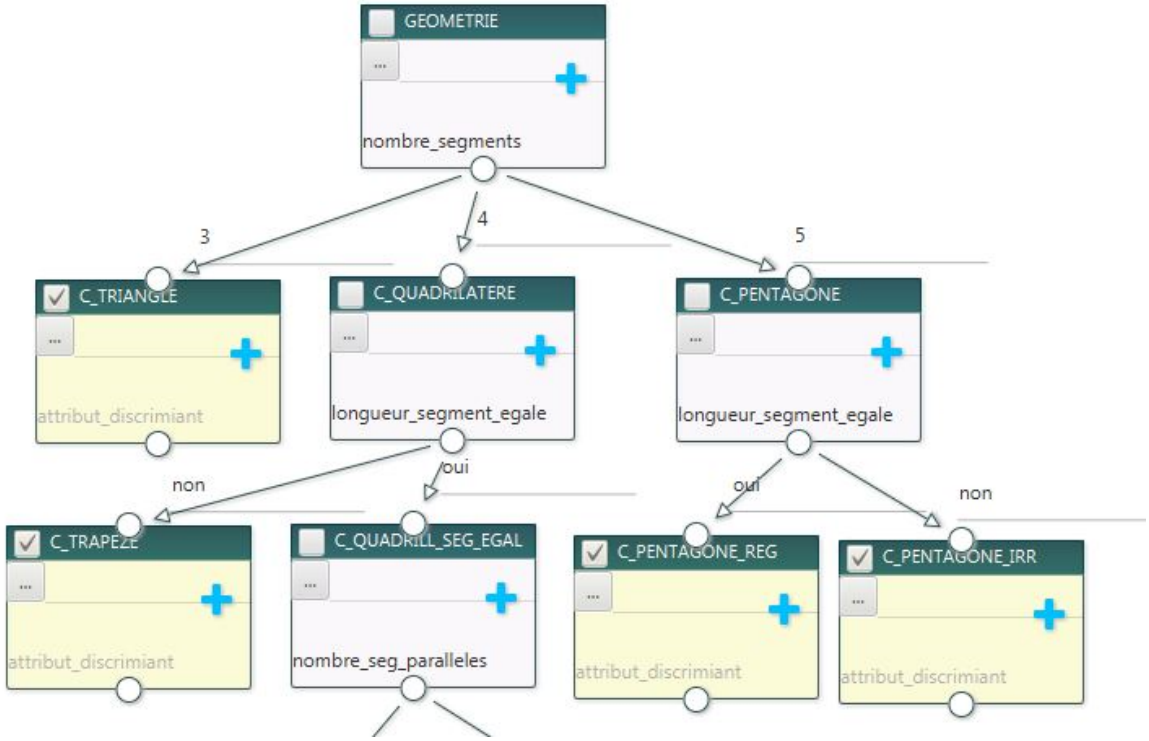

FIGURE 11 – Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie

Une partie de l'arbre de classification défini par le sujet 1.

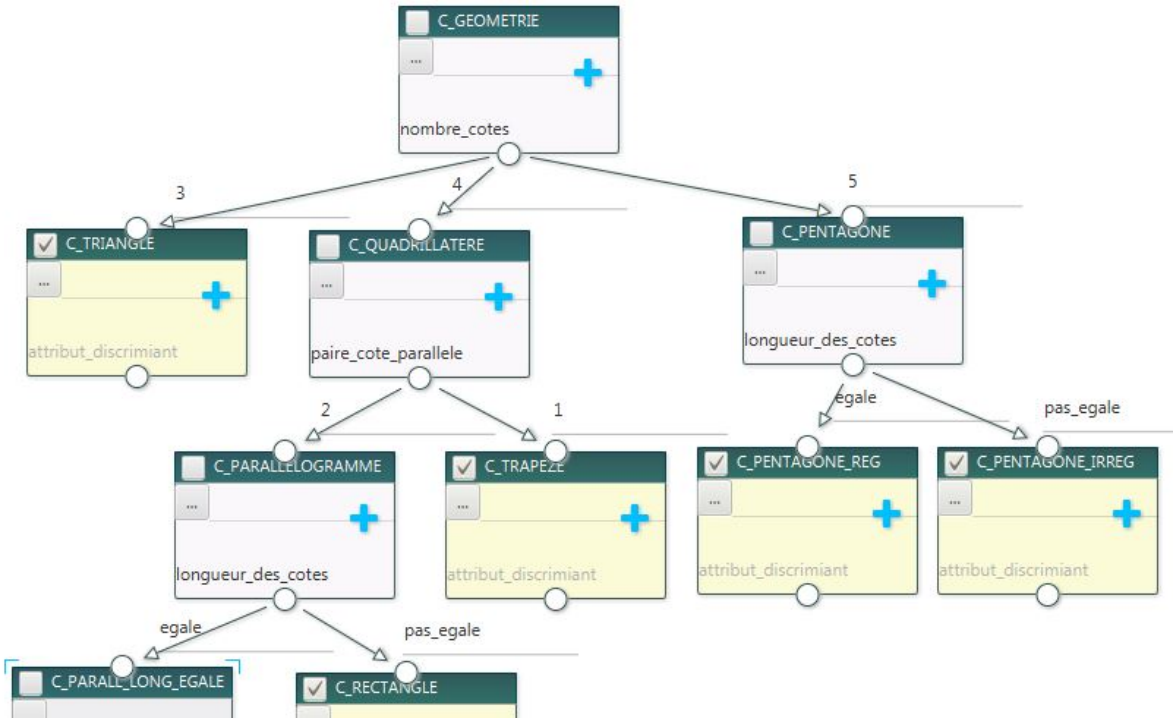

FIGURE 12 – Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie

Règles de reformulation

Nous présentons quelques règles de reformulation définies par le sujet 1.

/* Règle permettant de conclure sur la valeur de l'attribut *nombre_segments* ***/**
Si dans un problème il existe :
- une figure géométrique F
- S1 le segment_1 de F
- S2 le segment_2 de F
- S3 le segment_3 de F
- S4 le segment_4 de F
- S5 le segment_5 de F
- La longueur d'un des segments est différents des autres

Alors la valeur de l'attribut *nombre_segments* est égale à *5*
/**Règle permettant de conclure sur la valeur de l'attribut longueur_segment_egale****/**
Si dans un problème, il existe :
- une figure géométrique F
- la longueur C1 du segment_1 S1 de F
  est égale à la longueur C2 du segment_2 S2 de F
- la longueur C2 du segment_2 S2 de F
  est égale à la longueur C3 du segment_3 S3 de F
- la longueur C3 du segment_3 S3 de F
  est égale à la longueur C4 du segment_4 S4 de F
- la longueur C4 du segment_4 S4 de F
  est égale à la longueur C5 du segment_5 S5 de F
- la longueur C5 du segment_5 S5 de F
  est égale à la longueur C1 du segment_1 S1 de F

Alors la valeur de l'attribut *longueur_segment_egale* est égale à *oui* height

Tableau 25 – Exemple de règles de reformulation définies pour le calcul d'aire en géométrie

Techniques de résolution

Exemples de techniques de résolution :

/* technique permettant de calculer l'aire d'un triangle ***/**
- Soit F une figure
- soit C1 la longueur du segment_1 de F
- soit C2 la longueur du segment_1 de F
- soit C3 la longueur du segment_3 de F
- soit Sol le résultat de la fonction aire_triangle
- Solution égale Sol

Tableau 26 – Exemple de techniques de résolution définies pour le calcul d'aire en géométrie

Formation des adverbes en "ment"

Profil du sujet : Infographiste (non programmeur).

Méthode à enseigner. Le sujet a choisi d'enseigner la formation des adverbes en français terminés par « ment ».

L'expert a défini :

- 17 classes dont 11 sont opérationnelles
- 16 règles
- 11 techniques de résolution
- 40 problèmes
- 4 Fonctions intervenant dans la définition des règles et des techniques de résolution ajouter_chiane : permet de concaténer deux chaines de caractères enlever_chaine : enlever une sous chaine à une chaine de caractères dernière_lettre_mot : qui renvoie la dernière lettre d'un mot liste_de$_v$*oyelles* : $fonctionquirenvoielalistedesvoyelles$

Vocabulaire proposé pour le domaine

Types objets	**Caractéristiques objets**
mot	identifiant–>chaine de caractères adverbe–>chaine de caractères

Tableau 27 – vocabulaire pour la formation des adverbes français en "ment"

Problèmes à résoudre

Exemples de problèmes définis par l'expert :

Adverbe_faible : Donner l'adverbe du mot (faible) ».
Adverbe_apparent : Donner l'adverbe du mot (apparent) ».
Adverbe_suffisant : Donner l'adverbe du mot (suffisant) ».
Adverbe_vrai : Donner l'adverbe du mot (vrai) ».

Adverbe_assidu : Donner l'adverbe du mot (assidu) ».

Arbre de classification

La **Figure 13** présente une partie de l'arbre de classification proposé pour le domaine.

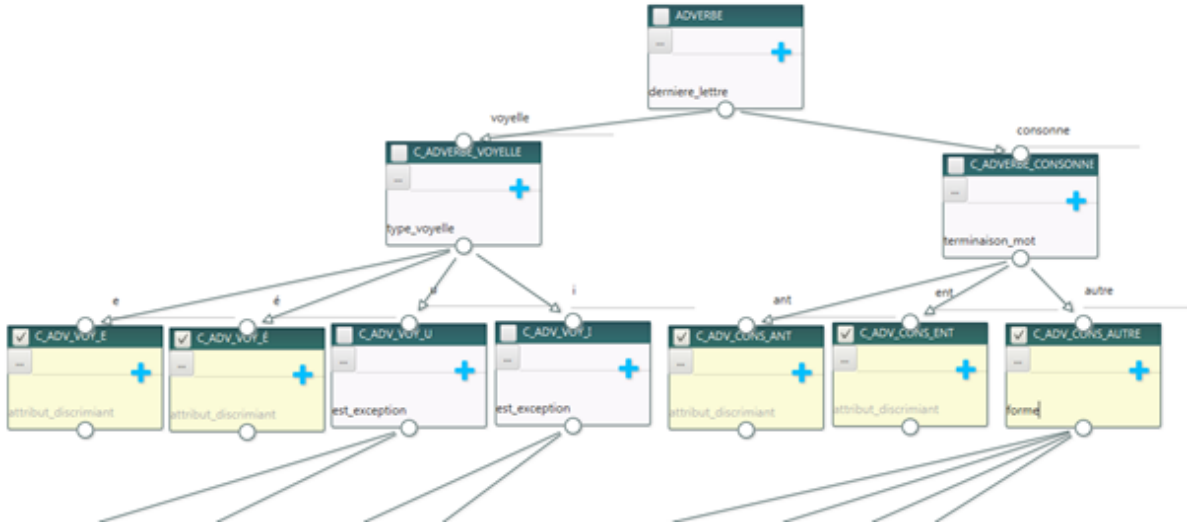

FIGURE 13 – Une partie de l'arbre de classification pour la formation des adverbes en "ment"

Règles de reformulation

/* Exemples de règle permettant de conclure sur la valeur de l'attribut derniere_lettre ***/**

Si dans un problème, il existe
- un mot M
- la dernière lettre du mot appartient à la liste des voyelles

Alors la valeur de l'attribut *derniere_lettre* est égale à *voyelle*

Si dans un problème, il existe
- un mot M
- la dernière lettre du mot n'appartient pas à la liste de s voyelles
Alors la valeur de l'attribut *derniere_lettre* est égale à *consonne*

Techniques de résolution

/* technique permettant de former l'adverbe des mots terminés par "ant" ***/**
- Soit M un mot
- soit M1 le radical du mot en enlevant la terminaison 'ant'
- Soit M2 le mot obtenu en ajoutant 'amment' au radical de M
- Solution est égale à M2

/* technique permettant de former l'adverbe des mots terminés par "ent" ***/**
- Soit M un mot
- soit M1 le radical du mot en enlevant la terminaison 'ent'
- Soit M2 le mot obtenu en ajoutant 'emment' au radical de M
- Solution est égale à M2

Critères retenus pour évaluer le processus d'acquisition et sa mise en œuvre dans AMBRE-KB

Dans le chapitre 6, nous avons identifié différentes facettes pour évaluer si AMBRE-KB permet d'éliciter les différentes connaissances à la conception d'un EIAH AMBRE. Nous détaillons ici les critères retenus pour évaluer chacune de ces facettes.

La première section détaille les critères retenus pour évaluer si AMBRE-KB permet d'éliciter les connaissances nécessaires à l'EIAH pour résoudre des problèmes. Dans la deuxième section, nous présentons les critères retenus pour évaluer si les interfaces proposées dans AMBRE-KB sont assez intuitives. La dernière section détaille les critères retenus pour évaluer l'assistance fournie AMBRE-KB.

Critères retenus pour évaluer l'élicitation des connaissances avec AMBRE-KB

Facette à évaluer	Définition du vocabulaire pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi ?
Méthodologie envisagée	Définition du vocabulaire de son domaine par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi

Tableau 28 – Critères retenus pour évaluer le vocabulaire d'un domaine

Facette à évaluer	Description des problèmes à résoudre pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de décrire les problèmes du domaine ?
Méthodologie envisagée	Définition des problèmes du domaine par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet de décrire l'ensemble des problèmes du domaine

Tableau 29 – Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné

Facette à évaluer	Élicitation des connaissances de classification pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de définir les connaissances de classification du domaine choisi ?
Méthodologie envisagée	Élicitation des connaissances de classification par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'éliciter les connaissances de classification du domaine

Tableau 30 – Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné

Facette à évaluer	Élicitation des connaissances de reformulation pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de définir les connaissances de reformulation du domaine choisi ?
Méthodologie envisagée	Élicitation des connaissances de reformulation par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'éliciter les connaissances de reformulation du domaine

Tableau 31 – Critères retenus pour évaluer l'élicitation des connaissances de reformulation pour un domaine donné

Facette à évaluer	Élicitation des connaissances de résolutionpour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de définir les connaissances de résolution du domaine choisi ?
Méthodologie envisagée	Élicitation des connaissances de résolution par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'éliciter les connaissances de résolution du domaine

Tableau 32 – Critères retenus pour évaluer l'élicitation des connaissances de résolution pour un domaine donné

Critères retenus pour évaluer les interfaces de AMBRE-KB

Facette à évaluer	Définition du vocabulaire pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour la définition du vocabulaire ?
Méthodologie envisagée	Définition du vocabulaire par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir le vocabulaire sans aide extérieure.

Tableau 33 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la définition du vocabulaire pour un domaine donné

Facette à évaluer	Description des problèmes à résoudre pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour la Description des problèmes à résoudre ?
Méthodologie envisagée	Description des problèmes à résoudre par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les problèmes du domaine sans aide extérieure.

Tableau 34 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la Description des problèmes à résoudre pour un domaine donné

Facette à évaluer	Élicitation des connaissances de classification pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour Élicitation des connaissances de classification à résoudre ?
Méthodologie envisagée	Élicitation des connaissances de classification par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les connaissances de classification sans aide extérieure.

Tableau 35 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de classification pour un domaine donné

Facette à évaluer	Élicitation des connaissances de reformulation pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour Élicitation des connaissances de reformulation
Méthodologie envisagée	Élicitation des connaissances de reformulation par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les connaissances de reformulation sans aide extérieure.

Tableau 36 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de reformulation pour un domaine donné

Facette à évaluer	Élicitation des connaissances de résolution pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour Élicitation des connaissances de résolution
Méthodologie envisagée	Élicitation des connaissances de résolution par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les connaissances de résolution sans aide extérieure.

Tableau 37 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de résolution pour un domaine donné

Critères retenus pour évaluer l'assistance fournie avec AMBRE-KB

Facette à évaluer	Flexibilité offerte avec AMBRE-KB
Point à analyser	Est-ce que la flexibilité fournie à l'auteur est utile et permet d'aider l'auteur durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	La flexibilité proposée dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation

Tableau 38 – Critères retenus pour évaluer la flexibilité proposée dans AMBRE-KB

Facette à évaluer	Guidage offerte avec AMBRE-KB
Point à analyser	Est-ce que le guidage fourni à l'auteur est utile et permet d'aider l'auteur durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	Le guidage proposé dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation

Tableau 39 – Critères retenus pour évaluer les fonctionnalités de guidage proposées dans AMBRE-KB

Facette à évaluer	Contrôle de connaissances offerte avec AMBRE-KB
Point à analyser	Est-ce que le Contrôle des connaissances proposé à l'auteur est utile et permet d'aider l'auteur durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	Le guidage proposé dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation

Tableau 40 – Critères retenus pour évaluer les fonctionnalités de contrôle de connaissances proposées dans AMBRE-KB

Critères retenus pour évaluer le feedback proposé avec AMBRE-KB

Facette à évaluer	Feedback offert avec AMBRE-KB
Point à analyser	Test de la résolution des problèmes à résoudre ?
Méthodologie envisagée	Test de la résolution des problèmes à résoudre + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	Le feedback proposé permet à l'auteur de comprendre ses erreurs lorsque la résolution échoue et de pouvoir les corriger

Tableau 41 – Critères retenus pour évaluer les fonctionnalités de feedback proposées dans AMBRE-KB

Questionnaire distribué aux sujets

1. **Date :** _____

2. **Domaine testé :** _____

3. **Profil :** _____

Définition du vocabulaire

4. **Avez-vous rencontré des problèmes pour définir le vocabulaire ?**
   - ☐ J'ai compris dès le début. 5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

5. **Est-ce que les interfaces proposées sont assez intuitives ?**
   - ☐ J'ai compris dès le début.5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

6. **Est-ce que vous avez pu définir toutes les connaissances nécessaires au vocabulaire du domaine que vous avez choisi ?**
   - ☐ Oui 5/5

- ☐ Sinon, merci de préciser les types de connaissances que vous souhaitiez pouvoir définir.

  _____

  _____

  _____

7. **Est-ce que vous avez des suggestions concernant le vocabulaire ou sa définition ?**

   _____

   _____

   _____

Description des problèmes à résoudre

8. **Avez-vous rencontré des problèmes pour décrire les problèmes à résoudre ?**
   - ☐ J'ai compris dès le début. 5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

9. **Est-ce que les interfaces proposées sont assez intuitives ?**
   - ☐ J'ai compris dès le début.5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

10. **Est-ce que vous avez pu définir toutes les connaissances nécessaires pour décrire les problèmes du domaine que vous avez choisi ?**
    - ☐ Oui 5/5
    - ☐ Sinon, merci de préciser les types de connaissances que vous souhaitiez pouvoir définir.

11. **Est-ce que vous avez des suggestions concernant la description des problèmes ?**

Élicitation des connaissances de classification

12. **Avez-vous rencontré des problèmes pour définir l'arbre de classification du domaine que vous avez choisi ?**
    - ☐ J'ai compris dès le début. 5/5
    - ☐ J'ai compris vers la fin.
    - ☐ Je n'ai pas compris.

13. **Est-ce que les interfaces proposées sont assez intuitives ?**
    - ☐ J'ai compris dès le début.5/5
    - ☐ J'ai compris vers la fin.
    - ☐ Je n'ai pas compris.

14. **Est-ce que vous avez des suggestions concernant la définition de l'arbre de classification ?**

Élicitation des connaissances de reformulation

15. **Avez-vous rencontré des problèmes pour définir les règles de reformulation du domaine que vous avez choisi ?**
    - ☐ J'ai compris dès le début. 1/5
    - ☐ J'ai compris vers la fin. 4/5
    - ☐ Je n'ai pas compris.

16. **Est-ce que les interfaces proposées sont assez intuitives ?**
    - ☐ J'ai compris dès le début.1/5
    - ☐ J'ai compris vers la fin. 4/5
    - ☐ Je n'ai pas compris.

17. **Est-ce que vous avez des suggestions concernant la définition des règles de reformulation ?**

    _____

    _____

    _____

Élicitation des connaissances de résolution

18. **Avez-vous rencontré des problèmes pour définir les techniques de résolution du domaine que vous avez choisi ?**
    - ☐ J'ai compris dès le début. 1/5
    - ☐ J'ai compris vers la fin. 4/5
    - ☐ Je n'ai pas compris.

19. **Est-ce que les interfaces proposées sont assez intuitives ?**
    - ☐ J'ai compris dès le début.1/5

- ☐ J'ai compris vers la fin. 4/5
- ☐ Je n'ai pas compris.

20. Est-ce que vous avez des suggestions concernant la définition des techniques de résolution ?

_____

_____

_____

Assistance fournie avec AMBRE-KB

21. Que pensez-vous de la flexibilité offerte par AMBRE-KB lors de la définition des connaissances ?

- ☐ Je n'ai pas remarqué.
- ☐ J'ai compris vers la fin. 5/5

22. Que pensez-vous des fonctionnalités de guidage proposées par AMBRE-KB ?

- ☐ Je n'ai pas remarqué.1/5
- ☐ Elles sont utiles mais peuvent être améliorées. 2/5
- ☐ Elles sont utiles et m'ont beaucoup aidé. 3/5

23. Que pensez-vous des fonctionnalités de contrôle des connaissances offertes par AMBRE-KB ?

- ☐ Elles sont insuffisantes et ne m'ont pas vraiment aidé.
- ☐ Elles sont utiles mais ne sont pas assez suffisantes.
- ☐ Elles sont utiles et m'ont beaucoup aidé. 5/5

24. Que pensez-vous des fonctionnalités de guidage proposées par AMBRE-KB ?

- □ Je n'ai pas compris les messages d ?erreur.
- □ J'ai compris les messages d'erreur mais ils ne sont pas utiles.
- □ Les messages d'erreur sont utiles et m ?ont permis de comprendre et de corriger mes erreurs. 2/5

Fonctionnement général de AMBRE-KB

25. Est-ce que vous avez compris le fonctionnement général de AMBRE-KB ?

- □ Oui.5/5
- □ Non.

Bibliographie

[1] Shaaron Ainsworth, Nigel Major, Shirley Grimshaw, Mary Hayes, Jean Underwood, Ben Williams, and David Wood. Redeem : Simple intelligent tutoring systems from usable tools. In *Authoring Tools for Advanced Technology Learning Environments*, pages 205–232. Springer, 2003.

[2] Vincent Aleven. Using background knowledge in case-based legal reasoning : a computational model and an intelligent learning environment. *Artificial Intelligence*, 150(1-2) :183–237, 2003.

[3] Vincent Aleven. Rule-based cognitive modeling for intelligent tutoring systems. In *Advances in intelligent tutoring systems*, pages 33–62. Springer, 2010.

[4] Vincent Aleven, Bruce M Mclaren, Jonathan Sewall, and Kenneth R Koedinger. A new paradigm for intelligent tutoring systems : Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2) :105–154, 2009.

[5] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, Martin Van Velsen, Octav Popescu, Sandra Demi, Michael Ringenberg, and Kenneth R Koedinger. Example-tracing tutors : Intelligent tutor development for non-programmers. *International Journal of Artificial Intelligence in Education*, 26(1) :224–269, 2016.

[6] Nathalie Aussenac-Gilles. *Conception d'une methodologie et d'un outil d'acquisition de connaissances expertes*. PhD thesis, Toulouse 3, 1989.

[7] Nathalie Aussenac-Gilles. *Méthodes ascendantes pour l'ingénierie des connaissances*. PhD thesis, Université Paul Sabatier-Toulouse III, 2005.

[8] Nathalie Aussenac-Gilles, Jean Charlet, and Chantal Reynaud. Les enjeux de l'Ingénierie des connaissances. In P. Marquis F. Sedes, J.M. Ogier, editor, *Information, Interaction, Intelligence - Le point sur le i(3)*. Cepadues, January 2012.

[9] N Balacheff. Contribution à la réflexion sur la recherche sur les environnements informatiques pour lâapprentissage humain. *Les technologies en éducation : Pers-*

pectives de recherche et questions vives. Paris : INRP–MSH-IUFM de Basse Normandie, pages 193–201, 2002.

[10] Nicolas Balacheff, Monique Baron, Cyrille Desmoulins, Monique Grandbastien, and Martial Vivet. Conception d'environnements interactifs dâapprentissage avec ordinateur. tendances et perspectives. *Actes des journées nationales du PRC IA*, pages 315–338, 1997.

[11] Emmanuel Bigand, François Madurell, Barbara Tillmann, and Marion Pineau. Effect of global structure and temporal organization on chord processing. *Journal of Experimental Psychology : Human Perception and Performance*, 25(1) :184, 1999.

[12] Stephen B Blessing. A programming by demonstration authoring tool for model-tracing tutors. In *Authoring Tools for Advanced Technology Learning Environments*, pages 93–119. Springer, 2003.

[13] John H Boose. A survey of knowledge acquisition techniques and tools. *Knowledge acquisition*, 1(1) :3–37, 1989.

[14] Jill Boucher. The theory of mind hypothesis of autism : Explanation, evidence and assessment. *British Journal of Disorders of Communication*, 24(2) :181–198, 1989.

[15] Alain Bronner, Thomas Huguet, Denis Bouhineau, and Jean-François Nicaud. Usages didactiques du logiciel aplusix pour l'enseignement de l'algebre. 2003.

[16] E Bruillard, Élisabeth Delozanne, Pascal Leroux, Paul Delannoy, Xavier Dubourg, Pierre Jacoboni, Jérôme Lehuen, Daniel Luzzati, and Philippe Teutsch. Quinze ans de recherche informatique sur les sciences et techniques éducatives au lium. *Revue Sciences et Techniques Educatives*, 7(1) :87–145, 2000.

[17] Jean Charlet. L'ingénierie des connaissances, entre science de l'information et science de gestion. In *Entre la connaissance et l'organisation, l'activité collective*. November 2004. Ce document est la version allongée du chapitre référencé.

[18] Jean Charlet, Manuel Zacklad, and Gilles Kassel. Ingénierie des connaissances. 2000.

[19] Michelene TH Chi, Paul J Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive science*, 5(2) :121–152, 1981.

[20] Christophe Choquet. *Ingénierie et réingénierie des EIAH-L'approche REDiM*. PhD thesis, Université du Maine, 2007.

[21] William J Clancey. Heuristic classification. *Artificial intelligence*, 27(3) :289–350, 1985.

[22] William J Clancey. From guidon to neomycin and heracles in twenty short lessons. *AI Magazine*, 7(3) :40, 1986.

[23] Albert T Corbett, Kenneth R Koedinger, and John R Anderson. Intelligent tutoring systems. *Handbook of human-computer interaction*, 5 :849–874, 1997.

[24] Stephane Crozat and Philippe Trigano. *An Ontological Approach for Design and Evaluation of Tutoring Systems*, pages 651–651. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[25] Denise Dellarosa Cummins. Role of analogical reasoning in the induction of problem categories. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 18(5) :1103, 1992.

[26] Jean-Marc David, Jean-Paul Krivine, and Reid Simmons. Second generation expert systems : a step forward in knowledge engineering. In *Second generation expert systems*, pages 3–23. Springer, 1993.

[27] André Didierjean. Apprendre à partir d'exemples : abstraction de règles et/ou mémoire d'exemplaires ? *L'année psychologique*, 101(2) :325–348, 2001.

[28] R Dieng, O Corby, F Gandon, A Giboin, J Golebiowska, N Matta, and M Ribière. Knowledge management : Méthodes et outils pour la gestion des connaissances, 2005.

[29] Rose Dieng. *Méthodes et outils d'acquisition des connaissances.* PhD thesis, INRIA, 1990.

[30] Rose Dieng. Comparison of conceptual graphs for modelling knowledge of multiple experts. *Foundations of Intelligent Systems*, pages 78–87, 1996.

[31] Nathalie Dubus. *Développement d'un système expert dans le processus de planification des ressources en eau au Burkina Faso.* PhD thesis, Université Joseph Fourier (Grenoble), 1994.

[32] Nathalie Duclosson, Stéphanie Jean-Daubias, and Stéphanie Riot. Ambre-enseignant : un module partenaire de l'enseignant pour créer des problèmes. 2005.

[33] M Duribreux, P Caulier, and B Houriez. Application industrielle d'une approche mixte de modélisation des connaissances. *), Ingénierie des connaissances : évolutions récentes et nouveaux défis, Eyrolles et France Télécom R&D, Paris, France*, 2000.

[34] Valérie Emin and Jean-Philippe Pernin. Un environnement graphique de scénarisation pédagogique dirigé par les intentions. In *actes de la conférence TICE 2010*, pages 16–pages. to be published, 2010.

[35] Michel Fayol. *L'enfant et le nombre : du comptage à la résolution de problèmes*. Delachaux et Niestlé Paris, 1990.

[36] Mary L Gick and Keith J Holyoak. Schema induction and analogical transfer. *Cognitive psychology*, 15(1) :1–38, 1983.

[37] JE Gilbert and CY Han. Case-based reasoning applied to instruction method selection for intelligent tutoring systems. In *Rin Workshop proceedings of ITS*, pages 11–15, 2000.

[38] Monique Grandbastien. Teaching expertise is at the core of its research. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10 :335–349, 1999.

[39] Viviane Guéraud and Anne Lejeune. Une approche auteur pour la scénarisation et le suivi de situations d'apprentissage. In *EIAH 2011-Conférence Environnements Informatiques pour l'Apprentissage Humain*, pages 11–17, 2011.

[40] Nathalie Guin. *Reformuler et classer un problème pour le résoudre. L'architecture SYRCLAD et son application à quatre domaines*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 1997.

[41] Nathalie Guin. *Des meÌta-modeÌles pour guider lâeÌlicitation des connaissances en EIAH : contributions aÌ lâenseignement de meÌthodes et aÌ la personnalisation des activiteÌs*. PhD thesis, Université Claude Bernard-Lyon 1, 2014.

[42] N Guin-Duclosson. Syrclad : une architecture de résolveurs de problèmes permettant d'expliciter des connaissances de classification, reformulation et résolution. *Revue d'Intelligence Artificielle*, 13(2) :225–282, 1999.

[43] N Guin-Duclosson, S Jean-Daubias, and S Nogry. Le projet ambre : utiliser un paradigme d'apprentissage fondé sur le ràpc pour faire acquérir à des élèves une méthode de résolution de problèmes. *Raisonnement à partir de cas*, 2007.

[44] Pitrat Jacques. Métaconnaissance, futur de l'intelligence artificielle. *Hermès, Paris*, 1990.

[45] TM Khan. Case-based evaluation for student modelling. In *Workshop*, volume 5, pages 16–22, 2000.

[46] Kenneth R Koedinger and Vincent Aleven. An interview reflection on âintelligent tutoring goes to school in the big cityâ. *International Journal of Artificial Intelligence in Education*, 26(1) :13–24, 2016.

[47] Kenneth R Koedinger, Vincent Aleven, Neil Heffernan, Bruce McLaren, and Matthew Hockenberry. Opening the door to non-programmers : Authoring intelligent

tutor behavior by demonstration. In *Intelligent tutoring systems*, volume 3220, pages 7–10. Springer, 2004.

[48] Pierre Laforcade, Thierry Nodenot, and Christian Sallaberry. Un langage de modélisation pédagogique basé sur uml. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation (STICEF)*, pages 89–116, 2005.

[49] Sébastien Lallé. *Assistance to build and compare knowledge diagnostic techniques*. Theses, Université de Grenoble, December 2013.

[50] Pascal Leroux. *Machines partenaires des apprenants et des enseignants-Étude dans le cadre d'environnements supports de projets pédagogiques*. PhD thesis, Université du Maine, 2002.

[51] Noboru Matsuda, Victoria Keiser, Rohan Raizada, Arthur Tu, Gabriel Stylianides, William W Cohen, and Kenneth R Koedinger. Learning by teaching simstudent : Technical accomplishments and an initial use with students. In *International Conference on Intelligent Tutoring Systems*, pages 317–326. Springer, 2010.

[52] Antonija Mitrovic, Brent Martin, Pramuditha Suraweera, Konstantin Zakharov, Nancy Milik, Jay Holland, and Nicholas McGuigan. Aspire : an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2) :155–188, 2009.

[53] Antonija Mitrovic, Pramuditha Suraweera, Brent Martin, Konstantin Zakharov, Nancy Milik, and Jay Holland. Authoring constraint-based tutors in aspire. In *International Conference on Intelligent Tutoring Systems*, pages 41–50. Springer, 2006.

[54] Riichiro Mizoguchi and Jacqueline Bourdeau. Using ontological engineering to overcome ai-ed problems : Contribution, impact and perspectives. *International Journal of Artificial Intelligence in Education*, 26(1) :91–106, 2016.

[55] E Shiri A Mohammad, Esma Aïmeur, and Claude Frasson. Sara : A case-based student modelling system. In *European Workshop on Advances in Case-Based Reasoning*, pages 425–436. Springer, 1998.

[56] Allen Munro. Authoring simulation-centered learning environments with rides and vivids. In *Authoring Tools for Advanced Technology Learning Environments*, pages 61–91. Springer, 2003.

[57] Tom Murray. Authoring intelligent tutoring systems : An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10 :98–129, 1999.

[58] Tom Murray. An overview of intelligent tutoring system authoring tools : Updated analysis of the state of the art. In *Authoring tools for advanced technology learning environments*, pages 491–544. Springer, 2003.

[59] Tom Murray, Stephen Blessing, and Shaaron Ainsworth. *Authoring tools for advanced technology learning environments : Toward cost-effective adaptive, interactive and intelligent educational software*. Springer Science & Business Media, 2003.

[60] Roger Nkambou, Jacqueline Bourdeau, and Valéry Psyché. Building intelligent tutoring systems : An overview. *Advances in Intelligent Tutoring Systems*, pages 361–375, 2010.

[61] Thierry Nodenot. *Contribution à l'Ingénierie dirigée par les modèles en EIAH : le cas des situations-problèmes coopératives*. PhD thesis, Université de Pau et des Pays de l'Adour, 2005.

[62] Sandra Nogry. Faciliter l'apprentissage à partir d'exemples en situation de résolution de problèmes. 2005.

[63] Sandra Nogry, Nathalie Guin, and Stéphanie Jean-Daubias. Ambre-add : An its to teach solving arithmetic word problems. *Technology, Instruction, Cognition & Learning*, 6(1), 2008.

[64] Mary S Riley et al. Development of children's problem-solving ability in arithmetic. 1984.

[65] Aline Robert, Janine Rogalski, and Renan Samurcay. *Enseigner des méthodes*. Institut de Recherche pour L'Enseignement des Mathematiques, 1987.

[66] Marc Rogalski. Enseigner des méthodes des mathematiques. *Recherche em Didactiques des Mathématiques*, 1(1) :1–10, 1990.

[67] Marc Rogalski. Les concepts de l'eiao sont-ils indépendants du domaine. *L'exemple d'enseignement de méthodes en analyse. Recherches en Didactiques des Mathématiques*, 14(1.2) :43–66, 1994.

[68] Brian H Ross and Patrick T Kennedy. Generalizing from the use of earlier examples in problem solving. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 16(1) :42, 1990.

[69] Roger C Schank. Dynamic memory : A theory of learning in people and computers, 1982.

[70] A.H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, 1985.

[71] Alan H Schoenfeld. Metacognitive and epistemological issues in mathematical understanding. *Teaching and learning mathematical problem solving : Multiple research perspectives*, pages 361–380, 1985.

[72] Alan H Schoenfeld. Problem solving in context (s). *The teaching and assessing of mathematical problem solving*, 3 :82–92, 1989.

[73] Alan H Schoenfeld. Toward a theory of teaching-in-context. *Issues in education*, 4(1) :1–94, 1998.

[74] Guus Schreiber. *Knowledge engineering and management : the CommonKADS methodology*. MIT press, 2000.

[75] Guus Schreiber, Bob Wielinga, and Joost Breuker. *KADS : A principled approach to knowledge-based system development*, volume 11. Academic Press, 1993.

[76] Kay G Schulze, Robert N Shelby, Donald J Treacy, Mary C Wintersgill, Kurt Vanlehn, and Abigail Gertner. Andes : An intelligent tutor for classical physics. *Journal of Electronic Publishing*, 6(1), 2000.

[77] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering : principles and methods. *Data & knowledge engineering*, 25(1-2) :161–197, 1998.

[78] Pramuditha Suraweera, Antonija Mitrovic, and Brent Martin. The role of domain ontology in knowledge acquisition for itss. In *Intelligent Tutoring Systems*, pages 215–230. Springer, 2004.

[79] Pierre Tchounikine. Précis de recherche en ingénierie des eiah. 2009.

[80] Gheorghe Tecuci and Harry Keeling. Developing an intelligent educational agent with disciple. *International Journal of Artificial Intelligence in Education*, 10(3-4) :221–237, 1999.

[81] Wouter R Van Joolingen and Ton De Jong. Simquest. In *Authoring tools for advanced technology learning environments*, pages 1–31. Springer, 2003.

[82] Michael van Lent and John E Laird. Learning procedural knowledge through observation. In *Proceedings of the 1st international conference on Knowledge capture*, pages 179–186. ACM, 2001.

[83] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The andes physics tutoring system : Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3) :147–204, 2005.

[84] Maria Virvou and Maria Moundridou. A web-based authoring tool for algebra-related intelligent tutoring systems. *Educational Technology & Society*, 3(2) :61–70, 2000.

[85] C Vogel. Kod : une méthode pour le recueil et la modélisation de connaissances. *Journées internationales sur les systèmes experts et leurs applications, Avignon, France*, 1990.

[86] Bob J Wielinga. Reflections on 25+ years of knowledge acquisition. *International Journal of Human-Computer Studies*, 71(2) :211–215, 2013.

[87] Bob J Wielinga, A Th Schreiber, and Jost A Breuker. Kads : A modelling approach to knowledge engineering. *Knowledge acquisition*, 4(1) :5–53, 1992.

[88] Beverly Woolf. *Building Intelligent Interactive Tutors, Student-Centered Strategies for Revolutionizing E-Learning*. 01 2008.

[89] Beverly Park Woolf. *Building intelligent interactive tutors : Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.

[90] Ming Zhou, Jianwen Ren, Jianxun Qi, Dongxiao Niu, and Gengyin Li. Common-kads methodology for developing power grid switching orders systems. *Emerging Technologies in Knowledge Discovery and Data Mining*, pages 87–98, 2007.

Publications durant la thèse

Publications disponibles depuis https://liris.cnrs.fr/membres/?idn=adiattar&onglet=publis

Conférences internationales avec comité de lectures et actes

[1] A. Diattara, N. Guin, V. Luengo and A. Cordier (2017). « Preliminary evaluations of AMBRE-KB, An Authoring Tool to Elicit Knowledge to be Taught Without Programming ». [Extension de l'article publié à CSEDU : **soumis**]

[2] A. Diattara, N. Guin, V. Luengo and A. Cordier .: An Authoring Tool to Elicit Knowledge to be Taught without Programming. In: Paula Escudeiro, Gennaro Costagliola, Susan Zvacek, J.U. and B.M.M. (ed.) International Conference on Computer Supported Education. pp. 82–91. , Porto, Portugal (2017). **Article nominé pour le prix du meilleur article**.

[3] A. Diattara, N. Guin, V. Luengo and A. Cordier (2016). « Towards an Authoring Tool to Acquire Knowledge for ITS Teaching Problem Solving Methods ». European Conference on Technology Enhanced Learning (EC-TEL 2016), 16 septembre 2016, Lyon (France), pp. 575-578.

Conférences nationales avec comité de lectures et actes

[4] A. Diattara, "Vers un outil auteur pour des EIAH destinés à l'apprentissage de méthodes de résolution de problèmes : acquisition des connaissances de la méthode à enseigner," in RJCEIAH, 2016.

[5] A. Diattara, N. Guin, M. Lefevre, B. Kong Win Chang, and V. Luengo, "Vers un outil auteur pour des EIAH destinés à l'apprentissage de méthodes : formalisation des connaissances à acquérir." in TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION POUR L'ENSEIGNEMENT, 2014, pp. 180–185.

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

« Awa DIATTARA »

Thèse dirigée par **Vanda Luengo, Professeur à l'université Pierre et Marie Curie**, et
codirigée par **Nathalie GUIN, Maître de conférences à l'Université Claude Bernard Lyon1**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Acquisition de connaissances pour des EIAH destinés à enseigner des méthodes de résolution de problèmes

Thèse soutenue publiquement le **« 20 Octobre 2017 »**,
devant le jury composé de :

M. Serge GARLATTI
Professeur à IMT Atlantique, rapporteur
Mme Marie-Hélène ABEL
Professeur à l'Université de Technologie de Compiègne, rapporteur
M. Abdelhamid CHAACHOUA
Professeur à l'Université Grenoble Alpes, examinateur

Remerciements

Au-delà des remerciements formels, je tiens à exprimer ma profonde gratitude à mes encadrantes : Nathalie Guin pour son suivi constant, ses conseils, sa disponibilité, sa générosité, les conditions de travail ; Vanda Luengo pour ses critiques constructives, sa disponibilité, ses conseils avisés lors des moments clés. Durant ces années de thèse, elles m'ont accompagnée, chouchoutée et soutenue. Je les remercie aussi pour la confiance accordée tout au long de cette thèse et de m'avoir fait découvrir le domaine des EIAH. Ce fut un réel plaisir de travailler avec elles. Et j'espère avoir acquis un peu de leur rigueur scientifique afin de devenir un jour, une enseignante chercheuse à la hauteur de leur mentor.

Je remercie également Amélie CORDIER pour sa disponibilité, pour son approche pédagogique et de m'avoir fait bénéficier de sa grande expérience et de ses conseils.

Mes remerciements aussi à Marie LEFEVRE pour ses remarques pertinentes qui ont permis d'améliorer mon travail.

J'adresse également mes remerciements à la région Auvergne Rhône Alpes qui a financé cette thèse.

Mes remerciements vont également à l'endroit de Madame Marie-Hélène ABEL et Monsieur Serge GARLATTI pour avoir rapporté sur cette thèse. Les jugements et remarques qu'ils ont fournis à l'égard de mon travail ont été pertinents et très enrichissants. J'adresse également mes remerciements à Monsieur Hamid CHAACHOUA pour m'avoir fait l'honneur d'examiner ce travail de recherche.

Je remercie l'ensemble des membres du LIRIS et du LIG, en particulier, les collègues des équipes TWEAK et MeTAH pour leur accueil chaleureux, leur sympathie, les échanges et conseils lors des réunions de présentation de travail, les moments agréables

passés pendant ces années de thèse.

Merci enfin à Isabelle, Brigitte et Catherine d'avoir rendu si facile le côté administratif de la recherche.

Ensuite, vient le tour de mes amis sans qui cette thèse aurait été interminable : Arame, Thierno, Lycka, Laure Esnault, Pascaline, Adamaz, Oumou salam, Nabou, Mamy, Adja, Assitan, Lucie, les amies de Mariama Bâ, les amis de MASS, les amis de Sanar, les amis de Microsoft, les amis de Montpellier, les amis de Lyon... et tous les autres que j'aurais oubliés de citer.

Mention spéciale à ma famille qui m'a toujours soutenue. Merci à mes parents qui m'ont permis d'arriver jusque-là. Mention spéciale à ma maman, qui a toujours cru en moi. C'est grâce à elle que j'ai pu tenir pendant toutes ces années d'étude. Mes remerciements également à ma belle-famille pour leur soutien.

Et pour finir, je remercie mon merveilleux mari, pour son amour inconditionnel, son soutien et ses encouragements. Mamour, merci de m'avoir suivi à Lyon, de m'avoir supporté et de me supporter encore, merci pour ta compréhension et ta patience.

Un grand merci à tous ceux qui de près ou de loin ont participé à la réussite de ce travail.

Résumé

Cette thèse aborde la problématique de l'acquisition des connaissances dans le cadre de la conception des Environnements Informatiques pour l'Apprentissage Humain (EIAH). Dans le contexte spécifique de ce travail, nous nous intéressons à des EIAH destinés à enseigner des méthodes de résolution de problèmes. De telles méthodes permettent, dans un domaine donné, de reconnaître la classe d'un problème et d'être capable de savoir quelle technique de résolution appliquer pour le résoudre. Le coût de conception de ces EIAH est cependant très élevé, en particulier du fait de l'élicitation des connaissances qui nécessite non seulement une expertise dans le domaine concerné mais également en programmation.

Afin de réduire le coût de conception de ces EIAH et permettre à des auteurs (enseignants, professionnels ou pédagogues plus experts) de pouvoir éliciter sans programmer les connaissances nécessaires à l'EIAH pour résoudre des problèmes, nous proposons un processus d'acquisition interactive de ces connaissances. Ce processus est mis en œuvre à travers la conception d'un outil auteur : AMBRE-KB. Pour ce faire, nous avons d'abord proposé des méta-modèles qui permettent de décrire la forme des connaissances à acquérir. Ces connaissances ne sont pas celles d'un expert dans un domaine, mais les connaissances telles qu'on voudrait qu'elles fonctionnent chez l'apprenant à l'issue de l'apprentissage. En s'appuyant sur ces méta-modèles, nous avons ensuite proposé un processus d'acquisition de ces connaissances qui assiste l'auteur lors de l'élicitation des connaissances nécessaires, en lui permettant de construire un modèle de connaissances spécifique à un domaine.

Deux expérimentations ont été menées afin de mettre à l'essai AMBRE-KB. Les résultats de la première expérimentation - réalisée sur trois domaines - ont conduit à faire évoluer les fonctionnalités et les interfaces de AMBRE-KB afin d'améliorer l'interactivité

du système, mais également permettre à l'auteur d'avoir un feedback sur le processus d'élicitation. Cette nouvelle version de AMBRE-KB a permis de mener une seconde expérimentation réalisée sur quatre domaines. Celle-ci a permis de mettre en évidence les points forts et les points faibles de AMBRE-KB :

— Pour un domaine donné, AMBRE-KB permet de générer les modèles de connaissances permettant au résolveur de résoudre les problèmes relevant de la méthode à enseigner

— Les interfaces proposées sont faciles à utiliser dans l'ensemble

— Les auteurs ont apprécié l'assistance fournie, en particulier le feedback

— Une amélioration de certaines interfaces et un feedback plus poussé faciliteraient davantage le processus d'élicitation.

Mots clés : EIAH – Outil auteur – Acquisition de connaissances – Enseignement de méthodes – Modèles de connaissances.

Abstract

The general issue we addressed in this thesis is the challenge of knowledge elicitation in Intelligent Tutoring Systems (ITS). In the context of this work, we are interested in ITS teaching problem solving methods. Teaching methods for solving problems consists in teaching students how to think about the problem before starting its resolution. In a given field, such a method is based on a categorization of problems. Knowing to recognize the class of a problem enables students to choose the resolution technique associated with this class. However, designing such ITS is tedious and costly, and specially requires expertise in the application domain and in programming.

In order to reduce the design cost of these ITS and to enable an author (for example a teacher) to be able to elicit knowledge needed without programming, we propose an interactive knowledge elicitation process. This process is implemented through the design of an authoring tool : AMBRE-KB. For that, we first propose meta-models for the knowledge to be acquired. This knowledge is not an expert knowledge, but knowledge such as we would want that it works at the end of the learning. Next, we propose a knowledge acquisition process based on these meta-models, which enable the author to be assisted in the elicitation process enabling him/her to build specific knowledge models for a given domain.

Two experiments were conducted to evaluate the knowledge acquisition process and its implementation in the AMBRE-KB tool. The results of the first experiment, conducted in three domains, led to modify the features and the interfaces of AMBRE-KB, so that to improve the interactivity of the tool, and to enable authors to have feedback about the knowledge elicitation. The second experiment based on the new version of AMBRE-KB, was conducted in four domains. The results of this experiment enable to highlight the strengths and the limits of AMBRE-KB :

— For a given domain, AMBRE-KB can produce the knowledge models needed by the solver to solve problems about the method to be taught

— On the whole, interfaces are easy to use

— The assistance provided is useful and particularly the feedback

— An improvement of the interfaces and of the interactivity of the system would facilitate even more the elicitation process.

Keywords :

ITS – Authoring tool – Knowledge acquisition – Teaching methods – Knowledge models

Table des matières

Introduction générale **21**

0.1 Introduction 22

0.2 Problématique et objectifs de la thèse 25

0.3 Contributions 27

0.4 Structure du document 27

I CONTEXTE ET ÉTAT DE L'ART **31**

1 Cadre des recherches : le projet AMBRE **32**

1.1 Principe des EIAH AMBRE 34

1.1.1 Des EIAH pour enseigner des méthodes 34

1.1.2 Un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC) 35

1.2 Comment appliquer une méthode de résolution de problèmes ? 37

1.2.1 Connaissances pour appliquer une méthode de résolution de problèmes 37

1.2.2 Processus de résolution de problèmes 41

1.3 Comment guider l'apprenant dans sa résolution de problèmes ? 42

1.3.1 Connaissances pour guider l'apprenant durant son apprentissage . 42

1.3.2 Processus de raisonnement 44

1.4 Conclusion 45

2 État de l'art **47**

2.1 De l'acquisition des connaissances à l'Ingénierie des Connaissances 49

2.2 Méthodes et techniques d'acquisition de connaissances en Ingénierie des Connaissances 50

2.2.1 Méthodes empiriques 50

2.2.2 Méthodes analytiques 52

2.3 IC et EIAH 58

2.4 Outils auteurs 59

2.4.1 Outils auteurs pour la création de parcours pédagogiques 59

2.4.2 Outils auteurs pour la création de tuteurs intelligents 60

2.5 Synthèse des principaux verrous 62

II CONTRIBUTIONS THÉORIQUES **66**

3 Des méta-modèles pour guider l'élicitation des connaissances **67**

3.1 Comment décrire les problèmes d'un domaine ? 69

3.1.1 Méta-modèle du vocabulaire permettant de décrire des problèmes 70

3.1.2 Méta-modèle permettant de décrire des problèmes 71

3.2 Comment définir les connaissances sur la méthode à enseigner ? 73

3.2.1 Méta-modèle des connaissances de classification 74

3.2.2 Méta-modèle des connaissances de reformulation 77

3.2.3 Méta-modèles des connaissances de résolution 82

3.3 Comment définir les connaissances permettant d'accompagner l'apprenant ? 86

3.3.1 Méta-modèle des connaissances d'aide 87

3.3.2 Méta-modèle des connaissances de diagnostic 89

3.3.3 Connaissances d'explications sur les erreurs 91

3.4 Conclusion 93

4 Un processus interactif d'acquisition des connaissances pour des EIAH destinés à enseigner des méthodes 95

4.1 Étapes du processus d'acquisition des connaissances 97

4.1.1 Définition du vocabulaire pour le domaine 98

4.1.2 Description des problèmes à résoudre 99

4.1.3 Élicitation des connaissances sur la méthode 100

4.1.4 Conception de l'interface de l'EIAH 104

4.1.5 Élicitation des connaissances destinées à accompagner l'apprenant 104

4.2 Feedback après l'élicitation des connaissances 104

4.3 Synthèse 106

III MISE EN ŒUVRE ET ÉVALUATIONS 109

5 AMBRE-KB : un outil auteur pour des EIAH destinés à enseigner des méthodes 110

5.1 Élicitation des connaissances avec AMBRE-KB 113

5.1.1 Définition du vocabulaire 115

5.1.2 Description des problèmes à résoudre 116

5.1.3 Élicitation des connaissances sur la méthode 118

5.2 Génération des modèles de connaissances 126

5.3 Tests des modèles de connaissances 130

5.4 Conclusion 135

6 Évaluations 137

6.1 Mise à l'essai de AMBRE-KB 139

6.1.1 Première expérimentation 139

6.1.2 Seconde expérimentation 143

6.2 Bilan des expérimentations 154

Introduction générale 156

6.3 Synthèse de nos travaux 157

6.4 Discussion 159

6.5 Perspectives 161

Annexes **163**

Bibliographie **206**

Table des figures

1 Exemple de résolution du problème P1 en utilisant une technique de résolution 24

1.1 Fonctionnement du résolveur de problèmes dans AMBRE [40] 38

1.2 Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif 39

2.1 KADS : cycle de vie d'un système expert [31] 53

3.1 Méta-modèle permettant de décrire le vocabulaire d'un domaine 71

3.2 Méta-modèle permettant de décrire un problème en s'appuyant sur le vocabulaire 72

3.3 Lien entre les connaissances de classification, de reformulation et de résolution 74

3.4 méta-modèle permettant de décrire une classe de problèmes 75

3.5 Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif 76

3.6 Lien entre connaissances de classification et connaissances de reformulation 78

3.7 Méta-modèle permettant de définir une fonction de calcul. 79

3.8 Méta-modèle permettant de décrire une règle 80

3.9 Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif, avec les règles de reformulation 81

3.10 Lien entre connaissances de reformulation et connaissances de résolution 83

3.11 Méta-modèle permettant de décrire une technique de résolution 84

3.12 Conjugaison du verbe Parler au présent de l'indicatif 85

3.13 Lien entre connaissances sur la méthode et celle destinées à accompagner l'apprenant 87

3.14 Exemple de fenêtre d'aide à deux niveaux dans l'étape d'adaptation dans l'EIAH AMBRE-add. le deuxième niveau active le coloriage facilitant la mise en correspondance du problème-type et du problème à résoudre . . 88

3.15 Aide à l'analogie à l'étape d'adaptation dnas AMBRE-add 89

3.16 Méta-modèle des connaissances d'aide 89

3.17 Aide à l'analogie à l'étape d'adaptation dans AMBRE-add 90

3.18 Exemple de diagnostic réalisé par le système 91

3.19 Méta-modèle permettant de décrire un message d'explications 92

3.20 Exemple de message d'explications généré à partir de la réponse de l'apprenant 93

4.1 Approche générale du processus d'élicitation 98

5.1 Approche générale de AMBRE-KB 112

5.2 Vue d'ensemble de AMBRE-KB 113

5.3 Une partie du vocabulaire pour le domaine des problèmes additifs 115

5.4 Description des objets du problème réunion_1 117

5.5 Définition d'une classe de problèmes en utilisant AMBRE-KB 119

5.6 Une partie de l'arbre de classification pour les problèmes additifs 120

5.7 Une partie de l'arbre de classification pour les problèmes additifs 122

5.8 représentation d'une partie des prémisses de la regle_reunion_operande dans AMBRE-KB 124

5.9 représentation de la partie conclusion de la regle_reunion_operande dans AMBRE-KB 125

5.10 représentation de la technique tech_réunion_opérande dans ambre-kb . 126

5.11 Exemple de feedback lorsque la résolution réussit. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème 130

5.12 Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a réussi 131

5.13 Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème 132

5.14 Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée parce que le système n'a pas trouvé de règle permettant de conclure sur la valeur d'un attribut 133

5.15 Exemple de feedback lorsque la résolution a échoué lors de l'application d'un technique de résolution. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème 134

5.16 Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a échoué lors de l'application d'une technique de résolution . 135

6.1 Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie 151

6.2 Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie 152

3 Définiiton du vocabulaire pour le pluriel des noms avec AMBRE-KB . . . 166

4 Définiton du problème pluriel_detail avec AMBRE-KB 167

5 Une partie de l'arbre de classification pour le pluriel des noms en "al", "ail", "ou", "eu, "au", "eau" 168

6 une aprtied e l'arbre de classification 172

7 Définition de la règle *regle_ail_exception* en utilisant AMBRE-KB 173

8 Définition de la technique associée à la classe *c_plu_ail_pas_ex* en utilisant AMBRE-KB 175

9 Arbre de classification pour les équations défini par le sujet 1 180

10 Arbre de classification pour les équations défini par le sujet 2 181

11 Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie 188

12 Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie 189

13 Une partie de l'arbre de classification pour la formation des adverbes en "ment" 192

Liste des tableaux

3.1 Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème 82

3.2 Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème 85

5.1 Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème 123

5.2 Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé 123

5.3 Exemple de technique permettant de résoudre les problèmes de la classe *réunion* 126

5.4 Modèle du problème *reunion_1* en Prolog généré par AMBRE-KB . . . 127

5.5 Une partie du modèle de connaissances représentant les connaissances de classification (cf. Figure 5.6) généré par AMBRE-KB en Prolog 128

5.6 Une partie du modèle de connaissances décrivant les règles généré par AMBRE-KB en Prolog 129

5.7 Une partie du modèle de connaissances décrivant les connaissances de résolution, généré par AMBRE-KB en Prolog 130

6.1 Critère d'évaluation pour la première expérimentation 140

6.2 Critère retenu pour évaluer le vocabulaire d'un domaine. 147

6.3 Critères retenus pour évaluer la flexibilité de AMBRE-KB. 147

6.4 Critères d'évaluation de l'ergonomie de l'interface permettant de définir le vocabulaire 148

6.5 Domaines testés 149

6.6 Résultats des expérimentations pour les équations par les sujets 1 et 2 . . 149

6.7 Vocabulaire défini par le sujet 1. 150

6.8 Vocabulaire défini par le sujet 2. 150

6.9 Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2 150

6.10 Résultats des expérimentations pour le pluriel des noms et la formation des adverbes en "ment" testés par le même sujet 153

6.11 Domaines testés 155

12 vocabulaire proposé pour le pluriel noms terminés par "al", "ail", "ou", "eu, "au", "eau" en français 165

13 Modèle descriptif du problème pluriel_bail généré par AMBRE-KB . . . 168

15 Critères retenus pour évaluer le vocabulaire d'un domaine 174

16 Exemple de modèles de connaissances (décrivant des techniques) pour le pluriel des noms générés par AMBRE-KB 176

17 Comparaison des résultats des expérimentations pour les équations par les sujets 1 et 2 177

18 Vocabulaire défini par le sujet 1. 178

19 Vocabulaire défini par le sujet 2. 179

20 Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé 182

22 Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2 185

23 vocabulaire pour le calcul d'aire proposé par le sujet 1 186

24 vocabulaire pour le calcul d'aire proposé par le sujet 2 187

25 Exemple de règles de reformulation définies pour le calcul d'aire en géométrie 190

26 Exemple de techniques de résolution définies pour le calcul d'aire en géométrie 190

27 vocabulaire pour la formation des adverbes français en "ment" 191

28 Critères retenus pour évaluer le vocabulaire d'un domaine 195

29 Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné 195

30 Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné 195

31 Critères retenus pour évaluer l'élicitation des connaissances de reformulation pour un domaine donné 196

32 Critères retenus pour évaluer l'élicitation des connaissances de résolution pour un domaine donné 196

33 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la définition du vocabulaire pour un domaine donné 196

34 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la Description des problèmes à résoudre pour un domaine donné 197

35 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de classification pour un domaine donné 197

36 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de reformulation pour un domaine donné 197

37 Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de résolution pour un domaine donné 198

38 Critères retenus pour évaluer la flexibilité proposée dans AMBRE-KB . . 198

39 Critères retenus pour évaluer les fonctionnalités de guidage proposées dans AMBRE-KB 198

40 Critères retenus pour évaluer les fonctionnalités de contrôle de connaissances proposées dans AMBRE-KB 199

41 Critères retenus pour évaluer les fonctionnalités de feedback proposées dans AMBRE-KB 199

Introduction générale

Ce chapitre donne un aperçu général du contenu de ce manuscrit. Il présente tout d'abord le sujet abordé et dégage la problématique ainsi que les objectifs de cette thèse. Les propositions et les travaux réalisés sont présentés par la suite. Une section décrivant l'organisation du document clôt le chapitre.

Sommaire

0.1 Introduction 22
0.2 Problématique et objectifs de la thèse 25
0.3 Contributions 27
0.4 Structure du document 27

0.1 Introduction

Les travaux présentés dans ce mémoire s'inscrivent dans le cadre des recherches menées sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH). Ce domaine de recherche est dédié à la conception d'environnements informatiques destinés à favoriser l'apprentissage - la construction de connaissances chez un apprenant [10] - dans des contextes variés. Il peut en effet s'agir d'un apprentissage en classe avec un enseignant, ou à distance lors d'une activité en toute autonomie ou en complément d'un enseignement, dans l'optique de s'entraîner, de s'évaluer, de remédier à des difficultés, ou d'approfondir certains sujets.

Le champ de recherche des EIAH est un domaine pluridisciplinaire, à la croisée de l'informatique, de la psychologie cognitive, de la didactique des disciplines et des sciences de l'éducation. Nos recherches relèvent des apports de l'informatique à la question de l'ingénierie des EIAH, champ de recherche qui porte sur les « *questions liées aux concepts, méthodes, théories, techniques et technologies utiles à la conception des EIAH* » [79]. Au sein de ce champ, nous nous intéressons plus particulièrement à la manière de représenter, au niveau de l'environnement informatique, les connaissances qui permettent d'accompagner l'apprenant au cours de son apprentissage. Il s'agit d'une part, de représenter explicitement les connaissances enjeu de l'apprentissage, et d'autre part, de mettre en œuvre des mécanismes permettant au système informatique interagissant avec l'apprenant de s'appuyer sur ces connaissances afin de susciter chez lui un apprentissage.

Les EIAH sont extrêmement variés. Dans le cadre de cette thèse, nous nous intéressons à des situations d'apprentissage individuel, dans des domaines où il est possible d'établir une classification des problèmes et des outils de résolution correspondants. Dans ces situations, des apprenants de divers niveaux interagissent avec l'environnement informatique autour d'une tâche de résolution de problèmes, sans collaboration ni interaction entre eux. Dans le contexte spécifique de ce travail, nous nous focalisons sur des EIAH destinés à enseigner des méthodes de résolution de problèmes [65, 67, 71, 72]. De telles méthodes, souvent issues de recherches en didactique des disciplines, permettent - dans un domaine donné - de reconnaître la classe d'un problème et ainsi d'en déduire la technique de résolution appropriée.

Afin d'illustrer nos propos, prenons le cas concret de la conjugaison française, où les problèmes à résoudre consistent à conjuguer à toutes les personnes un verbe V à un temps T donné. L'utilisation d'une méthode pour résoudre ces problèmes va alors consister à déterminer la classe du problème, afin de pouvoir appliquer par la suite la bonne technique de résolution : comment déterminer le radical du verbe ainsi que les terminaisons adéquates pour les différentes personnes au temps demandé.

Exemple de problème :
P1 « *Conjuguer le verbe parler au présent de l'indicatif* ».

Classe du problème à résoudre :
La classification des problèmes permet d'identifier la classe de ce problème : *conjuguer un verbe du premier groupe régulier au présent de l'indicatif.*

Technique de résolution à appliquer :
Pour les problèmes de cette classe, la technique de résolution à appliquer est la suivante :

- Déterminer le radical du verbe : enlever la terminaison *er* de l'infinitif du verbe (parler) pour obtenir son radical (parl) ;
- Déterminer la correspondance entre personnes et terminaisons : au présent de l'indicatif, pour les verbes réguliers du premier groupe, les terminaisons des différentes personnes *je, tu, il/elle, nous, vous, ils/elles* sont respectivement *e, es, e, ons, ez, ent*.
- Construire la forme conjuguée aux différentes personnes en utilisant le radical et les terminaisons identifiés.

La **Figure 1** illustre ainsi la solution au problème *P1* après application de la technique de résolution.

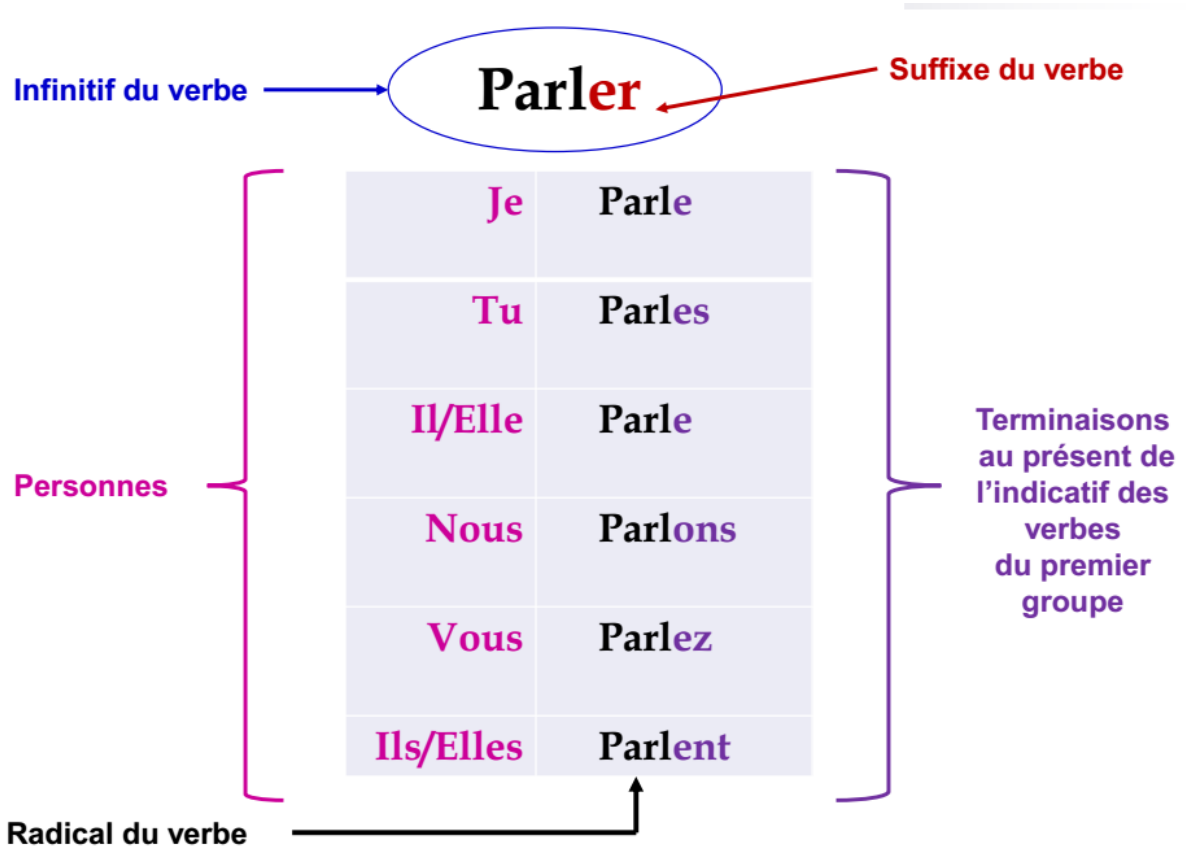

FIGURE 1 – Exemple de résolution du problème P1 en utilisant une technique de résolution

L'objectif des EIAH n'est pas bien sûr de remplacer l'enseignant, mais de lui fournir des outils destinés à accompagner son enseignement. Il peut s'agir d'environnements qu'il fait utiliser à ses apprenants soit en classe en leur compagnie, soit en totale autonomie chez eux, afin d'approfondir certains sujets, de s'entraîner, de s'évaluer, etc. Cependant, pour que ces environnements soient utiles aux enseignants, il faut qu'ils soient adaptés aux besoins et pratiques de chacun d'entre eux. Pour ce faire, il faut donc donner à l'enseignant les moyens de s'approprier l'environnement, soit en le configurant, soit en lui permettant de le concevoir lui-même.

Pour répondre à cette problématique, certains chercheurs comme Leroux [50] et Bruillard [16] ont proposé d'associer les enseignants à la conception des EIAH, afin de favoriser leur acceptabilité par leurs collègues. Toutefois, cela ne garantit pas pour autant de répondre au besoin individuel de chaque enseignant. D'autres chercheurs ont quant à eux proposé de donner la possibilité à chaque enseignant d'achever lui-même la

conception de son EIAH, grâce un module dédié et via lequel il peut paramétrer l'environnement destiné à ses apprenants [38]. Ce module lui permet ainsi de choisir ou de créer les activités que ses apprenants auront à effectuer : exemples AMBRE-enseignant [32] ou encore aplusix [15]. Cependant, comme le souligne [20] pour favoriser la réingénierie de tels outils, il faut également fournir à l'enseignant des outils lui permettant d'observer et d'analyser l'activité de ses élèves. D'autres encore se sont penchés sur les outils auteurs, afin de permettre à l'enseignant de concevoir lui-même les EIAH correspondant à ses besoins. Un outil auteur est un environnement informatique permettant de construire un EIAH, en partie ou dans sa globalité, avec un coût réduit voire nul en programmation

Dans le cadre spécifique de cette thèse, nous nous intéressons à des outils auteurs destinés à concevoir des EIAH dits centrés connaissances, c'est-à-dire des environnements informatiques dans lesquels l'ensemble des interactions entre l'apprenant et le système sont fondés sur une représentation explicite des connaissances [89]. Ces connaissances ne sont pas celles d'un expert dans un domaine, mais les connaissances telles qu'on voudrait qu'elles fonctionnent chez l'apprenant à l'issue de l'apprentissage [23]. Nous ciblons particulièrement des utilisateurs experts des connaissances de la méthode à enseigner (enseignant, professionnel, pédagogue plus expert, etc.), familiarisés avec l'outil informatique, avec ou sans compétences en programmation. Ces experts auront pour tâche d'expliciter l'ensemble des connaissances nécessaires à l'EIAH pour accompagner l'apprenant durant son apprentissage. L'enjeu est alors d'assister ces experts dans leur tâche en automatisant une partie du processus de création d'activités pédagogiques. Dans la suite de ce document, ces utilisateurs experts seront dénommés auteurs.

0.2 Problématique et objectifs de la thèse

Notre travail de recherche aborde la question de l'élicitation des connaissances dans le cadre de la conception des EIAH. Notre problématique de recherche concerne plus particulièrement l'élicitation de connaissances à travers la conception et le développement d'un outil auteur. La **formalisation des différents types de connaissances à éliciter** demeure une des premières questions auxquelles nous nous sommes confrontées. En effet, plusieurs types de connaissances interviennent lors de la conception d'un EIAH destiné à enseigner des méthodes. Outre les connaissances permettant la résolution des problèmes, il existe d'autres types de connaissances qui vont permettre de guider l'ap-

prenant durant son apprentissage, en lui fournissant aide, diagnostic de ses réponses et explications sur ses erreurs. Il s'agira donc dans un premier temps de proposer des méta-modèles de connaissances qui permettront, pour un domaine donné, d'exprimer les modèles de connaissances nécessaires à la conception d'un EIAH de ce type.

Une fois les différents méta-modèles de connaissances à acquérir formalisés, il faut ensuite réfléchir sur un **processus d'élicitation de ces connaissances**. Il s'agira alors de permettre à des non-programmeurs, en s'appuyant sur ces méta-modèles, de pouvoir éliciter les connaissances nécessaires à l'EIAH. Dans le cadre spécifique de cette thèse, nous nous focaliserons uniquement sur l'élicitation des connaissances permettant au système de résoudre des problèmes. Néanmoins, puisque les connaissances permettant de guider l'apprenant sont fortement dépendantes des tâches que ce dernier doit accomplir lors de sa résolution de problèmes, et donc de l'interface de l'EIAH, il faut gérer l'articulation entre la conception de l'interface de l'EIAH et l'élicitation de ces connaissances.

Une autre question très importante dans le cadre de ce travail est celle de l'**assistance** à fournir à l'auteur lors de l'élicitation des connaissances. En effet, pour permettre une acquisition aisée des connaissances, il est important d'assister l'auteur durant tout le processus d'élicitation en lui proposant des fonctionnalités d'aide, de guidage et de contrôle de ces connaissances. Enfin, une fois le processus d'élicitation terminé, une dernière question à laquelle il faut répondre est : **comment permettre à l'auteur d'avoir un feedback sur les connaissances élicitées ?** En effet, il est très important de donner à l'auteur les moyens de contrôler les connaissances élicitées, de savoir s'il a élicité toutes les connaissances nécessaires, et de pouvoir corriger et/ou compléter les connaissances manquantes le cas échéant.

Ces questions doivent être traitées dans l'objectif de proposer un processus d'acquisition des connaissances qui permettra à un auteur, informaticien ou non, de pouvoir expliciter les connaissances nécessaires à l'EIAH sans aucune programmation. Nous nous posons donc comme contrainte de proposer un processus interactif d'acquisition des connaissances, indépendant du domaine, qui puisse être implémenté dans un outil auteur, et dans lequel l'auteur est assisté tout du long. L'outil auteur à développer doit permettre à l'auteur d'éliciter les connaissances qui permettront au système de résoudre les problèmes pour un domaine donné. Il doit également permettre à l'auteur d'avoir un feedback sur le processus, de sorte à pouvoir corriger/compléter les connaissances au besoin.

0.3 Contributions

Afin de répondre au premier point de notre problématique, nous proposons des méta-modèles pour formaliser les différents types de connaissances à acquérir. Ces méta-modèles permettent, pour un domaine donné, de définir la forme des connaissances à acquérir pour la conception d'un EIAH destiné à enseigner une méthode de résolution de problèmes. La réponse au second point est un processus interactif d'acquisition de connaissances fondé sur ces méta-modèles et indépendant du domaine. Il est mis en ?uvre à travers la conception d'un outil auteur.

Pour un domaine donné, cet outil auteur permet d'acquérir les différents types de connaissances nécessaires à la résolution des problèmes, et génère les modèles de connaissances qui constituent les bases de connaissances de l'EIAH. Pour chaque type de connaissances, l'outil auteur propose également une assistance à l'auteur dans le processus d'élicitation, et lui permet d'obtenir un feedback une fois cette étape effectuée. Deux expérimentations pour évaluer le processus d'acquisition des connaissances et sa mise en ?uvre dans l'outil auteur ont été menées. La première expérimentation a permis de valider que pour un domaine donné, les modèles de connaissances générés par l'outil auteur permettent à l'EIAH de résoudre des problèmes. Les résultats de cette expérimentation ont également conduit à une extension de l'outil auteur afin d'améliorer l'interactivité du système, mais aussi permettre à l'auteur d'avoir un feedback sur les connaissances explicitées. Cette nouvelle version de l'outil auteur a permis de mener une seconde expérimentation qui s'est conclue avec des résultats satisfaisants concernant le processus d'élicitation, les interfaces proposées, ainsi que l'assistance fournie et en particulier le feedback

0.4 Structure du document

Après cette introduction générale, le reste de ce manuscrit s'articule autour de trois grandes parties.

La première partie est consacrée à la description du cadre applicatif de la thèse ainsi qu'à un état de l'art sur le sujet.

Chapitre 1 : ce chapitre présente le contexte applicatif de nos recherches : le projet AMBRE. Une première section décrit le principe des EIAH AMBRE en précisant ce qu'est l'enseignement de méthodes. Une autre section présente le Raisonnement à Partir de Cas (RàPC) avant de montrer comment il est utilisé dans le cadre du projet AMBRE pour faciliter l'apprentissage de méthodes de résolution de problèmes. Ce chapitre se termine par la présentation des différents types de connaissances nécessaires à la conception d'un EIAH AMBRE. Ces connaissances sont, dans un domaine donné, regroupées en deux grandes catégories : celles permettant de résoudre des problèmes, et celles destinées à accompagner l'apprenant en lui fournissant aide, diagnostic de ses réponses et explications sur ses erreurs.

Chapitre 2 : l'objectif de ce chapitre est de présenter un état de l'art sur les fondements théoriques de nos travaux de thèse. Après une présentation de quelques principales techniques d'acquisition de connaissances les plus répandues en Ingénierie des Connaissances (IC), nous montrons comment elles sont utilisées dans le domaine des EIAH par les outils auteurs les plus proches de notre problématique. Ce chapitre s'achève par une synthèse des principaux verrous ou manques actuels par rapport à notre problématique.

La deuxième partie de ce rapport expose les différentes contributions théoriques apportées dans cette thèse.

Chapitre 3 : ce chapitre présente les différents méta-modèles proposés pour guider l'élicitation des connaissances. Ces méta-modèles permettent de définir la forme des connaissances à acquérir et de contraindre le processus d'acquisition de connaissances proposé. Nous présentons dans un premier temps des méta-modèles pour guider l'élicitation des connaissances sur la méthode à enseigner, et dans un deuxième temps des méta-modèles des connaissances destinées à accompagner l'apprenant durant son apprentissage.

Chapitre 4 : ce chapitre décrit le processus d'acquisition des connaissances que nous avons proposé pour la conception d'EIAH destinés à enseigner une méthode de résolution dans un domaine donné. Il présente également les fonctionnalités d'assistance proposées afin d'aider l'auteur dans le processus d'élicitation de connaissances.

La troisième partie présente la mise en œuvre de nos contributions théoriques ainsi que leur évaluation.

Chapitre 5 : ce chapitre explique le fonctionnement général de l'outil auteur que nous proposons : AMBRE-KB. Nous présentons d'abord l'approche générale adoptée pour sa conception. Ensuite, nous illustrons l'utilisation de AMBRE-KB, en montrant comment éliciter, pour un domaine donné, les différentes connaissances nécessaires au système pour résoudre des problèmes.

Chapitre 6 : ce chapitre expose les différentes évaluations conduites pour valider nos contributions. Nous présentons notamment les expérimentations menées pour mettre à l'essai AMBRE-KB. L'objectif de ces expérimentations est de vérifier que les connaissances générées par AMBRE-KB permettent au système de résoudre les problèmes pour un domaine donné, mais aussi d'évaluer l'ergonomie des interfaces, l'assistance fournie avec AMBRE-KB, et le feedback proposé à l'auteur

Enfin, une **conclusion générale** fait le bilan des contributions, présente une discussion des résultats obtenus et propose des perspectives d'enrichissement et de continuité des travaux présentés dans ce manuscrit.

Première partie

CONTEXTE ET ÉTAT DE L'ART

Chapitre 1

Cadre des recherches : le projet AMBRE

Ce chapitre se consacre pour l'essentiel à la description du cadre applicatif de nos recherches : le projet AMBRE (Apprentissage de Méthodes Basé sur le Raisonnement à partir de l'Expérience) [43]. Ce projet de recherche pluridisciplinaire a pour objectif de concevoir des environnements d'apprentissage s'appuyant sur les étapes du cycle du raisonnement à partir de cas (RàPC) pour guider l'acquisition de méthodes de résolution de problèmes.

La première partie de ce chapitre se focalise sur le principe des EIAH AMBRE en précisant tout d'abord ce qu'est l'enseignement de méthodes. Elle se conclut par la présentation du RàPC et de son utilisation dans le cadre du projet AMBRE afin de faciliter l'apprentissage de méthodes de résolution de problèmes.

La seconde partie de ce chapitre est consacrée à la présentation des différents types connaissances nécessaires à la conception d'un EIAH AMBRE. Elle décrit d'abord les connaissances nécessaires pour résoudre des problèmes dans un domaine donné en appliquant une méthode de résolution, puis les connaissances destinées à accompagner l'apprenant durant son apprentissage en lui fournissant aide, diagnostic et explications sur ses erreurs.

Sommaire

1.1 Principe des EIAH AMBRE 34
1.1.1 Des EIAH pour enseigner des méthodes 34
1.1.2 Un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC) 35
1.2 Comment appliquer une méthode de résolution de problèmes ? 37
1.2.1 Connaissances pour appliquer une méthode de résolution de problèmes 37
1.2.2 Processus de résolution de problèmes 41
1.3 Comment guider l'apprenant dans sa résolution de problèmes ? 42
1.3.1 Connaissances pour guider l'apprenant durant son apprentissage 42
1.3.2 Processus de raisonnement 44
1.4 Conclusion 45

1.1 Principe des EIAH AMBRE

1.1.1 Des EIAH pour enseigner des méthodes

L'objectif du projet AMBRE est la conception d'environnements informatiques destinés à enseigner des méthodes pour résoudre des problèmes dans un domaine donné. Il se fonde sur des recherches en didactique des disciplines qui proposent d'enseigner explicitement des méthodes text [65, 71, 73].

Une méthode peut être définie comme « *la description d'un ensemble d'activités du sujet, portant sur l'analyse et le classement des problèmes à résoudre dans un domaine assez précis, l'utilisation des outils et des techniques disponibles, les stratégies et tactiques possibles, la gestion du temps des choix, des stratégies et de leur déroulement, la conscience de ces choix, les moyens de contrôle et de retour en arrière pour procéder à d'autres choix...* » [66]. Toutefois dans le cadre du projet AMBRE, nous restreignons la méthode en considérant qu'elle permet d'organiser une résolution de problèmes en se fondant sur le classement des problèmes et des outils de résolution. Notre objectif demeure que l'apprenant acquière cette méthode. Pour cela, il doit apprendre à construire progressivement des classes de problèmes, à reconnaître à quelle classe appartient un nouveau problème, à associer une technique de résolution à chaque classe de problèmes et à appliquer ces techniques de résolution.

Afin de permettre à un apprenant d'acquérir de telles méthodes, Rogalski [67] et Schöenfeld [70] proposent un enseignement de méthodes basé sur la catégorisation des problèmes et des techniques de résolution à l'aide d'un tuteur intelligent. Rogalski précise que le résolveur d'un tuteur « *donneur de leçons de méthodes* » doit fonctionner selon les méthodes qu'il veut enseigner, et non pas selon des méthodes expertes du domaine concerné. Suivant ces recommandations, les résolveurs utilisés dans le cadre du projet AMBRE fonctionnent selon les méthodes que nous souhaitons enseigner. Toutefois, toujours dans le cadre du projet AMBRE, cette méthode n'est pas présentée explicitement à l'apprenant. Cela s'explique par le fait que dans certains domaines, les termes employés pour définir les classes et les techniques de résolution ne sont pas connus des apprenants puisque, institutionnellement, ils ne sont pas utilisés.

Ainsi, afin d'aider l'apprenant à construire sa propre méthode, les EIAH AMBRE

proposent un processus d'apprentissage en deux étapes :

— Présenter à l'apprenant des problèmes-type résolus ;

— Inciter l'apprenant à mettre en œuvre un raisonnement par analogie pour résoudre de nouveaux problèmes.

En effet, comme le montrent de nombreuses recherches en Psychologie Cognitive [68], la présentation d'exemples et la résolution de problèmes peuvent aider l'apprenant à savoir comment et quand appliquer une technique de résolution. De plus, comme le soulignent d'autres recherches [25,27,36], le raisonnement par analogie permet d'acquérir des connaissances contextuelles et des connaissances abstraites. Par ailleurs, ce type de raisonnement n'est pas toujours spontané et l'acquisition de connaissances lors la résolution de problème par analogie n'est pas toujours automatique [36]. Ainsi, pour encourager l'apprenant à raisonner par analogie, les EIAH AMBRE le guident à travers le cycle du raisonnement à partir de cas.

1.1.2 Un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC)

Pour accompagner l'apprenant dans sa résolution des problèmes, les EIAH AMBRE proposent à l'apprenant un processus d'apprentissage inspiré du raisonnement à partir de cas (RàPC) [69]. Cependant, contrairement à d'autres systèmes [2, 37, 45, 55], dans AMBRE, le RàPC n'est pas utilisé par le système, mais proposé à l'apprenant comme une stratégie d'apprentissage. En effet, les EIAH AMBRE proposent d'aider l'apprenant en lui permettant de se référer à un cas proche pour résoudre un nouveau problème [62]. Cependant, contrairement aux autres systèmes, le cas est choisi par l'apprenant lui-même, et non par le système. Pour chaque problème, l'apprenant doit passer par les toutes étapes du RàPC, pour retrouver lui-même le cas le plus proche de son problème et en adapter la solution.

Une séance de travail avec un EIAH AMBRE se déroule en deux phases. La première consiste à la présentation par le système à l'apprenant, de scénarios de résolution de problèmes-types. Après plusieurs séances, l'apprenant aura rencontré un problème-type correspondant à chaque classe de problèmes représentée au niveau de la méthode. La seconde phase est alors consacrée à la résolution par l'apprenant de plusieurs problèmes. L'EIAH guide l'apprenant à travers cinq étapes inspirées du RàPC, le cycle

AMBRE :

- **Lecture de l'énoncé**. L'apprenant commence d'abord par lire l'énoncé afin d'identifier des éléments pertinents qui lui permettront de reformuler le problème.
- **Reformulation du problème**. Cette tâche consiste à identifier les éléments importants de l'énoncé (traits structuraux du problème). L'apprenant est guidé par le système pour construire une nouvelle formulation du problème posé, à partir de l'énoncé du problème. La reformulation obtenue est débarrassée de l'essentiel des traits de surface du problème initial, et devient une référence pour la suite de la résolution.
- **Choix d'un problème-type**. Cette étape consiste pour l'apprenant à comparer le problème à résoudre à ceux déjà rencontrés, en observant les similarités entre la reformulation du problème à résoudre et les reformulations des problèmes déjà rencontrés. L'apprenant doit choisir le problème prototype qui lui semble le plus proche de son problème à résoudre, en se fondant sur les reformulations (étape analogue à la remémoration du RàPC). Il identifie ainsi de manière implicite la classe du problème à résoudre.
- **Adaptation de la solution du problème-type au problème à résoudre**. L'apprenant doit adapter la solution du problème-type au problème à résoudre. L'objectif de cette étape est de l'aider à identifier, par généralisation, une technique de résolution adaptée aux problèmes de cette classe.
- **Classement du problème**. Enfin, cette dernière étape consiste pour l'apprenant à associer le problème qu'il vient de résoudre à un problème-type, de manière à constituer des groupes de problèmes correspondant aux classes, un groupe étant représenté par un problème-type (étape analogue à la mémorisation du RàPC).

Une phase de test et de révision est incluse à chaque étape du cycle AMBRE. En effet, à chaque étape, le système peut évaluer les réponses de l'apprenant et l'aider à les réviser. Afin de pouvoir mettre en œuvre tout cela, le système a besoin d'un certain nombre de connaissances que nous présentons dans la section suivante.

1.2 Comment appliquer une méthode de résolution de problèmes ?

1.2.1 Connaissances pour appliquer une méthode de résolution de problèmes

Au sein du projet AMBRE, enseigner une méthode de résolution consiste à identifier de quelle classe relève le problème afin de pouvoir choisir une technique de résolution adaptée aux problèmes de cette classe. Pour que l'EIAH puisse accompagner l'apprenant dans sa résolution des problèmes, il doit être capable d'appliquer une telle méthode pour résoudre des problèmes du domaine. Pour ce faire les EIAH AMBRE utilisent d'une part, des modèles de connaissances portant sur la méthode à enseigner, et d'autre part un moteur de raisonnement capable d'exploiter ces modèles de connaissances. Nous obtenons ainsi, pour chaque domaine, un résolveur de problèmes qui applique la méthode explicitée par les modèles de connaissances. Ces connaissances sur la méthode à enseigner sont de trois types : des connaissances de classification, des connaissances de reformulation et des connaissances de résolution [42].

Dans AMBRE, les problèmes à résoudre ne sont pas donnés au résolveur en langage naturel, mais en Prolog, sous la forme d'un modèle que nous appelons modèle descriptif [40]. Pour résoudre un problème dans un domaine donné, le système utilise les connaissances de classification et les connaissances de reformulation afin de (i) déterminer de quelle classe relève le problème, et (ii) construire un nouveau modèle du problème appelé modèle opérationnel. La résolution proprement dite consiste alors à appliquer au modèle opérationnel la technique de résolution associée à la classe pour trouver la solution (cf. **Figure 1.1**).

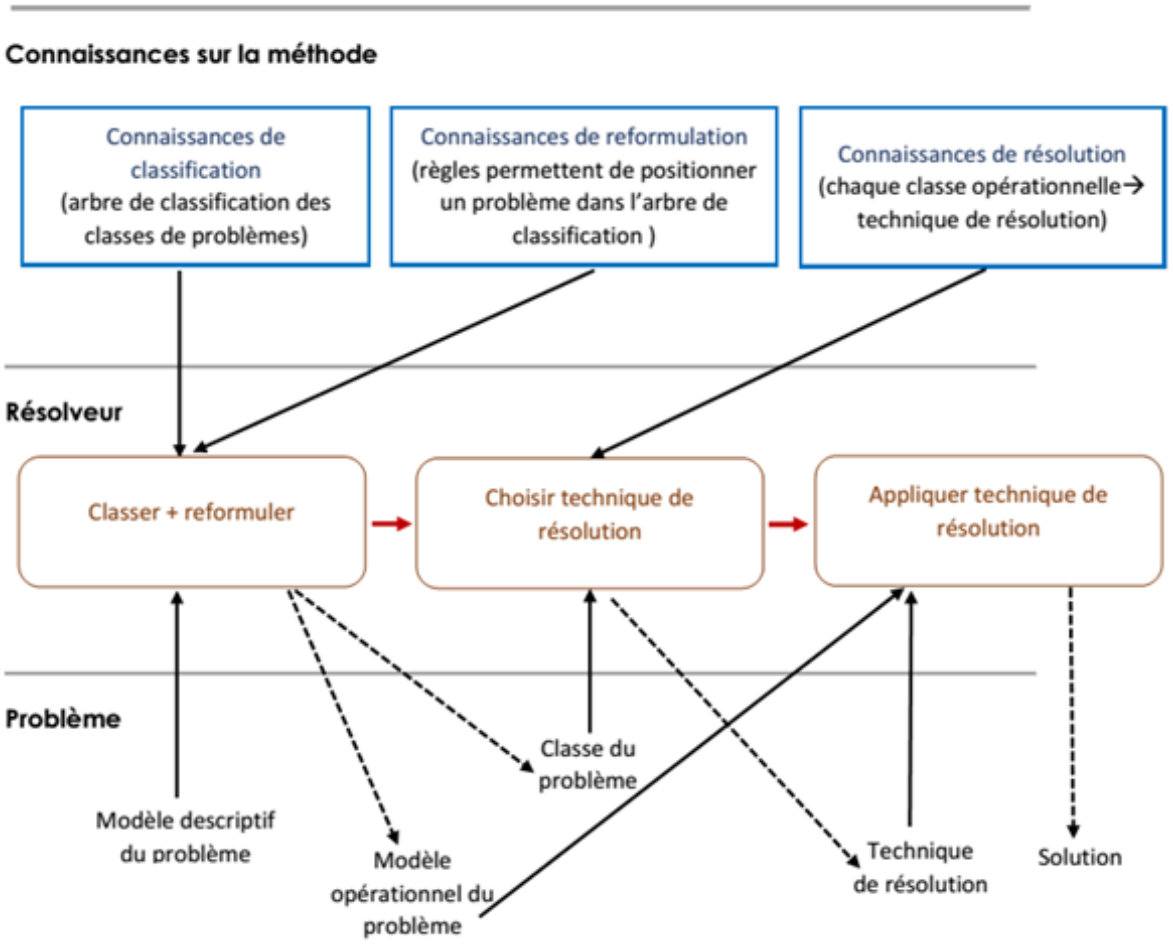

FIGURE 1.1 – Fonctionnement du résolveur de problèmes dans AMBRE [40]

1.2.1.1 Connaissances de classification

Dans chaque domaine d'application, les connaissances de classification sont représentées par une hiérarchie des classes de problèmes à résoudre. La classification représentée par cette hiérarchie est évidemment dépendante du domaine choisi. Toutefois, la structure de représentation ainsi que son utilisation restent les mêmes quel que soit le domaine.

Dans une telle hiérarchie, une classe C2 est sous-classe d'une classe C1 si tout problème de C2 est aussi un problème de C1. Une *classe racine* est définie comme étant la classe la plus générale, alors que les feuilles sont les classes les plus spécifiques.

Chaque classe possède un *attribut discriminant* portant une valeur différente dans chacune de ses sous-classes. D'autres attributs appelés *attributs du problème* peuvent également être définis. Ces attributs ne sont pas discriminants pour la classification, mais demeurent utiles pour la résolution. Les valeurs de ces attributs dépendent ainsi du problème à résoudre. Les classes qui sont suffisamment spécifiques pour qu'on puisse leur associer une technique de résolution sont appelées *classes opérationnelles*. La **Figure 1.2** par exemple représente une partie du graphe de classification pour la conjugaison française. *C_Conjugaison* représente la classe racine et a pour attribut discriminant *temps_simples*. Les valeurs discriminantes de cet attribut sont : présent, et futur. Cette classe possède donc deux sous-classes : une classe non opérationnelle *C_Présent* ainsi qu'une classe opérationnelle *C_Futur*. L'attribut discriminant de la classe C_Présent est *groupe*. Les valeurs discriminantes de cet attributs sont : *1er groupe* et *2ème groupe*. La classe C_Présent possède donc deux sous-classes : la classe *C_Présent_1er_Gr* qui est non opérationnelle et la classe *C_Présent_2ème_Gr* qui est opérationnelle.

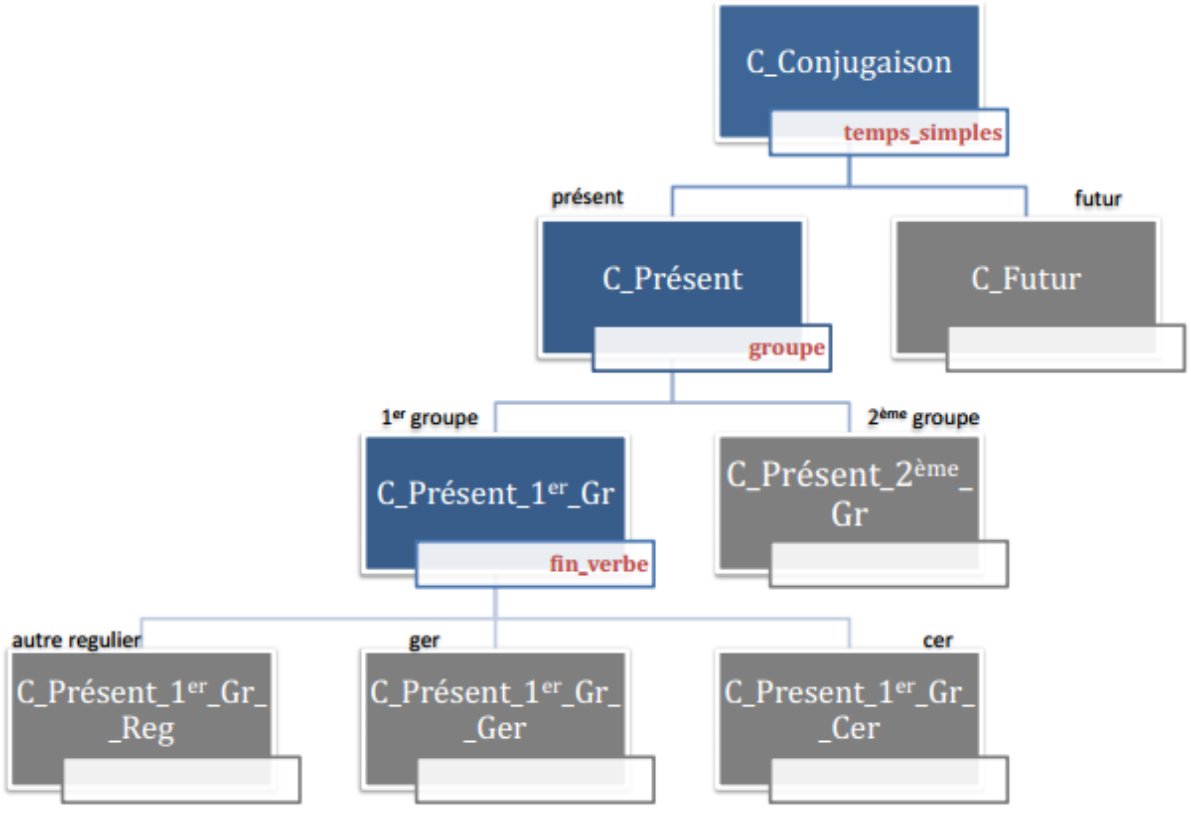

FIGURE 1.2 – Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif

Pour que le système puisse utiliser cette classification afin d'identifier la classe d'un problème, un ensemble de règles doivent être définies. Par exemple, si nous reprenons le problème P1 sur la conjugaison française (cf. Introduction générale) : « *Conjuguer le verbe parler au présent de l'indicatif* », le résolveur doit disposer de règles lui permettant, à partir de l'énoncé du problème, de calculer les valeurs des attributs discriminants afin de situer ce problème au niveau de la hiérarchie des classes. l'ensemble de toutes les règles définies pour un domaine constituent les connaissances de reformulation.

1.2.1.2 Connaissances de reformulation

Pour chaque attribut (discriminant ou non) défini dans la classification, une(des) règle(s) permettant de conclure sur sa/ses valeur(s) doit (doivent) être définie(s). Une règle est définie par son *nom*, un *ensemble de prémisses*, et un *ensemble de conclusions*. Les prémisses portent sur des éléments d'énoncé ou sur des attributs du problème qui pourront eux-mêmes nécessiter l'application de règles pour calculer leurs valeurs. Ces prémisses peuvent également faire appel à des fonctions de calcul Prolog. Les conclusions quant à elles permettent de conclure sur la valeur des attributs. On y retrouve également des appels à des fonctions de calcul.

La règle suivante permet par exemple de situer le problème *P1* au niveau de la classification (cf. **Figure 1.2**).

Règle 1 :
Si dans un problème on a :

- un verbe *V*
- la terminaison du verbe *V* est égale à *er*
- le verbe *V* est différent du verbe *aller*

Alors la valeur de l'attribut discriminant *groupe* est égale à *1er groupe*.

Cette règle stipule que si dans un problème il existe un verbe V, si la terminaison du verbe est *er* et si ce verbe est différent du verbe aller, alors la valeur de l'attribut groupe est égal à *1er groupe*.

Les connaissances de reformulation permettent de passer d'un modèle descriptif du problème à un modèle que nous appelons opérationnel - constitué des attributs ainsi que

de leurs valeurs - plus adapté à l'application d'une technique de résolution. L'ensemble des techniques de résolution constituent ce que nous nommons les connaissances de résolution.

1.2.1.3 Connaissances de résolution

Les techniques de résolution sont fortement liées aux classes de problèmes. En effet, pour connaître quelle technique employer, il faut savoir quelle technique est adaptée pour une classe de problèmes. Cependant c'est une expertise difficile à acquérir. Des recherches en Psychologie Cognitive ont montré que les experts se distinguent des novices par leur capacité, en situation de résolution de problèmes, à reconnaître la classe du problème afin d'y appliquer la technique de résolution la plus adaptée. Chi et al. [19] ont montré que les novices sont souvent en difficultés parce qu'ils n'ont pas les mêmes critères de classification que les experts.

Dans les EIAH AMBRE, une technique de résolution doit être associée à chaque classe opérationnelle définie dans l'arbre de classification. Une technique de résolution est définie par son nom, un ensemble de données portant sur le problème, un ensemble d'éléments de solution, et une fonction de calcul permettant de déterminer les éléments de solution à partir des données. Ces techniques sont spécifiques au domaine, en conjugaison par exemple, il s'agit d'une part du processus permettant d'obtenir la partie fixe pour l'ensemble des personnes et d'autre part de la liste des terminaisons pour les six personnes.

1.2.2 Processus de résolution de problèmes

Pour chaque domaine, une fois que les trois modèles de connaissances sont définis : connaissances de classification, connaissances de reformulation et connaissances de résolution, le résolveur de AMBRE met en œuvre une phase d'opérationnalisation pour résoudre des problèmes :

— **Classement du problème**. Pour classer un problème, le résolveur utilise l'arbre de classification - comme un guide - et les règles de reformulation. Il part de la racine, et affine le classement petit à petit en passant à une classe plus spécifique.

Pour descendre dans une sous-classe, il utilise les règles afin de déterminer la valeur de l'attribut discriminant de la classe et éventuellement les valeurs des attributs du problème. Il essaie ainsi d'aller le plus loin possible vers les feuilles, au moins jusqu'à une classe opérationnelle. Il obtient ainsi la classe la plus spécifique selon laquelle les connaissances permettent de modéliser le problème.

— **Construction du modèle opérationnel**. Ce modèle est construit au cours du classement. Vide au départ, il est enrichi à chaque fois que l'on descend dans une sous-classe. À la fin de l'opérationnalisation, on obtient un couple formé par la classe à laquelle appartient le problème, et le nouveau modèle du problème contenant assez d'informations permettant de le résoudre.

— **Construction de la solution du problème**. Pour trouver la solution au problème, le résolveur applique au modèle opérationnel la technique de résolution associée à la classe du problème (déterminée dans les étapes précédentes).

Le résolveur de problème et le processus de raisonnement dans son ensemble sont indépendants du domaine. En d'autres termes, leur fonctionnement reste le même quel que soit le domaine. Seul le contenu des trois bases de connaissances (connaissances de classification, connaissances de reformulation, connaissances de résolution) change suivant le domaine concerné. Cependant, le résolveur ne permet pas à lui seul de fournir de l'aide à l'apprenant, ni de diagnostiquer ses réponses ou de lui fournir des explications pertinentes sur ses erreurs. Pour assurer ces fonctions, l'EIAH a besoin de connaissances complémentaires qui lui permettent de mieux accompagner l'apprenant.

1.3 Comment guider l'apprenant dans sa résolution de problèmes ?

1.3.1 Connaissances pour guider l'apprenant durant son apprentissage

Afin de favoriser l'apprentissage, les EIAH AMBRE assistent l'apprenant dans la résolution de nouveaux problèmes en permettant d'une part de répondre à une demande d'aide de l'apprenant, et d'autre part de diagnostiquer ses réponses à chaque étape

de résolution et l'aider à réviser une réponse erronée en lui proposant des explications pertinentes. Ces connaissances sont donc fortement dépendantes des connaissances précédemment définies sur la méthode à enseigner et des étapes de résolution. Elles sont de trois types : les *connaissances d'aide*, les *connaissances de diagnostic* et les *connaissances d'explications* sur les erreurs commises.

1.3.1.1 Connaissances d'aide

Afin d'assister l'apprenant dans sa tâche de résolution, l'EIAH doit proposer à l'apprenant des aides de différentes natures en fonction de l'étape de résolution en cours et des erreurs commises. Dans AMBRE-add [63], un EIAH développé pour l'apprentissage de la résolution des problèmes additifs au primaire, des messages d'aide sont proposés à l'apprenant. Ces messages d'aide permettent de rappeler à l'apprenant ce qu'il doit faire en fonction de l'étape de résolution à laquelle il se trouve. Ils sont souvent structurés en deux niveaux. Lorsque le premier ne suffit pas à l'apprenant pour comprendre, ce dernier peut utiliser le second.

1.3.1.2 Connaissances de diagnostic

Les connaissances de diagnostic permettent de comparer la réponse de l'apprenant à celle fournie par le résolveur de AMBRE, afin de pouvoir déterminer si une réponse est correcte ou non. En effet, le résolveur donne une réponse correcte. Cependant, il peut arriver que l'apprenant donne une réponse différente de celle du résolveur quand bien même elle reste correcte. Il faut donc des connaissances pour gérer ces équivalences.

1.3.1.3 Connaissances d'explications sur les erreurs

Les connaissances d'explications font suite au diagnostic. Elles permettent de donner à l'apprenant des explications pertinentes en cas d'erreur. Il s'agit de définir un ensemble de réponses fausses attendues, et d'y associer des messages explicatifs. L'ensemble des connaissances que nous venons de décrire (connaissances sur la méthode à enseigner, connaissances d'aide, de diagnostic et d'explications, connaissances de communication) sont associés à un processus spécifique afin de permettre les fonctionnalités

recherchées, c'est-à-dire le diagnostic des réponses de l'apprenant, et la construction de messages d'aide et d'explications sur les erreurs.

1.3.2 Processus de raisonnement

Grâce aux connaissances précédemment décrites, le système peut répondre à trois types de demande : une demande de l'interface de l'EIAH, une demande d'aide de l'apprenant ou une demande de diagnostic (de l'apprenant ou de l'interface). Nous décrivons dans la section suivante les processus permettant de répondre à ces demandes.

1.3.2.1 Productions d'informations pour l'interface

Le processus de production d'informations destinées à l'interface utilise le résolveur, sa connaissance des problèmes ainsi que les connaissances de configuration de l'interface figurant dans les connaissances de communication. En début de séance par exemple, l'interface fait appel au résolveur pour présenter à l'apprenant des problèmes-types résolus. De même pour demander à l'apprenant de choisir un problème-type, l'interface a besoin de connaître les problèmes-types que l'apprenant a déjà vus. Elle doit aussi savoir quels problèmes ont déjà été résolus par l'apprenant, et à quels problème-type l'apprenant les a associés. L'interface peut également faire appel au résolveur pour constituer des éléments d'interface.

Dans AMBRE-add par exemple, pour construire les listes déroulantes utilisées par l'apprenant lors de la construction de la phrase de réponse, le système utilise les données du problème pour lui proposer les éléments corrects et incorrects : le sujet de la phrase (les prénoms de l'énoncé et d'autres noms), les objets (ceux de l'énoncé, par exemple les billes et ceux figurant dans d'autres problèmes tels que le problème-type), le verbe de la phrase (a, avait, a gagné, a perdu, a mangé, etc.).

1.3.2.2 Production d'aide

Lorsque l'apprenant demande de l'aide, l'interface de l'EIAH indique au système quel est l'avancement exact de l'apprenant dans le cycle AMBRE. Ce qui permet au

système de construire ainsi un message d'aide en utilisant les connaissances sur le cycle AMBRE et les connaissances liées à la langue naturelle. De même, afin de pouvoir mettre en évidence les éléments analogues du problème-type et du problème à résoudre, le système utilise les connaissances sur les liens entre ces éléments ainsi que les connaissances du résolveur sur les deux problèmes.

1.3.2.3 Diagnostic des réponses et production d'explications

Le diagnostic peut être effectué automatiquement par le système ou demandé par l'apprenant à la fin de chaque étape ou sous-étape de résolution. L'EIAH procède en deux temps lors d'un diagnostic. Dans un premier temps, le système utilise ses connaissances de diagnostic pour comparer la réponse de l'apprenant à celle du résolveur. Puis dans le second, il génère en cas d'erreur un message ou des informations que l'interface mettra en œuvre visuellement. Pour ce faire, il utilise les connaissances d'explications sur les erreurs en s'appuyant sur le résolveur ainsi que l'ensemble des connaissances permettant de communiquer avec l'apprenant.

1.4 Conclusion

Ce chapitre a présenté le cadre applicatif de nos recherches : le projet AMBRE. Après avoir décrit le principe des EIAH AMBRE en montrant comment le RàPC a été utilisé dans le cadre de ce projet pour soutenir la démarche d'apprentissage proposée à l'apprenant, nous avons présenté les différentes connaissances qui entrent en jeu lors de la conception de tels EIAH. Ce chapitre se termine par la présentation du processus de raisonnement associé aux différentes connaissances et permettant d'assurer les fonctionnalités attendues que sont la résolution des problèmes, l'aide, le diagnostic des réponses, et les explications sur les erreurs de l'apprenant.

Une des particularités du projet AMBRE est que les EIAH développés proposent une démarche d'apprentissage inspirée du raisonnement à partir de cas, alors que les résolveurs de problèmes se fondent sur une classification des problèmes du domaine. Face à la difficulté d'assister l'apprenant dans sa résolution de problèmes, la notion de problème prototype représentant une classe est utilisée, afin de faire le lien entre les

connaissances du système et le raisonnement de l'apprenant.

Un des problèmes majeurs de l'ingénierie des EIAH demeure leur coût de conception élevé. En particulier dans les EIAH centrés connaissances, ce phénomène s'explique en grande partie par la difficulté d'une élicitation des connaissances ayant recours à des langages de programmation : c'est le cas de Prolog dans le cadre des EIAH AMBRE. Cette nécessité de compétences en programmation de la part des concepteurs restreint ainsi l'ingénierie des EIAH à une communauté spécifique.

Notre problématique est alors de réduire ce coût de conception, tout en étendant la communauté des auteurs d'EIAH, pour une utilisation à plus grande échelle. La conception d'un outil auteur peut permettre à un auteur d'éliciter, sans recourir à de la programmation, les connaissances qui permettront à l'EIAH de résoudre les problèmes d'un domaine qu'il aura choisi. Nous ciblons des utilisateurs experts des connaissances de la méthode à enseigner (enseignant, professionnel ou pédagogue plus expert), et familiarisé avec l'outil informatique, avec ou sans compétences en programmation. Ainsi, l'outil auteur à développer doit permettre d'acquérir auprès de tels auteurs les différentes connaissances entrant en jeu lors de la conception de tels environnements. La mise en œuvre dudit outil auteur passe par la :

— Formalisation des différentes connaissances qui entrent en jeu lors de la conception de tels environnements ;

— Proposition, à travers la conception d'un outil auteur, des interfaces permettant d'accompagner l'auteur dans l'élicitation des connaissances (en lui proposant une assistance durant le processus).

Avant de répondre à notre problématique, dressons d'abord un état de l'art concernant les fondements théoriques des travaux abordés dans cette thèse.

Chapitre 2

État de l'art

Dans ce chapitre, nous nous intéressons à la problématique de l'acquisition des connaissances, plus particulièrement dans le cadre de la conception d'EIAH. Nous présentons dans un premier temps un petit historique sur l'évolution de l'acquisition des connaissances vers l'Ingénierie des Connaissances (IC) avant de décrire les méthodes et techniques d'acquisition de connaissances les plus répandues en IC. Nous montrons ensuite comment ces méthodes et techniques sont utilisées dans le domaine des EIAH et plus spécifiquement lors de la conception d'outils auteurs. Ce chapitre se termine par la synthèse des principaux verrous et manques actuels par rapport à notre problématique.

Sommaire

2.1 De l'acquisition des connaissances à l'Ingénierie des Connaissances 49
2.2 Méthodes et techniques d'acquisition de connaissances en Ingénierie des Connaissances 50
2.2.1 Méthodes empiriques 50
2.2.2 Méthodes analytiques 52
2.3 IC et EIAH 58
2.4 Outils auteurs 59
2.4.1 Outils auteurs pour la création de parcours pédagogiques . . 59
2.4.2 Outils auteurs pour la création de tuteurs intelligents 60
2.5 Synthèse des principaux verrous 62

2.1 De l'acquisition des connaissances à l'Ingénierie des Connaissances

L'acquisition des connaissances pour la conception de systèmes à base de connaissances (SBC) est l'un des principaux obstacles à leur développement. Les travaux dans ce domaine se sont clairement divisés en deux groupes ayant des objectifs relativement différents : *le transfert d'expertise* [6, 17] et l'*apprentissage automatique*. Le transfert d'expertise vise à mettre en place des moyens pour donner à un système informatique des connaissances issues d'un expert, de documents ou de manuels. L'apprentissage automatique, quant à lui, vise à doter les systèmes de capacités d'acquisition de nouvelles connaissances par eux-mêmes. Dans le cadre de ce travail, nous nous intéresserons essentiellement au transfert d'expertise dans le sens donné ci-dessus. Présentons d'abord un petit historique sur l'évolution de l'acquisition des connaissances vers l'Ingénierie des Connaissances (IC).

L'acquisition des connaissances est apparue comme un domaine de recherche à la fin des années 80 [8]. Pendant longtemps, les recherches dans ce domaine se sont focalisées sur la modélisation psychologique ou empirique des connaissances d'un expert dans le but de les coder dans un système expert [17]. Il s'agissait essentiellement d'une activité de transfert de connaissances entre des sources d'expertise (experts humains, documents, etc.) et un outil informatique, de façon à disposer d'un système à base de connaissances pouvant être consulté comme un expert.

Par la suite elle a migré vers l'organisation de modèles structurés vers des modèles conceptuels de différents types de connaissances nécessaires à la résolution des problèmes : on tente à partir d'une formulation linguistique du problème d'élaborer une représentation plus ou moins formelle du problème. En d'autres termes, il s'agissait de définir quelles connaissances sont pertinentes, pour ensuite les exploiter en formulant un modèle du problème. Les recherches en acquisition des connaissances ayant évolué ces dernières années, elles s'intéressent aujourd'hui à la mise en place de systèmes informatiques s'intégrant dans des tâches humaines faisant appel à des connaissances spécialisées [7]. Ces motivations ont fait évoluer les attentes de l'acquisition des connaissances vers l'Ingénierie des Connaissances où l'apport de méthodes et techniques reproductibles est devenu central.

2.2 Méthodes et techniques d'acquisition de connaissances en Ingénierie des Connaissances

Dès les premiers travaux sur l'acquisition des connaissances autour des années 1980, l'IC s'est positionnée comme un domaine actif proche de l'Intelligence Artificielle, mais aussi à la croisée de plusieurs disciplines concernées par la mise en ouvre, le repérage, l'analyse de connaissances comme la sémantique, l'ergonomie, la linguistique ou encore la sociologie.

Ce champ de recherche vise la définition de solutions techniques, méthodologiques et organisationnelles pour identifier des connaissances, les modéliser et les restituer au sein d'applications informatiques [7]. Plus formellement, l'IC se définit comme étant l'étude des concepts, des méthodes et des techniques permettant de modéliser et/ou d'acquérir les connaissances pour des systèmes réalisant ou aidant à réaliser des tâches se formalisant a priori peu ou pas [18]. Sa problématique et ses résultats sont marqués par l'évolution des systèmes qu'elle cherche à produire, qui sont des logiciels pouvant aider un utilisateur à mettre en œuvre des connaissances. Son usage couvre une large variété d'applications dont notamment : acquisition de connaissances auprès d'experts humains, acquisition de connaissances à partir d'un corpus de textes, recherche d'informations dans des bases de données ou sur le Web, mémoire d'entreprise, etc. Nous nous intéresserons dans cette thèse à l'acquisition de connaissances auprès d'experts humains.

Lors de la mise en œuvre des premiers systèmes experts, les méthodes utilisées pour effectuer le transfert de la connaissance depuis le monde réel vers le système artificiel étaient totalement empiriques. Elles se sont par la suite précisées et structurées (méthodes analytiques), voire même automatisées pour certaines. Nous détaillons ci-dessous quelques méthodes et techniques d'acquisition de connaissances les plus répandues en Ingénierie des Connaissances.

2.2.1 Méthodes empiriques

Les premiers développements de systèmes experts se sont faits sans aucune méthode précise quant à l'acquisition du savoir-faire des experts. Le recueil de la connaissance et

sa traduction étaient alors effectués par un ou plusieurs ingénieurs de la connaissance. Cette phase de transfert d'expertise est souvent considérée comme le « *goulot d'étranglement* » du développement d'un système expert. En effet, elle peut durer des mois, voire des années, et compte-tenu de la complexité des connaissances de l'expert qui parvient difficilement à expliciter ses processus mentaux, la connaissance extraite risque souvent d'être inexacte, incomplète, voire inconsistante [28] .

De nombreuses techniques de verbalisation et d'aide aux interviews ont été développées dans le but d'aider à mener correctement le processus d'acquisition de connaissances et ainsi faciliter le développement de bases de connaissances [6, 13, 29, 85] ; Ces techniques, souvent inspirées des travaux en psychologie cognitive, sont entre autres :

— Les **interviews**, généralement enregistrées et souvent non dirigées en début d'extraction.

— L'**analyse de protocoles verbaux**, où l'expert résout un problème en réfléchissant à haute voix.

— L'**observation directe**, où l'expert est filmé en train d'exercer son activité dans son environnement habituel. Les actions de l'expert, sa démarche générale, les cas et types de problèmes traités sont ainsi notés. Cette technique est un bon moyen d'accéder aux connaissances de contrôle qui échappent aux verbalisations.

— La **simulation du comportement** du futur système par l'expert.

— Les **commentaires sur des protocoles** antérieurs (reprise de cas).

— Le **tri** par l'expert des problèmes suivant leur degré de difficulté, le type de résolution attachée, etc.

— La **consultation de documents** : le cogniticien lit des documents conseillés par l'expert (documents généraux sur le domaine, documents particuliers rédigés par l'expert, etc.).

— L' **utilisation d'un questionnaire**, une fois qu'une partie de l'expertise a été identifiée.

— Le **brainwriting** pour regrouper et comparer les idées de plusieurs spécialistes : chacun enrichit les idées des autres en les commentant par écrit (une feuille par idée par exemple).

— Etc.

Dans chacune de ces techniques, le plus difficile était de minimiser les déformations de l'expertise. En effet, le décalage entre le langage dans lequel l'expert exprime ses connaissances et le niveau d'abstraction des formalismes classiques de représentation des connaissances apparaît très vite comme une difficulté, un facteur limiteur. Progressivement, les chercheurs se sont rendu compte que le problème n'était pas simplement un problème de transcription, mais fondamentalement un problème de modélisation. L'acquisition des connaissances se fixa alors pour nouveau but la construction de modèles [21, 26, 86]. À partir de ce moment, de nombreux travaux se sont engagés à définir des méthodes et outils pour assister le cogniticien chargé de mettre au point une base de connaissances : les méthodes analytiques.

2.2.2 Méthodes analytiques

Plus structurées que les méthodes empiriques, les méthodes analytiques nécessitent l'intervention d'un cogniticien. Un des points forts de ces méthodes est qu'elles contribuent à construire un modèle de connaissances intermédiaire entre le monde réel et le monde symbolique, permettant ainsi une meilleure appréhension des connaissances acquises. Nous allons présenter brièvement deux de ces méthodes, pertinentes par rapport à notre problématique : KADS et KOD. Ces méthodes ont pour intérêt d'être assez générales pour être utilisées dans plusieurs domaines, de prendre en compte tout le processus d'acquisition des connaissances, et de proposer chacune un modèle de la connaissance experte.

2.2.2.1 KADS

KADS (Knowledge and Analysis Design Support) [75, 87] est une méthodologie orientée vers la conception de systèmes à base de connaissances. C'est une méthode dirigée par les modèles qui propose :

— Un **cycle de vie** comprenant plusieurs phases : analyse, conception, implémentation, installation, utilisation, maintenance et enrichissement des connaissances (cf. **Figure 2.1**).

— Un **ensemble d'outils conceptuels** pour la modélisation des connaissances, in-

dépendamment du langage d'implémentation.

— Et un **support logiciel** avec l'outil SHELLEY [14] guidant la démarche du cogniticien (module de conseil pour l'utilisation de la méthodologie, modules permettant la structuration et l'analyse des interviews, la conceptualisation du domaine en entités et relations, la proposition de plusieurs modèles d'interprétation, etc.).

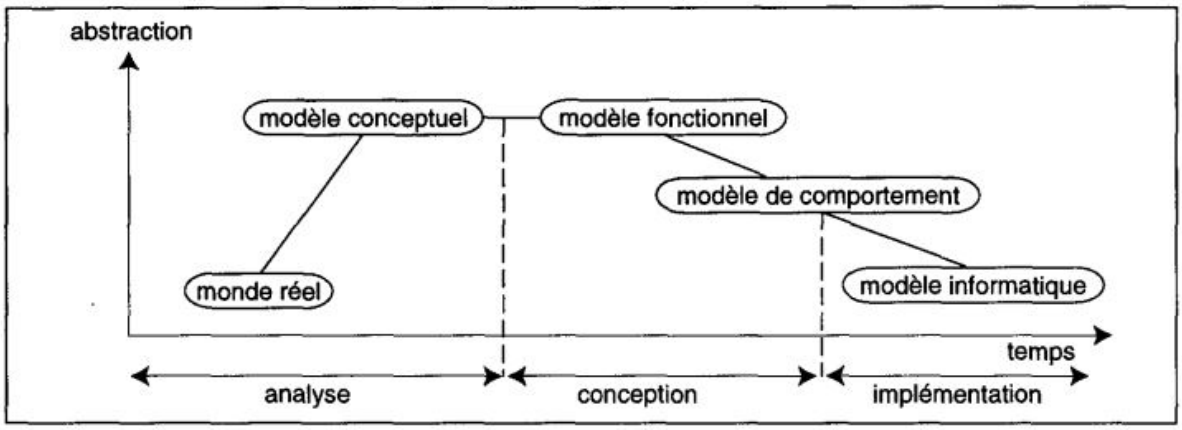

FIGURE 2.1 – KADS : cycle de vie d'un système expert [31]

Dans KADS, le processus de développement d'un SBC consiste à construire successivement plusieurs modèles : le **modèle d'expertise**, le **modèle de coopération** entre le futur système et ses utilisateurs, le **modèle externe** qui intègre les contraintes qui vont influer sur la conception et la réalisation du système final, le **modèle de conception fonctionnelle**, le **modèle logique** puis le modèle physique du système. Une bibliothèque de modèles de tâches génériques appelés modèles d'interprétation permet une acquisition structurée des connaissances.

La définition du modèle d'expertise dans KADS s'appuie sur une structure à quatre niveaux :

— Le **niveau domaine** qui décrit les connaissances statiques du domaine (indépendamment de son utilisation) en termes d'objets, de concepts, d'attributs et de relations entre ces divers éléments.

— Le **niveau inférence** permettant de décrire les inférences qui peuvent être faites sur les connaissances du niveau précédent. Il contient un ensemble de fonctions permettant de faire des inférences lors de la résolution de problèmes. Ces fonctions sont appelées sources de connaissances et leurs arguments méta-classes. Une méta-

classe indique comment un ensemble spécifique de concepts du domaine peut être utilisé.

— Le **niveau tâches** décrit comment appliquer les inférences possibles (les étapes du raisonnement) en fonction des buts. Il s'agira d'une part de préciser comment la structure d'inférence sera parcourue, et d'autre part d'associer les éléments de contrôle nécessaires.

— Le **niveau stratégie** contient les connaissances conduisant à l'élaboration du plan d'action. Il décrit le plan de résolution du problème posé ainsi que les connaissances nécessaires au contrôle de celui-ci. Le plan de résolution précise l'ordonnancement des buts à atteindre pour résoudre le problème et formalise le comportement de résolution du problème : par exemple comment résoudre le problème, comment réaliser le contrôle de l'exécution des buts, comment réagir en cas d'impasse.

Pour faciliter l'identification de ces différents niveaux, KADS offre :

— Des *techniques d'analyse de protocoles* pour identifier les éléments du niveau domaine.

— Un *ensemble de primitives d'inférence* pour l'identification des connaissances du niveau inférence.

— Une *bibliothèque de modèles d'interprétation* qui sont des modèles génériques pour la résolution d'un type de problème particulier et qui servent de cadre pour l'élaboration du modèle des tâches et de guide pour les séances de recueil des connaissances.

— Le niveau stratégie quant à lui ne bénéficie d'aucun outil, son identification étant laissée à l'initiative du cogniticien.

La méthodologie KADS a connu plusieurs évolutions. La version la plus récente est commonKADS [74, 90], développée par Cap gemini et qui propose de construire le modèle conceptuel autour de cinq modèles : le modèle d'organisation, le modèle des tâches, le modèle des agents, le modèle des connaissances (qui précise les types, les structures et les rôles de la connaissance) et le modèle de communication.

Conclusion

KADS permet de traiter tout le processus de construction d'un SBC, du recueil au développement du système. Pour faciliter l'analyse des connaissances, KADS propose une structure à quatre niveaux permettant la formalisation des connaissances, ainsi qu'une bibliothèque de modèles d'interprétation. L'utilisation des modèles génériques permet de faciliter la construction d'un modèle d'expertise : le cogniticien travaille par affinements successifs des éléments des structures génériques. Le recueil des connaissances est alors guidé par les modèles.

L'intérêt majeur de KADS est qu'elle offre un support pour guider l'acquisition des connaissances. L'autre intérêt réside dans la généricité et la réutilisabilité des modèles qui en résultent. Cependant, un des inconvénients de KADS est qu'il transforme l'activité d'analyse en une activité de choix de modèles qui nécessite une bonne expérience de manipulation des modèles d'interprétation. De plus, il est nécessaire de disposer d'une large bibliothèque de modèles et/ou de composantes génériques. En outre, l'adaptation d'un modèle existant à une application spécifique est généralement une tâche difficile. Nous constatons également qu'à la fin de la construction des différents modèles, KADS ne propose aucun lien pour passer des différents niveaux de connaissances vers les langages de représentation disponibles actuellement. Dans la section qui suit, nous présentons KOD, qui elle, spécifie les langages correspondant à chaque type de connaissances obtenu par cette méthode

2.2.2.2 KOD

KOD (Knowledge Oriented Design) [85] est une méthodologie d'acquisition de connaissances mise au point par Claude Vogel. Contrairement à KADS, le cycle de vie de KOD ne couvre que l'aspect "base de connaissances". L'atelier proposé par KOD comme support logiciel est la K-STATION. C'est une approche ascendante qui sépare le transfert d'expertise en deux étapes séquentielles :

— L' **analyse cognitive** : elle consiste à comprendre et à modéliser le savoir de l'expert. Cette étape s'appuie sur l'utilisation de techniques issues des travaux en linguistique et en anthropologie pour le découpage et l'analyse du discours de l'expert.

— La **réalisation informatique** : elle consiste à développer le SBC proprement dit, en utilisant les résultats obtenus lors de l'analyse cognitive.

La méthode KOD s'intéresse donc au recueil des connaissances d'un expert, à leur modélisation à partir de sources d'informations (discours de l'expert, documentations, observation, etc.), puis à la spécification de la base de connaissances du futur système, selon une analyse en plusieurs étapes des données recueillies. Ses fondements linguistiques la rendent bien adaptée pour l'acquisition de connaissances exprimées en langage naturel. Ainsi, elle propose un cadre méthodologique pour guider la collecte des termes et les organiser à partir d'une analyse terminologique. Grâce à ses fondements anthropologiques, le cogniticien dispose d'un cadre méthodologique facilitant l'analyse sémantique de la terminologie utilisée, pour produire un modèle cognitif. Enfin, KOD guide le travail du cogniticien depuis l'extraction de la connaissance jusqu'à l'élaboration du modèle informatique.

Pour mener l'analyse cognitive, KOD propose une décomposition en trois étapes donnant lieu à trois niveaux de modèles :

- Le **modèle pratique** : il permet de définir des dictionnaires de connaissances à partir d'une analyse textuelle des entretiens. Cette étape consiste à analyser le discours de l'expert en identifiant les termes pertinents et en les étiquetant selon une typologie clairement définie :
  - Les *taxèmes*, qui décrivent les objets du monde physique.
  - Les *actèmes*, qui décrivent le changement d'état causé par un sujet sur un destinataire.
  - Les *schémèmes* : des schémas d'interprétation (déduction, induction) qui se manifestent par des contraintes et des inférences. Ils permettent d'établir de nouvelles propositions à partir de celles déjà acquises.
- Le **modèle cognitif** : il permet de structurer et valider les connaissances acquises. Il se compose de trois structures principales :
  - Une *taxinomie*, qui organise les taxèmes selon la relation *est-un*.
  - Les *actinomies*, qui articulent la structuration des schémas mentaux de l'expert permettant de répondre à des situations déjà connues.
  - Et des *schémas d'interprétation*, constitués par des ensembles de règles d'inférence associées correspondant aux stéréotypes de raisonnement de l'expert, qui se déclenchent selon les observations du domaine.
- Le **modèle informatique** : il permet la mise en oeuvre des connaissances dans un système informatique. Il est constitué d'objets, de méthodes et de règles. La

taxonomie devient une hiérarchie d'objets, les actinomies donnent les méthodes, et les schémas donnent des règles. Vogel considère que les différents modèles de programmation actuels se réfèrent aux paradigmes suivants :

- La programmation par objets, qui permet d'exprimer les données et les taxonomies ;
- La programmation par acteurs, qui permet d'exprimer les traitements et les actionomies ;
- La programmation par la logique, qui permet d'exprimer les contraintes et les règles.

Conclusion

L'un des avantages de la méthode KOD est qu'elle laisse la liberté aux experts d'exprimer leurs perceptions ainsi que leurs tâches sans les contraindre [11] Cette approche était largement utilisée pour l'élaboration des premières générations des systèmes experts. Aujourd'hui, elle reste particulièrement adaptée à des domaines où il n'existe pas de modèles de connaissances prédéfinis, et où il manque de la formalisation. L'exploration d'un nouveau domaine en est un exemple. La K-STATION, qui se présente comme une plate-forme logicielle, offre un ensemble d'outils pour aider à la mise en œuvre de la méthode KOD. Certains de ces outils, de type hypertexte, permettent de gérer les fichiers de retranscription, de conceptualiser les informations qu'ils contiennent à l'aide des types de structures constituées.

Cependant, comme la plupart des approches, KOD présente un certain nombre d'inconvénients [30] : le coût élevé en termes de temps et d'expertise dans le processus d'élicitation ainsi que dans le processus de validation, aussi bien pour les experts du domaine que pour les ingénieurs de la connaissance. Les modèles issus de cette approche souffrent aussi d'un manque d'abstraction, de généricité et sont donc difficilement réutilisables [33].

Comme KADS, KOD détaille également les différentes phases du projet de construction d'un SBC et propose un modèle de connaissances à l'expert. En revanche, elle ne propose rien qui permet et de distinguer les différents niveaux de connaissances de l'expert, ce qui peut nuire à l'explicabilité du système ainsi construit. On peut également lui reprocher sa lourdeur, le manque de méthode dans la conduite des entretiens, le coût et la lenteur de leur structuration systématique ainsi que la formalisation presque automatique de connaissances dont beaucoup ne sont pas utiles à la conception du système.

2.3 IC et EIAH

Ingénierie des Connaissances (IC) et EIAH sont deux domaines distincts. Un système à base de connaissances et un EIAH sont deux objets de nature différente. En effet, des problèmes spécifiques sont soulevés lors de la conception d'un EIAH : perception et interprétation des actions de l'apprenant, contrôle de l'interaction, prise en compte des caractéristiques individuelles des apprenants et de l'évolution de leurs connaissances, évaluation des apprentissages, etc. [10].

Cependant, comme le souligne [79], ces deux domaines sont suffisamment voisins par la nature de certaines problématiques qu'ils abordent : représentation et modélisation des connaissances en IC, versus modélisation des connaissances du domaine d'apprentissage, des connaissances de l'apprenant, etc. en EIAH ; résolution de problème en IC, versus résolution d'exercices, diagnostic des réponses de l'apprenant, etc. en EIAH. Alors que l'IC fournit sur ces aspects des outils (conceptuels, techniques, méthodologiques) utiles à la construction d'EIAH, le domaine des EIAH quant à lui présente des spécificités dont la prise en compte a notablement contribué à l'évolution de l'IC. On peut citer par exemple les travaux de Clancey [22] consacrés à la recherche d'une utilisation pédagogique de MYCIN. Les travaux en EIAH sur la notion de connaissances ou l'étude des interactions épistémiques sont également pertinents pour l'IC. En effet, l'avènement des TIC a amené de nouveaux points de convergence entre les deux domaines. On peut citer par exemple la construction et l'utilisation d'ontologies [24, 54, 78], la modélisation de documents pédagogiques en XML, les hypermédias adaptatifs, etc. Enfin, l'IC et le domaine des EIAH se rejoignent très largement sur les travaux sur l'apprentissage et le travail collaboratif (CSCL : Computer Supported Collaborative Learning, CSCW : Computer Supported Collaborative Work), deux problématiques que certains proposent même de confondre [9]. Dans le domaine des EIAH, les outils auteurs ont été proposés pour aider les concepteurs d'EIAH, entre autres, dans l'élicitation des connaissances nécessaires à l'EIAH, sans aucune ou peu de connaissances en programmation. Nous présentons dans la section suivante les méthodes et techniques d'acquisition de connaissances issues de l'IC et utilisées lors de la conception d'outils auteurs en EIAH.

2.4 Outils auteurs

Les outils auteurs sont des environnements informatiques permettant de construire tout ou partie des EIAH avec un coût réduit ou nul en programmation. De manière générale, les outils auteurs ont pour objectif de minimiser, d'une part le coût de conception des EIAH, et d'autre part la nécessité d'une grande expertise chez l'auteur [3]. Ils permettent par exemple d'éliciter les connaissances nécessaires à l'EIAH, de construire graphiquement l'interface de l'EIAH en utilisant des Toolkits et UI-Builder, de créer les exercices à résoudre, de proposer de l'aide à l'apprenant, de diagnostiquer ses réponses, etc. [57,59]. Les recherches sur les outils auteurs ont débuté il y a une quinzaine d'années. Ce champ permet d'espérer résoudre le problème de la difficulté et du coût de conception des EIAH. En effet, comme le soutient Woolf [89], une plus grande offre d'outils auteurs permettrait de supporter le développement plus rapide d' EIAH, les rendant ainsi plus variés.

À l'heure actuelle, plusieurs outils auteurs - plus de deux douzaines - ont été proposés dans la littérature. Ils sont souvent divisés en deux grandes catégories [57,60] :

— Ceux s'attachant à construire des parcours pédagogiques à partir de ressources existantes ;

— Ceux visant la construction d'EIAH de type tuteurs intelligents, fondés sur une représentation des connaissances permettant la résolution de problèmes et la construction de rétroactions pertinentes pour l'apprenant.

2.4.1 Outils auteurs pour la création de parcours pédagogiques

Dans la catégorie d'outils auteurs destinés à créer des parcours pédagogiques, nous retrouvons les travaux autour des scénarios d'apprentissage [34,39] et de la modélisation pédagogique [48]. Les travaux sur les outils auteurs permettant de créer des hypermédias adaptatifs [34] tels que les interbook, métalinks, et ceux permettant d'expliciter des stratégies d'enseignement [1] font également partie de cette catégorie.

Énormément d'outils ont été créés dans cette catégorie. Toutefois, de manière générale, la limite notoire de ces outils est le manque de connaissances sur le domaine et sur l'apprenant [88]. Ces outils ne répondent donc pas à notre problématique, plutôt centrée

sur la représentation des connaissances. C'est la raison pour laquelle nous ne présentons pas individuellement ces outils. Nous nous focalisons plutôt sur les outils destinés à créer des tuteurs intelligents, qui sont les plus proches de notre problématique.

2.4.2 Outils auteurs pour la création de tuteurs intelligents

Les outils auteurs développés dans cette catégorie visent la construction d'EIAH de type *tuteurs intelligents*, fondés sur une représentation des connaissances. Dans cette catégorie, certains outils se sont focalisés sur l'acquisition de connaissances procédurales. On y retrouve des systèmes opérationnels tels que Knomic [82], Disciple [80], Demonstr8 [12], RIDES [56], XAIDA [60], SIMQUEST [81], etc.

Knomic par exemple, est un système orienté learning-by-observation qui permet d'acquérir des connaissances procédurales dans un environnement de simulation. Pour ce faire, l'expert effectue ses tâches dans l'environnement simulé en annotant les points où il/elle a changé un but qu'il n'a pas réussi ou a abandonné. Le système génère ensuite les traces d'observations correspondantes. Knomic observe ces traces et utilise un algorithme de généralisation pour apprendre les conditions des actions, buts et opérateurs. Demonstr8 quant à lui est un outil auteur destiné à créer des model-tracing tuteurs pour le domaine de l'arithmétique. Afin de réduire le coût de conception de ces tuteurs, cet outil utilise la programmation par démonstration. Il propose un outil graphique pour concevoir l'interface de l'EIAH. Le système définit automatiquement chaque élément de l'interface comme un élément de mémoire de travail (WME), et les WME sont définis manuellement. Le système génère ensuite des règles de production en observant la résolution des problèmes. Cependant, ces outils se limitent à l'acquisition de connaissances procédurales. Ils ne permettent donc pas d'acquérir toutes les connaissances nécessaires aux EIAH de type AMBRE. En effet, dans AMBRE, en plus des connaissances de reformulation (sous forme de règles), les systèmes à base de connaissances sous-jacents utilisent des connaissances de classification de problèmes (sous forme d'arbre), des connaissances sur l'analogie entre les problèmes, des connaissances de génération de problèmes, des connaissances de communication avec l'apprenant, etc.

D'autres outils auteurs se sont concentrés sur la création de tuteurs à base de contraintes. Dans ASPIRE [52,53] par exemple, les auteurs utilisent des ontologies pour recueillir les connaissances expertes : (1) construction d'une ontologie du domaine, (2)

acquisition de contraintes syntaxiques directement à partir de l'ontologie, (3) mise en place d'une boîte de dialogue avec l'auteur afin de déduire des contraintes syntaxiques, (4) et utilisation d'un algorithme d'apprentissage pour déduire les contraintes sémantiques. Cependant, ces outils sont limités à des domaines où l'ensemble des connaissances peuvent être représentées sous forme de contraintes. Une contrainte est représentée par le motif <Cr,Cs>, tel que Cr représente le critère de pertinence, et Cs le critère de satisfaction. Plus spécifiquement, Cr spécifie l'ensemble des situations pour lesquelles la contrainte est pertinente, et Cs l'ensemble des situations pour lesquelles la contrainte est satisfaite. La sémantique représentée par une contrainte est que si Cr est satisfait dans une situation, alors Cs devrait aussi l'être, sinon il y a erreur.

Les outils auteurs développés autour de CTAT (Cognitive Tutor Authoring Tools) permettent la création de deux types de tuteurs : les *cognitives tutors* [46] et les *example-tracing tutors* [4, 5, 46, 47]. Les premiers sont fondés sur un modèle cognitif à base de règles de production, et portent en général sur des tâches de résolution. La résolution est conduite pas à pas, et le comportement de l'élève est analysé à chaque étape, puis corrigé s'il s'écarte de la procédure prévue. Cependant, les cognitives tutors se limitent aux domaines ou la tâche de résolution de problèmes s'effectue pas à pas et ou l'ensemble des connaissances du domaine peut s'exprimer sous forme de règles de production. Ils ne permettent donc pas l'apprentissage de connaissances déclaratives comme les concepts, ni l'apprentissage à partir d'exemples [89]. De plus, la création de outils nécessite des compétences en programmation. Les example-tracing tutors [4] ont quant à eux l'avantage d'être développés rapidement et sans programmation. Pour construire un tuteur de ce type, l'enseignant commence d'abord par définir l'interface de l'apprenant à l'aide d'outils graphiques. Il définit ensuite, à l'aide d'un graphe de résolution, les stratégies de résolution des apprenants et leurs misconceptions. Cependant, les example-tracing tutors présentent les mêmes inconvénients que les cognitive tutors concernant la résolution des problèmes qui est conduite pas à pas, et par conséquent ne répondent pas au principe des EIAH AMBRE qui demandent à l'apprenant de prendre du recul avant d'entamer la résolution.

2.5 Synthèse des principaux verrous

Nous avons vu dans ce chapitre que la littérature en IC abonde de méthodologies d'acquisition de connaissances. La méthodologie commonKADS a particulièrement retenu notre attention en ce sens qu'elle est basée sur une démarche d'acquisition de connaissances dirigée par les modèles. Cependant, ses composants génériques permettant de résoudre des problèmes très généraux ne sont pas forcément adaptés à ce que nous cherchons à enseigner en EIAH. En effet, dans le domaine des EIAH, la démarche d'élicitation se particularise par le fait que :

— Il ne s'agit pas de modéliser les connaissances de l'expert, mais les connaissances telles qu'on voudrait qu'elles fonctionnent chez l'élève après l'apprentissage ;

— Il existe d'autres types de connaissances à modéliser, telles que les connaissances de diagnostic des compétences de l'apprenant, ou les connaissances permettant de lui fournir des rétroactions ou de l'assistance. Or ces connaissances ne sont pas des connaissances de résolution de problèmes proprement dites : par exemple, lorsque l'on parle de diagnostic des connaissances de l'apprenant en EIAH, il ne s'agit pas de la même tâche qu'un diagnostic de panne que l'on pourrait modéliser avec la méthode CommonKads.

C'est pourquoi, compte tenu des problématiques spécifiques au champ des EIAH, nous ne retiendrons de cette méthode que sa démarche.

La méthodologie KOD, quant à elle, présente l'avantage d'être bien adaptée à l'acquisition de connaissances exprimées en langage naturel. Cependant, les modèles issus de cette approche souffrent d'un manque d'abstraction, de généricité, et sont donc difficilement réutilisables. De plus, KOD ne propose rien permettant de distinguer les différents niveaux de connaissances de l'expert, ce qui peut nuire à l'explicabilité du système. Un système destiné à enseigner une stratégie de résolution (comme ceux produits dans AMBRE) doit avoir accès à une représentation explicite des connaissances et du processus mis en jeu. Cela permet de faciliter la production d'explications au sein de l'EIAH ; celles-ci sont d'autant plus faciles à donner que les connaissances sont explicitées. Par conséquent, nous ne retiendrons de cette méthode que son approche ascendante.

À l'heure actuelle, plus de deux douzaines d'outils auteurs ont été proposés dans le domaine des EIAH. Ces outils diffèrent pour la plupart par le type de domaine ou de tâches prises en compte, le degré par lequel ils facilitent la création d'EIAH, et les

possibilités offertes pour représenter ou non les connaissances ou les compétences à enseigner [58]. Chacun de ces outils a ses forces et faiblesses. C'est pourquoi nous proposons de discuter des limites des approches existantes en considérant deux critères d'analyse liés à notre problématique de recherche : l'approche de résolution de problème adoptée, et la possibilité de représenter toutes les connaissances dont on a besoin (cf. Chapitre 1).

Approche de résolution de problèmes

Les outils développés autour de CTAT sont les plus aboutis actuellement en terme de création d'EIAH centrés connaissances. Cependant, ils se limitent à des domaines ou la tâche de résolution de problèmes s'effectue pas à pas. Ils ne répondent donc pas à notre problématique d'outils auteurs pour l'enseignement de méthodes, qui demande à l'apprenant de prendre du recul et de raisonner globalement sur le problème à résoudre.

Possibilité de représenter les connaissances dont a besoin

La principale limite des outils auteurs destinés à l'acquisition de connaissances procédurales est qu'ils ne permettent pas de représenter toutes les connaissances nécessaires à la conception des EIAH auxquels nous nous intéressons (arbre de classification, règles, techniques de résolution). En effet, ces outils se limitent à des domaines où l'ensemble des connaissances peuvent être représentées sous forme de connaissances procédurales. La limite des outils auteurs basés sur les contraintes repose sur le fait que les connaissances du domaine sont représentées par un ensemble de contraintes. En effet, les connaissances nécessaires à la conception d'EIAH destinés à l'enseignement de méthodes ne peuvent pas être représentées sous forme de contrainte.

Notre analyse de l'existant ne nous a pas permis de trouver d'approche permettant de répondre complètement à notre problématique : comment permettre à un auteur d'éliciter les connaissances nécessaires à conception d'un EIAH destiné à enseigner des méthodes. L'approche que nous proposons consiste donc à :

— Proposer des méta-modèles de connaissances pour formaliser les connaissances qui entrent en jeu lors de la conception des EIAH destinés à enseigner des méthodes. Pour cela, nous adopterons une approche ascendante pour la construction des méta-modèles ;

— Proposer un processus d'élicitation des connaissances en nous fondant sur ces méta-modèles (approche dirigée par les méta-modèles). Il s'agira en particulier de proposer des interfaces permettant d'accompagner l'auteur dans l'élicitation des connaissances.

Deuxième partie

CONTRIBUTIONS THÉORIQUES

Chapitre 3

Des méta-modèles pour guider l'élicitation des connaissances

Dans ce chapitre, nous décrivons les méta-modèles que nous avons proposés pour formaliser les connaissances nécessaires à la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes. Ces méta-modèles sont indépendants du domaine. Ils permettent d'une part de définir la forme des connaissances à acquérir, et d'autre part de contraindre le processus d'acquisition de ces connaissances décrit dans le chapitre 4.

Une première partie présente les méta-modèles permettant de décrire les problèmes à résoudre, alors que la suivante se consacre aux méta-modèles des connaissances portant sur la méthode à enseigner. Ce chapitre se termine par la description des méta-modèles des connaissances permettant d'accompagner l'apprenant durant son apprentissage.

Sommaire

3.1 Comment décrire les problèmes d'un domaine ? 69
3.1.1 Méta-modèle du vocabulaire permettant de décrire des problèmes 70
3.1.2 Méta-modèle permettant de décrire des problèmes 71
3.2 Comment définir les connaissances sur la méthode à enseigner ? 73
3.2.1 Méta-modèle des connaissances de classification 74
3.2.2 Méta-modèle des connaissances de reformulation 77
3.2.3 Méta-modèles des connaissances de résolution 82
3.3 Comment définir les connaissances permettant d'accompagner l'apprenant ? 86
3.3.1 Méta-modèle des connaissances d'aide 87
3.3.2 Méta-modèle des connaissances de diagnostic 89
3.3.3 Connaissances d'explications sur les erreurs 91
3.4 Conclusion 93

Les travaux menés dans le cadre du projet AMBRE et présentés dans le chapitre 1 (cf. chapitre 1) de ce rapport permettent la conception d'EIAH destinés à l'enseignement de méthodes. Ces EIAH proposent à l'apprenant un processus d'apprentissage inspiré du raisonnement à partir de cas : le cycle AMBRE [43]. Ils reposent sur des systèmes à base de connaissances leur permettant de résoudre des problèmes en appliquant la méthode à enseigner, de diagnostiquer les réponses de l'apprenant, et de lui fournir de l'aide et des explications pertinentes sur ses erreurs. La conception de tels EIAH est cependant très coûteuse, en particulier du fait que les connaissances doivent être explicitées en Prolog, un langage informatique de représentation des connaissances utilisé par ces systèmes.

Afin de remédier à ce problème, nous souhaitons permettre à des auteurs (enseignant, professionnel ou pédagogue plus expert), de pouvoir expliciter eux-mêmes les connaissances nécessaires à l'EIAH, sans recourir à de la programmation. Il devient dès lors nécessaire d'identifier et de formaliser les différents types de connaissances entrant en jeu dans la conception d'un EIAH AMBRE. Pour ce faire, nous avons proposé des méta-modèles décrits par des schémas XML. Ces méta-modèles constituent ainsi une réponse au problème de formalisation des connaissances posé dans l'introduction générale de ce manuscrit. Pour la construction de nos méta-modèles, nous avons adopté une méthode ascendante [77] en comparant les bases de connaissances des différents EIAH AMBRE existants.

3.1 Comment décrire les problèmes d'un domaine ?

Dans AMBRE, les problèmes à résoudre ne sont pas donnés au système en langue naturelle, mais plutôt en Prolog sous la forme d'un modèle que nous appelons modèle descriptif du problème [40]. Ce modèle décrit une situation concrète qui est celle représentée dans l'énoncé. Il contient des traits de surface du problème considéré : par exemple dans les problèmes additifs, ce sont les éléments concrets tels que billes, les jetons, les bonbons, etc. Ils forment l'aspect contextuel du problème appelé aussi encodage linguistique.

Pour permettre au résolveur de AMBRE de résoudre des problèmes, il faut d'abord pouvoir décrire ces problèmes, c'est-à-dire les éléments de l'énoncé du problème. De plus, la description des règles et des techniques de résolution (cf. Chapitre 1) nécessite souvent

aussi de pouvoir se référer aux éléments de l'énoncé. Une question importante à laquelle il fallait répondre était alors :
Comment décrire les éléments descriptifs de l'énoncé des problèmes ?
Notre réponse à cette question a été de proposer que pour chaque domaine, la description des problèmes soit fondée sur un vocabulaire permettant de décrire les objets du domaine présents dans les énoncés. Ce vocabulaire va servir d'élément d'ancrage autour duquel s'organise la description des problèmes à résoudre, les règles permettant de les situer au niveau de la classification des problèmes, et les techniques intervenant dans leur résolution. De ce fait nous avons proposé un méta-modèle permettant de décrire un vocabulaire pour un domaine donné.

3.1.1 Méta-modèle du vocabulaire permettant de décrire des problèmes

Pour créer le vocabulaire d'un domaine, il faut donc créer les types d'objets qui vont le constituer. Chacun d'entre eux doit posséder un *identifiant* et une *liste de caractéristiques*. Chaque caractéristique a un *nom* et un *type* qui peut être simple (chaîne de caractères, booléen, nombre, etc.) ou complexe (un objet représentant une relation avec un autre objet). Par exemple, dans les problèmes additifs que nous présentons dans les sections suivantes, pour décrire les listes d'éléments possédés par chaque personne, un objet *liste_éléments_possédes* avec les caractéristiques suivantes peut être créé : *identifiant* de type chaîne de caractères, *taille_des_éléments_possédés* de type nombre entier, et le *type_des_éléments_possédés* de type chaîne de caractères.

La **Figure 3.1** présente le méta-modèle proposé pour définir le vocabulaire pour un domaine donné. Pour chaque domaine choisi (cf *nom_du_domaine*), il faut alors définir le vocabulaire, c'est-à-dire créer les types d'objets nécessaires pour décrire les problèmes de ce domaine. Créer un type d'objet revient à définir :

— Son **identifiant**

— La liste de ses **caractéristiques**. Pour chaque caractéristique, il faut préciser le nom (cf nom_caractéristique) ainsi que le type (cf type_caractéristique) qui peut être : type_simple ou type_objet.

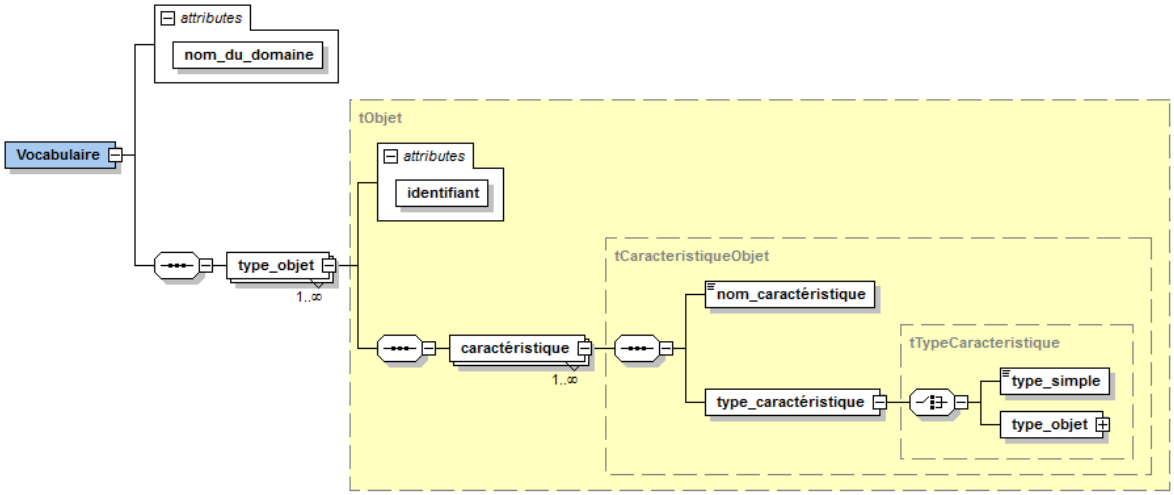

FIGURE 3.1 – Méta-modèle permettant de décrire le vocabulaire d'un domaine

Une fois défini, le vocabulaire pourra alors être utilisé dans la description des différents problèmes à résoudre par le système et par l'apprenant.

3.1.2 Méta-modèle permettant de décrire des problèmes

La description des problèmes à résoudre s'appuie sur le vocabulaire défini pour le domaine choisi. Ainsi, pour chaque problème, il faut choisir, dans l'ensemble des types d'objets du vocabulaire défini pour ce domaine, ceux qui sont nécessaires pour décrire l'énoncé, puis les instancier, nous les appelons objets du problèmes. Il faut ensuite spécifier la question à résoudre pour ce problème en précisant la caractéristique sur laquelle porte l'inconnue. La **Figure 3.4** représente le méta-modèle proposé pour décrire un problème en s'appuyant sur un vocabulaire.

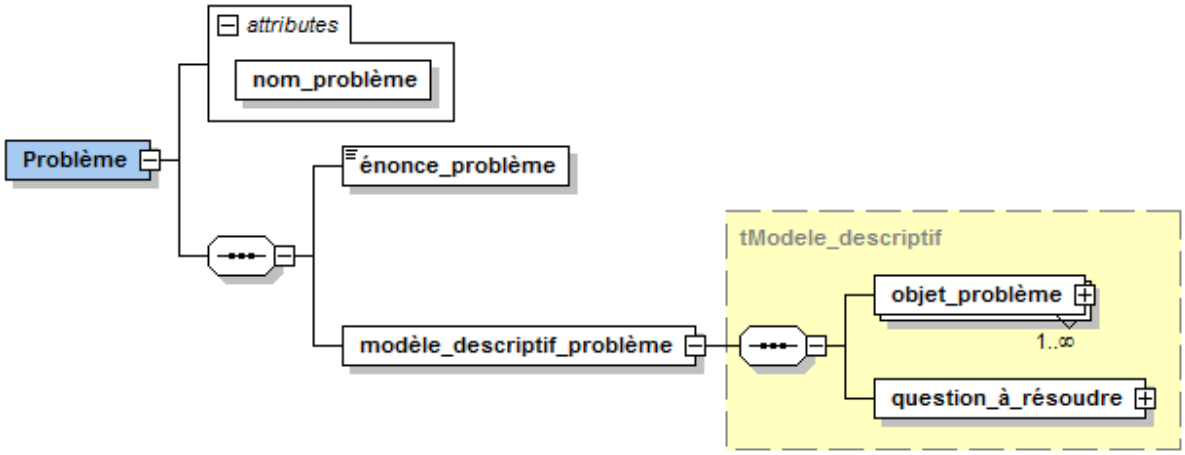

FIGURE 3.2 – Méta-modèle permettant de décrire un problème en s'appuyant sur le vocabulaire

Prenons le cas des problèmes additifs étudiés à l'école élémentaire. Ces problèmes décrivent une situation concrète comme celle représentée dans l'exemple du problème *P2* suivant :
« *Paul a 31 billes rouges. En ajoutant les billes rouges de Pierre, Paul et Pierre en ont 44 à eux deux. Trouver le nombre de billes rouges de Pierre* ».
Pour pouvoir décrire ce genre de problèmes, le vocabulaire à définir doit contenir des objets permettant de décrire les personnes impliquées, les ensembles formés, etc. En utilisant le méta-modèle pour la définition du vocabulaire (cf. **Figure 3.1**), nous pouvons, entre autres, définir les types d'objets suivants pour décrire les problèmes additifs :

— **Personne** : pour désigner les personnes impliquées dans les problèmes. Une personne a un identifiant et possède une liste d'éléments.

— **Liste_d'éléments** : pour désigner les listes d'éléments possédés par chaque personne. Chaque liste est caractérisée par sa taille, et le type des éléments qui la constituent.

— **Liste_de_personnes** : pour désigner la liste des personnes impliquées, avec des listes d'éléments de même type. Cette liste de personnes possède une liste d'éléments caractérisée par sa taille et le type des éléments possédés par les personnes de la liste.

Ainsi, en utilisant le méta-modèle de description des problèmes, le problème *P2* pourrait par exemple être décrit comme suit :

Énoncé du problème P2 :
Paul a 31 billes rouges. En ajoutant les billes rouges de Pierre, Paul et Pierre en ont 44 à eux deux. Trouvez le nombre de billes rouges de Pierre.

Modèle descriptif du problème P2 :

- Paul possède une liste d'éléments E1 de taille 31 et de type "billes rouges".
- Pierre possède une liste d'éléments E2 de taille X et de type "billes rouges".
- Nous disposons d'une liste de personnes P comportant deux personnes : Paul et Pierre. La liste P possède une liste d'éléments E de taille 44 et de type "billes rouges".
- La question à résoudre consiste à calculer la taille X de la liste d'éléments E2 possédé par Pierre.

Dans AMBRE, lorsque le résolveur doit résoudre un problème, il effectue une phase d'opérationnalisation afin d'identifier la classe du problème et d'appliquer la technique de résolution associée. Pour cela, il utilise les connaissances sur la méthode à enseigner.

3.2 Comment définir les connaissances sur la méthode à enseigner ?

Dans notre contexte, une méthode permet d'organiser la résolution de problèmes en se basant sur une classification des problèmes et des outils de résolution. Résoudre un problème en utilisant une méthode consiste alors à identifier la classe dont relève ce problème afin de choisir une technique de résolution adaptée. Dans AMBRE, les connaissances sur la méthode, c'est-à-dire intervenant dans la résolution de problèmes, sont de trois types : des connaissances de classification, des connaissances de reformulation et des connaissances de résolution. Pour un domaine donné, l'association de ces trois types de connaissances au processus de raisonnement conduit à l'obtention d'un résolveur qui fonctionne selon la méthode modélisée.

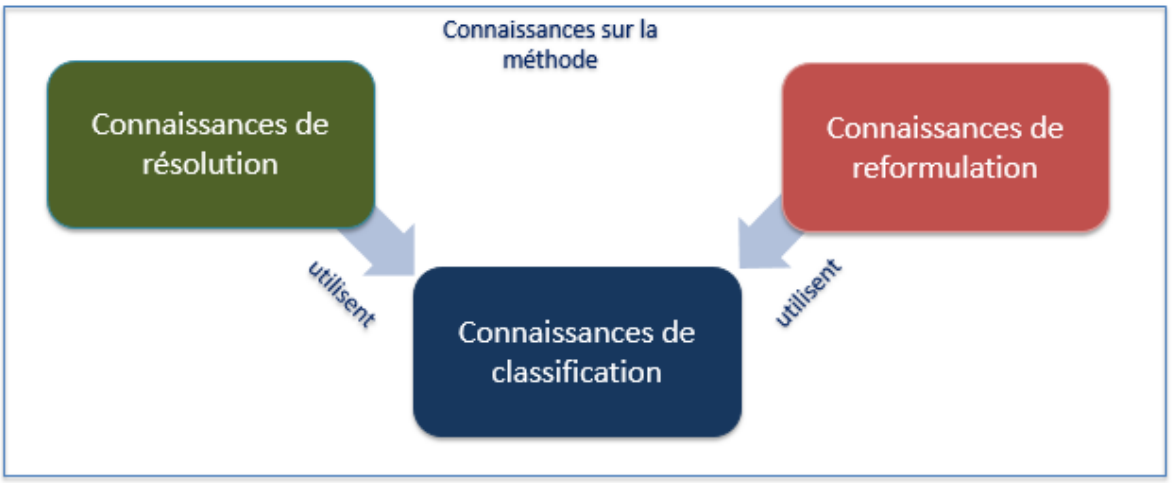

FIGURE 3.3 – Lien entre les connaissances de classification, de reformulation et de résolution

Dans les travaux précédents de l'équipe, une étude exploratoire a été menée sur la formalisation des connaissances sur la méthode (B. K. W. Chang et A. K. W. Chang en collaboration avec N. Guin et M. Lefevre, 2013). Afin de compléter cette étude, nous avons proposé des méta-modèles permettant de décrire les connaissances sur la méthode à enseigner : classification, reformulation, résolution.

3.2.1 Méta-modèle des connaissances de classification

Comme J. Pitrat [44], nous jugeons intéressant de changer la représentation d'un problème afin de se ramener à un problème pour lequel nous disposons d'outils efficaces pour le résoudre. La classification des problèmes du domaine est une connaissance de niveau méta qui permet de classer un problème donné et ainsi choisir une meilleure représentation et une meilleure technique de résolution. Elle permet également de donner une idée sur la difficulté du problème à résoudre. C'est cette classification des problèmes que nous nommons connaissances de classification.

Pour définir ces connaissances, une hiérarchie des classes de problèmes à résoudre doit être définie pour chaque domaine d'application. Cette hiérarchie se présente sous la forme d'un arbre de classification où une classe C2 est une sous-classe d'une classe C1 si tout problème de C2 est aussi un problème de C1. Dans un tel arbre, une classe

racine est définie comme étant la classe la plus générale, les feuilles étant les classes les plus spécifiques. Les classes suffisamment spécifiques qu'on puisse leur associer une technique de résolution sont des classes dites opérationnelles. La **Figure 3.4** présente le méta-modèle proposé pour représenter une classe de problème. Créer une classe revient ainsi à définir :

— Son **nom**.

— Son **statut** : opérationnelle ou non opérationnelle.

— La **valeur de l'attribut qui discrimine cette classe par rapport à son parent**, sauf pour la classe racine qui n'a pas de parent.

— Un **attribut discriminant** permettant de discriminer les sous-classes de cette classe, sauf pour les feuilles qui n'ont pas de sous-classes.

— D'autres **attributs du problème**, s'il y en a besoin pour la résolution des problèmes de cette classe.

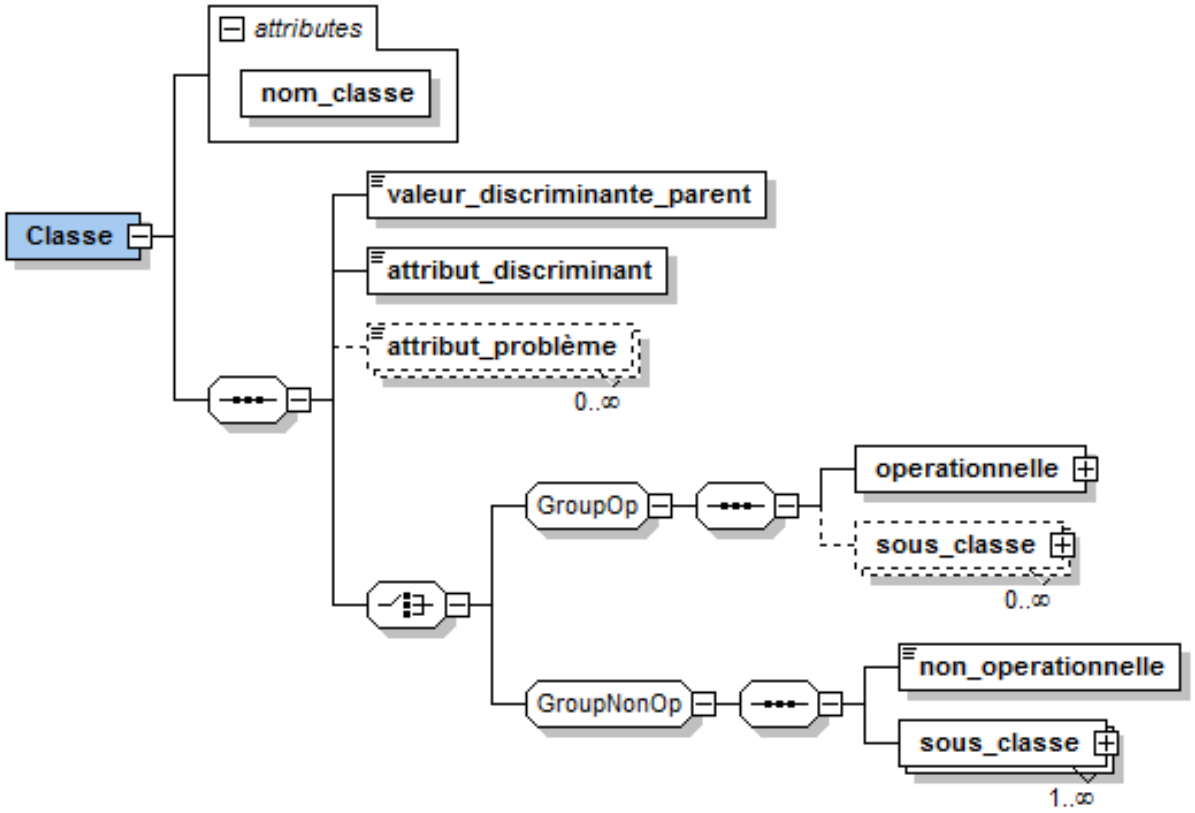

FIGURE 3.4 – méta-modèle permettant de décrire une classe de problèmes

Reprenons notre exemple sur la conjugaison française (cf. Introduction générale) : Problème *P1* : « *Conjuguer le verbe Parler au présent de l'indicatif* ».

Pour résoudre un problème, le système a besoin de déterminer d'abord la classe à laquelle appartient ce problème, afin de choisir par la suite la technique de résolution la plus adaptée. La **Figure 3.5** représente une partie de l'arbre de classification pour la conjugaison française. *C_Conjugaison* représente la classe racine et a pour attribut discriminant *temps_simples*. Les valeurs discriminantes de cet attribut sont : *présent*, et *futur*. Cette classe possède donc deux sous-classes : une classe non opérationnelles *C_Présent* ainsi qu'une classe opérationnelle *C_Futur*. L'attribut discriminant de la classe *C_Présent* est groupe. Les valeurs discriminantes de cet attribut sont : *1er groupe* et *2ème groupe*. La classe *C_Présent* possède donc deux sous-classes : la classe *C_Présent_1er_Gr* qui est non opérationnelle et la classe *C_Présent_2ème_Gr* qui est opérationnelle.

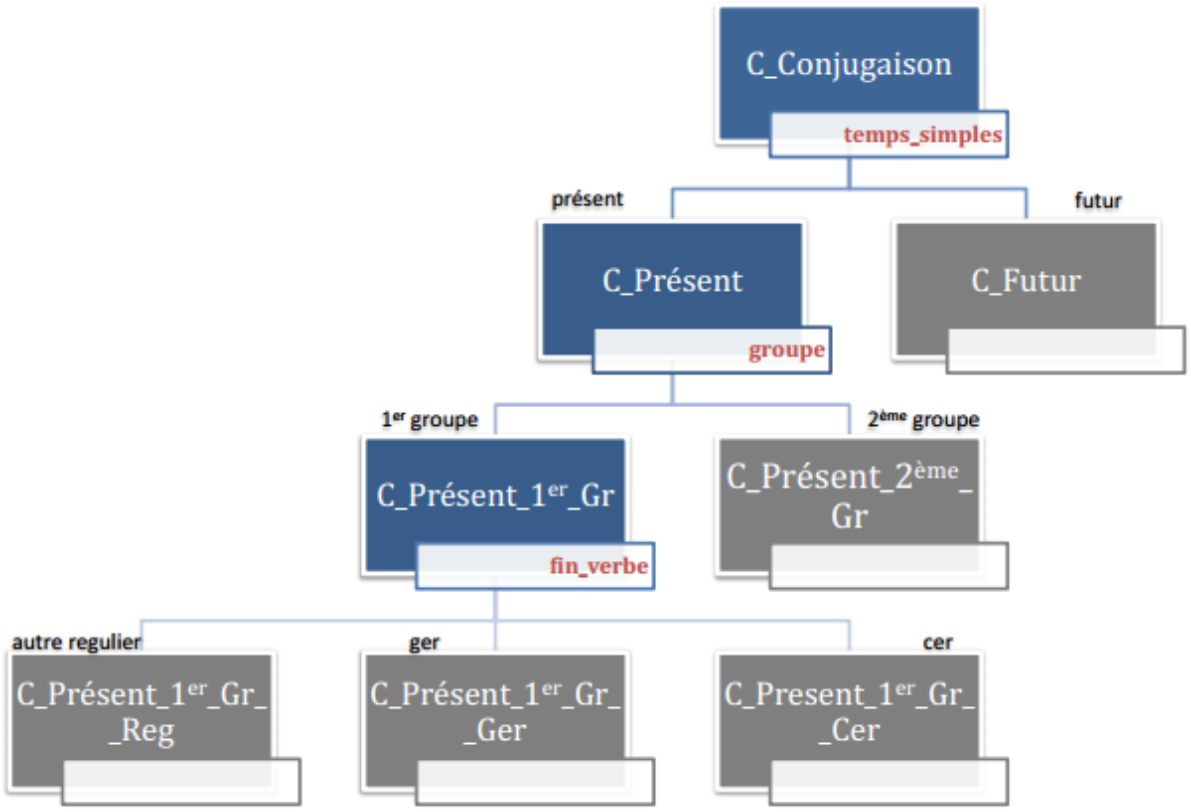

FIGURE 3.5 – Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif

Pour classer un problème dans AMBRE, le résolveur utilise l'arbre de classification comme guide. Partant de la racine de l'arbre, il affine progressivement le classement du problème pour passer à une classe plus spécifique. En d'autres termes, il parcourt l'arbre en profondeur jusqu'aux feuilles ou au moins jusqu'à une classe opérationnelle. Lorsqu'une classe opérationnelle est atteinte, le système essaie quand même de descendre vers une sous-classe pour avoir un classement le plus précis possible. Cette descente se termine soit lorsqu'une feuille est atteinte, soit parce que la valeur de l'attribut discriminant n'est pas celle qui permet de descendre vers les cas particuliers. On obtient alors la classe la plus spécifique selon laquelle les connaissances permettent de modéliser le problème.

Descendre dans une sous-classe revient alors à déterminer la valeur de l'attribut discriminant. Cependant, les attributs intervenants dans la classification sont souvent des attributs relevant de la théorie du domaine. Ces attributs ne sont la plupart du temps pas présents dans les modèles descriptifs des problèmes. Il est donc nécessaire de recourir à des connaissances permettant de déterminer la valeur de ces attributs. Ces connaissances constituent ce que nous nommons : connaissances de reformulation.

3.2.2 Méta-modèle des connaissances de reformulation

Pour chaque attribut (discriminant ou non) de l'arbre de classification, l'auteur doit définir des règles si-alors. Celles-ci permettent, à partir de l'énoncé du problème, de calculer la/les valeur(s) des attributs afin de situer ce problème dans l'arbre de classification. L'ensemble de ces règles constitue les connaissances de reformulation (cf. **Figure 3.6**).

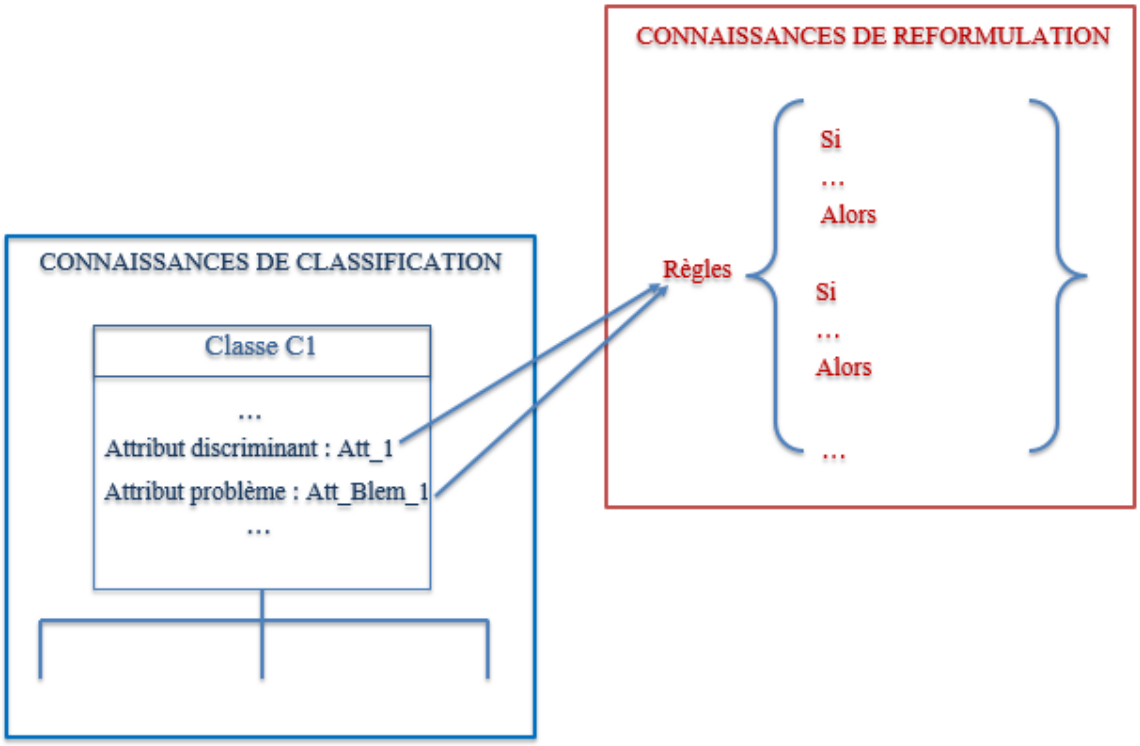

FIGURE 3.6 – Lien entre connaissances de classification et connaissances de reformulation

Nous avons vu dans le chapitre 1 que chaque règle est définie par :

— Son **nom** ;

— Un **ensemble de prémisses** portant sur des éléments de l'énoncé, c'est-à-dire les différents objets du problème ou des contraintes sur ces objets. Les prémisses peuvent également porter sur des attributs du problème qui pourront eux-mêmes nécessiter l'application de règles pour calculer leurs valeurs. On peut également y trouver des appels à des fonctions de calcul. Les connaissances représentées dans les prémisses sont des traitées avec l'opérateur logique ET.

— Un **ensemble de conclusions** permettant de calculer ou de modifier les valeurs des attributs du problème. On peut y retrouver également des appels à des fonctions de calcul.

Pour décrire les prémisses d'une règle, nous avons utilisé le méta-modèle du vocabulaire défini plus haut (cf. section 1.1) pour décrire les éléments de l'énoncé. Un verrou à lever dans le cadre de ce travail était également de pouvoir définir des fonctions de

calcul que l'on retrouve parfois dans les conclusions des règles, mais également lors de la définition des techniques de résolution. De ce fait, nous avons proposé un méta-modèle permettant de décrire des fonctions de calcul. Pour créer une fonction, il faut définir :

— Son **nom**.

— Sa **description en langue naturelle**. Il peut également spécifier dans cette partie les valeurs possibles de la fonction.

— Son **nombre de paramètres** ainsi que leur type.

— Son **type de retour**.

Les **Figures 3.7** et **3.8** représentent respectivement le méta-modèle permettant de décrire une fonction de calcul, et celui permettant de décrire une règle.

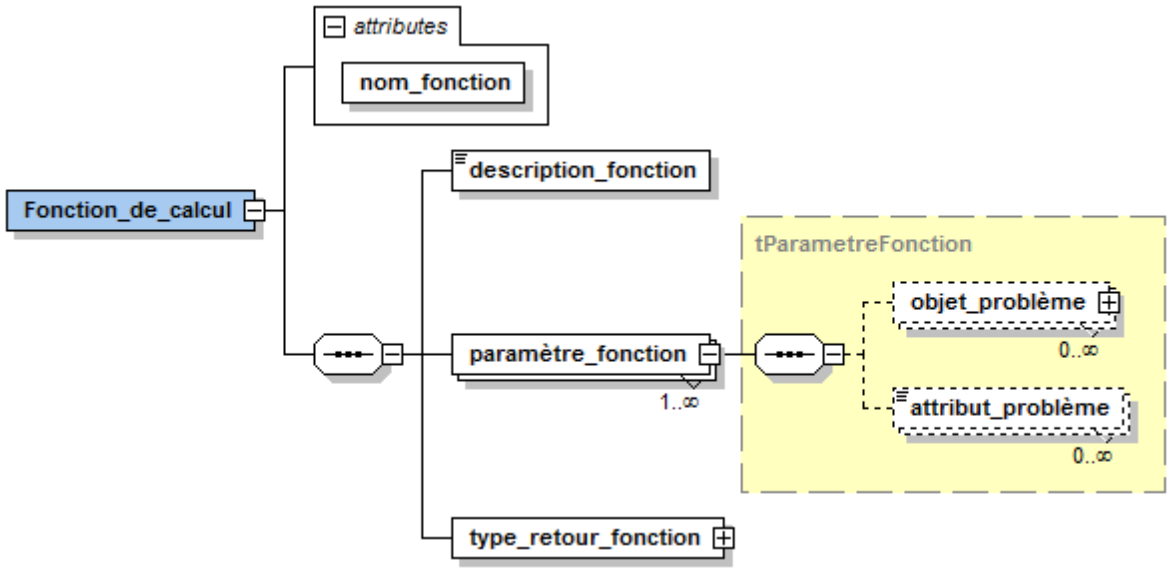

FIGURE 3.7 – Méta-modèle permettant de définir une fonction de calcul.

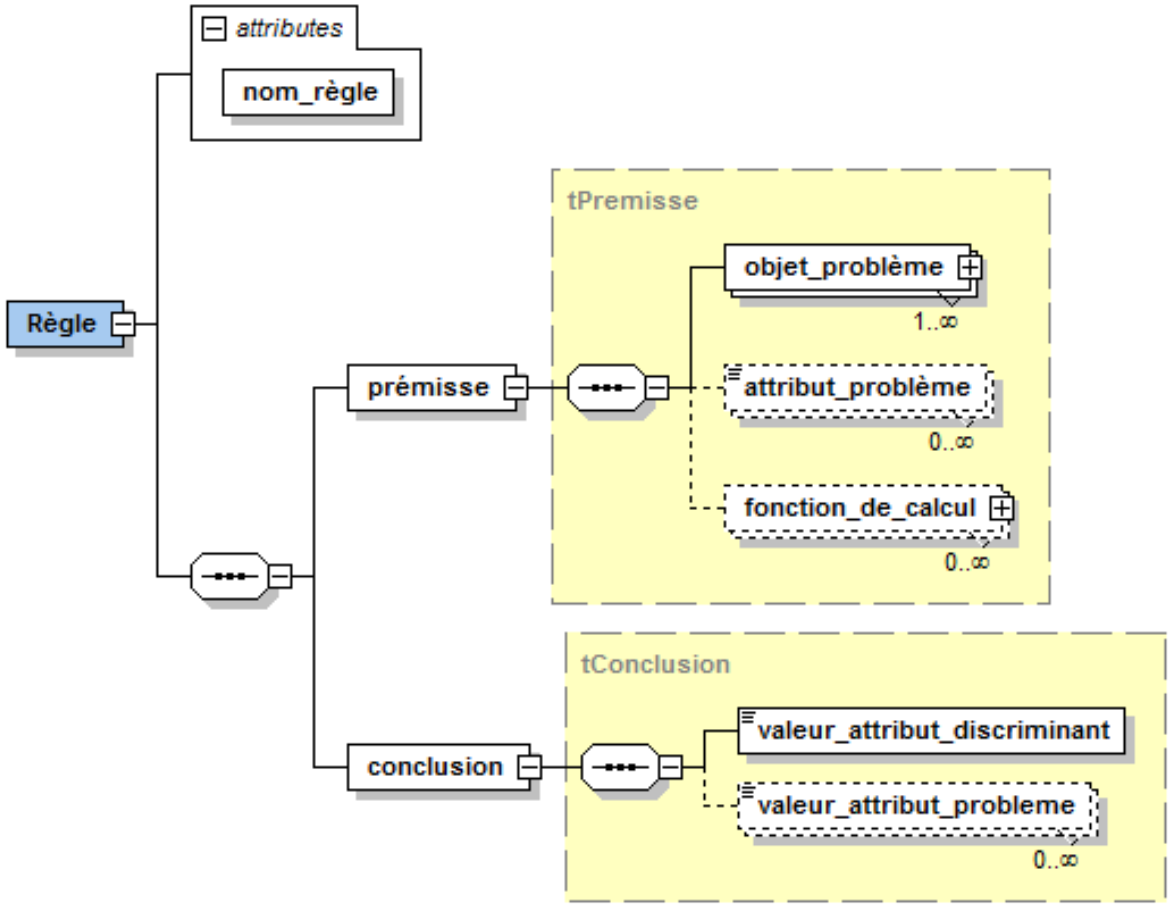

FIGURE 3.8 – Méta-modèle permettant de décrire une règle

Reprenons notre exemple P1, la conjugaison française : « *Conjuguer le verbe Parler au présent de l'indicatif* ».

Afin de déterminer la classe de ce problème, le système utilise l'arbre de classification (cf. **Figure 3.5**) ainsi que les règles de reformulation définies par l'auteur. Celui-ci doit donc définir des règles permettant de conclure sur la valeur des attributs discriminants : *temps_simples*, *groupe*, *fin_verbe*. Soient les règles *regle_temps*, *regle_groupe*, et *regle_fin_verbe* permettant de conclure respectivement sur ces attributs (cf. **Figure 3.9**)
.

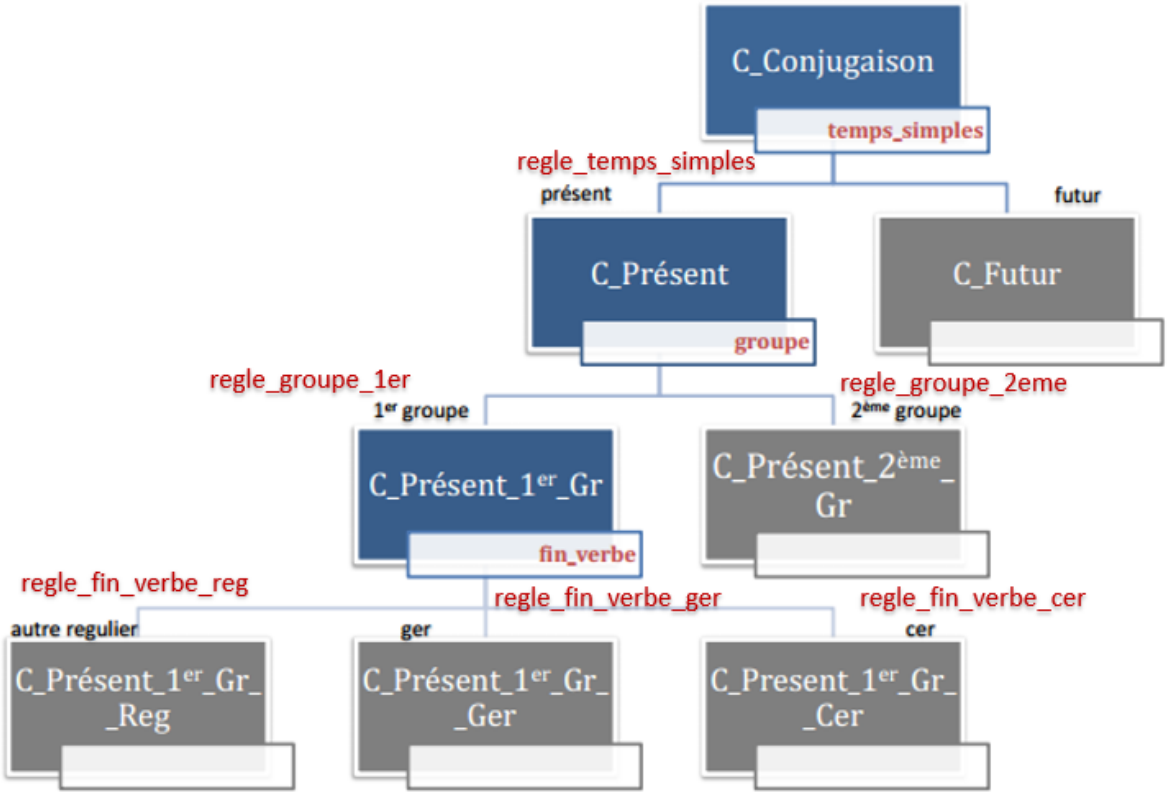

FIGURE 3.9 – Une partie de l'arbre de classification pour la conjugaison des verbes aux temps simples de l'indicatif, avec les règles de reformulation

/***règle permettant de conclure sur la valeur de l'attribut temps_simples***/ Si dans un problème, il existe : - un verbe V - un temps T Alors la valeur de l'attribut *temps_simples* est égale à *T*.
/***règle permettant de conclure sur la valeur de l'attribut groupe***/ Si dans un problème, il existe : - un verbe *V* - la terminaison du verbe V est égale à *er* - le verbe V est différent du verbe *aller* Alors la valeur de l'attribut *groupe* est égale à *1er groupe*.
/***règle permettant de conclure sur la valeur de l'attribut fin_verbe***/ Si dans un problème, il existe : - un verbe V - V est un verbe du premier groupe - le verbe V est différent du verbe aller Alors la valeur de l'attribut *fin_verbe* est égale à *autre régulier*.

Tableau 3.1 – Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème

En utilisant ces connaissances, le système parcourt respectivement les classes : *C_Conjugaison*, *C_Présent*, *C_Présent_1er_Gr*, pour arriver à la classe opérationnelle *C_present_1er_Gr_Reg*. Puisque cette classe est une feuille, le système en conclut que ce problème appartient à cette classe.

Les connaissances de reformulation permettent de passer du modèle descriptif d'un problème à un modèle opérationnel constitué d'un ensemble d'attributs avec leurs valeurs et plus adapté à l'utilisation d'une technique de résolution. L'ensemble des techniques de résolution constituent les connaissances de résolution.

3.2.3 Méta-modèles des connaissances de résolution

Pour que la classification soit utile dans la résolution des problèmes, une technique de résolution doit être associée à chaque classe opérationnelle de l'arbre de classification. (Cf. **Figure 3.10**).

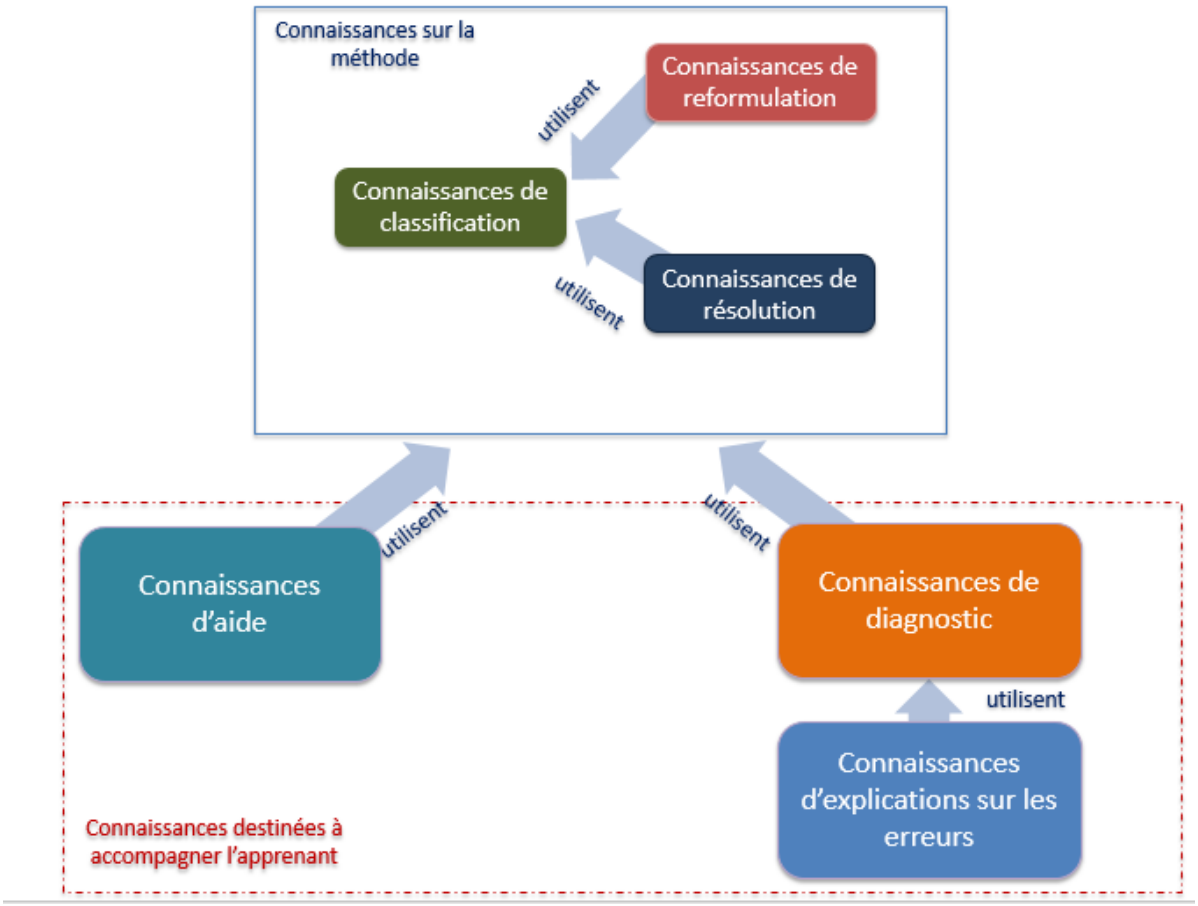

FIGURE 3.10 – Lien entre connaissances de reformulation et connaissances de résolution

La **Figure 3.11** représente le méta-modèle proposé pour décrire une technique. Chaque technique de résolution est définie par :

— Son **nom** ;

— Un **plan de résolution**. Il s'agit d'une description de type algorithmique des actions (étapes de résolution) à réaliser pour résoudre un problème. L'analyse de ce plan doit permettre de trouver la solution du problème. En général, c'est le type de plan qu'on demande aux élèves de fournir dans une copie pour justifier le résultat d'un problème. Dans notre cas, la description des actions portent sur les objets et/ou attributs du problème.

— Une **fonction de calcul** permettant de déterminer les éléments de solution à partir des données issues de l'énoncé.

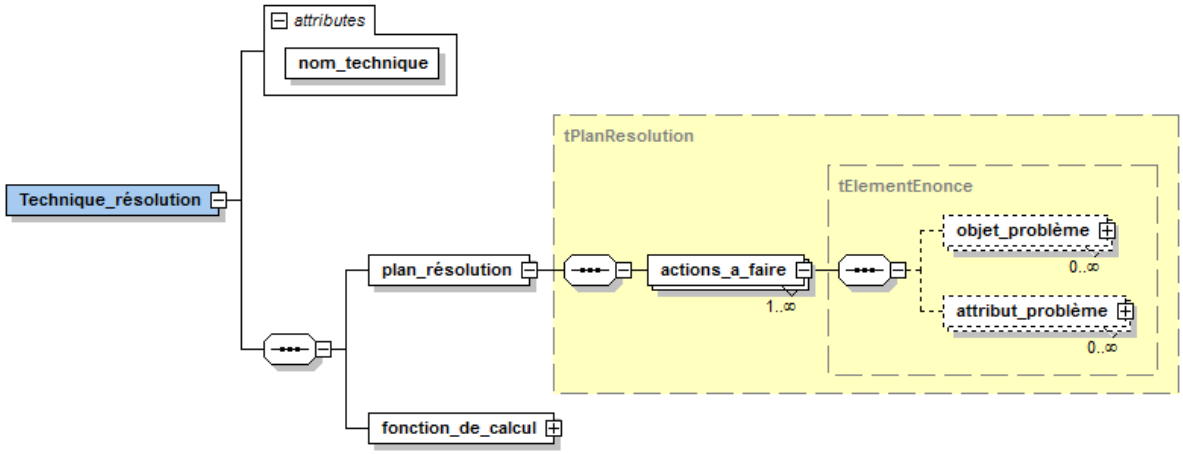

FIGURE 3.11 – Méta-modèle permettant de décrire une technique de résolution

Afin de guider l'auteur dans la formalisation des connaissances de résolution, une question cruciale soulevée est la suivante :
Comment permettre à un auteur de définir des fonctions de calcul intervenant dans l'explicitation des techniques de résolution ?
Notre réponse à cette question a été de proposer le méta-modèle illustré par la **Figure 3.7**.

Les techniques de résolution sont spécifiques au domaine. Pour la conjugaison française, il s'agit d'une part du processus permettant de déterminer le radical du verbe pour l'ensemble des personnes, et d'autre part la liste des terminaisons pour les six personnes.

Reprenons notre exemple sur la conjugaison française : « *Conjuguer le verbe parler au présent de l'indicatif* ».
Après la détermination de la classe du problème C_Présent_1erGroupe_Régulier (section 3.2), Il s'agit alors d'appliquer la technique associée à cette classe pour trouver la solution.

Technique associée à la classe C_Présent_1erGroupe_Régulier :

— Déterminer le radical du verbe : enlever le suffixe ou la terminaison du verbe (er dans notre cas) de l'infinitif du verbe (parler) pour obtenir son radical (parl) ;

— Déterminer la correspondance entre personnes et terminaisons en fonction du temps : au présent de l'indicatif les terminaisons des différentes personnes je, tu, il ou elle, nous, vous, ils ou elles sont respectivement *e, es, e, ons, ez, ent*.

Le **Tableau 3.2.3** et la **Figure 3.12** illustrent ainsi la solution à notre problème après application de cette technique de résolution.

Soit infinitif	= Parler
Soit radical	= Infinitif - "er"
	= "parl"
Soit liste_radical	= [radical,radical,radical,radical,radical,radical]
	= [parl,parl,parl,parl,parl,parl]
Soit liste_terminaisons	= [e,es,e,ons,ez,ent]
Soit liste_personnes	= [je,tu,il/elle,nous, vous,ils/elles]
Soit forme_conjuguée	= fusion des liste : [liste_radical, liste_terminaison]
Soit solution	= Concaténation des liste : [liste_personnes, forme_conjuguée]

Tableau 3.2 – Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème

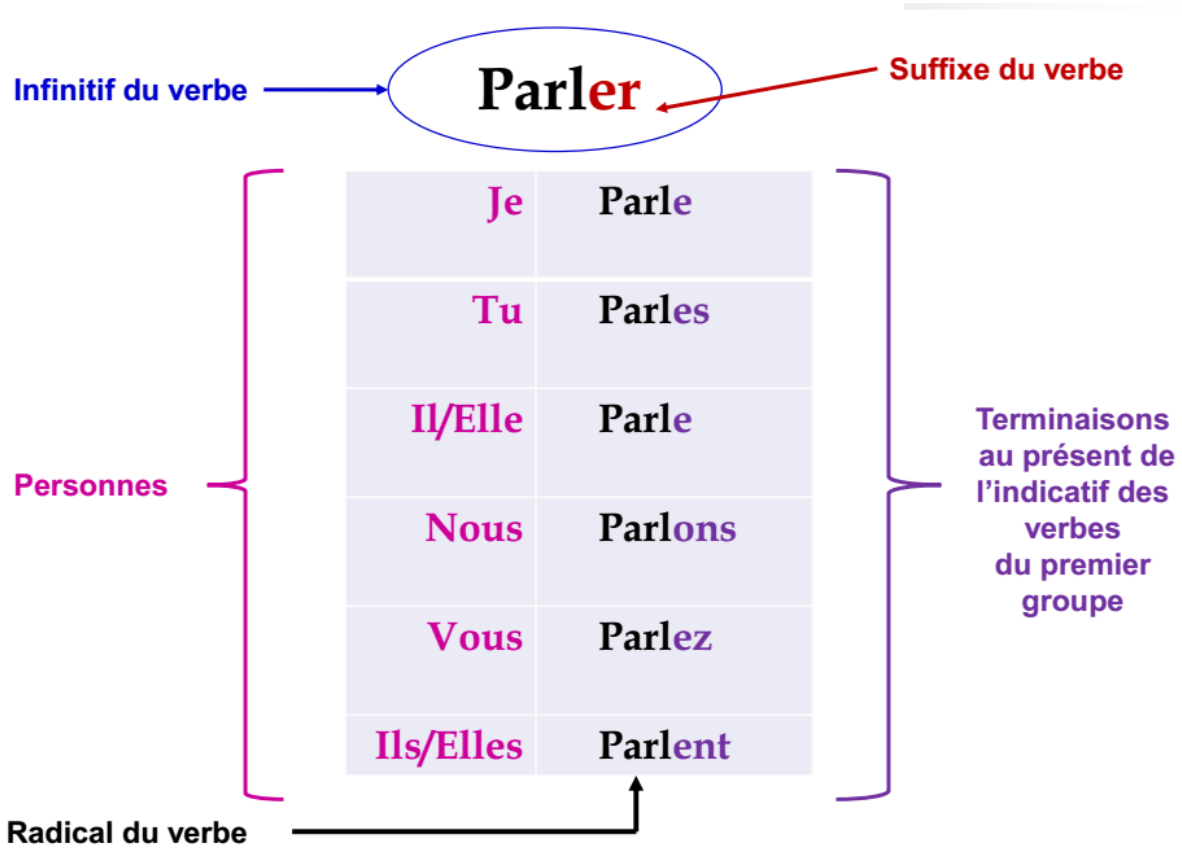

FIGURE 3.12 – Conjugaison du verbe Parler au présent de l'indicatif

Les connaissances sur la méthode que nous venons de présenter (connaissances de

classification, connaissances de reformulation et connaissances de résolution) sont les différentes connaissances qui interviennent lors de la résolution d'un problème par le résolveur de AMBRE. Elles ne permettent cependant pas de fournir de l'aide à l'apprenant, ni de diagnostiquer ses réponses, ni de lui fournir des explications sur ses erreurs. Pour assurer ces fonctionnalités, le système a besoin de connaissances complémentaires que nous présentons dans la section 3.3.

3.3 Comment définir les connaissances permettant d'accompagner l'apprenant ?

Pour que le système puisse assister l'apprenant dans sa résolution de problème et ainsi favoriser l'apprentissage, l'auteur doit définir des connaissances qui permettent au système d'une part de répondre à une demande d'aide de l'apprenant, et d'autre part de diagnostiquer ses réponses pour chaque étape de la résolution et de l'aider à réviser une réponse erronée en lui proposant des explications pertinentes. Dans AMBRE, ces connaissances sont de trois types : des connaissances d'aide, des connaissances de diagnostic et des connaissances d'explications sur les erreurs (cf. **Figure 3.13**).

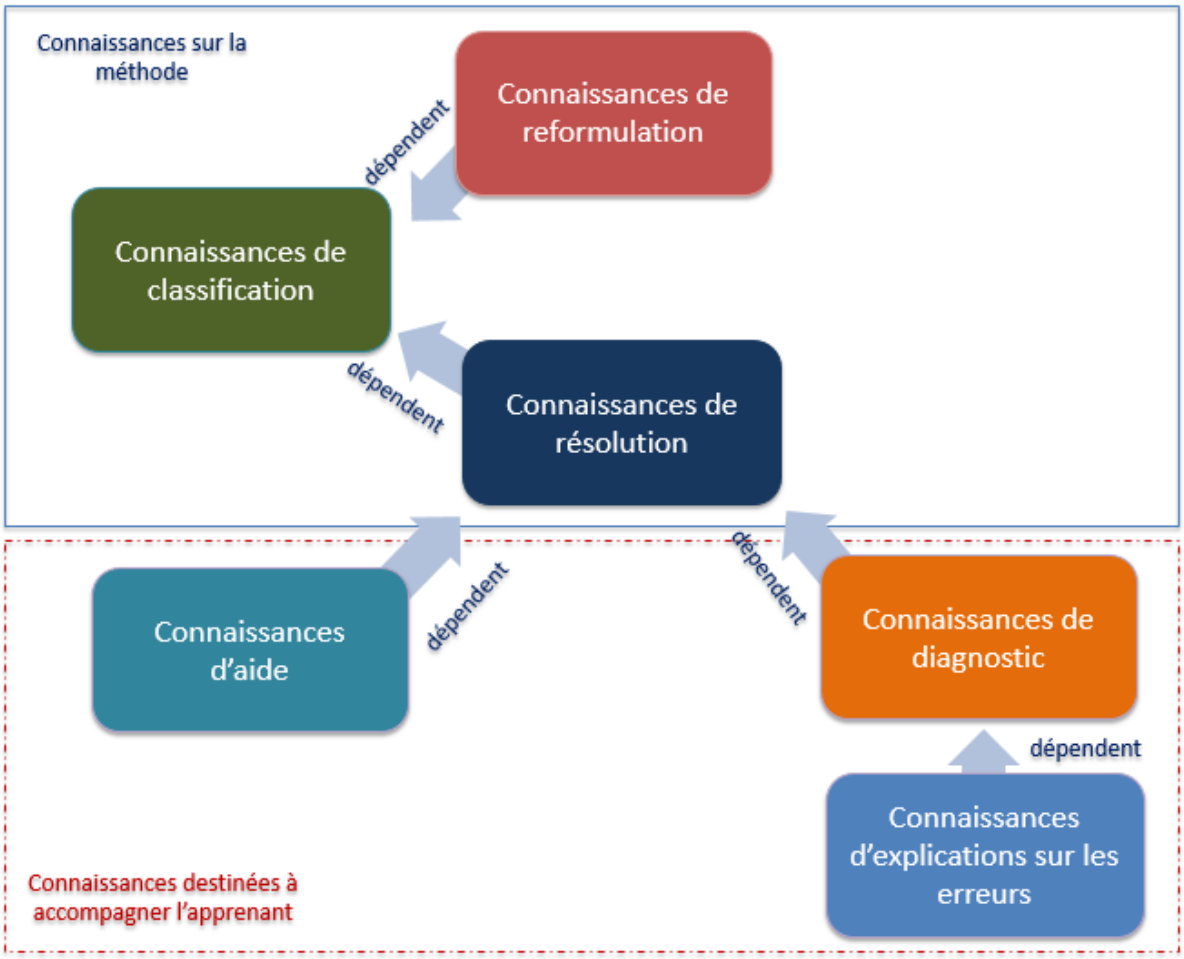

FIGURE 3.13 – Lien entre connaissances sur la méthode et celle destinées à accompagner l'apprenant

Afin de permettre à un auteur de pouvoir expliciter ces connaissances, nous avons proposé des méta-modèles pour chacun de ces types de connaissances.

3.3.1 Méta-modèle des connaissances d'aide

Pour que l'EIAH puisse accompagner l'apprenant de manière efficace durant son apprentissage, il doit pouvoir lui proposer des aides de différentes natures en fonction de l'étape de résolution en cours. Dans AMBRE, l'apprenant peut demander de l'aide à tout moment de la résolution. Lorsque le système reçoit une demande d'aide de la part de l'apprenant, il lui propose deux types d'aide en fonction de l'état d'avancement en

utilisant soit :

— Des **messages** : ces messages d'aide sont souvent des phrases permettant de rappeler à l'apprenant ce qu'il doit faire. Ils sont généralement de deux niveaux : lorsque le premier ne suffit pas à l'apprenant pour comprendre, ce dernier peut passer au second. Par exemple dans l'EIAH AMBRE-add, pour demander de l'aide l'apprenant peut utiliser l'icône en forme de bouée de sauvetage (cf. **Figure 3.14**).

— Les **éléments de l'interface** : dans certains cas des éléments de l'interface sont utilisés afin de rappeler à l'apprenant ce qu'il doit faire. Pour cela, l'EIAH peut renvoyer l'apprenant vers des pages de rappel ou vers des outils favorisant la comparaison entre problèmes-type et problèmes à résoudre à travers la coloration des éléments de l'énoncé et de la reformulation ayant la même fonction dans les deux problèmes (cf. **Figure 3.15**).

FIGURE 3.14 – Exemple de fenêtre d'aide à deux niveaux dans l'étape d'adaptation dans l'EIAH AMBRE-add. le deuxième niveau active le coloriage facilitant la mise en correspondance du problème-type et du problème à résoudre

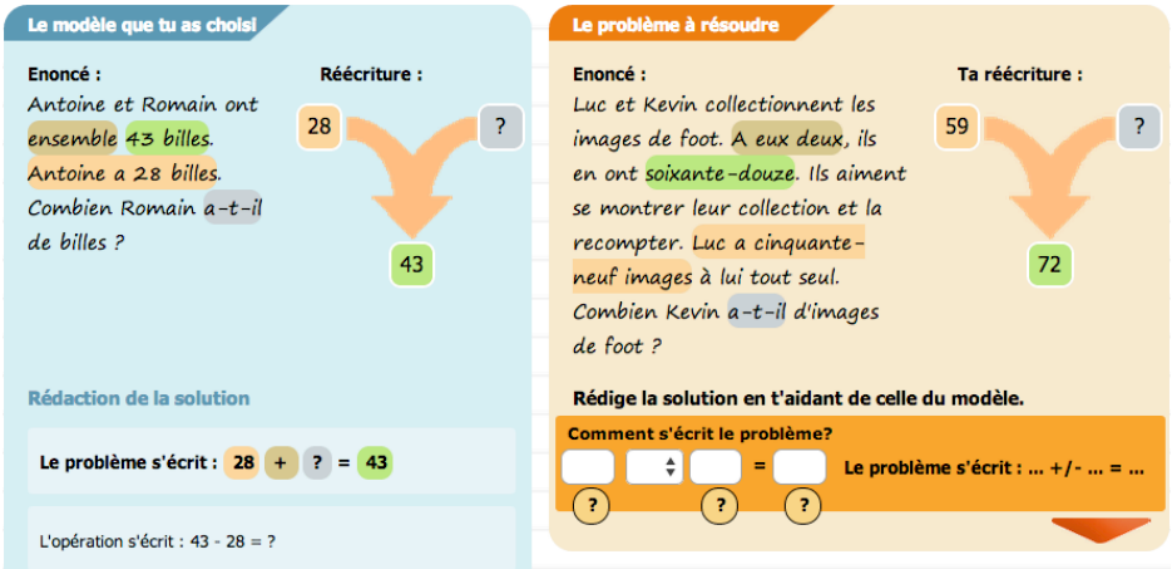

FIGURE 3.15 – Aide à l'analogie à l'étape d'adaptation dnas AMBRE-add

La **Figure 3.16** détaille le méta-modèle que nous avons proposé pour décrire ces connaissances d'aide.

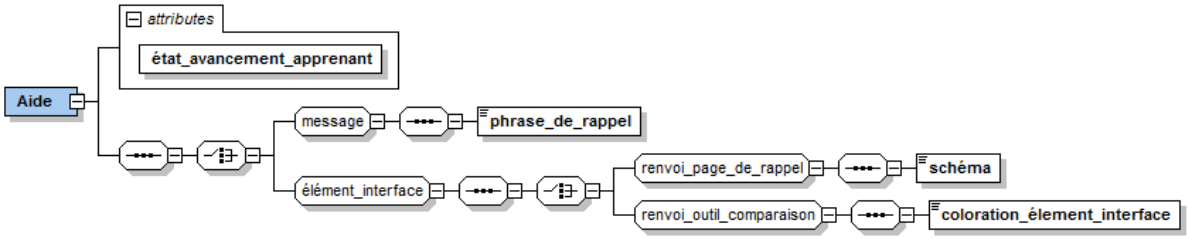

FIGURE 3.16 – Méta-modèle des connaissances d'aide

3.3.2 Méta-modèle des connaissances de diagnostic

Les connaissances de diagnostic permettent la comparaison entre les réponses de l'apprenant et celles fournies par le système afin de pouvoir déterminer si une réponse est correcte ou non. Dans AMBRE-add par exemple, le diagnostic peut être effectué automatiquement par le système à chaque étape ou à la fin d'une sous-étape du cycle

AMBRE, suivant un paramétrage défini par l'enseignant. Il peut également être demandé par l'apprenant à tout moment en cliquant sur l'icône représentant un feu tricolore (cf. **Figure 3.15**) en haut à droite).

Pour définir les connaissances de diagnostic, des points de diagnostic doivent être définis à chaque étape de résolution (cf. **Figure 3.17**). Définir un point de diagnostic revient à préciser l'attribut du problème concerné et à définir un ensemble de fonctions permettant de conclure sur le résultat de la comparaison entre la réponse de l'apprenant et celle du résolveur. Trois cas sont possibles : la réponse de l'apprenant peut être conforme à celle du résolveur, elle peut correspondre à une réponse fausse prévue/non prévue ou à une réponse correcte équivalente à la réponse du résolveur. Dans le cas où la réponse de l'apprenant est fausse, un message d'erreur à afficher à lui doit être prévu.

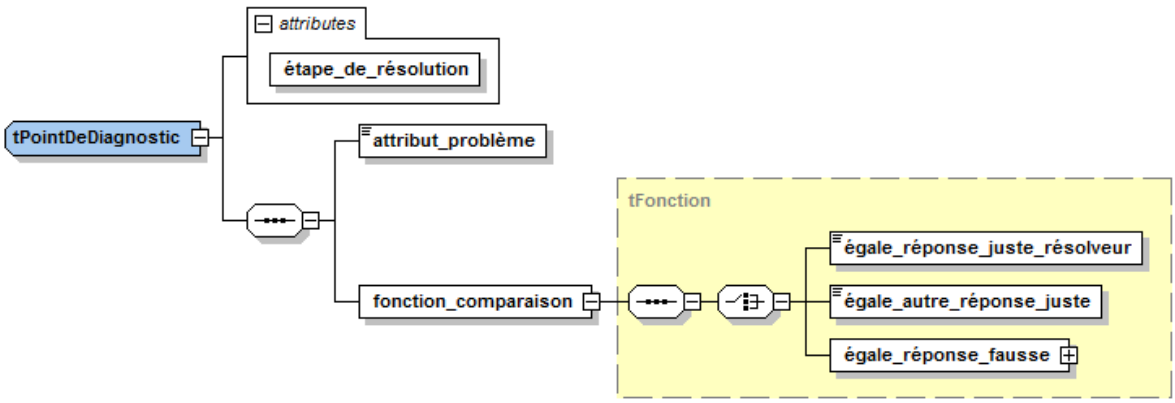

FIGURE 3.17 – Aide à l'analogie à l'étape d'adaptation dans AMBRE-add

La **Figure 3.18** est tirée de l'EIAH AMBRE-add. Dans cet exemple, le diagnostic est réalisé automatiquement par le système et le résultat matérialisé à travers la coloration de certains éléments de l'interface (couleur verte lorsque la réponse est correcte, et rouge lorsqu'elle est fausse).

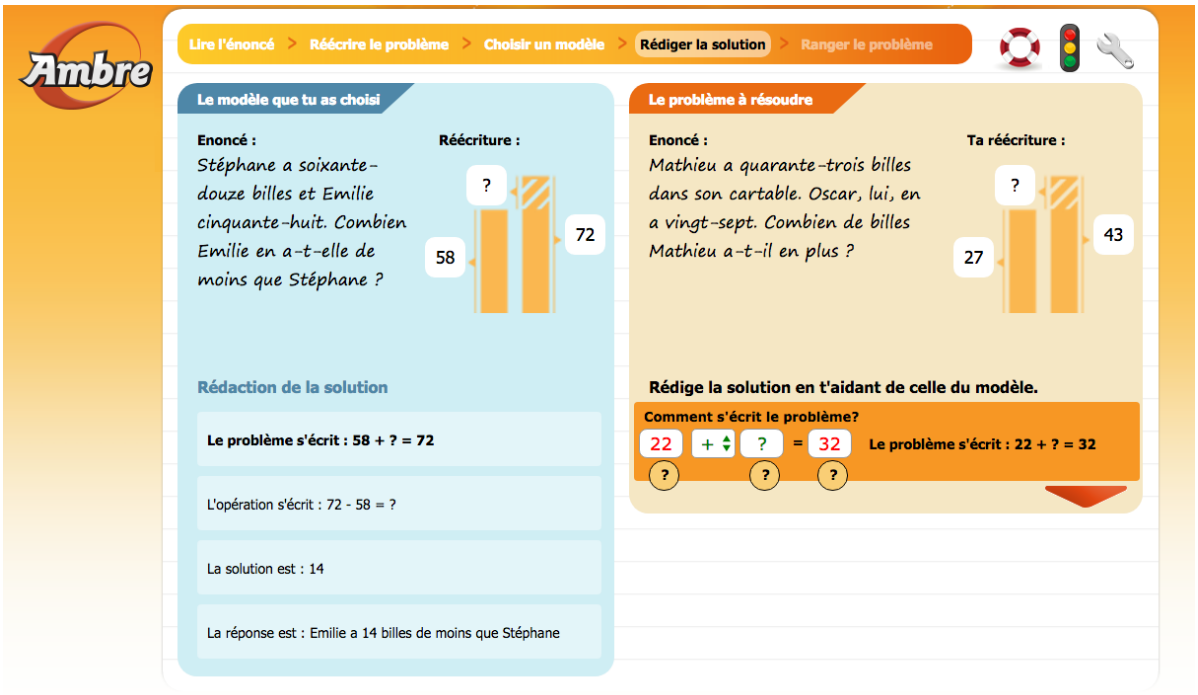

FIGURE 3.18 – Exemple de diagnostic réalisé par le système

3.3.3 Connaissances d'explications sur les erreurs

Les connaissances d'explications font suite au diagnostic. Elles permettent de donner à l'apprenant des explications pertinentes en cas d'erreur. Il s'agit alors de définir un ensemble de réponses fausses attendues et d'y associer des messages explicatifs. La **Figure 3.19** représente le méta-modèle proposé pour décrire ces messages d'explications.

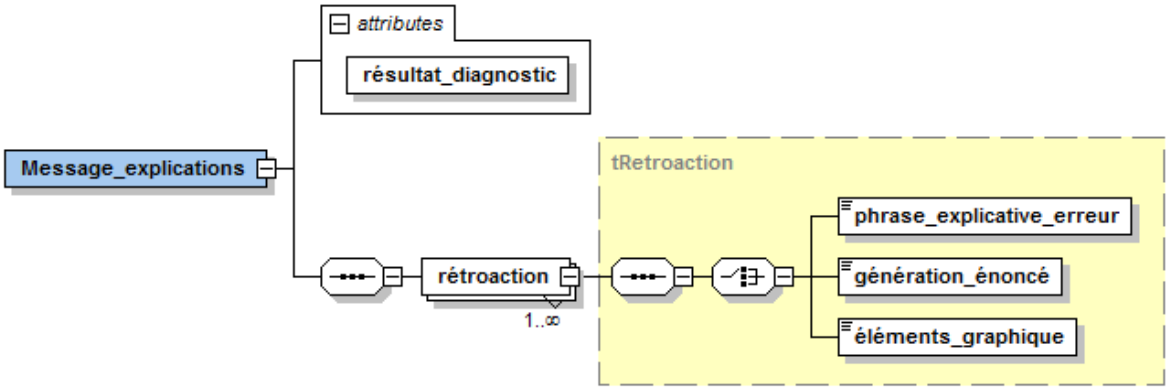

FIGURE 3.19 – Méta-modèle permettant de décrire un message d'explications

Ces messages représentent les différentes rétroactions à envoyer à l'apprenant. Elles peuvent se présenter sous plusieurs formes :

— Une **phrase explicative de l'erreur** : par exemple lors de la rédaction de la phrase de réponse dans AMBRE-add, il est fréquent que les apprenants écrivent une phrase qui est dans l'énoncé. Dans ce cas, le système explique « Ce que tu as écrit est vrai, mais ce n'est pas la réponse à la question ».

— Un **énoncé généré par le système** correspondant à la réponse de l'apprenant. Cette génération d'énoncé peut souvent lui permettre de comprendre en quoi sa réponse est erronée. Cette fonctionnalité de génération d'énoncé est par exemple utilisée dans AMBRE-add pour expliquer à l'apprenant pourquoi sa reformulation à l'aide d'un schéma n'est pas correcte : le système génère un énoncé de problème correspondant au schéma donné par l'apprenant (cf. Figure 14).

— Des **éléments graphiques** : le recours aux éléments graphiques peut se faire lorsque le système ne sait pas interpréter la réponse de l'apprenant ou lorsque ce format est plus facile à comprendre que des explications textuelles complexes. Par exemple, le système colorie en vert des éléments de la réponse qui sont corrects et en rouge ceux qui sont erronés (cf. **Figure 3.20**). Lorsque la réponse de l'apprenant est incorrecte, comme sur la Figure 12 par exemple, l'apprenant peut cliquer sur le point d'interrogation à côté de chaque champ encadré en rouge pour afficher le message d'erreur.

FIGURE 3.20 – Exemple de message d'explications généré à partir de la réponse de l'apprenant

3.4 Conclusion

Dans ce chapitre, nous avons présenté les différentes connaissances identifiées comme étant nécessaire à expliciter lors de la conception d'un EIAH AMBRE. Ces connaissances peuvent être classées en deux grandes catégories : les connaissances sur la méthode à enseigner permettant de résoudre des problèmes, et celles destinées à accompagner l'apprenant durant son apprentissage en lui fournissant aide, diagnostic de ses réponses et explications pertinentes sur ses erreurs. Pour formaliser ces différentes connaissances, nous avons proposé des méta-modèles décrits par des schémas XML. Ces méta-modèles permettent de décrire la forme des connaissances à acquérir, cherchent être indépendants du domaine.

L'objectif de ce travail était surtout d'identifier les types de connaissances à acquérir et de les formaliser. Même si les méta-modèles proposés ont été proposés en s'appuyant sur les connaissances des EIAH AMBRE, leur utilisation reste ouverte à d'autres types d'applications. Par exemple, le méta-modèle portant sur la description des problèmes

peut être utilisé dans des domaines nécessitant de modéliser des problèmes.

Dans [84] par exemple les auteurs présentent un outil auteur web permettant d'assister l'enseignant dans la création d'exercices portant sur les équations algébriques. Cet outil repose sur un résolveur de problèmes qui utilise les informations du problème et accompagne l'apprenant durant sa résolution en lui fournissant un feedback. Les problèmes sont décrits en termes de variables, d'unités de mesures, de formules. Les méta-modèles du vocabulaire et des problèmes que nous avons proposés peuvent, par exemple, être utilisés dans le cadre de cet outil auteur web, afin de mieux assister l'auteur dans la définition des connaissances. Le méta-modèle permettant de décrire les fonctions peut être utilisé pour cet outil afin de faciliter la description des formules. De même, les méta-modèles permettant d'accompagner l'apprenant peuvent être utiles ici pour donner des explications pertinentes aux apprenants sur leurs erreurs.

Dans Andes [76, 83], un EIAH destiné à l'enseignement de la Physique, l'auteur doit définir formellement le problème, de sorte que le système puisse le résoudre. Utiliser un méta-modèle permettant de décrire les problèmes peut être très utile pour faciliter l'élicitation. Les méta-modèles des connaissances sur la méthode peuvent quant à eux être utilisés dans le domaine de l'aide à prise de décision, et plus particulièrement pour la conception d'outils de prévision. Par exemple, le méta-modèle des connaissances de classification peut être utilisé pour organiser les différentes informations, le méta-modèle des connaissances pour décrire les règles, et les techniques pour représenter les différentes prévisions en fonction des cas.

Nous nous sommes appuyées sur ces méta-modèles pour mettre en place un processus d'acquisition des connaissances décrit dans le chapitre 4.

Chapitre 4

Un processus interactif d'acquisition des connaissances pour des EIAH destinés à enseigner des méthodes

Ce chapitre présente le processus interactif que nous avons proposé pour acquérir les connaissances nécessaires à conception d'un EIAH AMBRE. Il se consacre pour l'essentiel à l'élicitation des connaissances portant sur la méthode à enseigner, ainsi que l'assistance fournie à l'auteur durant le processus. Cependant, même si l'élicitation des connaissances destinées à guider l'apprenant n'est pas présentée ici, nous montrons comment elle est prise en compte dans le processus. Ce chapitre se termine par le feedback proposé à l'auteur à la fin de l'élicitation.

Sommaire

4.1 Étapes du processus d'acquisition des connaissances 97
4.1.1 Définition du vocabulaire pour le domaine 98
4.1.2 Description des problèmes à résoudre 99
4.1.3 Élicitation des connaissances sur la méthode 100
4.1.4 Conception de l'interface de l'EIAH 104
4.1.5 Élicitation des connaissances destinées à accompagner l'apprenant 104
4.2 Feedback après l'élicitation des connaissances 104
4.3 Synthèse 106

L'acquisition des connaissances pour la conception d'EIAH est l'un des principaux obstacles à leur développement. Cette difficulté est liée au fait qu'elle repose en général sur des experts humains mais aussi sur un processus d'explicitation de leurs connaissances. Outre le fait qu'il est difficile d'amener l'expert à expliciter ses connaissances, cette pratique s'avère également longue et ardue. En effet, elle nécessite de recommencer l'explicitation des connaissances pour chaque nouveau domaine. La mise en place d'un processus interactif pourrait favoriser une acquisition plus aisée de ces connaissances. C'est dans ce sens que nous proposons un processus pour permettre à un auteur d'éliciter sans programmer les connaissances nécessaires à la conception d'un EIAH AMBRE.

4.1 Étapes du processus d'acquisition des connaissances

La **Figure 4.1** résume l'approche générale que nous avons adoptée pour la mise en place d'un processus d'élicitation des connaissances nécessaires à la conception d'un EIAH AMBRE. Ce processus permet à un auteur, expert des connaissances qu'il souhaite enseigner, de pouvoir expliciter les connaissances nécessaires à l'EIAH sans aucune programmation. Ce processus s'appuie sur les méta-modèles indépendants du domaine et proposés dans le Chapitre 3 : un méta-modèle pour décrire le vocabulaire, un méta-modèle pour décrire les problèmes, des méta-modèles pour les connaissances portant sur la méthode à enseigner, et des méta-modèles pour les connaissances destinées à accompagner l'apprenant. Pour rappel, ces méta-modèles permettent de décrire les différents types des connaissances que l'auteur doit définir pour construire des modèles de connaissances spécifiques à un domaine. Ces méta-modèles contraignent ainsi les différentes étapes du processus d'acquisition des connaissances.

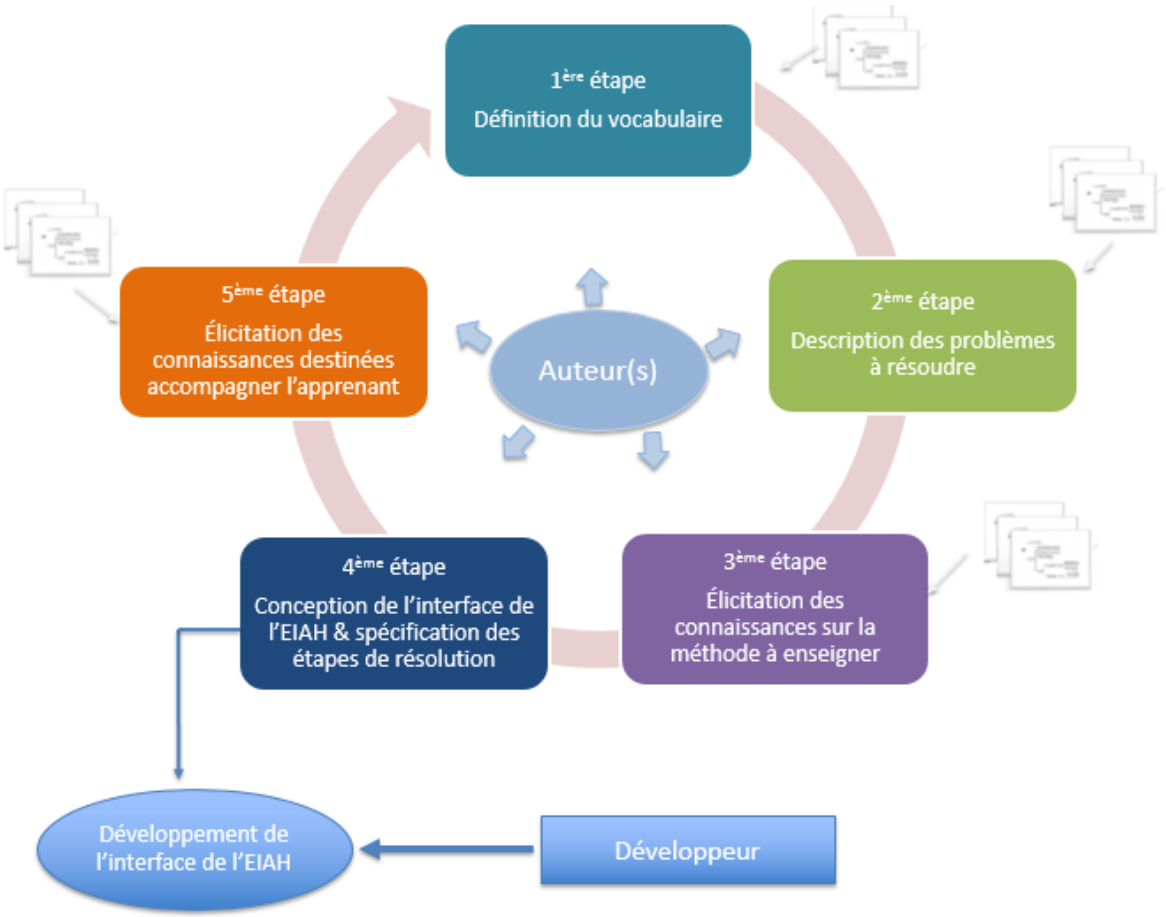

FIGURE 4.1 – Approche générale du processus d'élicitation

4.1.1 Définition du vocabulaire pour le domaine

L'objectif ici est de permettre à l'auteur de pouvoir définir les connaissances nécessaires à la conception d'un EIAH destiné à enseigner des méthodes dans un domaine qui l'intéresse. Les EIAH de ce type reposent sur une classification des problèmes à résoudre et des techniques de résolution. Pour ce faire, l'auteur doit d'abord commencer par bien cibler le domaine d'application en respectant ce critère d'existence d'une classification. Une fois le domaine d'application choisi, et avant d'entamer l'élicitation des connaissances, il est important de bien réfléchir au type de problèmes à faire résoudre à l'apprenant (et donc au système). Ces éléments sont en effet nécessaires pour que l'auteur puisse définir le vocabulaire du domaine choisi, qui joue un rôle central dans le processus d'élicitation des connaissances que nous proposons.

Pour définir le vocabulaire, l'auteur dispose d'une interface lui permettant de créer, modifier ou supprimer des types d'objets. Pour chaque type d'objet créé, il doit pouvoir ajouter des caractéristiques (nom et type des caractéristiques), les modifier et les supprimer au besoin.

Pour faciliter la définition du vocabulaire, le système doit proposer une grande flexibilité à l'auteur (sans trop le contraindre) lors de l'élicitation des connaissances. Nous entendons par flexibilité la capacité du système à offrir des modes d'interaction multiples pour répondre aux besoins, préférences et expériences de l'utilisateur. Ainsi, le système doit, dans la mesure du possible, laisser le choix à l'auteur quant à l'ordre de définition des connaissances. Par exemple, après avoir réfléchi sur le champ d'application de la méthode à enseigner et donc de l'ensemble des problèmes qui seront proposé à l'apprenant, l'auteur doit pouvoir choisir de définir l'ensemble des objets dont il aura besoin pour décrire tous les problèmes. Il doit également pouvoir choisir de définir les types d'objets au fur et à mesure qu'il en a besoin pour décrire chaque problème.

Une fois le vocabulaire défini, l'auteur doit l'utiliser d'une part pour décrire les différents problèmes à résoudre, et d'autre part pour éliciter les connaissances sur la méthode.

4.1.2 Description des problèmes à résoudre

Dans cette étape, l'auteur utilise le vocabulaire précédemment défini pour décrire les problèmes à résoudre. Pour ce faire, le système doit lui proposer une interface basée sur le méta-modèle permettant de décrire des problèmes (cf. Chapitre 3). Ainsi, pour chaque problème, il doit pouvoir :

— Définir l'**énoncé du problème** qui va être présenté à l'apprenant lors de la résolution d'un problème. Pour cela, il doit disposer d'un champ texte à remplir. L'énoncé décrit dans ce champ sera présenté tel quel à l'apprenant.
— Définir les **objets du problème**. L'auteur doit pouvoir choisir, par les types d'objets précédemment définis dans son vocabulaire, ceux qui sont nécessaires pour décrire son problème, puis les instancier en mettant des valeurs aux identifiants, ainsi qu'aux caractéristiques, tout en respectant le type de ces dernières.
— Spécifier la **question à résoudre** pour ce problème. Le système doit également

permettre à l'auteur de préciser la caractéristique qui constitue l'inconnue à calculer pour le problème concerné.

Pour faciliter la description des problèmes, proposer des fonctionnalités de guidage peut être utile à l'auteur. Nous entendons par guidage l'ensemble des moyens proposés à l'auteur afin de l'orienter et de le conseiller durant tout le processus d'élicitation. Par exemple, proposer des fonctionnalités d'auto-complétions peut être très utile parce qu'elles permettent à l'auteur, au fur et à mesure de la définition des connaissances, de savoir ce qui a déjà été défini et de le réutiliser au besoin. Par exemple, connaître les objets déjà définis pour pouvoir les réutiliser permet une élicitation plus facile des connaissances, avec moins de risque d'erreurs.

Le fait que les modèles de connaissances soient fondés sur des méta-modèles permet aussi de pouvoir proposer des fonctionnalités de contrôle de ces connaissances. Celles-ci vont permettre d'une part de vérifier que toutes les connaissances nécessaires ont été explicitées, et d'autre part que ces connaissances sont bien conformes aux différents méta-modèles. Par exemple, le système doit permettre à l'auteur vérifier qu'il a défini toutes les connaissances nécessaires pour décrire un problème. Il s'agit de mettre en place une fonctionnalité permettant de vérifier d'une part que pour chaque objet, les types des caractéristiques ont bien été respectés. Par exemple lorsqu'une caractéristique est de type « nombre entier », le système va contraindre le champ de sorte que l'auteur ne puisse saisir que des nombres. D'autre part, il s'agira de vérifier aussi que pour chaque problème, l'auteur a bien indiqué la caractéristique sur laquelle porte l'inconnue à calculer.

4.1.3 Élicitation des connaissances sur la méthode

Pour les connaissances sur la méthode, l'auteur doit définir trois types de connaissances en s'appuyant sur les méta-modèles correspondants (cf. chapitre 3) : les connaissances de classification, les connaissances de reformulation et les connaissances de résolution.

4.1.3.1 Élicitation des connaissances de classification

Pour définir les connaissances de classification, il revient à l'auteur de construire un arbre de classification des problèmes à résoudre pour le domaine choisi. Pour cela, il doit disposer d'une interface lui permettant de construire cet arbre au fur et à mesure. Pour la construction de cet arbre, l'auteur doit avoir deux possibilités : définir toutes les classes de problèmes d'abord, puis les organiser ensuite en une hiérarchie ou alors définir les classes une par une en les hiérarchisant au fur et à mesure. Pour chaque classe créée, il doit pouvoir définir un nom, un attribut discriminant permettant de discriminer ses sous-classes. Des attributs du problème peuvent également être définis s'ils ont un sens pour les problèmes de cette classe. L'auteur doit également pouvoir indiquer le statut de chaque classe : les classes suffisamment spécifiques pour qu'une technique de résolution puisse être appliquée doivent être opérationnelles et les autres non opérationnelles.

L'auteur doit, à tout moment de la construction de son arbre, pouvoir vérifier si les connaissances qu'ils a définies jusque-là sont conformes au méta-modèle des connaissances de classification (cf. Chapitre 3). Pour cela, le système doit pouvoir vérifier, par exemple, que l'auteur a défini des noms pour toutes les classes, qu'il a défini des attributs discriminants pour toutes les classes non opérationnelles, que toutes les classes non opérationnelles ont au moins une sous-classe. En effet, afin de classer un problème lors de la résolution, le résolveur essaie d'aller le plus loin possible dans la classification : au moins jusqu'à une classe opérationnelle, du fait que les classes non opérationnelles ne possèdent pas de technique de résolution. C'est pourquoi d'ailleurs les feuilles de l'arbre doivent être des classes opérationnelles pour que la classification soit utile pour résoudre des problèmes.

Le système doit également assister l'auteur durant la construction de l'arbre en lui proposant des fonctionnalités de guidage. Par exemple : colorer toutes les classes que l'auteur aura définies comme étant opérationnelles. Cela permet de lui rappeler qu'il doit associer des techniques de résolution à ces classes, mais également d'avoir un aperçu des classes qui sont opérationnelles et de celles qui ne le sont pas. Lors de la définition des attributs discriminants, le système doit proposer des fonctionnalités d'auto-complétion pour permettre à l'auteur d'avoir un aperçu des attributs qu'il a déjà définis et d'en utiliser un s'il le désire. Lorsque l'auteur définit le même attribut pour deux sous-classes, le système doit lui proposer soit de définir cet attribut comme un attribut du problème au niveau de la classe parent la plus proche, soit de renommer l'un des deux attributs.

En choisissant de définir l'attribut comme un attribut du problème dans la classe parent, cela lui permettra par exemple de définir les règles associées une seule fois et de pouvoir les utiliser pour les deux classes.

Une fois que l'auteur a défini une partie de l'arbre de classification, il peut entamer la définition des règles. Celles-ci vont permettre de calculer les valeurs des attributs discriminants, et ainsi de positionner les problèmes dans la classification. L'ensemble des règles constitue les connaissances de reformulation.

4.1.3.2 Élicitation des connaissances de reformulation

Pour chaque attribut (discriminant ou non) défini dans l'arbre de classification, l'auteur doit associer des règles permettant de conclure sur sa/ses valeur(s). L'auteur doit donc disposer d'une interface permettant de les définir. Pour chaque règle, il doit pouvoir définir un nom, un ensemble de prémisses portant sur des éléments de l'énoncé du problème et un ensemble de conclusions permettant de conclure sur la/les valeur(s) des attributs concernés.

Comme pour les connaissances de classification, le système doit offrir à l'auteur de la flexibilité lors de la définition des connaissances de reformulation. Par exemple, l'auteur doit pouvoir définir les règles au fur et à mesure de la construction de son arbre de classification. Ainsi pour chaque classe définie, une fois les attributs définis, l'auteur doit pouvoir spécifier le(s) règles(s) qui permettront de déterminer sa/ses valeur(s). L'auteur doit également avoir le choix de définir son arbre entièrement, et ensuite procéder à la définition des règles. Le système doit permettre également de modifier une règle, la supprimer ou la dupliquer à partir d'une autre en complétant/modifiant certaines valeurs.

Des fonctionnalités de contrôle des connaissances de reformulation doivent également être proposées. Celles-ci permettront à l'auteur au fur et à mesure de vérifier si les connaissances qu'il a explicitées sont conformes au méta-modèle des connaissances de reformulation (cf. chapitre 3). Il s'agit notamment de vérifier qu'il a associé au moins une règle à tous les attributs (discriminants ou non) de l'arbre de classification, de vérifier que chaque règle comporte une partie conclusion donnant des informations permettant de conclure sur la/les valeurs(s) des attributs correspondants. Ces connaissances de re-

formulation sont très importantes parce qu'elles vont permettre au système de passer d'un modèle initial du problème, à un nouveau modèle plus adapté à l'application d'une technique de résolution. L'ensemble des techniques de résolution constituent les connaissances de résolution.

4.1.3.3 Élicitation des connaissances de résolution

Pour définir les connaissances de résolution, l'auteur doit associer à chaque classe opérationnelle de l'arbre de classification, une technique de résolution. Pour ce faire, il doit disposer d'une interface lui permettant pour chaque classe, de définir sa technique en décrivant le plan de résolution des problèmes de cette classe. Il s'agir pour l'auteur de décrire l'ensemble des actions que le système doit réaliser pour résoudre un problème donné. Dans certains domaines, il peut également être amené à définir une fonction de calcul permettant de déterminer les éléments de solution à partir des données fournies dans l'énoncé des problèmes. Pour l'assister dans cette élicitation, le système doit proposer des fonctionnalités d'auto-complétion sur les objets des problèmes, ainsi qu'un contrôle sur les connaissances élicitées.

De manière similaire aux règles, l'auteur doit pouvoir choisir de définir les techniques de résolution au fur et à mesure de la définition des classes de l'arbre. Dans ce cas, il peut définir une classe opérationnelle et sa technique de résolution de manière simultanée. Il doit également pouvoir définir l'ensemble des classes, puis l'ensemble des techniques et enfin associer les techniques aux classes correspondantes. Le système doit également permettre de modifier une technique, la supprimer ou la dupliquer en complétant/modifiant certaines valeurs.

Une fois ces trois types de connaissances définies, le système est apte à résoudre les problèmes du domaine. Toutefois, afin de permettre au système de pouvoir fournir de l'aide à l'apprenant, de diagnostiquer ses réponses et de lui fournir des explications pertinentes sur ses erreurs, l'auteur doit définir des connaissances complémentaires : les connaissances destinées à guider l'apprenant (cf. section 4.5).

4.1.4 Conception de l'interface de l'EIAH

Avant d'entamer l'élicitation des connaissances destinées à guider l'apprenant (étape 5), l'auteur doit concevoir l'interface de l'EIAH, et particulièrement les différentes tâches à effectuer par l'apprenant pour résoudre les problèmes. En effet, les connaissances destinées à guider l'apprenant sont fortement dépendantes des connaissances portant sur la méthode (étape 4.1.3), mais aussi des différentes étapes de résolution proposées dans l ?EIAH. Le développement de cette interface doit bien sûr être faite par un informaticien.

4.1.5 Élicitation des connaissances destinées à accompagner l'apprenant

En s'appuyant sur les connaissances sur la méthode (étape 3), les différentes tâches à effectuer (étape 4.1.4), ainsi que les méta-modèles destinés à accompagner l'apprenant, l'auteur définit les connaissances nécessaires permettant de fournir à l'apprenant de l'aide, un diagnostic de ses réponses et des explications pertinentes sur ses erreurs.

Une fois le processus d'élicitation terminé, le système doit permettre à l'auteur d'avoir un feedback sur les connaissances élicitées.

4.2 Feedback après l'élicitation des connaissances

Permettre à l'auteur d'avoir un feedback sur le processus d'élicitation des connaissances est très important. Le feedback présenté ici est différent de la fonctionnalité de contrôle des connaissances présentée dans les sections précédentes. En effet, dans AMBRE-KB, pour chaque type de connaissances à éliciter (vocabulaires du domaine, problèmes à résoudre, connaissances de classification, connaissances de reformulation, connaissances de résolution), l'auteur dispose d'une fonctionnalité lui permettant de vérifier qu'il a explicité toutes les connaissances nécessaires, et le cas échéant d'avoir un message lui permettant de comprendre son erreur et de la corriger. Par exemple, lorsque l'auteur définit son arbre de classification, il peut utiliser cette fonctionnalité pour véri-

fier qu'il a associé des noms à toutes les classes, des attributs discriminants à toutes les classes non opérationnelles, que les feuilles de l'arbre sont toutes des classes opérationnelles, etc. Ces fonctionnalités de contrôle de connaissances sont proposées au moment de l'élicitation des connaissances, et permettent de réduire les risques d'erreurs.

La fonctionnalité de feedback quant à elle est proposée après l'élicitation de toutes les connaissances nécessaires à l'EIAH pour résoudre les problèmes du domaine. Cette fonctionnalité permet à l'auteur de tester la résolution des problèmes qu'il a définis. Pour cela, il doit d'abord, en utilisant AMBRE-KB, générer les modèles correspondants aux connaissances qu'il a explicité, et qui seront utilisés par le système pour résoudre les problèmes du domaine. Ensuite, pour chaque problème défini, il dispose d'une fonctionnalité lui permettant de tester la résolution. Ainsi, pour chaque problème, le système doit envoyer un certain nombre d'informations concernant sa résolution. Deux cas sont possibles : soit le système parvient à résoudre le problème, soit le système ne parvient pas à résoudre le problème. Dans les deux cas, il doit envoyer un certains nombre d'informations concernant la résolution du problème.

Lorsque le système parvient à résoudre un problème, il doit, par exemple, afficher la solution du celui-ci. Dans ce cas, cela voudrait dire que l'auteur a bien défini toutes les connaissances nécessaires pour résoudre ce problème. Il peut alors consulter des informations concernant les différentes étapes de résolution telles que la classe du problème, la liste des classes parcourues, les attributs calculés, et pour chacun d'eux la liste des règles essayées, la règle exécutée et la technique de résolution appliquée. Les règles essayées correspondent aux règles qui une partie des prémisses sont satisfaites, mais l'ensemble des prémisses ne sont pas respectées pour permettre à la règle d'être déclenchée.

Il peut arriver aussi que le système n'arrive pas à résoudre un problème soit parce qu'il n'a pas assez de connaissances, soit parce qu'il y a des erreurs dans les connaissances explicitées. Par exemple, la résolution d'un problème peut échouer lorsque le système n'a pas pu trouver de règles pour conclure sur la valeur d'un attribut, ou lorsque les règles essayées ne lui permettent pas de conclure sur la valeur de cet attribut. Dans ce cas, le système doit afficher un message pour indiquer que la classification a été bloquée en renseignant la classe à laquelle elle est bloquée. Il doit également afficher la liste des règles essayées. La résolution peut également échouer lors de l'application d'une technique de résolution. Dans ce cas, le système doit afficher un message pour le spécifier à l'auteur. Pour cela, il doit afficher la classe du problème pour permettre à l'auteur de vérifier la

technique associée à cette classe et de la corriger ou la compléter.

4.3 Synthèse

Dans ce chapitre, nous avons présenté le processus interactif d'élicitation des connaissances que nous avons proposé. Ce processus s'appuie sur des méta-modèles indépendants du domaine permettant d'une part de décrire la forme des connaissances à acquérir, et d'autre part de contraindre le processus d'élicitation associé.

L'avantage de cette approche est de permettre à des auteurs de définir ensuite des modèles, qui du fait de leur conformité aux méta-modèles seront exploitables par l'EIAH. Par exemple, dans AMBRE, un moteur de raisonnement (résolveur) fondé sur ces méta-modèles a été mis en place. Celui-ci est capable d'exploiter ainsi les modèles de connaissances qui sont conformes à ces méta-modèles. Pour résoudre un problème, le résolveur de AMBRE utilise le modèle des connaissances de classification (comme un guide) et le modèle des connaissances de reformulation pour trouver la classe du problème et construire un nouveau modèle du problème à résoudre, plus adapté à l'application d'une technique de résolution. Il utilise ensuite le modèle des connaissances de résolution pour appliquer la technique de résolution a associée à cette classe afin de trouver la solution au problème donné. De la même manière, le système utilise les modèles des problèmes pour fournir des informations à l'interface. Par exemple, dans AMBRE-add [63], pour construire les listes déroulantes utilisées par l'apprenant lors de la construction de la phrase de réponse, le système utilise les données du problème : le sujet de la phrase (les prénoms de l'énoncé et des autres noms), les objets (ceux de l'énoncé, par exemple billes, fleurs, autocollants, etc.), le verbe de la phrase, etc. De même, pour fournir des fonctionnalités d'aide, de diagnostic et d'explications sur les erreurs à l'apprenant - toujours dans AMBRE-add - le système utilise dans un premier temps, les modèles des connaissances de diagnostic pour comparer les réponses de l'élève à celles du résolveur. Ensuite, il utilise les modèles des connaissances d'explications pour construire un message d'explications ou des informations que l'interface mettra en œuvre visuellement.

Notre approche est donc cohérente avec celle de T. Nodenot [61], qui préconise que pour soulager les chercheurs en EIAH non informaticiens du poids des développements, il est urgent de leur demander de développer et réutiliser des modèles plutôt que du

code. Selon lui, pour faciliter le dialogue entre ces modèles et les modèles que savent produire les informaticiens, et dans le but de rendre les modèles exploitables par les informaticiens, il est nécessaire d'expliciter les méta-modèles qui permettent d'interpréter les divers modèles proposés. D'un point de vue Ingénierie des Connaissances (IC), nous nous inscrivons dans une démarche d'acquisition guidée par les modèles, puisque les méta-modèles guident le processus d'Ingénierie des Connaissances.

De plus, le fait que la résolution d'un problème soit séparée en différents processus (modélisation, classement, application d'une technique) permet de mieux distinguer les démarches et les types de difficultés propres à chaque processus. Un autre avantage de ce découpage est de permettre une réutilisabilité plus aisée des connaissances contenues dans la classification, et surtout de les faire évoluer plus facilement et avec moins de risques d'erreurs.

Notre processus est mis en œuvre à travers la conception d'un outil auteur d'un outil auteur que nous présentons dans le chapitre suivant.

Troisième partie

MISE EN ŒUVRE ET ÉVALUATIONS

Chapitre 5

AMBRE-KB : un outil auteur pour des EIAH destinés à enseigner des méthodes

Afin de mettre en œuvre les différentes contributions théoriques présentées dans les chapitres précédents, nous avons développé l'outil auteur AMBRE-KB (AMBRE-Knowledge Builder). En particulier, AMBRE-KB met en œuvre le processus d'acquisition des connaissances portant sur la méthode à enseigner (connaissances de classification, connaissances de reformulation et connaissances de résolution) et présenté dans le chapitre 4.

Dans la première section de ce chapitre, nous décrivons l'approche adoptée pour la conception de AMBRE-KB. La deuxième section illustre l'utilisation de AMBRE-KB en montrant, pour un domaine donné, comment éliciter les connaissances qui permettent au système de résoudre des problèmes. Nous présentons également, pour chacun des types de connaissances à acquérir, les fonctionnalités d'assistance proposées à l'auteur pour l'accompagner durant ce processus d'élicitation. Enfin, nous terminons ce chapitre en montrant comment l'auteur peut tester les modèles de connaissances générés par AMBRE-KB.

Sommaire

5.1 Élicitation des connaissances avec AMBRE-KB 113
5.1.1 Définition du vocabulaire 115
5.1.2 Description des problèmes à résoudre 116
5.1.3 Élicitation des connaissances sur la méthode 118
5.2 Génération des modèles de connaissances 126
5.3 Tests des modèles de connaissances 130
5.4 Conclusion 135

L'objectif de AMBRE-KB est de permettre à des auteurs (enseignant, professionnel, pédagogue plus expert) - experts de la méthode qu'ils souhaitent enseigner dans un domaine donné - de pouvoir éliciter, sans programmer, les connaissances nécessaires à la conception d'un EIAH AMBRE pour ce domaine. Pour ce faire, nous avons proposé un processus interactif d'acquisition des connaissances (cf. chapitre 4) mis en œuvre via les interfaces de AMBRE-KB. Ces interfaces sont contraintes par les méta-modèles présentés au chapitre 3, et qui décrivent la forme des connaissances que l'auteur doit définir pour construire des modèles de connaissances spécifiques au domaine qu'il aura choisi.

Pour chaque domaine, l'auteur doit définir les différents modèles de connaissances à l'aide de AMBRE-KB ; le système génère ensuite une version Prolog de ces modèles de connaissances qui constituent les bases de connaissances de l'EIAH (cf. **Figure 5.1**).

FIGURE 5.1 – Approche générale de AMBRE-KB

5.1 Élicitation des connaissances avec AMBRE-KB

Afin de permettre à un auteur de décrire les différents types de connaissances décrits au chapitre 4, nous avons proposé une interface dont le menu horizontal est organisé en onglets. Chaque onglet dispose d'un menu vertical à gauche et d'un espace de travail à droite (cf. **Figure 5.2**).

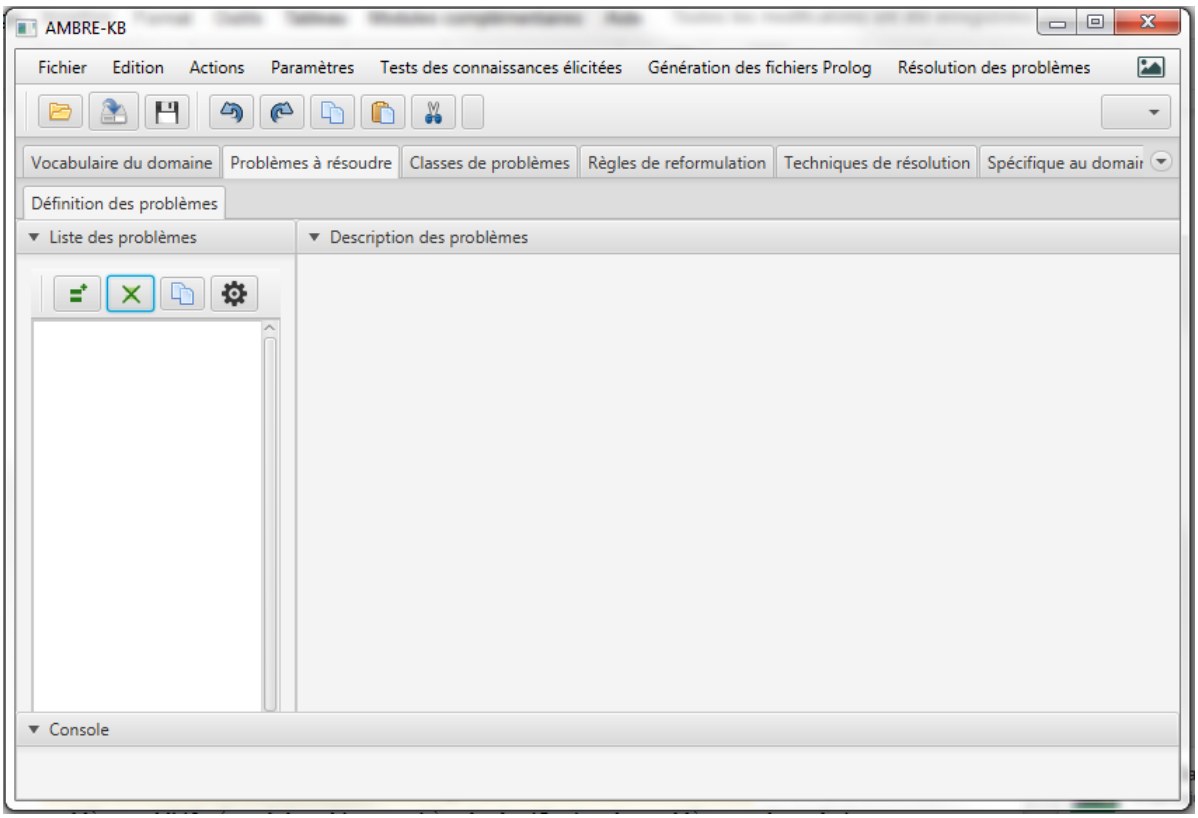

FIGURE 5.2 – Vue d'ensemble de AMBRE-KB

Pour expliquer le fonctionnement de cette interface, nous allons prendre un exemple complet avec le domaine des problèmes additifs, un des domaines les plus largement étudiés en didactique des mathématiques, psychologie cognitive, et linguistique [35]. Un EIAH AMBRE a déjà été conçu pour ce domaine, mais nous avons choisi de tester AMBRE-KB dans ce domaine pour voir comment les bases de connaissances auraient pu être définies avec notre outil auteur. Le domaine des problèmes additifs répond donc au critère de classification des problèmes et des techniques de résolution. Riley, Greeno

et Heller [64] ont proposé une classification des problèmes additifs qui distingue trois catégories de problèmes :

— Les problèmes de type « **réunion** » qui portent sur des situations statiques. Par exemple : « *Paul et pierre ont ensemble 10 billes* » ou encore « *Paul a 6 billes, Pierre en 4* ». La question consiste ici à trouver le total possédé par les deux ou un état partiel.

— Les problèmes de type « **changement** » qui décrivent une transformation appliquée à un état initial pour aboutir à un état final. Exemples : « *Sophie avait 3 fleurs pendant la récréation. On lui en a offert 4.* » ou encore « *Pierre a maintenant 7 autocollants* ». L'inconnue à chercher dans ce genre de problème peut concerner l'état initial, la transformation (qui peut être additive ou soustractive) ou l'état final.

— Les problèmes de types « **comparaison** » qui comparent des quantités statiques à l'aide de formules du type «*de plus que*», «*de moins que*». Exemples : « *Marie a 3 billes. Anna a 8 billes* ». Ou encore « *Hélène a 9 autocollants de plus que Laure* ». L'inconnue peut porter ici sur la grande partie, la petite partie, ou la différence.

En outre, le domaine des problèmes additifs est intéressant pour nous, en ce sens que les problèmes décrivent une situation concrète ; la plupart du temps, les élèves sont en difficulté pour résoudre ce genre de problèmes, parce qu'ils n'arrivent pas à représenter correctement la situation décrite dans l'énoncé [64].

Prenons par exemple le problème de type réunion suivant, que nous appelons *réunion_1*. Ce problème décrit un jeu de billes : « *Julien et Sophie ont des billes. Julien en a 3. Ensemble ils en ont 12. Combien Sophie a-t-elle de billes* » ?

Pour un problème de ce type par exemple, nous nous attendons à une résolution en plusieurs étapes :

— Décrire le problème à l'aide d'une « opération à trou » : $3 + ? = 12$

— Écrire comment s'effectue le calcul : 12 - 3 = ?

— Effectuer le calcul : 9

— Écrire la réponse à la question : *Sophie a 9 billes.*

Afin de pouvoir décrire ce genre de problème dans AMBRE-KB, l'auteur doit d'abord définir un vocabulaire pour le domaine des problèmes additifs. Pour cela, il doit

cliquer sur l'onglet « *Vocabulaire du domaine* » (cf. **Figure 5.2**).

5.1.1 Définition du vocabulaire

En cliquant sur l'onglet « *Vocabulaire du domaine* », une interface apparaît permettant ainsi à l'auteur de définir les *types d'objets*. Pour chaque type d'objet créé, le système ajoute automatiquement une caractéristique nommée *identifiant* permettant de le distinguer des autres types d'objets. Le type de cette caractéristique est une chaîne de caractères, sa valeur devra être renseignée par l'auteur à chaque fois qu'il sera utilisé pour définir des problèmes, règles ou techniques de résolution. L'auteur peut également associer d'autres caractéristiques à chaque type d'objet. Pour chacune d'elle, l'auteur doit définir un *nom* et un *type* qui peut être simple (chaîne de caractères, booléen, nombre, type énuméré, etc.) ou complexe (type objet). Il doit également spécifier si cette caractéristique est obligatoire ou pas. Définir une caractéristique comme étant obligatoire signifie que sa valeur devra être renseignée pour tous les problèmes dans lesquels le type d'objet aura été utilisé. Cette fonctionnalité est également importante, parce qu'une fois qu'une caractéristique est définie comme étant obligatoire, le système peut, à la demande de l'auteur, faire des contrôles lors de la définition des problèmes pour vérifier que sa valeur a été bien renseignée, et dans le cas contraire un message d'erreur est envoyé à l'auteur pour le lui rappeler.

La **Figure 5.3** montre une partie du vocabulaire proposé pour le domaine des problèmes additifs. Nous avons la liste des différents objets (à gauche), et pour chaque objet la liste de ses caractéristiques (au centre). Les types d'objets *personne* et *liste de personnes* par exemple, permettent de définir les personnes impliquées dans le jeu, et pour chacune d'elles, le type d'éléments possédés ainsi que leur nombre. Nous avons au centre la liste des caractéristique du type d'objet personne qui est sélectionné.

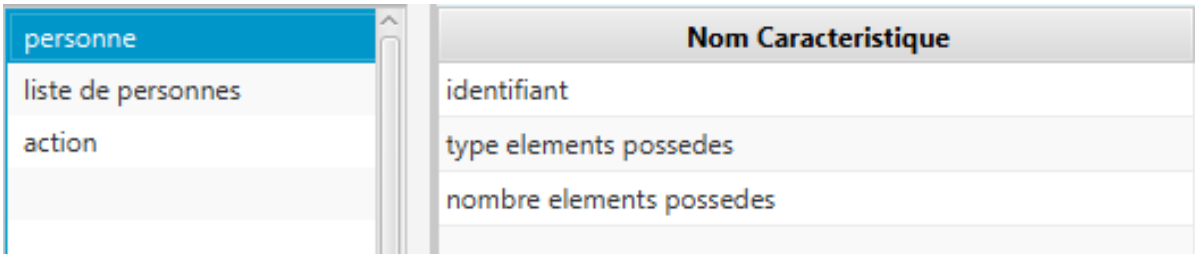

FIGURE 5.3 – Une partie du vocabulaire pour le domaine des problèmes additifs

Pour définir le vocabulaire d'un domaine, l'auteur a deux possibilités. Il peut choisir de définir d'abord l'ensemble des types objets dont il aura besoin pour décrire les problèmes à résoudre. Il peut également décider de définir les types d'objets au fur et à mesure qu'il en a besoin pour décrire les problèmes.

Le système permet à l'auteur à tout moment de vérifier si son vocabulaire est conforme au méta-modèle du vocabulaire (cf. chapitre 3). Il vérifie par exemple qu'il n'y a pas de duplication au niveau du nom des objets, et que pour chaque caractéristique définie, l'auteur a bien associé un type.

Une fois que l'auteur a défini au moins une partie du vocabulaire pour le domaine qu'il a choisi, il peut commencer à décrire des problèmes à résoudre. Pour cela, il doit cliquer sur l'onglet « *Problèmes à résoudre* » (cf. **Figure 5.2**).

5.1.2 Description des problèmes à résoudre

En cliquant sur l'onglet dédié à la description des problèmes, l'auteur dispose d'une interface permettant de définir de nouveaux problèmes, de les modifier/supprimer, d'en créer de nouveaux à partir de ceux déjà définis (dupliquer des problèmes). Pour chaque problème, l'auteur dispose d'un menu lui permettant de choisir les types d'objets lui permettant de décrire l'énoncé de son problème. Ensuite, il doit nommer chaque objet ainsi que chacune de ses caractéristiques. Le problème *réunion_1*, précédemment présenté, peut être défini par exemple comme suit :

- Nous disposons d'une liste E de deux personnes : *Julien* et *Sophie*
- Le nombre d'éléments possédés par *Julien* est *5*, et le type des éléments possédés est : *billes*
- Le nombre d'éléments possédés par *Sophie* est *X*, et le type des éléments possédés est *billes*
- Le nombre d'éléments possédés par l'ensemble des deux personnes est 12
- La question à résoudre consiste à calculer le nombre d'éléments *X* possédés par *Sophie*.

E représente ici l'identifiant de l'objet « *liste de personnes* », « *Sophie* » et « *Julien* » représentent les identifiants des objets *personnes* (éléments de la liste de personnes).

La **Figure 5.4** représente la description des objets du problème *réunion_1* en utilisant AMBRE-KB. Nous disposons d'un ensemble de deux personnes : Sophie et Julien. Julien possède 5 éléments de type billes. Sophie possède X éléments de types billes. Le nombre total d'éléments possédés par Sophie et Julien est 12. Nous cherchons ici le nombre d'éléments possédés par Sophie.

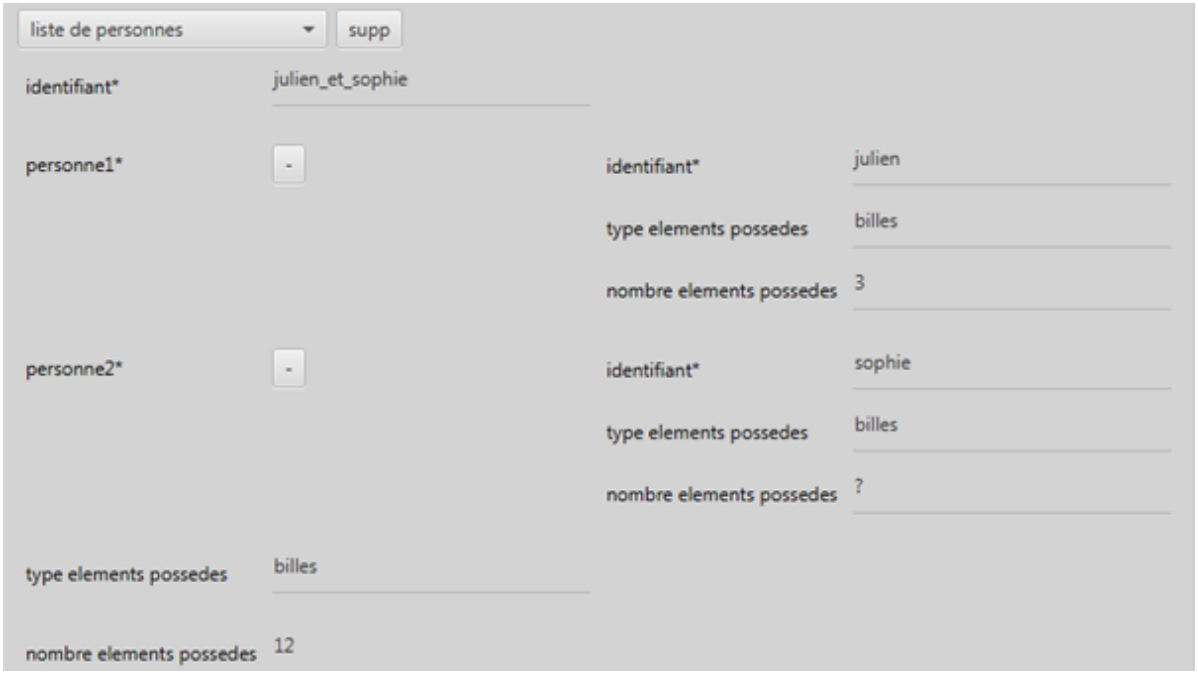

FIGURE 5.4 – Description des objets du problème réunion_1

Comme pour le vocabulaire, le système permet à l'auteur, à tout moment de la description des problèmes, de vérifier si les connaissances définies sont bien conformes au méta-modèle des problèmes (cf. chapitre 3). Par exemple, pour définir les caractéristiques des objets choisis, le système contraint les champs à remplir de sorte à permettre à l'auteur de respecter le type attendu. Pour définir par exemple, la caractéristique *nombre elements possedes* par une personne qui est de type *nombre entier*, l'auteur ne peut définir que des nombres entiers ; sauf si cette caractéristique est l'inconnue que l'on cherche et dans ce cas il met un point d'interrogation pour avertir le système que l'inconnue à calculer porte sur cette caractéristique. Le système vérifie également que les valeurs de toutes les caractéristiques définies comme étant obligatoires sont renseignées. Des fonctionnalités d'auto-complétion sont également proposées à l'auteur au fur et à mesure de la définition des objets du problème, de sorte qu'il puisse les réutiliser en cas de besoin. Une fois la description des problèmes terminée, l'auteur peut commencer à définir les

règles et les techniques de résolution. Les règles permettent d'identifier, à partir de l'énoncé des problèmes, des éléments permettant au système de les situer dans l'arbre de classification, et les techniques permettent au système de calculer les solutions des problèmes. L'ensemble formé par l'arbre de classification, les règles et les techniques constitue les connaissances sur la méthode à enseigner.

5.1.3 Élicitation des connaissances sur la méthode

Pour définir les connaissances sur la méthode, l'auteur doit définir trois types de connaissances : les connaissances de classification, les connaissances de reformulation et les connaissances de résolution. Nous présentons l'élicitation de ces connaissances dans les sections qui suivent.

5.1.3.1 Élicitation des connaissances de classification

Pour définir les connaissances de classification, l'auteur dispose d'un onglet spécifique « *Connaissances de classification* » (cf. **Figure 5.2**). En cliquant sur cet onglet, le système lui propose une interface permettant de définir un arbre de classification des problèmes. Cette interface dispose d'un menu permettant d'ajouter de nouvelles classes, de modifier ou de supprimer des classes. Pour chaque classe (cf. **Figure 5.5**) l'auteur doit :

— Définir le nom en cliquant sur (1),

— Définir l'attribut discriminant en cliquant sur (2)

— Définir statut de la classe. Par défaut toutes les classes sont non opérationnelles. Pour changer le statut d'une classe, l'auteur doit cliquer sur (3). Le système colore en jaune toutes les classes que l'auteur aura définies comme étant opérationnelles.

— Ajouter des sous-classes en cliquant sur le bouton "+" (4)

Pour chaque classe, l'auteur peut également définir des attributs du problème. Pour ce faire, pour chaque attribut, il doit définir le nom en cliquant sur (5), puis sur (6) pour le valider. Le système ajoute alors l'attribut et efface le champ (5) pour qu'il puisse rajouter d'autres attributs.

FIGURE 5.5 – Définition d'une classe de problèmes en utilisant AMBRE-KB

Pour définir un arbre de classification, le système offre plusieurs modes d'interaction, afin de répondre aux besoins et préférences de l'auteur. Par exemple, l'auteur peut choisir de construire son arbre de la racine vers les feuilles ou vice-versa. Il peut aussi choisir de définir toutes les classes et de les relier ensuite entre elles. Le système offre également la possibilité à l'auteur de définir une classe à partir d'autres classes : classe mère, classe fille, classe sœur. La **Figure 5.6** représente une partie de l'arbre de classification pour les problèmes additifs. *C_PROBLEME* représente la classe de l'arbre. Elle a pour attribut discriminant : *type du problème*. *Reunion*, *changement* et *comparaison* représentent les valeurs discriminantes de cet attribut. Les classes colorées en jaunes représentent des classes *opérationnelles* auxquelles il faudra associer une technique de résolution, les autres classes sont *non opérationnelles*.

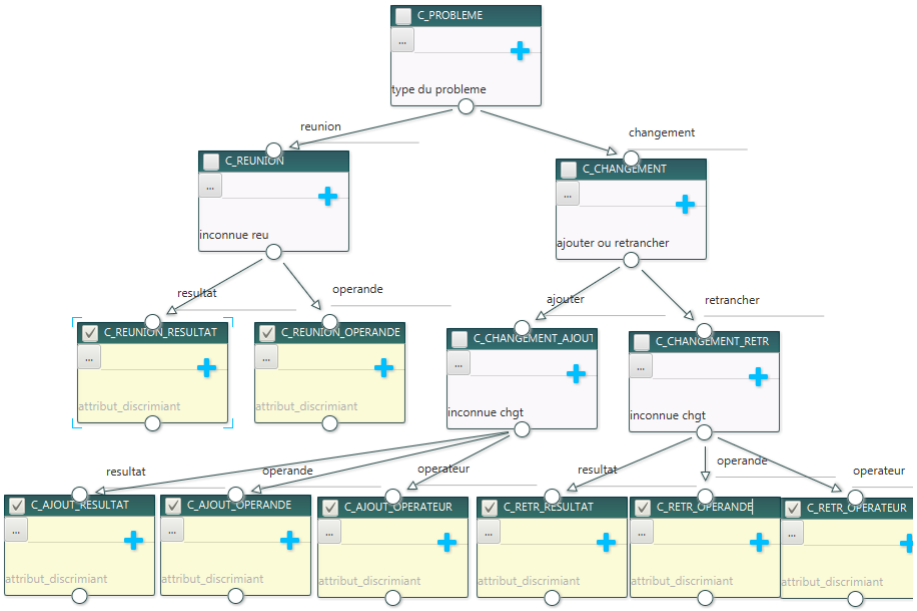

FIGURE 5.6 – Une partie de l'arbre de classification pour les problèmes additifs

L'auteur est également guidé pour la construction de l'arbre de classification. Par exemple, le fait de colorer les classes opérationnelles permet de rappeler à l'auteur qu'il doit associer des techniques de résolution à ces classes, mais également d'avoir un aperçu des classes qui sont opérationnelles et de celles qui ne le sont pas. De même, lorsque l'auteur définit une sous-classe, le système lui propose par exemple un nom pour cette classe en fonction de la valeur de l'attribut discriminant correspondant. Cette fonctionnalité permet à l'auteur d'aller plus vite dans la construction de l'arbre de classification, mais également de pouvoir repérer facilement la classe parent d'une classe en fonction des attributs discriminants.

Lorsque l'auteur définit un même attribut pour des sous-classes (par exemple, l'attribut *inconnue chgt* les classes *C_CHANGEMENT_AJOUT* et *C_CHANGEMENT_RET* cf. Figure 5.6), le système lui propose de définir cet attribut au niveau de leur parent commun le plus proche (plus proche ancêtre commun). Cette fonctionnalité va, par exemple, permettre à l'auteur de définir une seule fois les règles permettant de conclure sur la valeur de cet attribut, et de pouvoir par la suite les utiliser pour les deux classes. Dans notre exemple, les règles permettant de conclure sur la valeur de l'attribut *inconnue chgt* seront définies une seule fois.

Le système permet à l'auteur, à tout instant, de vérifier que son arbre de classi-

fication est conforme au méta-modèle correspondant (cf. Chapitre 3). Le système vérifie, par exemple, que toutes les classes terminales de l'arbre sont opérationnelles, que toutes les classes non opérationnelles ont au moins une sous-classe, etc. Lorsque l'auteur veut définir un nouvel attribut discriminant par exemple, le système lui propose des fonctionnalités d'auto-complétion pour lui donner la possibilité de vérifier les attributs discriminants déjà définis et d'en réutiliser un s'il le désire.

Une fois qu'une partie de l'arbre est définie, l'auteur peut (s'il le désire) commencer à définir les règles qui permettent au résolveur de situer les problèmes dans l'arbre de classification. Pour ce faire, l'auteur doit aller sur l'onglet « *Règles de reformulation*» (cf. **Figure 5.2**)..

5.1.3.2 Élicitation des connaissances de reformulation

Afin de faciliter l'élicitation des règles de reformulation, le système charge automatiquement dans l'interface des règles tous les attributs définis dans l'arbre de classification. Ainsi, en cliquant sur l'onglet « *Règle de reformulation* » l'auteur peut directement associer à chaque attribut les règles permettant de conclure sur sa(ses) valeur(s).

Comme pour les connaissances de classification, le système offre plusieurs modes d'interaction pour définir les règles. Par exemple, l'auteur peut choisir de définir toutes les règles au fur et à mesure de la définition de l'arbre de classification. En d'autres termes, il s'agit pour chaque classe, lors de la définition d'un attribut discriminant ou d'un attribut du problème, de définir la(s) règle(s) qui permettra(tront) de déterminer la(les) valeur(s) correspondante(s). L'auteur peut également choisir de définir entièrement l'arbre de classification, et ensuite procéder à la définition des règles. Pour l'aider dans la définition des règles, le système offre également à l'auteur de définir une règle à partir d'une autre déjà définie, en la complétant ou en la modifiant.

Le système vérifie que les connaissances explicitées sont conformes au méta-modèle correspondant. Pour cela, il contrôle que, pour chaque règle, l'auteur a défini une partie prémisse et une partie conclusion qui permet de conclure sur la valeur de l'attribut. Pour chacun des attributs, le système vérifie également qu'au moins une règle a été associée afin de permettre de déterminer sa valeur. Pour guider l'auteur, le système lui propose au fur et à mesure de la définition des règles des fonctionnalités d'auto-complétion : par

exemple dans les parties prémisses, pouvoir visualiser les objets qu'il a déjà définis et les réutiliser au besoin.

Reprenons notre exemple sur les problèmes additifs.
Les règles *regle_reunion_1*, *regle_changement_1* permettent de conclure sur la valeur de l'attribut discriminant *type du problème*. Les règles *regle_reunion_operateur* et *regle_reunion_operande* permettent de conclure sur la valeur de l'attribut *inconnue reu* (cf. **Figure 5.7**).

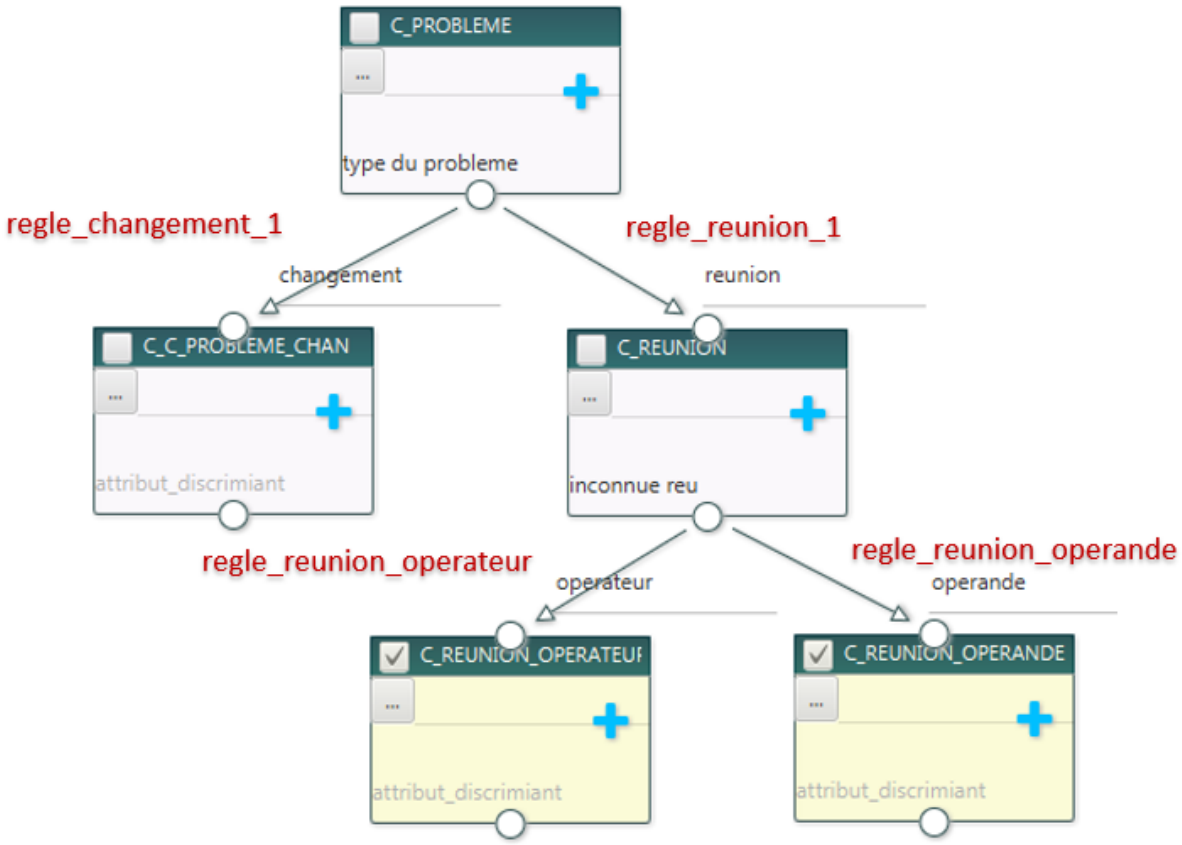

FIGURE 5.7 – Une partie de l'arbre de classification pour les problèmes additifs

La règle *regle_réunion_1* par exemple peut être représentée comme suit :

Si dans un problème on a :
- une liste de personnes L
- une personne P1 appartenant à L
- le type des éléments possédés par P1 est T1
- le nombre d'éléments possédés par P1 est N1
- une personne P2 appartenant à L
- le type d'éléments possédés par P2 est T2
- le nombre d'éléments possédés par P2 est N2
- le type d'éléments possédés par L est T
- le nombre d'éléments possédés par L est N
- T1, T2 et T sont de même catégorie
- deux parmi N1, N2 et N3 sont connus

Alors la valeur de l'attribut discriminant *type du problème* est égale à *réunion*.

Tableau 5.1 – Exemple de règle permettant de conclure sur la valeur de l'attribut type du problème

Soient *nombre_total_possede* et nombre *partiel_possede* deux attributs du problème définis au niveau de la classe *c_reunion*, la règle *regle_reunion_operande* permettant de conclure sur la valeur de l'attribut *inconnue_reu* peut être représentée comme suit :

Si dans un problème on a :
- une liste de personnes L
- une personne P1 appartenant à L
- le type des éléments possédés par P1 est T1
- le nombre d'éléments possédés par P1 est N1
- une personne P2 appartenant à L
- le type d'éléments possédés par P2 est T2
- le nombre d'éléments possédés par P2 est N2
- le type d'éléments possédés par L est T
- le nombre d'éléments possédés par L est N
- T1, T2 et T sont de même catégorie
- l'inconnue à calculer porte sur N2

Alors la valeur de l'attribut discriminant *inconnue réu* est égale à *opérande*,
la valeur de l'attribut du problème nombre total possédé est égale à N,
et la valeur de l'attribut du problème nombre partiel possédé est égale à N1.

Tableau 5.2 – Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé

Les **Figures 5.8** et **5.9** représentent respectivement les prémisses et conclusion de la règle regle_reunion_operande définie en utilisant AMBRE-KB.

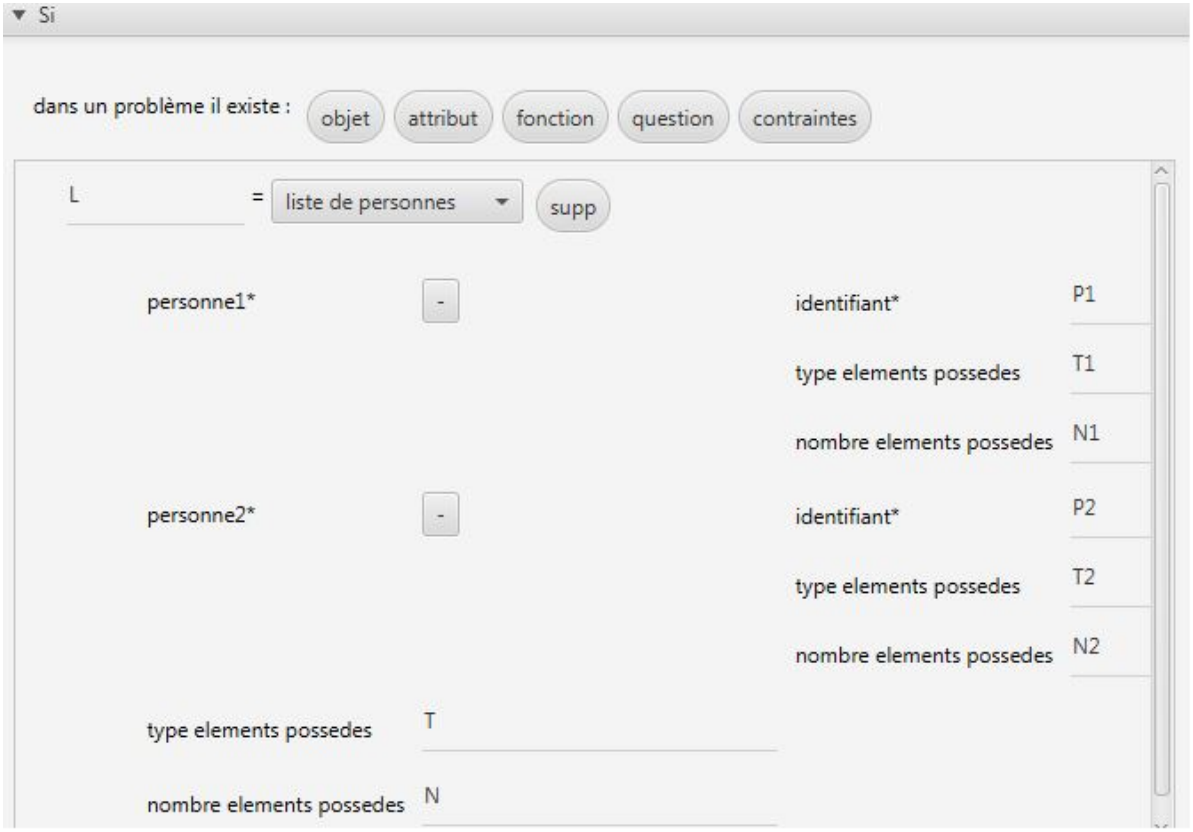

FIGURE 5.8 – représentation d'une partie des prémisses de la regle_reunion_operande dans AMBRE-KB

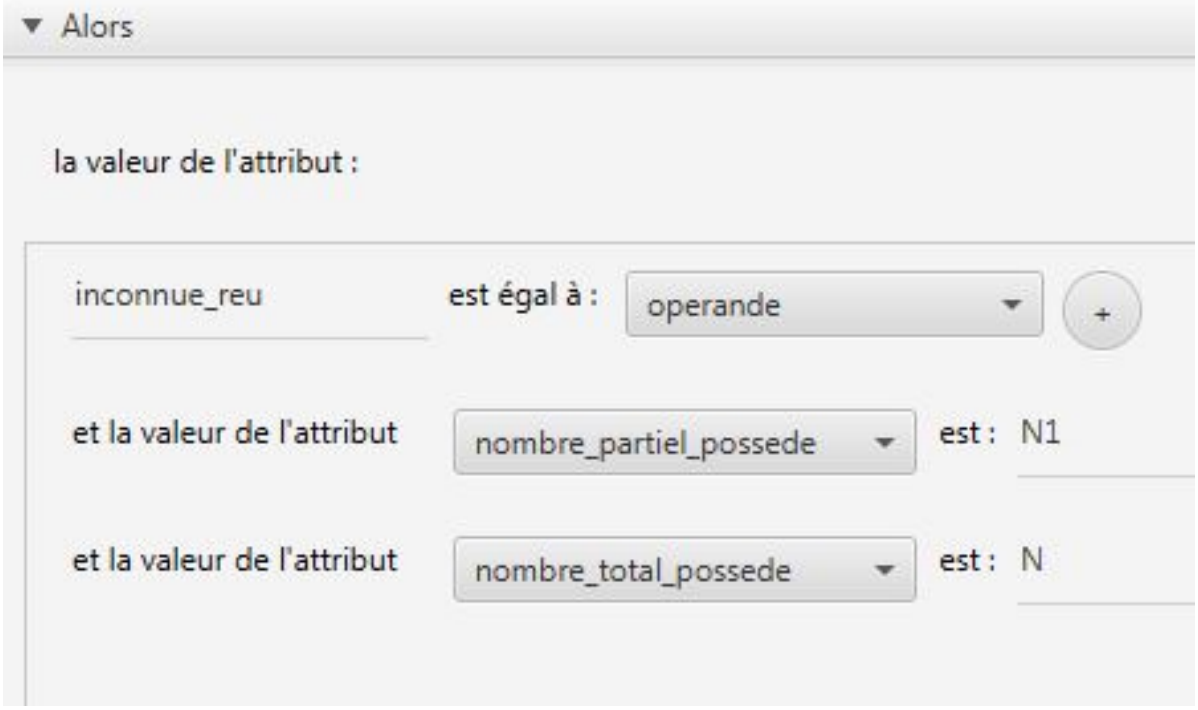

FIGURE 5.9 – représentation de la partie conclusion de la regle_reunion_operande dans AMBRE-KB

D'autre part, pour associer des techniques de résolution aux classes opérationnelles de l'arbre de classification, l'auteur doit cliquer sur l'onglet « *Techniques de résolution* » (cf. **Figure 5.2**).

5.1.3.3 Élicitation des connaissances de résolution

Lorsque l'auteur a cliqué sur l'onglet « *Technique de résolution* » une nouvelle interface s'ouvre, lui permettant ainsi de décrire les techniques de résolution. Pour faciliter l'élicitation de ces connaissances, le système charge automatiquement dans l'interface toutes les classes que l'auteur aura définies comme étant opérationnelles au niveau de l'arbre de classification. Ainsi, l'auteur peut associer directement une technique de résolution à chacune de ces classes. Pour ce faire, il peut choisir de définir les techniques au fur et à mesure de la définition des classes opérationnelles dans l'arbre de classification. Il peut également choisir de définir toutes les classes dans un premier temps et ensuite leur associer des techniques de résolution.

La technique *tech_reunion_opérande* permet par exemple de résoudre les problèmes de la classe réunion opérande.

Technique tech_réunion_opérande :

Soit P un problème,
Soit N la valeur de l'attribut *nombre total possédé* dans P,
Soit N1 la valeur de l'attribut *nombre partiel possédé* dans P,
Soit N2 le résultat de la soustraction entre N et N1,
Alors la **solution du problème** P est égale à N2.

Tableau 5.3 – Exemple de technique permettant de résoudre les problèmes de la classe *réunion*

La **Figure 5.10** montre une représentation de cette technique dans AMBRE-KB.

FIGURE 5.10 – représentation de la technique tech_réunion_opérande dans ambre-kb

5.2 Génération des modèles de connaissances

Une fois le processus d'élicitation terminé, en cliquant sur le menu « *générer les modèles de connaissances* » dans AMBRE-KB, l'auteur peut générer les modèles de

connaissances correspondants aux différentes connaissances explicitées.

Les **Figures 5.8**, **5.9**, **5.10**, et **5.11** représentent respectivement des extraits : du modèle descriptif du problème réunion1, du modèle des connaissances de classification, du modèle des connaissances de reformulation, et du modèle des connaissances de résolution.

```
fait(probleme(reunion_1)). /* soit le problème reunion_1*/
fait(enonce_naturel(reunion_1,Julien et Sophie ont des billes. Julien en a 3.
Ensemble ils en ont 12. Combien Sophie a-t-elle de billes ')).
/* énoncé de reunion_1*/
fait(est_un(julien_et_sophie,liste_de_personnes)).
/* julien_et_sophie une liste de personnes*/
fait(personne1(julien_et_sophie,julien)). /* julien une personne
appartenant à la liste de personnes*/
fait(est_un(julien,personne)). /*julien est une personne*/
fait(type_elements_possedes(julien,billes)). /* type éléments possédés
par julien est billes*/
fait(nombre_elements_possedes(julien,3)). /* nombre éléments possédés
par julien est 3 billes*/
fait(personne2(julien_et_sophie,sophie)). /* sophie une personne*/
/* appartenant à la liste de personnes*/
fait(est_un(sophie,personne)). /* sophie est une personne*/
fait(type_elements_possedes(sophie,billes)).
/* type éléments possédés
/*par sophie est billes*/
fait(nombre_elements_possedes(sophie,X)).
/* nombre éléments possédés*/
/*par sophie est X billes*/
fait(type_elements_possedes(julien_et_sophie,billes)). /* type éléments
possédés par julien et sophie est billes*/
fait(nombre_elements_possedes(julien_et_sophie,12)). /* nombre éléments
possédés par julien et sophie est 12 billes*/
fait(calculer(reunion_1,X)).
/* résoudre le problème reunion_1 consiste
/*à calculer la valeur de x*/
```

Tableau 5.4 – Modèle du problème *reunion_1* en Prolog généré par AMBRE-KB

```
/********** La Classe racine C_PROBLEME *********/
racine_arbre(c_probleme).
statut(c_probleme,non_operationnelle).
test_fils(c_probleme,P,type_du_probleme(P,_)).
valeur_fils(c_probleme,P,c_reunion) :-eval_prem(type_du_probleme(P,[reunion])).
valeur_fils(c_probleme,P,c_changement) :-eval_prem(type_du_probleme(P,[changement])).
attribut(c_probleme,P,type_du_probleme(P,_),type_du_probleme,[regle_changement_1,regle_reu
/********* La Classe C_REUNION **********/
pere(c_reunion,c_probleme).
statut(c_reunion,non_operationnelle).
test_fils(c_reunion,P,inconnue_reu(P,_)).
valeur_fils(c_reunion,P,c_reunion_resultat) :-eval_prem(inconnue_reu(P,[resultat])).
valeur_fils(c_reunion,P,c_reunion_operande) :-eval_prem(inconnue_reu(P,[operande])).
attribut(c_reunion,P,inconnue_reu(P,_),inconnue_reu,[regle_reunion_operateur,regle_reunion_op
attribut(c_reunion,P,nombre_total_possede(P,_),nombre_total_possede,[]).
attribut(c_reunion,P,nombre_partiel_possede(P,_),nombre_partiel_possede,[]).
/******** La Classe C_REUNION_OPERANDE*********/
pere(c_reunion_operande,c_reunion).
statut(c_reunion_operande,operationnelle).
...
```

Tableau 5.5 – Une partie du modèle de connaissances représentant les connaissances de classification (cf. Figure 5.6) généré par AMBRE-KB en Prolog

```
/* représentation de la règle regle_reunion_1 en Prolog*/
regle(regle_reunion_1) :-
        si([probleme(P),
                est_un(E,liste_de_personnes),
                personne1(E,P1),
                est_un(P1,personne),
                type_elements_possedes(P1,T1),
                nombre_elements_possedes(P1,X1),
                personne2(E,P2),
                est_un(P2,personne),
                type_elements_possedes(P2,T2),
                nombre_elements_possedes(P2,X2),
                type_elements_possedes(E,T3),
                nombre_elements_possedes(E,X3),
                eval_prolog(sont_meme_categorie(T1,T2)),
                eval_prolog(sont_meme_categorie(T1,T3)),
                eval_prolog(deux_sont_connus(X1,X2,X3))]),
        alors([ajouter(type_du_probleme(P,[reunion]))]).
```

```
/* représentation de la règle regle_reunion_operande en Prolog*/
regle(reunion_reunion_operande) :-
        si([probleme(P),
                est_un(E,liste_de_personnes),
                personne1(E,P1),
                est_un(P1,personne),
                type_elements_possedes(P1,T1),
                nombre_elements_possedes(P1,X1),
                personne2(E,P2),
                est_un(P2,personne),
                type_elements_possedes(P2,T2),
                nombre_elements_possedes(P2,X2),
                type_elements_possedes(E,T3),
                nombre_elements_possedes(E,X3),
                calculer(P,X2)]]),
        alors([ajouter(inconnue_reu(P,[operande])),
                ajouter(nombre_total_possede(P,X3)),
                ajouter(nombre_partiel_possede(P,X1))]).
```

Tableau 5.6 – Une partie du modèle de connaissances décrivant les règles généré par AMBRE-KB en Prolog

```
/* technique à appliquer pour résoudre les problèmes de la classe c_reunion_operande*/
appliquer_resolution(P,c_reunion_operande) :-
eval_prem(nombre_total_possede(P,N)),
eval_prem(nombre_partiel_possede(P,N1)),
soustraction(N,N1,N2),
ajouter_bf(solution(P,[N2])).
```

Tableau 5.7 – Une partie du modèle de connaissances décrivant les connaissances de résolution, généré par AMBRE-KB en Prolog

5.3 Tests des modèles de connaissances

Une fois le processus d'élicitation terminé et les modèles de connaissances générés, AMBRE-KB offre à l'auteur la possibilité d'avoir un feedback sur les connaissances explicitées, en testant la résolution de problèmes qu'il a définis. Pour cela, l'auteur doit choisir un problème et lancer la résolution. Le résolveur effectue alors une phase d'opérationnalisation qui lui permet de résoudre le problème sélectionné. Lorsque la résolution du problème a réussi, le système affiche à l'auteur un message lui permettant de voir que tout s'est bien passé, ainsi que la solution du problème. Il lui propose aussi de consulter la trace de la résolution (les règles appliquées, les attributs calculés ainsi que leurs valeurs, les règles essayées, et celles exécutées,la technique de résolution appliquée).

Les **Figures 5.11** et **5.12** montrent des exemples de feedback renvoyés par AMBRE-KB en testant la résolution du problème *reunion_1*.

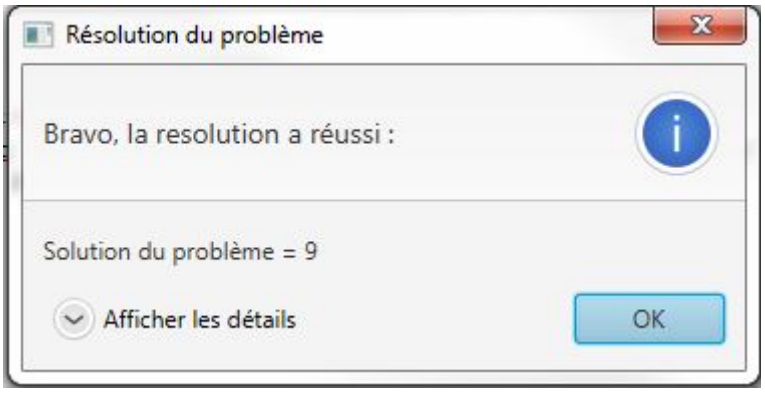

FIGURE 5.11 – Exemple de feedback lorsque la résolution réussit. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème

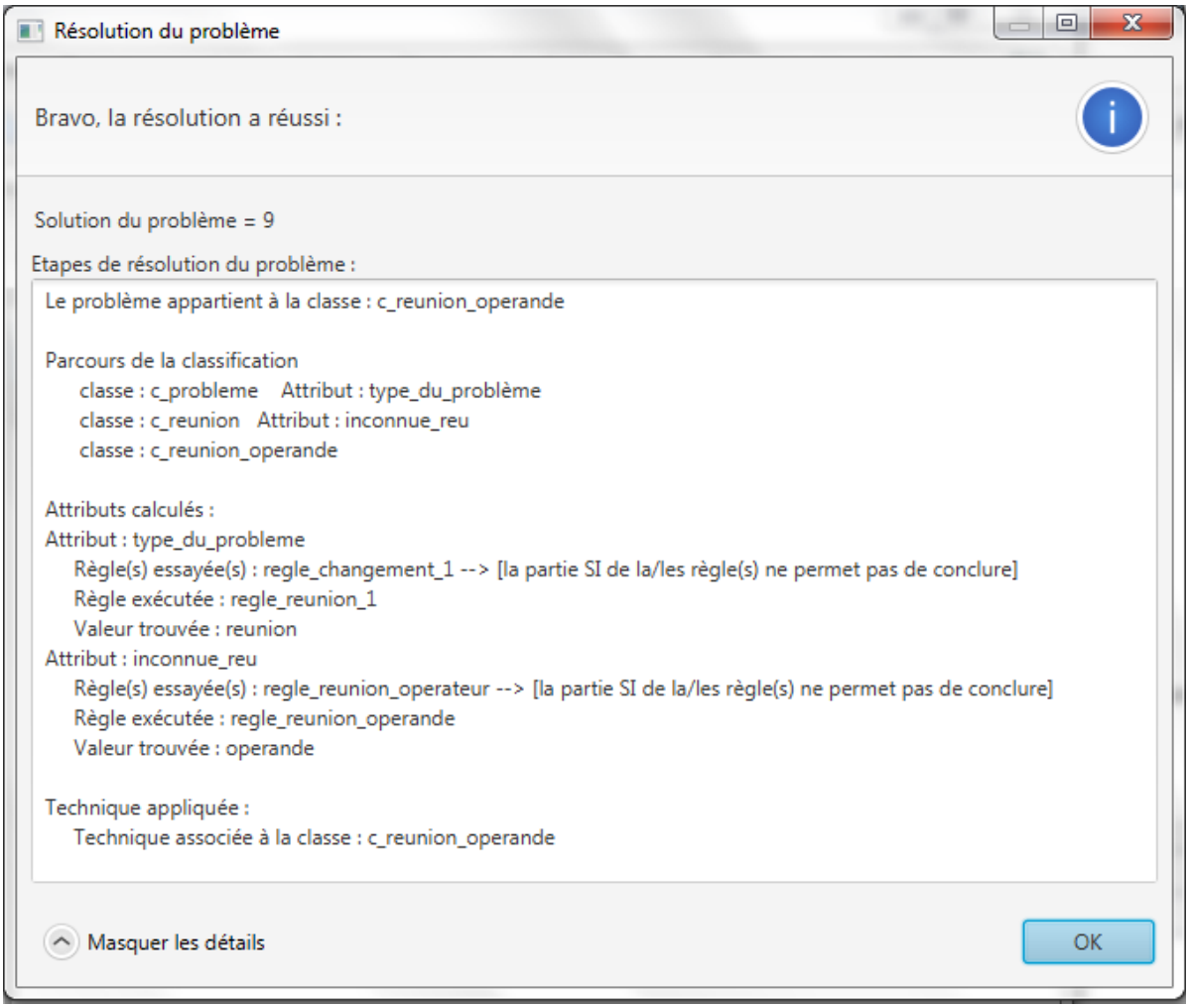

FIGURE 5.12 – Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a réussi

De même, lorsque le résolveur ne parvient pas à résoudre un problème, il affiche un message informant l'auteur que la résolution a échoué. Par exemple, la résolution d'un problème peut échouer lorsque le système n'arrive pas à trouver de règles permettant de conclure sur la valeur d'un attribut discriminant. Dans ce cas, le système envoie un message d'erreur pour indiquer à l'auteur que la classification est bloquée au niveau de la classe de cet attribut.

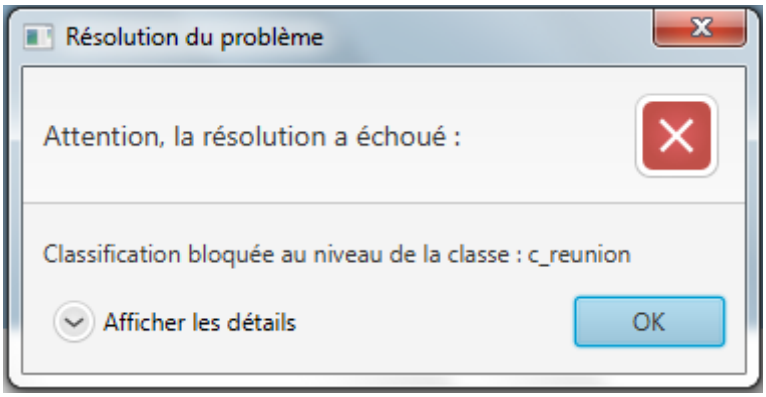

FIGURE 5.13 – Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème

En cliquant sur "*afficher les détails*" (cf. **Figure 5.13**), le système affiche :

— les classes rencontrées, avec pour chacune d'elle la valeur de l'attribut discriminant

— les attributs calculés. Pour chaque attribut, le système affiche les règles essayées (s'il en existe) et pourquoi elles ne se sont pas déclenchées. Les règles essayées représentent les règles dont l'ensemble des prémisses ne sont pas satisfaites, elles ne permettent donc pas au système de conclure. Si des règles ont été exécutées, il les affiche également, ainsi que la valeur trouvée.

— l'attribut auquel la classification est bloquée

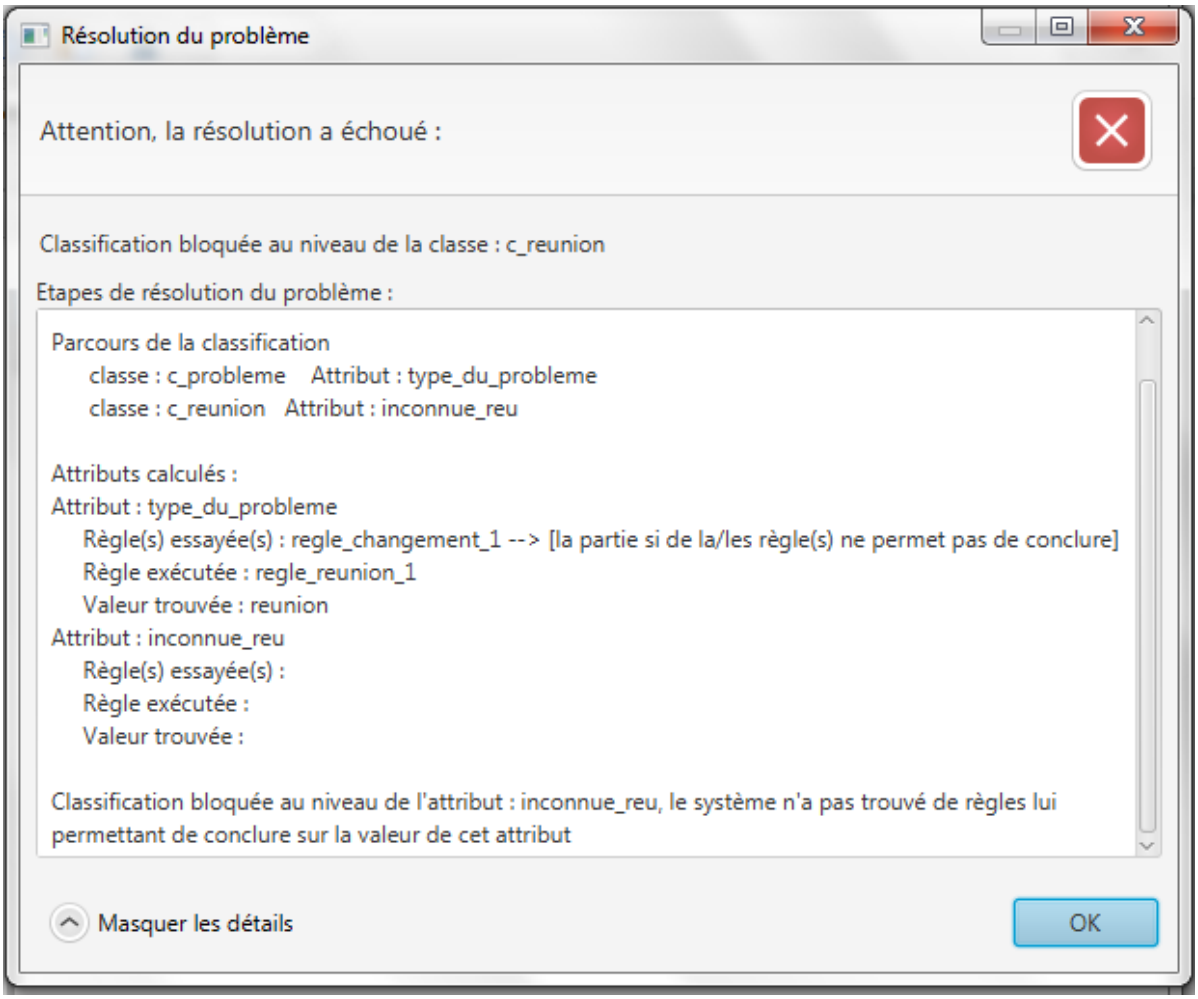

FIGURE 5.14 – Exemple de feedback lorsque la résolution a échoué parce que la classification est bloquée parce que le système n'a pas trouvé de règle permettant de conclure sur la valeur d'un attribut

La résolution d'un problème peut également échouer lorsque le résolveur identifie la classe du problème, mais que la technique de résolution a associé est incomplète ou comporte des erreurs. Dans ce cas, le système envoie un message d'erreur permettant à l'auteur de comprendre que la résolution a échoué à cause d'un problème rencontré lors de l'application d'une technique de résolution.

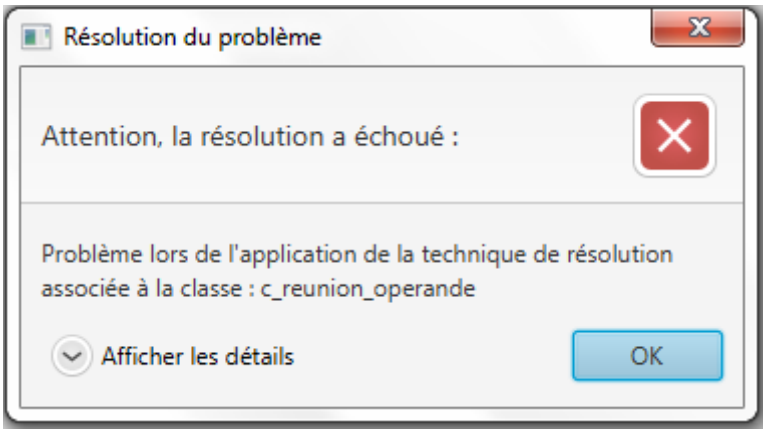

FIGURE 5.15 – Exemple de feedback lorsque la résolution a échoué lors de l'application d'un technique de résolution. En cliquant sur "*afficher les détails*" le système affiche les différentes étapes de résolution du problème

En cliquant sur "*afficher les détails*" (cf. **Figure 5.15**), le système affiche les différentes étapes de résolution : la classe du problème, les classes rencontrées, les attributs calculés et pour chacun d'eux, les règles essayées, celles exécutées et les valeurs calculées. Ces informations permettent à l'auteur de savoir qu'il doit corriger ou compléter la technique de résolution associée à la classe pour pouvoir corriger son erreur. La **Figure 5.16** montre un exemple de feedback plus détaillé.

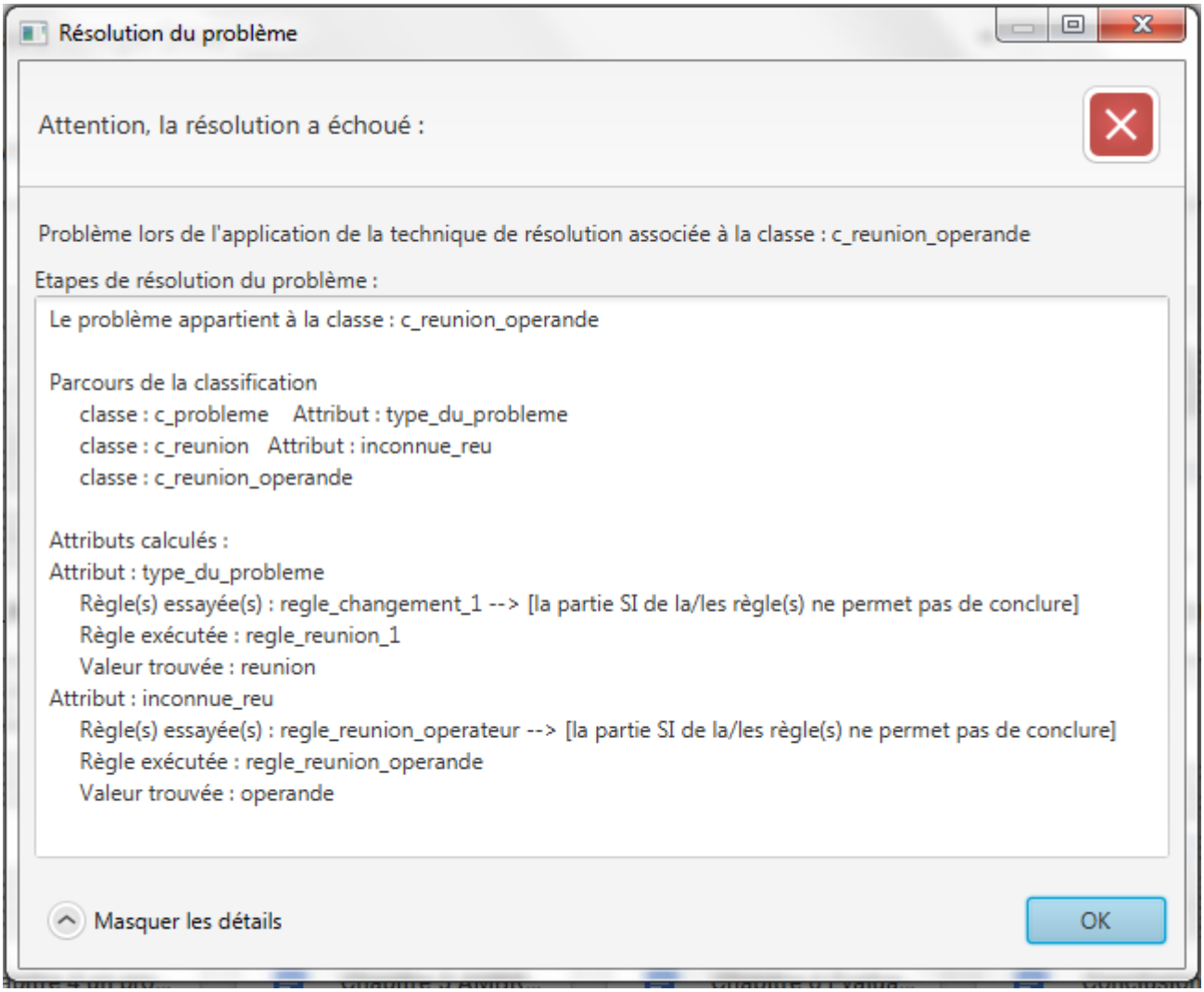

FIGURE 5.16 – Exemple de feedback avec les étapes de résolution lorsque la résolution du problème a échoué lors de l'application d'une technique de résolution

5.4 Conclusion

Nous avons présenté dans ce chapitre notre outil auteur, AMBRE-KB, dont l'objectif est d'assister un auteur dans la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes. En particulier, AMBRE-KB permet d'acquérir les connaissances nécessaires au système pour résoudre des problèmes dans un domaine donné (connaissances de classification, connaissances de reformulation et connaissances de résolution). L'élicitation des connaissances permettant d'accompagner l'apprenant

(connaissances d'aide, de diagnostic et d'explications sur les erreurs) n'est pas encore prise en compte.

AMBRE-KB repose sur un processus interactif d'élicitation des connaissances. Ce processus est basé sur des méta-modèles décrivant la forme des différentes connaissances à acquérir, et permettant de contraindre les interfaces dédiées à éliciter les connaissances. AMBRE-KB assiste également l'auteur durant le processus en lui offrant une grande flexibilité, notamment concernant l'ordre de définition des connaissance, en le guidant, et en lui permettant de contrôler au fur et à mesure que les connaissances explicitées sont conformes aux différents méta-modèles. À la fin du processus d'élicitation, le système offre à l'auteur la possibilité d'avoir un feedback en lui permettant de tester la résolution des problèmes qu'il a définis, et de corriger ses erreurs au besoin.

Nous avons utilisé le domaine des problèmes additifs pour illustrer le fonctionnement de AMBRE-KB. Un EIAH AMBRE existe déjà pour ce domaine, mais nous avons choisi de tester AMBRE-KB à nouveau dans ce domaine pour voir comment les bases de connaissances auraient pu être créées en utilisant notre outil auteur. Les modèles de connaissances générés par AMBRE-KB ont permis de résoudre les problèmes du domaine, mais des tests supplémentaires doivent être réalisés dans de nouveaux domaines pour évaluer l'outil.

Nous présentons dans le chapitre 6 les différentes expérimentations menées pour évaluer la mise en œuvre de l'outil auteur AMBRE-KB

Chapitre 6

Évaluations

Cette thèse a abouti à plusieurs propositions visant à la mise en place d'un outil auteur destiné à assister l'auteur dans l'élicitation des connaissances nécessaires à la conception d'EIAH dédiés à enseigner des méthodes de résolution de problèmes dans des domaines spécifiques. Ces propositions ont été mises en œuvre de manière opérationnelle à travers l'outil auteur AMBRE-KB. Ce chapitre présente les évaluations conduites pour évaluer le processus d'acquisition de connaissances ainsi que sa mise en œuvre dans l'outil auteur AMBRE-KB.

Sommaire

6.1 Mise à l'essai de AMBRE-KB 139

6.1.1 Première expérimentation 139

6.1.2 Seconde expérimentation 143

6.2 Bilan des expérimentations 154

Les travaux présentés dans cette thèse portent sur l'acquisition des connaissances nécessaires à la conception d'EIAH destinés à enseigner, pour un domaine donné, une méthode de résolution de problèmes. Les questions que nous nous sommes posées sont les suivantes :

1. **Comment permettre à un auteur (enseignant, professionnel ou pédagogue plus expert), pour un domaine donné, d'éliciter, sans programmer, les connaissances nécessaires à l'EIAH pour résoudre les problèmes de ce domaine relevant de la méthode à enseigner ?**
2. **Comment assister l'auteur durant ce processus d'élicitation ?**

Pour répondre à ces questions, nous avons d'abord proposé des méta-modèles pour décrire la forme des connaissances à acquérir. Nous avons ensuite proposé un processus interactif d'acquisition interactif des connaissances, fondé sur ces méta-modèles. Enfin, nous avons mis en œuvre ce processus en implémentant l'outil auteur AMBRE-KB. Même si l'élicitation des connaissances destinées à accompagner l'apprenant durant la résolution (connaissances d'aide, de diagnostic et d ?explications sur les erreurs) n'est pas encore prise en compte par AMBRE-KB, cet outil permet d'assister l'auteur durant l'élicitation des connaissances portant sur la méthode à enseigner, en le guidant tout au long du processus, en lui offrant une grande flexibilité notamment sur l'ordre de définition des connaissances, et en lui permettant de contrôler les connaissances au fur et à mesure qu'il les explicite. En effet, pour un domaine donné, celles-ci (les connaissances portant sur la méthode) permettent à elles seules de pouvoir résoudre les problèmes relevant de cette méthode.

Afin de mettre à l'essai AMBRE-KB, nous avons donc mené deux expérimentations portant sur l'élicitation des connaissances de la méthode à enseigner. Ce chapitre décrit de manière détaillée comment ces expérimentations ont été conduites, ainsi que les résultats obtenus. Il se termine par un bilan de ces expérimentations.

6.1 Mise à l'essai de AMBRE-KB

6.1.1 Première expérimentation

Nous avons mené une première expérimentation afin de mettre à l'essai AMBRE-KB. Cette expérimentation était divisée en deux étapes :

— **1ère étape**. Elle consistait à faire utiliser AMBRE-KB par des auteurs pour éliciter les connaissances à enseigner pour le domaine qu'ils ont choisi. Ces auteurs avaient pour tâche d'une part de définir de définir un vocabulaire pour leur domaine et d'utiliser ce vocabulaire pour décrire les problèmes qu'ils souhaitent faire résoudre par l'apprenant et le système. Ils devaient aussi, d'autre part définir les connaissances portant sur la méthode à enseigner (connaissances de classification, connaissances de reformulation et connaissances de résolution). Cette étape se terminait lorsqu'ils avaient fini d'explicité leurs connaissances.

— **2ème étape**. La deuxième étape était alors réalisée sans l'auteur. Il s'agissait de générer les modèles de connaissances correspondant aux connaissances explicitées par l'auteur en utilisant AMBRE-KB, puis de tester ces modèles indépendamment de AMBRE-KB, en utilisant le résolveur de AMBRE. Nous avons testé, pour chaque domaine, la résolution des problèmes définis par l'auteur, en utilisant le résolveur et les modèles de connaissances générés. Notre hypothèse était la suivante : « si le résolveur de AMBRE parvient à résoudre les problèmes des domaines à partir des modèles de connaissances générés par AMBRE-KB, alors AMBRE-KB est capable d'acquérir les connaissances nécessaires à un EIAH AMBRE pour résoudre les problèmes d'un domaine ».

Notre objectif pour cette expérimentation était double : nous souhaitions d'une part tester AMBRE-KB avec un domaine déjà existant (un EIAH AMBRE existant) afin de voir comment les bases de connaissances peuvent être créées à partir de AMBRE-KB. Le domaine des problèmes additifs a été ainsi testé. D'autre part, nous voulions tester le fonctionnement de AMBRE-KB sur de nouveaux domaines. Deux domaines ont ainsi été testés : le *calcul d'aire* et la *pâtisserie*.

Le **Tableau 6.1** résume le critère d'évaluation retenu pour cette expérimentation.

Point à analyser	Pour un domaine donné, les modèles de connaissances générés par AMBRE-KB permettent-ils au résolveur de résoudre des problèmes ?
Méthodologie envisagée	Pour chaque domaine, tester la résolution des problèmes en utilisant les modèles de connaissances générés par AMBRE-KB, et en lançant le résolveur en dehors de AMBRE-KB.
Condition de validation	Si le résolveur parvient à résoudre les problèmes d'un domaine, alors AMBRE-KB est capable d'acquérir les connaissances nécessaires à l'EIAH pour résoudre les problèmes de ce domaine.

Tableau 6.1 – Critère d'évaluation pour la première expérimentation

6.1.1.1 Protocole expérimental

Cette expérimentation s'est déroulée en deux étapes avec 3 sujets, dans trois domaines : deux nouveaux domaines (géométrie et pâtisserie) et un domaine (les problèmes additifs) dans lequel un EIAH AMBRE existe déjà (EIAH AMBRE-add). Les deux sujets qui ont testé sur les deux nouveaux domaines (géométrie et pâtisserie) ont d'abord commencé par choisir leur domaine. Pour cela, ils avaient pour contrainte de respecter le critère de classification des problèmes et des techniques de résolution imposé par l'architecture des EIAH AMBRE. Une fois qu'ils ont choisi leur domaine, ils ont commencé par réfléchir sur le type d'exercices à faire résoudre par le système et l'apprenant. Cette étape leur a permis de proposer le vocabulaire pour le domaine. Ensuite, ils ont dessiné sur papier une partie de l'arbre de classification de leur domaine. Ils ont également explicité sur papier quelques problèmes à résoudre, quelques règles, quelques techniques de résolution. En s'inspirant des problèmes à résoudre définis dans l'EIAH AMBRE-add, le troisième sujet a également proposé un vocabulaire pour le domaine. L'expérimentation s'est ensuite déroulée en deux étapes pour les trois sujets :

Étape 1 : Dans cette étape, les auteurs utilisent AMBRE-KB pour définir le vocabulaire de leur domaine, les problèmes à résoudre ainsi que les connaissances de classification, de reformulation et de résolution.

Étape 2 : Une fois que les auteurs ont terminé d'éliciter les connaissances pour leur domaine, à l'aide de AMBRE-KB, nous générons les modèles de connaissances corres-

pondants. Ensuite, en lançant le résolveur indépendamment de AMBRE, nous testons la résolution des différents problèmes définis.

6.1.1.2 Résultats

A l'issue de cette expérimentation, nous avons obtenu les résultats suivants :

— **Connaissances de classification** : pour tous les domaines les auteurs ont pu définir toutes les connaissances nécessaires

— **Autres connaissances** (vocabulaire, connaissances de reformulation, connaissances de résolution) : les interfaces proposées par AMBRE-KB n'ont pas permis de représenter toutes les connaissances nécessaires

— **Pas de génération de modèles de connaissances** : les interfaces proposées n'ayant pas permis d'expliciter toutes les connaissances, il n'y a pas eu de génération de fichiers, et par conséquent la résolution des problèmes n'a pas pu être testée.

En outre, les observations sur l'utilisation du système par les auteurs suggèraient plus d'interactivité de la part du système. En effet, durant le processus d'élicitation des connaissances, AMBRE-KB ne proposait rien permettant aux auteurs de vérifier s'ils avaient bien explicité les connaissances ou pas. De plus, à la fin de l'expérimentation, les utilisateurs ont manifesté le besoin d'avoir eux-mêmes un feedback sur le processus d'élicitation, c'est-à-dire de pouvoir vérifier si les connaissances ont été bien explicitées, et le cas échéant d'avoir un retour qui leur permettrait de comprendre l'origine de leur erreur afin de pouvoir la corriger. Ils ont posé les questions suivantes : comment est-ce que le système résout les problèmes ? comment savoir si la résolution des problèmes s'est bien passée ? et, le cas échéant, quelle est la cause de l'échec de la résolution et comment la corriger ?

En résumé, cette expérimentation ne nous a pas permis de valider notre hypothèse de départ. En effet, les interfaces proposées n'ayant pas permis aux auteurs d'expliciter toutes les connaissances nécessaires, les modèles de connaissances nécessaires à la résolution des problèmes n'ont pas pu être générés par le système. De plus, le système ne proposant aucun mécanisme d'interaction, ni de feedback, les auteurs n'avaient aucun moyen pour vérifier s'ils avaient bien défini les connaissances, si les connaissances définies étaient incomplètes ou s'ils avaient commis des erreurs lors de l'élicitation de celles-ci.

6.1.1.3 Recommandations et modifications

Les résultats de la première expérimentation ont conduit à modifier AMBRE-KB afin d'intégrer des fonctionnalités permettant d'assurer l'interactivité du système, mais aussi de permettre à l'auteur d'avoir un feedback sur les connaissances explicitées, ainsi que sur la résolution des problèmes par le résolveur.

Pour améliorer l'interactivité de AMBRE-KB, une nouvelle fonctionnalité a été mise en place pour chaque type de connaissances. Ainsi, après avoir explicité chaque type de connaissances (connaissances de classification, ou connaissances de reformulation, ou connaissances de résolution), l'auteur peut tester si les connaissances élicitées sont bien conformes au méta-modèle. Par exemple, pour les connaissances de classification, AMBRE-KB vérifie que les feuilles de l'arbre sont toutes des classes opérationnelles. Il vérifie également que chaque classe non opérationnelle possède au moins une sous classe, etc.

De plus, comme les interfaces proposées ne permettaient pas d'expliciter toutes les connaissances (notamment de reformulation et de résolution, et le vocabulaire), il était nécessaire de faire évoluer les méta-modèles correspondants, et donc les interfaces qui sont contraintes par ceux-ci. Par exemple, pour le vocabulaire, les auteurs souhaitaient pouvoir énumérer toutes les valeurs possibles pour une caractéristique, mais l'interface proposée par AMBRE-KB ne le permettait pas. Nous avons donc fait évoluer les interfaces pour intégrer les nouvelles fonctionnalités.

Une nouvelle fonctionnalité a également été développée pour permettre à l'auteur d'avoir un feedback sur les connaissances explicitées ainsi que sur la résolution des problèmes par le résolveur. Elle permet à l'auteur de tester lui-même la résolution des problèmes qu'il a définis à partir de AMBRE-KB. Cette fonctionnalité consiste à fournir un ensemble d'informations après la résolution du problème. Lorsque le résolveur réussit à résoudre un problème, ces informations peuvent être, par exemple, la classe du problème, les règles essayées, celles exécutées, les valeurs des attributs déterminés, la solution du problème, etc. Lorsque la résolution d'un problème échoue, le résolveur doit envoyer un ensemble d'informations lui permettant de comprendre son erreur et de pouvoir la corriger : par exemple, la classe à laquelle le résolveur s'est arrêté, les règles essayées, celles exécutées, etc.

Pour faciliter le processus d'élicitation, les auteurs ont également manifesté le besoin d'avoir des fonctionnalités d'auto-complétion. Cela leur permettrait en effet, au fur et à mesure du processus d'élicitation, d'avoir une idée des connaissances déjà explicitées et de pouvoir les réutiliser au besoin. Par exemple, ces fonctionnalités peuvent être très utiles lors de la description des problèmes par exemple : connaître les objets déjà définis pour pouvoir les réutiliser. Dans les problèmes additifs par exemple, où l'on a des listes de personnes, cela peut être utile lorsqu'on commence d'abord par définir les types d'objet *personne*, de pouvoir les réutiliser lorsqu'on définit les *listes de personnes*. La nouvelle version de AMBRE-KB, incluant ces différentes améliorations, a permis à nouveau de mettre à l'essai l'outil, dans une seconde expérimentation.

6.1.2 Seconde expérimentation

À partir de la version modifiée de AMBRE-KB, une seconde expérimentation a été menée.

6.1.2.1 Protocole expérimental

Pour des raisons pratiques, l'auteur est filmé pendant toute la durée de l'expérimentation. Le matériel utilisé se compose d'un ordinateur (avec souris et clavier) sur lequel est installé AMBRE-KB, une caméra, un magnétophone, un chronomètre et un questionnaire.

6.1.2.2 Déroulement

L'expérimentation s'est déroulée en quatre étapes.

Étape 1 : La première phase a consisté à présenter à l'auteur :

- Le principe de AMBRE ainsi que les méta-modèles de connaissances : la forme et le rôle des différents types de connaissances à acquérir.
- Le fonctionnement général de AMBRE-KB.

À la fin de cette étape, l'auteur est invité à commencer à réfléchir sur un domaine sur lequel il va tester l'outil.

Étape 2 : Lors de cette étape, l'auteur commence par choisir le domaine sur lequel il va tester AMBRE-KB. Il réfléchit ensuite aux types de problèmes qu'il souhaite faire résoudre à l'apprenant. Cette étape va en effet lui permettre de proposer un vocabulaire pour son domaine.

Une fois qu'il a proposé un vocabulaire pour son domaine, il est important que l'auteur réfléchisse également sur les règles et techniques. Cette étape est tout aussi importante que la précédente, parce qu'elle va nous permettre d'anticiper la phase de test des fichiers générés. En effet, le résolveur de AMBRE étant codé en Prolog, tous les modèles de connaissances doivent être générés en Prolog, de même que les toutes les fonctions utilisées par le résolveur pour appliquer une technique de résolution. Pour définir une technique pour un domaine, généralement, l'auteur a besoin de définir des fonctions spécifiques, comme par exemple : calculer la somme de deux nombres (les problèmes additifs par exemple), la racine carrée d'un nombre (les équations par exemple), etc. Pour que le résolveur puisse tester la résolution d'un problème il doit disposer de toutes les fonction (en Prolog) définies par l'auteur. Dans AMBRE-KB, l'auteur va définir la spécification de la fonction (description de la fonction ainsi que de ses paramètres) et nous nous chargeons de développer les fonctions correspondantes. À la fin de cette phase, l'auteur est prêt à utiliser AMBRE-KB pour éliciter les connaissances de son domaine.

Étape 3 : Lors de cette étape, l'auteur utilise en toute autonomie AMBRE-KB pour expliciter les différentes connaissances qu'il avait explicitées sur papier. Nous nous positionnons ainsi en retrait afin d'observer l'interaction de l'auteur avec le logiciel, tout en notant sur une fiche les temps passés sur chaque type de connaissance, les difficultés rencontrées au cours de l'utilisation de AMBRE-KB, les gestes et comportements non verbaux de l'auteur ainsi que ses remarques et questions. Lorsque les questions posées par l'auteur portent sur le fonctionnement de AMBRE-KB, nous l'envoyons dans un premier temps sur l'aide proposée. Toutefois, si l'aide n'est pas suffisante, nous lui donnons les indications nécessaires pour qu'il puisse poursuivre l'élicitation.

Lorsque nous remarquons que l'auteur n'avance plus dans le processus d'élicitation, nous considérons qu'il est bloqué ou éprouve des difficultés à comprendre le fonction-

nement de l'outil. Dans ce cas, nous lui posons une question pour savoir ce qu'il veut faire exactement. Si la réponse à cette question correspond à une fonctionnalité prise en compte par l'outil, nous le guidons en lui montrant comment faire. Si ce qu'il veut faire n'est pas pris en compte par l'outil, nous considérons que cette tâche ne peut être achevée, et dans ce cas il passe à l'élicitation des autres types de connaissances. Au terme de l'explicitation des connaissances, l'auteur peut générer les modèles de connaissances correspondants en utilisant AMBRE-KB. Ces différents modèles constituent les bases de connaissances de l'EIAH dont le résolveur a besoin pour résoudre les problèmes.

Étape 4 : Dans cette étape, l'auteur teste la résolution des problèmes dans AMBRE-KB. Deux cas sont possibles :

— Soit le résolveur parvient à résoudre un problème, et il affichera ainsi la classe du problème, les règles exécutées, la valeur des différents attributs et la solution du problème.

— Soit le résolveur ne parvient pas à résoudre un problème, et il affichera ainsi un message permettant à l'auteur de comprendre son erreur pour pouvoir ensuite la corriger.

Une fois que l'auteur a fini de tester la résolution des problèmes, il remplit un questionnaire portant sur le fonctionnement de AMBRE-KB : l'élicitation des différents types de connaissances, les interfaces proposées, l'assistance fournie par le système ainsi que le feedback, etc., ce qui conclut la séance. Les questionnaires (cf. Annexes) présentés aux auteurs contenaient 31 questions réparties en quatre grandes catégories :

— **Profil de l'auteur et domaine testé (2 questions)**.

— **Elicitation des différentes connaissances (17 questions)** : définition du vocabulaire, description des problèmes, élicitation des connaissances de classification, élicitation des connaissances de reformulation, élicitation des connaissances de résolution. Les auteurs disposaient de 10 questions, avec 3 degrés de compréhension pour donner leur impression : « *j'ai compris dès le début* », « *j'ai compris vers la fin* », et « *je n'ai pas compris* » ; 2 questions à réponses directes (*oui* ou *non*) avec la possibilité de commenter leurs réponses lorsqu'elles sont négatives ; 5 questions ouvertes leur permettant de s'exprimer sur les interfaces proposées et de proposer des suggestions d'amélioration.

— **Principes du processus interactif d'acquisition de connaissances** mis en œuvre dans AMBRE-KB. Les auteurs disposaient de 7 questions pour juger l'as-

sistance fournie par AMBRE-KB : 4 questions à choix multiples et 3 questions leur permettant de s'exprimer sur l'utilité et l'efficacité de l'assistance et de proposer des suggestions.

— **Évaluation générale de AMBRE-KB**. Les auteurs disposaient de 5 questions pour évaluer le fonctionnement général de AMBRE-KB : 2 questions à choix multiples et 3 questions ouvertes leur permettant de s'exprimer et de faire des suggestions.

6.1.2.3 Critères d'évaluation

Est-ce que AMBRE-KB permet d'éliciter les connaissances nécessaires à l'EIAH pour résoudre des problèmes ?

L'objectif de cette partie est de tester si AMBRE-KB permet d'éliciter toutes les connaissances nécessaires à l'EIAH pour résoudre des problèmes. Pour ce faire les facettes suivantes ont été identifiées : *définition du vocabulaire du domaine choisi, description des problèmes à résoudre, élicitation des connaissances de classification, élicitation des connaissances de reformulation et élicitation des connaissances de résolution.* Pour chacune d'elles nous avons considéré les éléments suivants :

— **Point à analyser** : est-ce-que AMBRE-KB permet à l'auteur d'expliciter toutes les connaissances nécessaires à chaque facette pour le domaine qu'il aura choisi ?

— **Méthodologie envisagée** : élicitation des connaissances avec AMBRE-KB par l'auteur, puis test de la résolution des problèmes avec le résolveur.

— **Condition de validation** : notre hypothèse est qu'AMBRE-KB permet d'éliciter toutes les connaissances nécessaires pour chaque facette à évaluer.

Le **Tableau 6.2** montre un exemple des critères d'évaluation(cf. Annexes) du vocabulaire pour un domaine donné.

Facette à évaluer	Définition du vocabulaire pour le domaine choisi.
Point à analyser	Est-ce que AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi ?
Méthodologie envisagée	Définition du vocabulaire de son domaine par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire.
Condition de validation	AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi.

Tableau 6.2 – Critère retenu pour évaluer le vocabulaire d'un domaine.

Afin d'évaluer les principes du processus interactif d'acquisition des connaissances mise en œuvre dans AMBRE-KB, nous avons identifié trois facettes à évaluer : flexibilité, guidage et contrôle des connaissances pendant l'élicitation des connaissances, puis feedback à la fin du processus d'élicitation. Pour chacune de ces facettes, nous avons considéré les éléments suivants :

— **Point à analyser** : est-ce que l'aide fournie est utile pour l'auteur ?

— **Méthodologie envisagée** : élicitation des connaissances avec AMBRE-KB par l'auteur, puis réponse aux questions concernant l'aide dans le questionnaire.

— **Condition de validation** : l'aide est utile et a aidé l'auteur.

Le **Tableau 6.3** montre un exemple des critères d'évaluation de la flexibilité offerte par AMBRE-KB lors de la définition des connaissances.

Facette à évaluer	Flexibilité offerte par AMBRE-KB.
Point à analyser	Est-ce que la flexibilité fournie à l'auteur est utile durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire.
Condition de validation	La flexibilité proposée dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation.

Tableau 6.3 – Critères retenus pour évaluer la flexibilité de AMBRE-KB.

1.2.2.2. Est-ce que les interfaces proposées dans AMBRE-KB sont assez intuitives ?

L'objectif dans cette partie est de tester la satisfaction de l'auteur : est-ce-que les interfaces proposées sont assez intuitives ? Nous optons pour une évaluation empirique qui consiste à observer l'auteur lors de l'utilisation du logiciel, et à recueillir ses impressions à travers un questionnaire.

À l'image de l'évaluation décrite en section 1.2.2, différentes facettes ont également été identifiées pour cette évaluation : *définition du vocabulaire du domaine choisi*, *description des problèmes à résoudre*, *élicitation des connaissances de classification*, *élicitation des connaissances de reformulation* et *élicitation des connaissances de résolution*. Pour chacune d'elles, nous avons ainsi considéré les éléments suivants :

— **Point à analyser** : ergonomie des interfaces proposées.

— **Méthodologie envisagée** : élicitation des connaissances avec AMBRE-KB par l'auteur, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)

— **Condition de validation** : l'auteur a pu définir le type de connaissances sans aide extérieure.

Le **Tableau 6.4** montre par exemple les critères d'évaluation de la fonctionnalité de définition du vocabulaire pour un domaine donné.

Facette à évaluer	Définition du vocabulaire pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour la définition du vocabulaire
Méthodologie envisagée	Définition du vocabulaire par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir le vocabulaire sans aide extérieure

Tableau 6.4 – Critères d'évaluation de l'ergonomie de l'interface permettant de définir le vocabulaire

6.1.2.4 Résultats

Quatre domaines ont été testés, mais 5 sujets ont participé à l'expérimentation. L'un d'entre eux a testé deux domaines (*Pluriel des noms* et *Formation des adverbes en "ment"*) et les deux autres domaines ont été testés chacun par deux sujets (cf. **Tableau 6.5**).

	Domaines			
	Équation	Calcul d'aire	Pluriel des noms	Adverbes en "ment"
Nombres de sujets	2	2	1	1

Tableau 6.5 – Domaines testés

Nous présentons d'abord les résultats des domaines testés par deux sujets à la fois, nous terminons par les autres domaines testés par le même sujet.

Le **Tableau 6.6** décrit le nombre de connaissances produites par des deux sujets pour la résolution des équations.

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	4	3
Classes	6	7
Règles	5	5
Techniques	4	4
Problèmes créés	20	12
% Problèmes résolus au total sans le feedback	75%	41%
% Problèmes résolus au total avec le feedback	100%	100%

Tableau 6.6 – Résultats des expérimentations pour les équations par les sujets 1 et 2

Pour la résolution des équations, les deux sujets ont défini des connaissances différentes : ils ont proposé des arbres de classification différents, ainsi que des vocabulaires différents. Le fait que les vocabulaires soient différents explique le fait que les problèmes à résoudre, les règles et les techniques de résolution ne sont pas définies de la même façon.

Les **Tableaux 18** et **6.8** par exemple montrent le vocabulaire défini respectivement par le sujet 1 et le sujet 2. Le système a pu résoudre 75% des problèmes définis par le sujet 1 sans résolveur, puis 100% en utilisant le feedback. Pour le sujet 2, le système a pu résoudre 41% sans utiliser le feedback puis 100% en utilisant le résolveur.

Types objets	**Caractéristiques objets**
Équation	Nom_inconnu –>chaine de caractères Valeur_inconnu–>nombre entier Liste_termes–>type énuméré(deux_termes,trois_termes)
Deux_termes	Premier_terme–>terme_inconnu Deuxième_terme –>terme_constante
Terme_inconnu	coefficient–>nombre entier
Terme_constante	valeur_constante–>nombre entier

Tableau 6.7 – Vocabulaire défini par le sujet 1.

Types objets	**Caractéristiques objets**
Équation	Terme_1–>type énuméré(terme_inconnu,terme_constante) Terme_2–>type énuméré(terme_inconnu,terme_constante) Terme_3–>type énuméré(terme_inconnu,terme_constante)
Terme_inconnu	Coefficient–>nombre entier Nom_variable–>chaine de caractères Degré–>nombre entier Valeur–>nombre entier
Terme_constante	valeur_constante –> nombre entier

Tableau 6.8 – Vocabulaire défini par le sujet 2.

Le **Tableau 22** décrit le nombre de connaissances produites par deux sujets pour le calcul d'aire en géométrie.

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	5	8
Classes	12	12
Règles	11	11
Techniques	7	7
Problèmes créés	28	18
% Problèmes résolus au total sans le feedback	57%	61%
% Problèmes résolus au total avec le feedback	85%	89%

Tableau 6.9 – Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2

Pour le calcul d'aire en géométrie, même si les sujets ont défini des arbres de classification avec le même nombre de classes, les classifications définies sont différentes.

Les **Figures 6.1** et **6.2** représentent une partie des arbres de classifications définis par les deux auteurs :

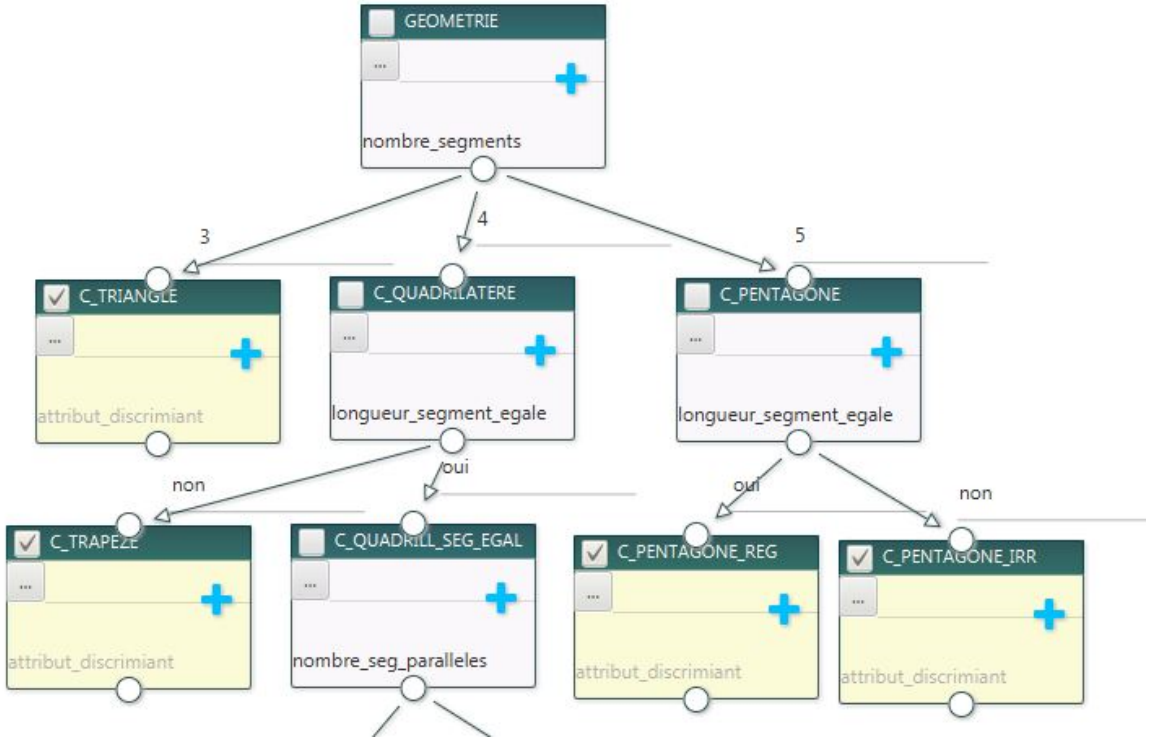

FIGURE 6.1 – Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie

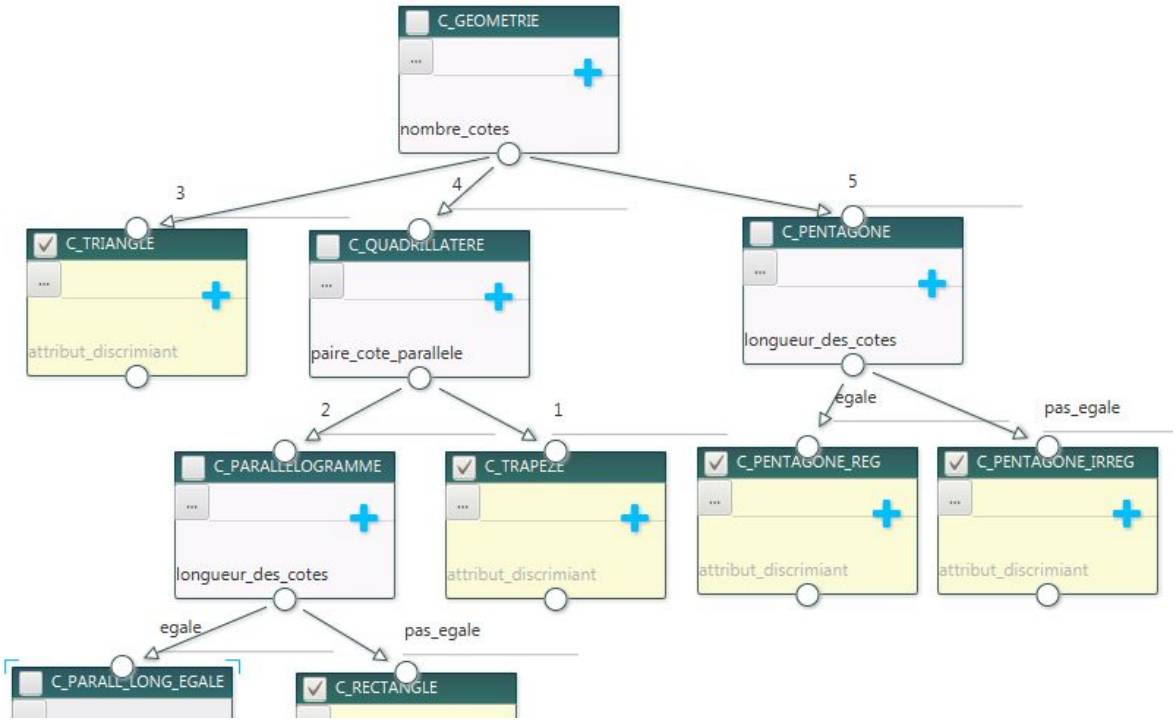

FIGURE 6.2 – Une partie de l'arbre de classification définie par le sujet 2 pour le calcul d'aire en Géométrie

Les sujets ont également défini des vocabulaires différents, et donc des règles et techniques de résolution différentes. Pour le sujet 1, le système a pu résoudre 57% des problèmes définis sans utiliser le feedback, et 85% en utilisant le feedback. Les 15% de problèmes qui ont échoué appartenaient à la même classe. L'auteur n'a pas pu tester leur résolution parce qu'il a défini au dernier moment une nouvelle fonction dont il avait besoin pour décrire la technique associée à cette classe. Comme le résolveur ne disposait pas de la version Prolog de cette fonction, la résolution de ces problèmes a achoué. Pour rappel, lors de la première étape de l'expérimentation, l'auteur doit donner la description de toutes les fonctions dont il aura besoin, de sorte que l'on puisse définir une version Prolog utilisable par le résolveur lors de la résolution des problèmes. Pour le sujet 2, le système a pu résoudre 61% des problèmes sans le feedback, et 89% avec le feedback. 11% de problèmes ont échoué parce que le système n'arrivait pas à trouver de règles lui permettant de conclure sur la valeur d'un attribut discriminant, mais comme l'auteur n'a pas corrigé l'erreur, la résolution de ces problèmes a échoué.

Le tableau **6.10** résume les connaissances décrites pour le pluriel des noms et la formation des adverbes en "ment" testés par le même sujet.

	Pluriel	Adverbes
vocabulaire (types d'objets)	1	1
Classes	13	17
Règles	9	16
Techniques	8	11
Problèmes créés	40	25
% Problèmes résolus au total sans le feedback	66%	87%
% Problèmes résolus au total avec le feedback	90%	100%

Tableau 6.10 – Résultats des expérimentations pour le pluriel des noms et la formation des adverbes en "ment" testés par le même sujet

Le pluriel des noms et la formation des adverbes ont été testés par le même sujet. Il a d'abord commencé par la formation des adverbes et a terminé par le pluriel des noms. Pour le premier domaine, le système a pu résoudre 52% des problèmes sans utiliser le feedback, puis 80% en utilisant le résolveur. Pour les 20% autres problèmes, le sujet a affirmé qu'il avait compris les messages d'erreur mais il a volontairement décidé de ne pas les corriger et de s'en tenir aux problèmes résolus.

Pour le pluriel des noms, le système a pu résoudre 87% des problèmes sans utiliser le feedback, mais une fois qu'il a utilisé le feedback, il a pu résoudre le reste des problèmes.

6.1.2.5 Analyse des résultats

Les 5 auteurs ont affirmé avoir pu définir toutes les connaissances nécessaires à l'EIAH pour résoudre les problèmes du domaine qu'ils avaient choisi. Ils ont également affirmé que les interfaces leur convenaient dans l'ensemble et étaient faciles à utiliser. Toutefois, l'un d'entre eux a quand même affirmé avoir rencontré quelques difficultés au début de l'élicitation des connaissances de reformulation car, selon lui, les interfaces permettant de définir les règles ne sont pas très intuitives au début, mais une fois le principe compris il devient facile de définir les connaissances de reformulation.

Concernant la définition du vocabulaire, deux des auteurs ont observé une différence entre le vocabulaire qu'ils avaient défini sur papier au début de l'expérimentation et celui qu'ils ont finalement explicité en utilisant l'outil. Cela s'explique par le fait que AMBRE-KB ne leur permettait pas de représenter les connaissances telles qu'ils les avaient explicitées. Néanmoins, avec une petite modification de la version papier,

ils ont pu parvenir à définir un vocabulaire pour leur domaine avec les fonctionnalités déjà implémentées dans AMBRE-KB. Un des auteurs a affirmé que la spécification de la question à résoudre lors de la description d'un problème n'était pas très intuitive et qu'il préférerait avoir un menu explicite pour la définir.

Quatre des auteurs ont déclaré avoir utilisé l'aide proposée. Et tous ont jugé que l'assistance proposée les a aidés durant le processus d'élicitation. Ils ont particulièrement apprécié le feedback proposé par l'outil et ont affirmé son aide pour mieux comprendre le fonctionnement de l'outil. Néanmoins, l'un d'entre eux a déclaré que cette assistance pourrait être enrichie en augmentant l'interactivité du système avec des pop-ups plus fréquentes durant le processus.

6.2 Bilan des expérimentations

Nous avons mené deux expérimentations avec 8 sujets pour mettre à l'essai AMBRE-KB. La première expérimentation a été réalisé avec 3 sujets et a porté un domaine déjà existant (EIAH AMBRE existant au moment de l'expérimentation) et deux nouveaux domaines : le calcul d'aire en géométrie et la pâtisserie. L'objectif de cette expérimentation était de mettre à l'essai AMBRE-KB afin de tester si AMBRE-KB était capable d'acquérir les connaissances nécessaires à un EIAH AMBRE pour résoudre des problèmes. Les résultats de cette expérimentation n'ont pas permis de valider notre hypothèse de départ, mais elle s'est conclue avec des remarques intéressantes qui ont permis d'étendre l'outil AMBRE-KB pour améliorer l'interactivité du système, et intégrer des fonctionnalités permettant à l'auteur d'avoir un feedback sur le processus d'élicitation et la résolution des problèmes.

Avec la nouvelle version de AMBRE-KB, l'outil a été mis à l'essai dans une seconde expérimentation. Cette-ci a été réalisée par 5 sujets et quatre domaines ont été testés : la résolution des équations du second degré (testé par deux sujets), le calcul d'aire en géométrie (testé par deux sujets), le pluriel des noms et la formation des adverbes en "ment", testés par le même sujet. De cette expérimentation, nous retenons les conclusions suivantes :

— Pour un domaine donné, AMBRE-KB permet de générer les modèles de connaissances qui permettent au résolveur de résoudre les problèmes.

— Les interfaces proposées sont dans l'ensemble faciles à utiliser.

— Les auteurs ont apprécié l'assistance fournie, en particulier le feedback sur le processus d'élicitation.

— Pour tous les domaines, au moins 40% des problèmes ont pu être résolus par sans utiliser le feedback, et au moins 80% en utilisant le feedback (cf. **Tableau 6.11**).

	Équation		**Calcul d'aire**		**Pluriel**	**Adverbes**
	Sujet 1	Sujet 2	Sujet 1	Sujet 2	Sujet	Sujet
sans le feedback	75%	41%	57%	61%	66%	87%
avec le feedback	100%	100%	85%	89%	90%	100%

Tableau 6.11 – Domaines testés

Parmi les sujets de cette deuxième expérimentation, deux d'entre eux sont des informaticiens, et trois des non-informaticiens. Les premiers ont rencontré moins de difficulté pour comprendre les principes mis en œuvre dans l'outil. Cependant, tous les sujets ont pu, sans programmer, éliciter les connaissances nécessaires à l'EIAH pour résoudre les problèmes dans le domaine qu'ils avaient choisi. Nous pouvons donc en conclure que AMBRE-KB est apte à acquérir les connaissances nécessaires à un EIAH AMBRE pour résoudre les problèmes portant sur la méthode à enseigner. L'analyse des questionnaires nous permet également d'affirmer que les interfaces de AMBRE-KB sont dans l'ensemble assez intuitives et facile à appréhender. Celle permettant de définir des connaissances de reformulation pourrait cependant être améliorées, de même que celle destinée à la description des problèmes.

Le fait que les auteurs aient pu éliciter toutes les connaissances dont ils avaient besoin pour la méthode qu'ils souhaitaient enseigner permet également de conclure que les méta-modèles sous-jacents à AMBRE-KB sont suffisants, même s'il est possible qu'en utilisant AMBRE-KB pour d'autres domaines, l'on constate qu'il est nécessaire de les enrichir.

Conclusion générale

Dans cette dernière partie du manuscrit, nous présentons d'abord une synthèse de nos contributions. Nous discutons ensuite de ces contributions ainsi que des limites soulevées par ce travail. Nous terminons enfin en présentant les différentes perspectives de recherche qui pourraient être intéressantes pour la poursuite de ce travail.

Sommaire

6.3 Synthèse de nos travaux 157
6.4 Discussion 159
6.5 Perspectives 161

6.3 Synthèse de nos travaux

Dans cette thèse, nous nous sommes intéressés à la problématique de l'acquisition des connaissances dans le cadre de la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes dans des domaines spécifiques. L'objectif de notre travail a été de réduire le coût de conception de ces EIAH, dû en partie à l'élicitation des connaissances, mais également de permettre à des auteurs de pouvoir éliciter, sans programmer, les connaissances permettant à l'EIAH de résoudre des problèmes. Notre problématique s'est donc décomposée en deux points :

— **Comment permettre à un auteur d'éliciter, sans programmer, les connaissances permettant au système de résoudre des problèmes dans un domaine donné ?**

— **Comment accompagner l'auteur dans le processus d'élicitation ?**

Pour répondre à ces questions, notre première contribution a consisté à formaliser les différents types de connaissances qui interviennent dans la résolution d'un problème, dans le cadre d'un EIAH destiné à enseigner des méthodes. Pour cela, nous avons proposé des méta-modèles de connaissances. Ceux-ci sont décrits par des schémas XML et permettent de définir les connaissances à acquérir. Une fois les différents types de connaissances formalisés, nous avons ensuite proposé un processus interactif d'acquisition de ces connaissances. Ce processus, indépendamment du domaine, est fondé sur les méta-modèles et permet à un auteur de construire les modèles de connaissances spécifiques pour le domaine qu'il aura choisi.

Nous avons par la suite mis en œuvre ce processus en implémentant l'outil auteur AMBRE-KB. Cet outil permet non seulement d'acquérir les connaissances nécessaires à la résolution des problèmes dans un domaine, mais également d'assister l'auteur dans ce processus, en lui offrant une grande flexibilité lors de la définition des connaissances, en le guidant et en lui permettant de tester les connaissances durant le processus d'élicitation s'il le souhaite. Il offre également la possibilité à l'auteur d'avoir un feedback sur le processus d'élicitation en lui permettant de tester la résolution des problèmes. Dans le cas où la résolution d'un problème a réussi, le système lui affiche les différentes informations utilisées pour le résoudre. Ces informations sont notamment la classe du problème, les attributs calculés, les règles essayées, celles exécutées, les valeurs des différents attributs discriminants impliqués, la technique de résolution de résolution appliquée, ainsi que la solution du problème. Lorsque la résolution échoue, le système envoie un message

d'erreur afin de permettre à l'auteur de comprendre l'origine de son erreur et de pouvoir par la suite la corriger.

Enfin, nous avons évalué le processus d'acquisition des connaissances et sa mise en 'uvre dans AMBRE-KB en menant deux expérimentations. Ces expérimentations ont consisté dans un premier temps à vérifier que les modèles de connaissances générés par AMBRE-KB permettent à l'EIAH de résoudre des problèmes pour un domaine donné. Les résultats de cette expérimentation ont permis d'améliorer l'outil auteur, de sorte à pouvoir effectuer une deuxième expérimentation. Cette dernière a permis de mettre en évidence les points forts et les points faibles de AMBRE-KB. Ainsi, de cette expérience, nous retenons les conclusions suivantes :

— AMBRE-KB permet d'acquérir les connaissances nécessaires au résolveur pour résoudre les problèmes portant sur la méthode à enseigner.

— Les interfaces proposées sont assez intuitives et facile à utiliser dans l'ensemble, même si celles dédiées à l'élicitation des règles et à la description des problèmes peuvent être améliorées.

— L'assistance fournie durant le processus est utile. Le feedback proposé est particulièrement apprécié et contribue à faciliter la compréhension du fonctionnement de l'outil.

Les propositions effectuées dans le cadre de cette thèse et leur mise œuvre opérationnelle répondent donc à notre problématique générale. Nous restons convaincus de l'utilité du processus d'acquisition de connaissances proposé dans la conception d'EIAH destinés à enseigner des méthodes de résolution de problèmes. De plus, l'approche proposée pour la description des problèmes, et consistant à proposer un vocabulaire, qui est ensuite utilisé pour décrire des problèmes peut être réutilisé dans tout domaine où on a besoin de représenter des problèmes. En effet, nous partons du principe que dans un problème, on retrouve généralement les éléments descriptifs de l'énoncé, ainsi que la question à résoudre. Ce qui pourrait différencier les problèmes dans des domaines différents, ce sont les différents types de connaissances qu'on peut retrouver au niveau des éléments descriptifs de l'énoncé. Dans AMBRE, par exemple, dans les éléments descriptifs de l'énoncé, on retrouve les traits de surface du problème ; dans [84], les problèmes sont décrits en termes de variables, d'unités de mesures, de formules. Il s'agira donc, selon les domaines, de faire évoluer le méta-modèle du vocabulaire que nous avons proposé pour prendre en compte les types de connaissances nécessaires pour décrire les

problèmes de ce domaine.Nous soutenons également que le fait que les connaissances puissent être explicitées sans programmer, motivera plus d'auteurs (enseignants, professionnels, pédagogues plus experts) à davantage s'impliquer dans la conception de ce type de logiciels.

6.4 Discussion

Positionnement en Ingénierie des Connaissances
D'un point de vue Ingénierie des Connaissances, nous nous positionnons dans une démarche d'ingénierie dirigée par les modèles comme la méthodologie commonKADS. Cette méthodologie propose de construire un modèle conceptuel autour de cinq modèles génériques qui décrivent comment résoudre des problèmes très généraux tels que diagnostic et prédiction du comportement, évaluation, conception, configuration, planification, affectation et ordonnancement. Dans le domaine des EIAH, ces types de tâches très génériques ne sont pas forcément adaptés à ce que l'on cherche à enseigner. En effet, dans le domaine des EIAH, pour modéliser les connaissances du domaine, il ne s'agit pas de modéliser les connaissances de l'expert, mais les connaissances telle qu'elles devraient fonctionner chez l'apprenant après l'apprentissage. D'autre part, en EIAH, il faut modéliser d'autres types de connaissances, comme les connaissances de diagnostic des compétences de l'apprenant, ou les connaissances permettant de fournir des rétroactions ou de l'assistance. La démarche que nous proposons même si elle est similaire à celle de commonKADS s'applique à des problématiques plus spécifiques au champ des EIAH.

Positionnement par rapport aux outils auteurs existants en EIAH
Les Model Tracing Tutors développés à l'Université Carnegie Mellon de Pittsburgh sont les plus proches de notre problématique. En effet, même si les modèles sous-jacents sont très différents, l'approche centrée connaissances du projet AMBRE s'inscrit dans le même courant. En effet, les EIAH développés à Pittsburgh se fondent sur le modèle ACT-R qui considère que la tâche de résolution de problèmes s'effectue pas à pas, un élément de connaissance étant mobilisé à chaque pas de résolution, et que l'ensemble des connaissances du domaine peut s'exprimer sous forme de règles de production. Ce modèle assez restrictif limite les domaines sur lequel on peut concevoir des Model Tracing Tutors. L'enseignement de méthodes propose un modèle cognitif dans lequel l'apprenant raisonne globalement sur le problème à résoudre. C'est également un modèle restrictif

puisqu'il ne s'applique qu'aux domaines dans lesquels on peut établir une classification des problèmes et des techniques de résolution, mais il pourrait compléter l'offre de CTAT. Une des perspectives de cette thèse peut d'ailleurs consister à intégrer AMBRE-KB dans la plateforme CTAT. Comme le souligne Nathalie Guin [41], le succès des tuteurs intelligents et des outils auteurs CTAT permettant de créer les model tracing tutors renforce notre conviction qu'une approche centrée connaissances est souhaitable en EIAH.

Évaluation et validation des méta-modèles

Les méta-modèles de connaissances proposés n'ont pas été validés de manière formelle. Les méta-modèles portant sur la méthode à enseigner (méta-modèles des connaissances de classification, connaissances de reformulation et connaissances de résolution) ont été évalués en créant, pour chaque domaine testé, des modèles de connaissances fondés sur ces méta-modèles et utilisables par le résolveur pour tester la résolution des problèmes du domaine. Le fait que le résolveur puisse résoudre des problèmes en utilisant ces modèles assure donc leur cohérence. Nous ne prétendons pas que AMBRE-KB permet de représenter n'importe quel type de domaine, mais il permet d'acquérir les connaissances nécessaires à la conception d'EIAH AMBRE.

Positionnement par rapport à la méthodologie utilisée

Notre travail a impliqué des choix de formalisation et de méthode qui relèvent de prises de décisions de notre part. Nous souhaitions permettre à des auteurs de pouvoir concevoir eux-mêmes des EIAH dans les domaines qui les intéressent sans aucune compétence en programmation. L'approche que nous avons adoptée est de leur proposer des interfaces fondées sur les méta-modèles des connaissances à expliciter pour les assister durant tout le processus d'élicitation. D'autres approches ont préféré utiliser des techniques d'IA comme l'apprentissage automatique pour découvrir des connaissances. Même si les techniques d'IA peuvent s'avérer efficaces, notre objectif était plus d'impliquer les enseignants dans la conception de ces EIAH. Il serait donc nécessaire que les enseignants puissent comprendre de tels processus d'apprentissage automatique, de sorte qu'ils puissent valider les connaissances à acquérir par l'apprenant. De plus, cela ne faciliterait pas la confiance et l'appropriation des EIAH par les enseignants, qui pour la plupart ont encore une réaction méfiante vis-à-vis de ces outils. Les techniques d'apprentissage automatiques peuvent néanmoins s'avérer efficaces si elles sont intégrées à un processus interactif d'élicitation des connaissances, comme nous l'avons vu dans la

thèse de Sébastien Lallé [49].

6.5 Perspectives

Ce travail s'est focalisé sur l'acquisition des connaissances permettant la résolution des problèmes pour un domaine donné. L'une de nos perspectives à court terme est d'intégrer dans AMBRE-KB l'élicitation des connaissances destinées à accompagner l'apprenant, en lui fournissant aide, diagnostic et explications sur ses erreurs. Nous avons regroupé ces connaissances en trois grandes catégories : les connaissances d'aide, les connaissances de diagnostic, et les connaissances d'explications sur les erreurs. Pour chacune de ces connaissances, nous avons déjà proposé des méta-modèles qui permettent de définir la forme des connaissances à acquérir. Cependant, ces méta-modèles n'ont pas été testés dans le cadre de ce travail, puisque nous nous sommes focalisés sur les connaissances de la méthode dans le cadre du développement de AMBRE-KB. Néanmoins, l'élicitation desdites connaissances est déjà prise en compte dans le processus d'acquisition proposé. Il reste donc à valider ces méta-modèles, et à intégrer dans AMBRE-KB les fonctionnalités qui permettront d'éliciter ces connaissances. Il s'agira donc dans un premier temps de proposer des interfaces qui permettront à l'auteur de pouvoir éliciter ces connaissances, et dans un second temps, comme pour les connaissances de la méthode, de proposer à l'auteur des fonctionnalités d'assistance pour faciliter le processus d'élicitation, mais également des moyens de tester ces connaissances, une fois explicitées.

Les connaissances destinées à guider l'apprenant sont fortement dépendantes des connaissances sur la méthode à enseigner, mais également des étapes de résolution. Dans le processus d'acquisition de connaissances que nous avons proposé, pour gérer cette articulation entre la conception de l'interface de l'EIAH et l'élicitation de ces connaissances, nous avons proposé à l'auteur de concevoir l'interface et en particulier les différentes tâches de résolution des problèmes, le développement de cette interface étant ensuite réalisé par un informaticien. Une perspective pourrait être d'intégrer dans AMBRE-KB la possibilité pour l'auteur de construire lui-même cette interface en utilisant des composants graphiques via le drag and drop. L'approche utilisée dans CTAT avec les example-tracing-tutors [4], qui permet à un enseignant de concevoir l'interface par du drag and drop, pourrait être intéressante dans le cadre de ce travail.

Dans AMBRE, pour le moment, les connaissances représentées dans les règles sont traitées comme des conjonctions (ET). Il serait également intéressant de faire évoluer le résolveur de sorte qu'il puisse, par exemple, raisonner sur des connaissances représentant des disjonctions (OU) au niveau des prémisses des règles. Dans la même lancée, au niveau de l'élicitation des connaissances, il serait intéressant aussi de permettre à l'auteur de pouvoir définir des contraintes, des connaissances de comparaison (par exemple de plus que, de moins que), des connaissances contraires, etc.

En outre, dans la version actuelle de AMBRE-KB, avant d'entamer l'élicitation des connaissances, l'auteur doit définir la spécification de toutes les fonctions dont il a besoin pour décrire des règles et des techniques de résolution ; celles-ci sont ensuite traduites en Prolog pour permettre au résolveur de pouvoir les utiliser pour la résolution des problèmes ; cela peut constituer un frein à l'auteur car il a aura toujours besoin d'un expert Prolog pour pouvoir tester la résolution des problèmes qu'il a définis. Il a sera certes très difficile d'anticiper sur cela puisque c'est l'auteur qui spécifie les fonctions dont il a besoin pour la méthode qu'il souhaite enseigner. Néanmoins, intégrer dans AMBRE-KB un ensemble de fonctions utilitaires (par exemple des fonctions mathématiques permettant de calculer la somme, multiplication, division de nombres, etc, ou encore des fonctions permettant de concaténer/supprimer des chaînes de caractères) pourrait constituer une première étape pour faciliter le processus d'élicitation pour certains domaines.

Pour poursuivre le travail lié à l'assistance apportée à l'auteur, une piste à explorer est celle de la généralisation de donnaissances à partir d'exemples. On pourrait ainsi envisager une acquisition interactive des connaissances, l'auteur définissant des exemples à la place de connaissances abstraites, le système lui proposant ensuite une généralisation des connaissances qu'il pourrait accepter ou modifier. Par exemple, pour la définition des connaissances de classification, cela faciliterait une partie du travail de l'auteur en lui permettant de construire la hiérarchie à partir d'un ensemble de classes opérationnelles. Dans ce cas, l'auteur définirait uniquement des cas (des problèmes avec leur technique de résolution), préciserait les attributs discriminants et ceux qui instancient le problème par rapport à la classe, et le système construirait ensuite la hiérarchie de classes. Ce procédé est un premier exemple de généralisation de connaissances qui pourrait être mis en place. Un autre exemple peut consister à s'inspirer de l'approche utilisée dans SimStudent [51] qui repose sur la généralisation de connaissances abstraites à partir d'exemples.

Annexes

Connaissances décrites pour les Domaines testés

Nous avons décrit dans le chapitre 6 les différents domaines testés pour évaluer AMBRE-KB. Nous détaillons ici les connaissances décrites par les sujets.

Pluriel des noms terminés par "al","ail","ou", "eu", "eau" en français

Profil du sujet : Infographiste (non programmeur).

Méthode à enseigner : l'expert à choisi d'enseigner la formation du pluriel des noms terminés par "al", "ail", "ou", "eu, "au", "eau" en français. Le **Tableau 3.2.3** résume la méthode à enseigner.

Le sujet a défini :

- 1 type d'objet pour le vocabulaire
- 13 classes, dont 8 sont opérationnelles
- 9 règles
- 8 techniques de résolution
- 24 problèmes à résoudre
- 11 fonctions intervenant dans la définition des règles et des techniques de résolution
    - **Concatener_chaine** : permettant de concaténer des chaines de caractères

— **Pluriel_ail_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **Pluriel_al_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **Pluriel_ou_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **Pluriel_eu_au_eau_exception** : permettant de former le pluriel d'un mot en ail qui est une exception (en ajoutant aux au radical du mot)

— **liste_exception_nom_ail** : renvoie la liste des mots terminés en « ail » qui sont des exceptions

— **liste_exception_nom_al** : renvoie la liste des mots terminés en « al » qui sont des exceptions

— **liste_exception_nom_ou** : renvoie la liste des mots terminés en « ou» qui sont des exceptions

— **liste_exception_nom_eu** : renvoie la liste des mots terminés en « eu » qui sont des exceptions

— **liste_exception_nom_au** : renvoie la liste des mots terminés en « au » qui sont des exceptions

— **liste_exception_nom_eau** : renvoie la liste des mots terminés en « ail » qui sont des exceptions

Vocabulaire proposé pour le domaine

Types objets	**Caractéristiques objets**
nom	identifiant–>chaine de caractères fin_nom–>chaine de caractères pluriel_nom–>chaine de caractères

Tableau 12 – vocabulaire proposé pour le pluriel noms terminés par "al", "ail", "ou", "eu, "au", "eau" en français

Définition du vocabulaire en utilisant AMBRE-KB

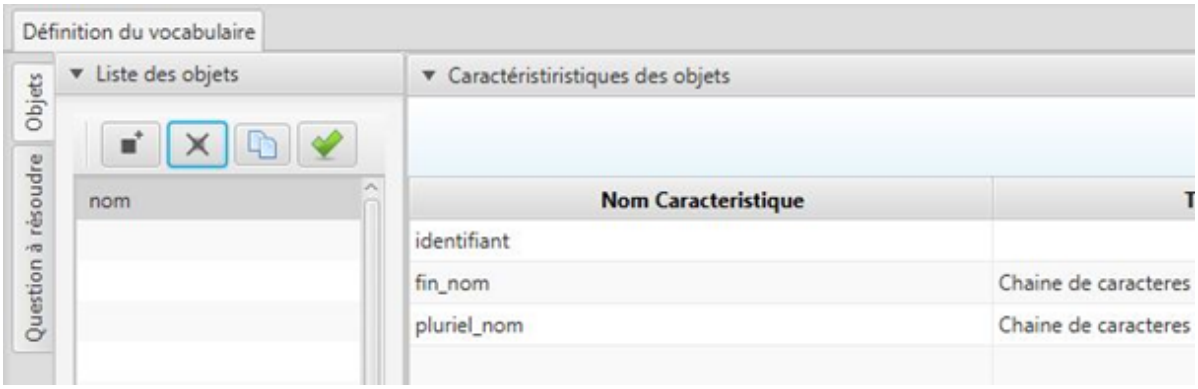

FIGURE 3 – Définiiton du vocabulaire pour le pluriel des noms avec AMBRE-KB

Problèmes à résoudre

Exemples de problèmes définis par le sujet

pluriel_detail : "Donner le pluriel du nom : (détail).
pluriel_bail : "Donner le pluriel du nom : (bail).
pluriel_travail : "Donner le pluriel du nom : (travail).
pluriel_bal : "Donner le pluriel du nom : (bal).
pluriel_chacal : "Donner le pluriel du nom : (chacal).
pluriel_jeu : "Donner le pluriel du nom : (jeu).
pluriel_bureau : "Donner le pluriel du nom : (bureau).
pluriel_joyau : "Donner le pluriel du nom : (joyau).
pluriel_pou : "Donner le pluriel du nom : (pou).

Exemple de définition d'un problème en utilisant AMBRE-KB

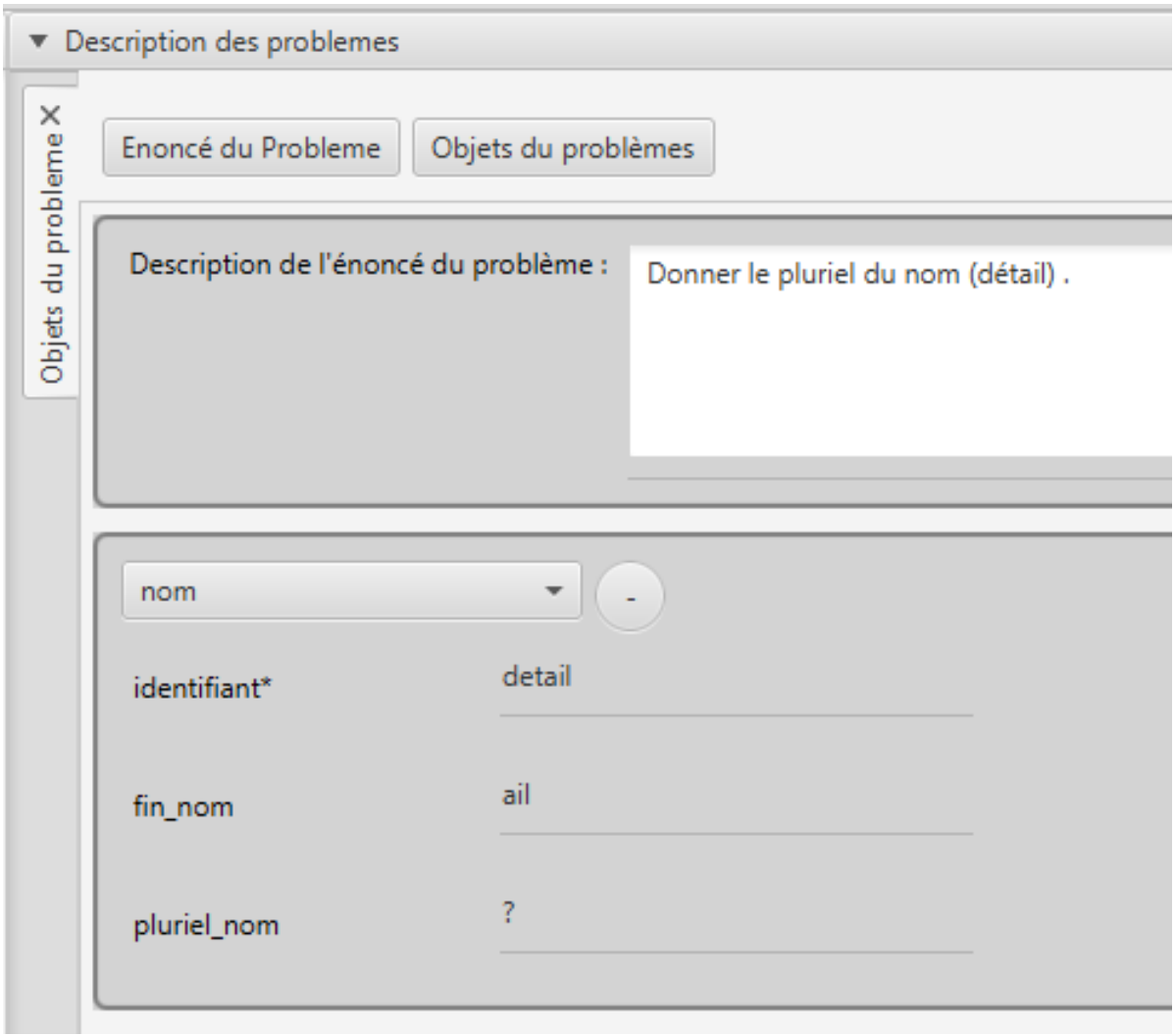

FIGURE 4 – Définiton du problème pluriel_detail avec AMBRE-KB

Exemple de modèles générés de connaissances par AMBRE-KB

```
/**Modèle descriptif représentant le problème pluriel_detail**/
fait(probleme(pluriel_bail)). /**soit le problème pluriel_bail**/
fait(enonce_naturel(pluriel_bail,'Donner le pluriel du nom (bail).')).
/**soit l'énoncé du problème pluriel_bail**/
fait(est_un(bail,nom)).**soit bail un nom**/
fait(fin_nom(bail,ail)). /**soit ail la fin du nom bail**/
fait(pluriel_nom(bail,Question)). /**soit Question le pluriel du nom bail**/
fait(inconnu(pluriel_bail,Question)). /**la question a résoudre consiste
/**à trouver la valeur de Question**/
```

Tableau 13 – Modèle descriptif du problème pluriel_bail généré par AMBRE-KB

Arbre de classification

.0.0.1 Définition de l'arbre de classification avec AMBRE-KB

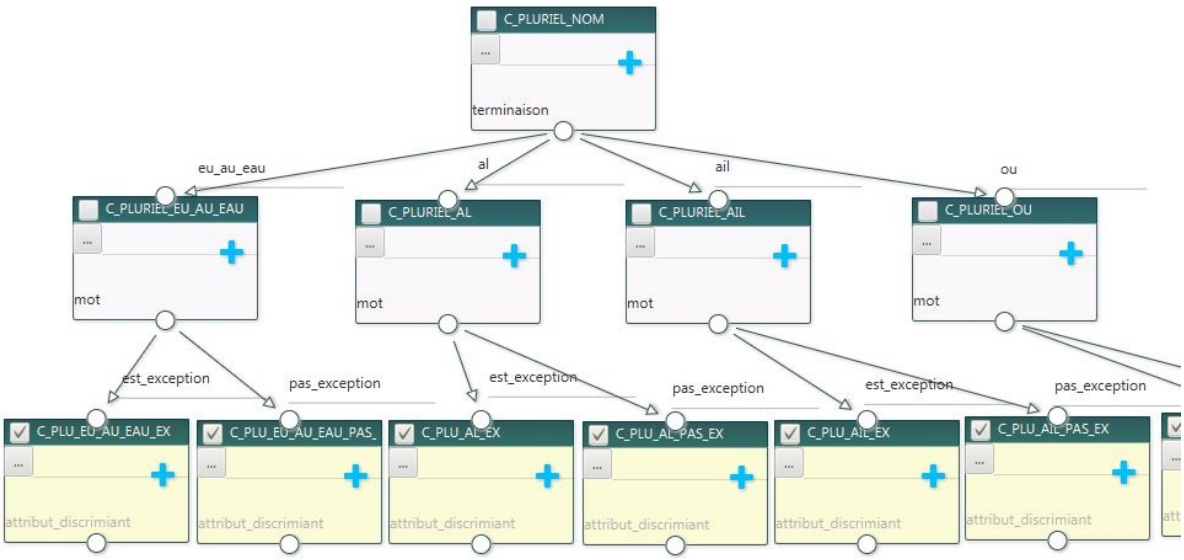

FIGURE 5 – Une partie de l'arbre de classification pour le pluriel des noms en "al", "ail", "ou", "eu, "au", "eau"

Modèle décrivant les connaissances de classification, généré par AMBRE-KB

```
racine_arbre(c_pluriel_nom). /*classe recine*/
statut(c_pluriel_nom,non_operationnelle). /* statut de la classe*/
test_fils(c_pluriel_nom,P,terminaison(P,_)).
attribut(c_pluriel_nom,P,terminaison(P,_),terminaison,[regle_terminaison]).
attribut(c_pluriel_nom,P,mot(P,_),mot,
        [regle_terminaison,regle_ail_exception,
        regle_ail_pas_exception,regle_al_exception,regle_al_pas_exception,
        regle_eu_au_eau_exception,regle_eu_au_eau_pas_exception,
regle_ou_exception,regle_ou_pas_exception]).
valeur_fils(c_pluriel_nom,P,c_pluriel_eu_au_eau) :-
        eval_prem(terminaison(P,[eu_au_eau])).
valeur_fils(c_pluriel_nom,P,c_pluriel_al) :-
        eval_prem(terminaison(P,[al])).
valeur_fils(c_pluriel_nom,P,c_pluriel_ail) :-
        eval_prem(terminaison(P,[ail])).
valeur_fils(c_pluriel_nom,P,c_pluriel_ou) :-
        eval_prem(terminaison(P,[ou])).
infos(c_pluriel_nom) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_pluriel_eu_au_eau ***/
pere(c_pluriel_eu_au_eau,c_pluriel_nom).
statut(c_pluriel_eu_au_eau,non_operationnelle).
        test_fils(c_pluriel_eu_au_eau,P,mot(P,_)).
valeur_fils(c_pluriel_eu_au_eau,P,c_c_plu_eu_au_eau_ex) :-
eval_prem(mot(P,[est_exception])).
valeur_fils(c_pluriel_eu_au_eau,P,c_c_plu_eu_au_eau_pas_ex) :-
        eval_prem(mot(P,[pas_exception])).
infos(c_pluriel_eu_au_eau) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_pluriel_al ***/
pere(c_pluriel_al,c_pluriel_nom).
statut(c_pluriel_al,non_operationnelle).
test_fils(c_pluriel_al,P,mot(P,_)).
valeur_fils(c_pluriel_al,P,c_c_plu_al_ex) :-
        eval_prem(mot(P,[est_exception])).
```

```
valeur_fils(c_pluriel_al,P,c_c_plu_al_pas_ex) :-
        eval_prem(mot(P,[pas_exception])).
infos(c_pluriel_al) :-
        writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_AL_EX ***/
pere(c_c_plu_al_ex,c_pluriel_al).
statut(c_c_plu_al_ex,operationnelle).
test_fils(c_c_plu_al_ex,P,(P,_)).
infos(c_c_plu_al_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_AL_PAS_EX ***/
pere(c_c_plu_al_pas_ex,c_pluriel_al).
statut(c_c_plu_al_pas_ex,operationnelle).
test_fils(c_c_plu_al_pas_ex,P,(P,_)).
/*** La Classe c_pluriel_ail ***/
pere(c_pluriel_ail,c_pluriel_nom).
statut(c_pluriel_ail,non_operationnelle).
test_fils(c_pluriel_ail,P,mot(P,_)).
attribut(c_pluriel_ail,P,mot(P,_),mot,
        [regle_ail_pas_exception,regle_ail_exception]).
valeur_fils(c_pluriel_ail,P,c_plu_ail_pas_ex) :-
        eval_prem(mot(P,[pas_exception])).
valeur_fils(c_pluriel_ail,P,c_plu_ail_ex) :-
        eval_prem(mot(P,[est_exception])).
/*** La Classe C_C_PLU_EU_AU_EAU_EX ***/
pere(c_c_plu_eu_au_eau_ex,c_pluriel_eu_au_eau).
statut(c_c_plu_eu_au_eau_ex,operationnelle).
test_fils(c_c_plu_eu_au_eau_ex,P,(P,_)).
/*** La Classe C_C_PLU_EU_AU_EAU_PAS_EX ***/
pere(c_c_plu_eu_au_eau_pas_ex,c_pluriel_eu_au_eau).
statut(c_c_plu_eu_au_eau_pas_ex,operationnelle).
test_fils(c_c_plu_eu_au_eau_pas_ex,P,(P,_)).
infos(c_c_plu_eu_au_eau_pas_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_pluriel_ou ***/
```

```
pere(c_pluriel_ou,c_pluriel_nom).
statut(c_pluriel_ou,non_operationnelle).
test_fils(c_pluriel_ou,P,mot(P,_)).
valeur_fils(c_pluriel_ou,P,c_c_plu_ou_ex) :-
        eval_prem(mot(P,[est_exception])).
valeur_fils(c_pluriel_ou,P,c_c_plu_pas_ex_) :-
        eval_prem(mot(P,[pas_exception])).
infos(c_pluriel_ou) :-
        writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_OU_EX ***/
pere(c_c_plu_ou_ex,c_pluriel_ou).
statut(c_c_plu_ou_ex,operationnelle).
test_fils(c_c_plu_ou_ex,P,(P,_)).
infos(c_c_plu_ou_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_plu_ail_pas_ex ***/
pere(c_plu_ail_pas_ex,c_pluriel_ail).
statut(c_plu_ail_pas_ex,operationnelle).
test_fils(c_plu_ail_pas_ex,P,(P,_)).
infos(c_plu_ail_pas_ex) :-
        writeln('aucune information sur le probleme').
/*** La Classe c_plu_ail_ex ***/
pere(c_plu_ail_ex,c_pluriel_ail).
statut(c_plu_ail_ex,operationnelle).
test_fils(c_plu_ail_ex,P,(P,_)).
infos(c_plu_ail_ex) :-writeln('aucune information sur le probleme').
/*** La Classe C_C_PLU_PAS_EX ***/
pere(c_c_plu_pas_ex_,c_pluriel_ou).
statut(c_c_plu_pas_ex_,operationnelle).
test_fils(c_c_plu_pas_ex_,P,(P,_)).
infos(c_c_plu_pas_ex_) :-
        writeln('aucune information sur le probleme').
/*** regle de propagation des attributs de mere en fille ***/
attribut(Classe,P,Att,Pred,Liste) :-
        pere(Classe,Pere),
```

attribut(Pere,P,Att,Pred,Liste).

Règles de reformulation

Les règles suivantes permettent par exemple de conclure sur la valeur des attributs : terminaison et mot (cf. **Figure 6**).

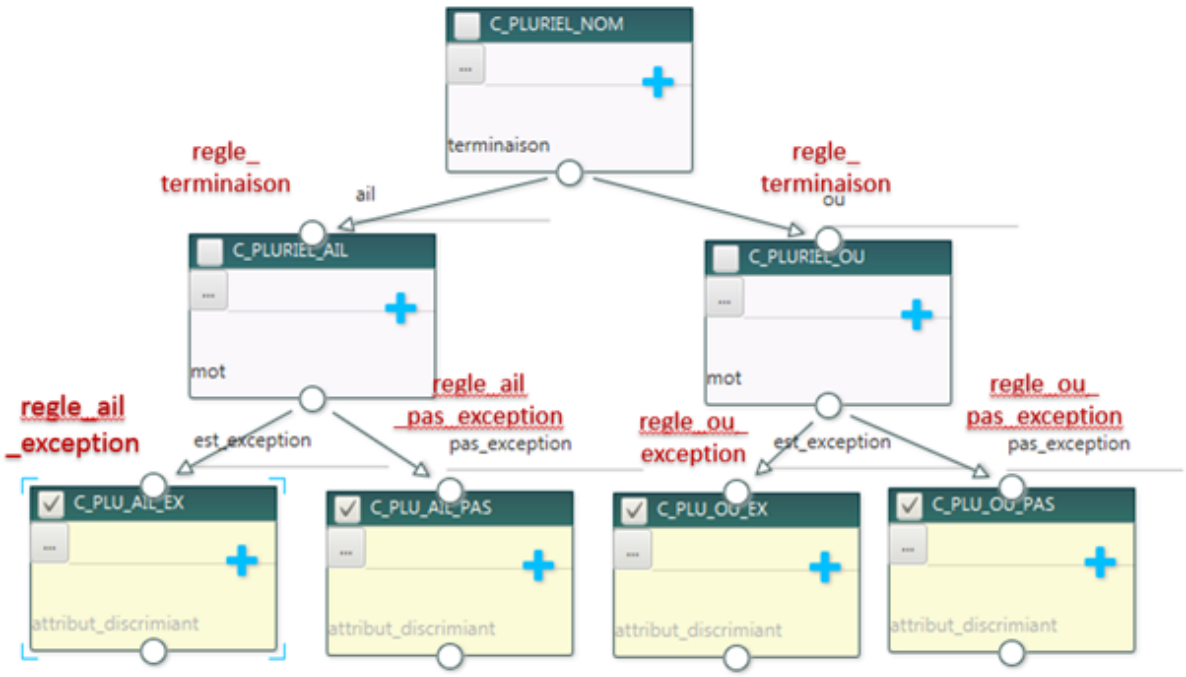

FIGURE 6 – une aprtied e l'arbre de classification

Exemple de règles permettant de conclure sur la valeur de l'attribut *terminaison*

regle_terminaison
Si dans un problème, il existe :
- un nom M
- la fin du nom est F
Alors la valeur de l'attribut *terminaison* est égale à *F*.

Exemples de règles permettant de conclure sur la valeur de l'attribut *mot*

regle_ail_exception
Si dans un problème, il existe :
- un nom M
- une liste L des exception des mots terminés par ail
- M appartient à la liste L
Alors la valeur de l'attribut *mot* est égale à *est_exception*.

regle_ail_pas_exception
Si dans un problème, il existe :
- un nom M
- une liste L des exception des mots terminés par ail
- M n'appartient pas à la liste L
Alors la valeur de l'attribut *mot* est égale à *pas_exception*.

Exemple de définition d'une règle en utilisant AMBRE-KB

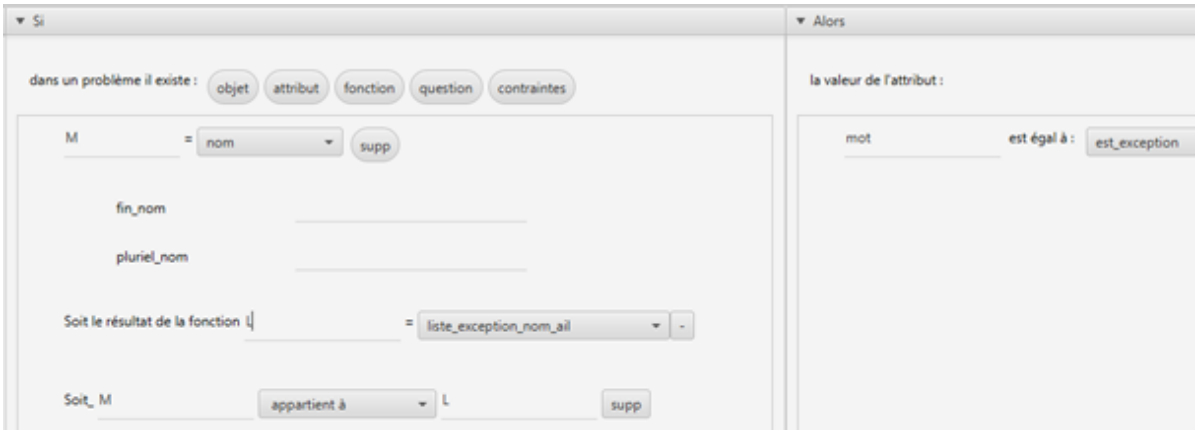

FIGURE 7 – Définition de la règle *regle_ail_exception* en utilisant AMBRE-KB

Exemple de modèles des connaissances de reformulation, généré par AMBRE-KB

/*** règle permettant de conclure sur la valeur de l'attribut terminaison ***/ regle(regle_terminaison) :-/*soit regle_terminaison nom de la règle*/ **si**([probleme(P),/*si dans un problème P*/ est_un(M,nom),/* M est un nom*/ fin_nom(M,F)]),/* T la fin du nom*/ alors([ajouter(terminaison(P,[F]))]). /*la valeur de l'attribut terminaison égale ou*/
/*** règle permettant de conclure sur la valeur de l'attribut mot ***/ regle(regle_ail_exception) :- /*soit regle_ail_exception nom de la règle*/ **si**([probleme(P), est_un(M,nom), eval_prolog(liste$_e$*xception*$_n$*om*$_a$*il*(*L*)), eval_prolog(member(M,L))]), **alors**([ajouter(mot(P,[est_exception]))]). /*la valeur de l'attribut mot égale est_exception*/
regle(regle_ail_pas_exception) :- **si**([probleme(P), est_un(M,nom), eval_prolog(liste$_e$*xception*$_n$*om*$_a$*il*(*L*)), /* L est la liste des exception en ail*/ eval_prolog(not member(M,L))]),/* M appartient à L*/ **alors**([ajouter(mot(P,[pas_exception]))]). /*la valeur de l'attribut mot égale pas_exception*/

Tableau 15 – Critères retenus pour évaluer le vocabulaire d'un domaine

Techniques de résolution

Quelques techniques définies par l'expert

Technique associée à la classe *c_plu_ail_pas_ex*

- soit un nom M
- soit Pluriel le nom obtenu en ajoutant 's' à M
- le pluriel du mot M est égal à Pluriel

Technique associée à la classe *c_plu_ail_ex*

- soit un nom M

- soit pluriel le résultat de la fonction permettant de former le pluriel des mot en ail qui sont des exception - le pluriel du mot M est égal à Pluriel

.0.0.2 Exemple de définition d'une technique de résolution en utlisant AMBRE-KB

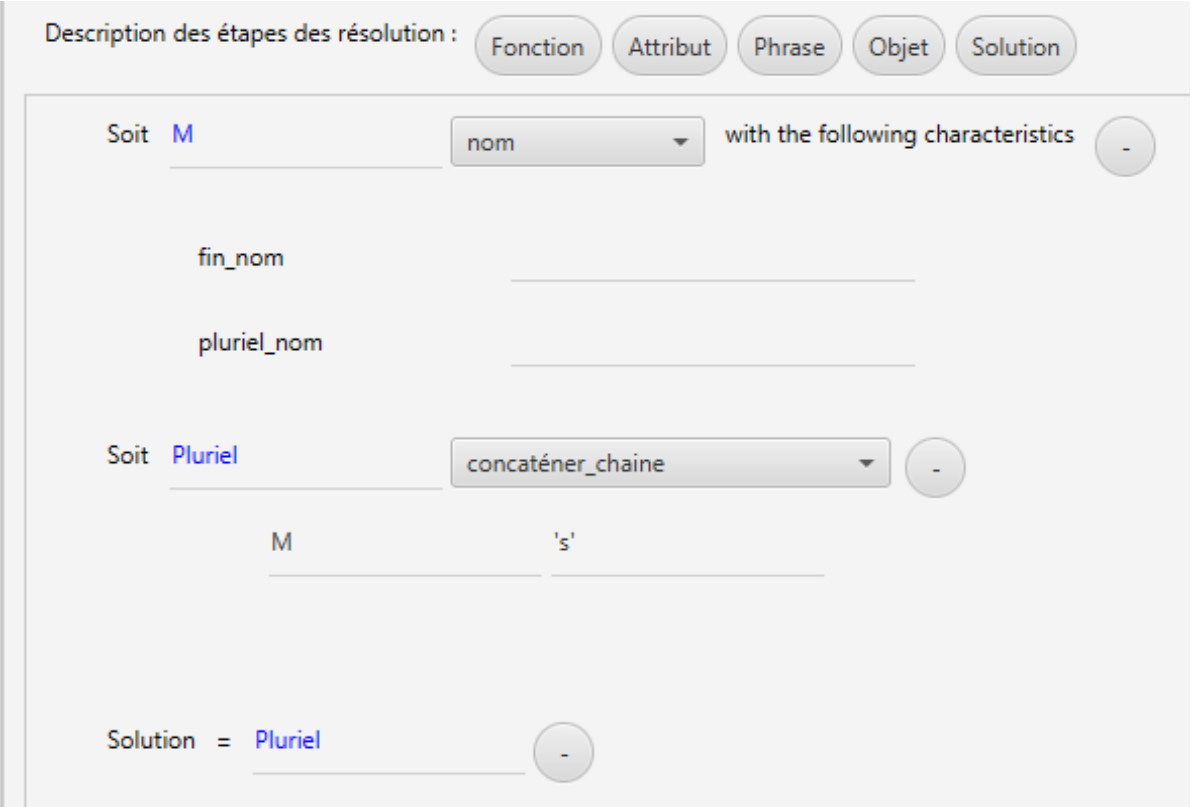

FIGURE 8 – Définition de la technique associée à la classe *c_plu_ail_pas_ex* en utilisant AMBRE-KB

Exemple de modèle décrivant des techniques de résolution, généré par AMBRE-KB

Technique associée à la classe *c_plu_ail_pas_ex* appliquer_resolution(P,c_plu_ail_pas_ex) :- eval_prem(est$_u n(M, nom)$), concatener_chaine(M,Chaine,Pluriel), ajouter_bf(solution(P,Pluriel)), write('').
Technique associée à la classe *c_plu_ail_ex* appliquer_resolution(P,c_plu_ail_ex) :- eval_prem(est_un(M,nom)), pluriel_ail_exception(M,Pluriel), ajouter_bf(solution(P,Pluriel)), write('').

Tableau 16 – Exemple de modèles de connaissances (décrivant des techniques) pour le pluriel des noms générés par AMBRE-KB

Résolution des équations à une seule inconnue

Ce domaine a été testé par deux sujets.

Profil du sujet 1 : Mathématicien (non programmeur).

Profil du sujet 1 : Informaticien (programmeur).

Méthode à enseigner : les sujets ont choisi d'enseigner la résolution des équations à une seule inconnue, du premier et second degré.
Les équations à résoudre ici sont toujours données sous les formes suivantes :
« *Coefficient_a*Inconnu +coefficient_b*Inconnu + Constante = 0* »
« *Coefficient_a*Inconnu + Constante = 0*»

Le **Tableau 6.6** décrit le nombre de connaissances produites par des deux sujets pour la résolution des équations.

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	4	3
Classes	6	7
Règles	5	5
Techniques	4	4
Problèmes	20	12
% Problèmes résolus sans feedback	75%	41%
% Problèmes résolus avec feedback	100%	100%

Tableau 17 – Comparaison des résultats des expérimentations pour les équations par les sujets 1 et 2

9 fonctions en plus ont été définies par les sujets. Celles-ci interviennent dans la définition des règles et des techniques de résolution pour le domaine :

— **Multiplication** : permettant de calculer le produit de deux nombres

— **Division** : qui renvoie le quotient de la division de deux nombres

— **Racine_carre** : permettant de calculer la racine carré d'un nombre

— **Determinant** : qui calcule le déterminant de trois valeurs (coefficient_b * coefficient_b - 4*coefficient_a*constante)

— **Determinant_nul_solution** : qui renvoie la solution lorsque le déterminant est nul (- coefficient_b/2*coefficient_a)

— **Determinant_positive_solution_1** : permettant de calculer une des solutions lorsque le déterminant est positif : (- coefficient_b ' racine_carré (determinant) /2*coefficient_a)

— **Determinant_positive_solution_2** : permettant de calculer une des solutions lorsque le déterminant est positif : (- coefficient_b + racine_carré (determinant) /2*coefficient_a)

— **Liste_nombres** : permettant de créer une liste à partir de deux nombres (par exemple, lorsque le déterminant est positif, renvoyer les deux solutions sous la forme d'une liste).

— **Phrase_reponse** : permet de construire une phrase (utilisée lorsque le déterminant est négatif, pour dire qu'il n'y a pas de solution).

Vocabulaires proposés pour le domaine

Vocabulaire défini par le sujet 1

Types objets	**Caractéristiques objets**
Équation	Nom_inconnu –>chaine de caractères Valeur_inconnu–>nombre entier Liste_termes–>type énuméré(deux_termes,trois_termes)
Deux_termes	Premier_terme–>terme_inconnu Deuxième_terme –>terme_constante
Terme_inconnu	coefficient–>nombre entier
Terme_constante	valeur_constante–>nombre entier

Tableau 18 – Vocabulaire défini par le sujet 1.

Vocabulaire défini par le sujet 2

Types objets	**Caractéristiques objets**
Équation	Terme_1–>type énuméré(terme_inconnu,terme_constante) Terme_2–>type énuméré(terme_inconnu,terme_constante) Terme_3–>type énuméré(terme_inconnu,terme_constante)
Terme_inconnu	Coefficient–>nombre entier Nom_variable–>chaine de caractères Degré–>nombre entier Valeur–>nombre entier
Terme_constante	valeur_constante –> nombre entier

Tableau 19 – Vocabulaire défini par le sujet 2.

La question à résoudre pour les différents problèmes consistait à trouver la valeur de valeur_variable.

Problèmes à résoudre

Exemples de problèmes définis par les sujets :

degre_1_1 : « Résoudre l'équation suivante : 2x-5=0. »
degre_1_2 : « Résoudre l'équation suivante : 7y+9=0. »
degre_2_nul : « Résoudre l'équation suivante : y2+2*Y+1=0. »
degre_2_positif : « Résoudre l'équation suivante : z2+6z+7=0=0. »
degre_2_negatif : « Résoudre l'équation suivante : 3*x2+3*x+3=0. »

Arbre de classification

Arbre de classification défini par le sujet 1.

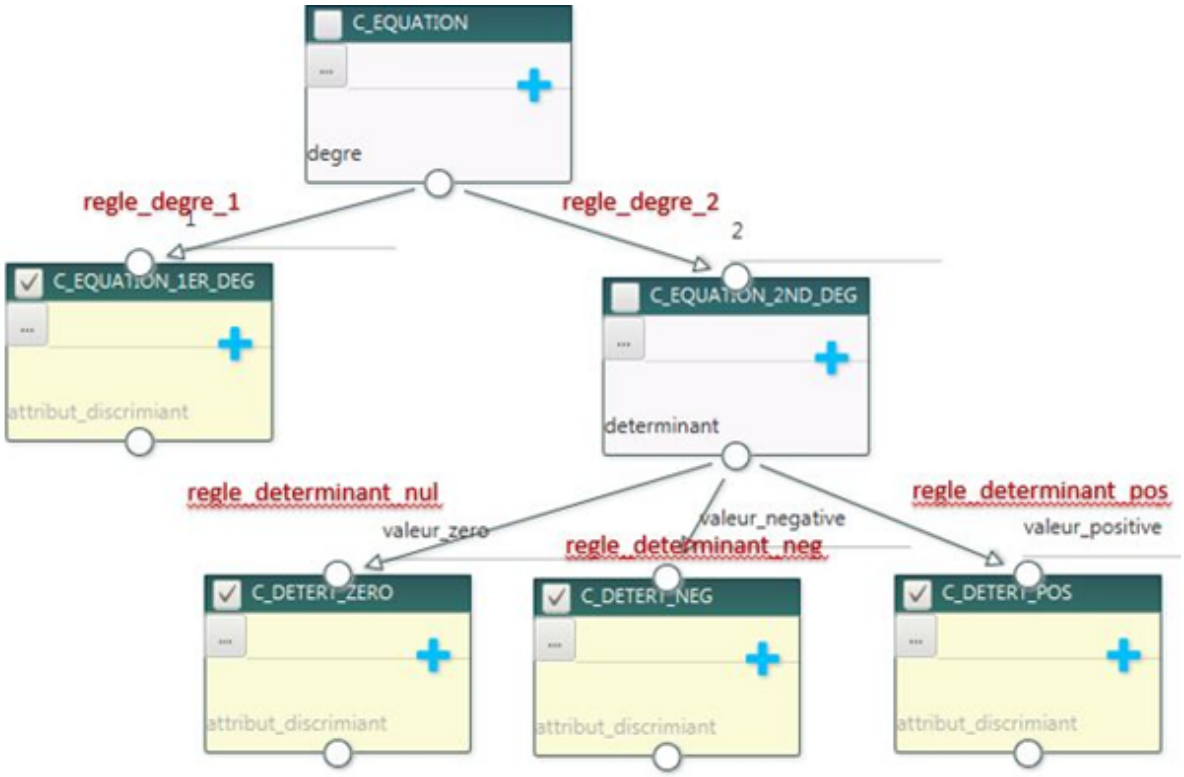

FIGURE 9 – Arbre de classification pour les équations défini par le sujet 1

Arbre de classification défini par le sujet 2.

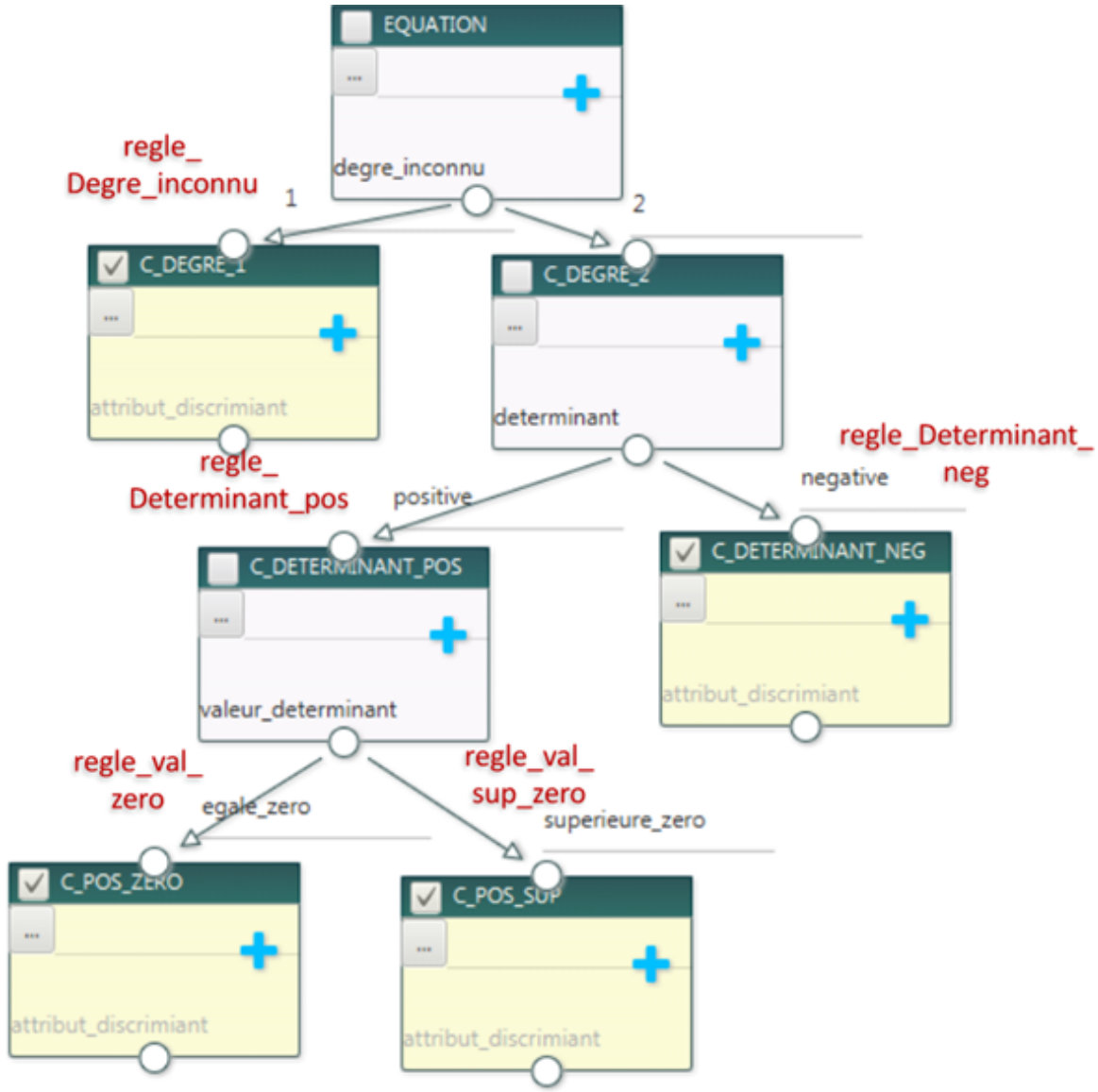

FIGURE 10 – Arbre de classification pour les équations défini par le sujet 2

Règles de reformulation

Exemples de règles définies par l'auteur

Exemples de règles de reformulation définies par les sujets.

regle_degre : **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le degré D, **Alors** la valeur de l'attribut degré est égal à D.
regle_determiannt_pos **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le coefficient est A, un deuxième terme T2 de type terme_inconnu dont le coefficient est B, un troisième terme de type T3 terme_constante dont la valeur de la constante est C, F le résultat de la fonction qui calcule le déterminant (B2-4*A*C) F>0, **alors** la valeur de l'attribut déterminant est égale à valeur_negative.
regle_determiannt_nul : **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le coefficient est A, un deuxième terme T2 de type terme_inconnu dont le coefficient est B, un troisième terme de type T3 terme_constante dont la valeur de la constante est C, F le résultat de la fonction qui calcule le déterminant (B2-4*A*C), F=0, **Alors** la valeur de l'attribut déterminant est égale à valeur_zero.
regle_determiannt_neg **Si** dans un problème, il existe : un premier terme T1 de type terme_inconnu dont le coefficient est A, un deuxième terme T2 de type terme_inconnu dont le coefficient est B, un troisième terme de type T3 terme_constante dont la valeur de la constante est C, F le résultat de la fonction qui calcule le déterminant (B2-4*A*C), F<0, **Alors** la valeur de l'attribut déterminant est égale à valeur_negative.

Tableau 20 – Exemple de règle permettant de conclure sur la valeur de l'attribut discriminant inconnue reu, et des attributs du problème : nombre_total_possédé et nombre partiel possédé

Techniques de résolution

Exemple de techniques de résolution définies par les sujets.

```
/*Technique pour les résoudre les équations du premier degré*/
appliquer_resolution(P,c_equation_1er_deg) :-
        eval_prem(est_un(T1,premier_terme)),
        eval_prem(term1(T1,Term1)),
```

```
        eval_prem(est_un(Term1,terme_inconnu)),
        eval_prem(coefficient(Term1,A)),
        eval_prem(nom_variable(Term1,X)),
        eval_prem(degre_variable(Term1,D)),
        eval_prem(est_un(T3,troisieme_terme)),
        eval_prem(term3(T3,Term3)),
        eval_prem(est_un(Term3,terme_constante)),
        eval_prem(valeur_constante(Term3,C)),
        division(- C,A,Div),
        ajouter_bf(solution(P,[Div])),
        write('').
/*Technique pour les résoudre les équations du second
degré dont le déterminant est nul*/
appliquer_resolution(P,c_2nd_deter_zero) :-
        eval_prem(est_un(T1,premier_terme)),
        eval_prem(term1(T1,Term1)),
        eval_prem(est_un(Term1,terme_inconnu)),
        eval_prem(coefficient(Term1,A)),
        eval_prem(nom_variable(Term1,X)),
        eval_prem(est_un(T2,deuxieme_terme)),
        eval_prem(term2(T2,Term2)),
        eval_prem(est_un(Term2,terme_inconnu)),
        eval_prem(coefficient(Term2,B)),
        eval_prem(nom_variable(Term2,X)),
        eval_prem(est_un(T3,troisieme_terme)),
        eval_prem(term3(T3,Term3)),
        eval_prem(est_un(Term3,terme_constante)),
        eval_prem(valeur_constante(Term3,C)),
        discriminant_nul_solution(A,B,Sol),
        ajouter_bf(solution(P,[Sol])),
        write('').
```

```
/*Technique pour les équations du 1er degré
        Soit E une équation
        Soit nom_inconnu de E = X
        Soit valeur_inconnu de E = Val
```

Soit la liste L des termes de E = deux_termes
Soit premier_terme de L = T1
Soit T1 est un terme_inconnu
Soit coefficient de T1 = C1
Soit deuxieme_terme de L= T2
Soit T2 est un terme_constante
Soit valeur_constante de T2 = C2
Val = - C2/C1
Solution = valeur_inconnu.

/*Technique pour les résoudre les équations du second degré dont le déterminant est négatif*/
Soit E une équation
Soit nom_inconnu de E = X
Soit Valeur_inconnu de E = Val
Soit la liste L des termes de E = trois_termes
Soit premier_terme de L = T1
Soit T1 est un terme_inconnu
Soit coefficient de T1 = C1
Soit troisieme_terme de L= T3
Soit T3 est un terme_constante
Soit valeur$_c onstante de T3 = C3$
Soit le résultat de la fonction déterminant = 0
Val = - C2/(2*C1)
Solution = Val

Calcul d'aire en géométrie

Ce domaine a été testé par deux sujets.

Profil du sujet 1 : Statisticien (non programmeur).

Profil du sujet 2 : Informaticien (non programmeur).

Méthode à enseigner : les sujets ont choisi d'enseigner le calcul d'aire des figures géométriques ayant 3, 4 ou 4 cotés.

Le **Tableau 22** résume les connaissances décrites par les deux sujets :

	Sujet 1	Sujet 2
vocabulaire (types d'objets)	5	8
Classes	12	12
Règles	11	11
Techniques	7	7
Problèmes	28	18
% Problèmes résolus sans feedback	57%	61%
% Problèmes résolus avec feedback	85%	89%

Tableau 22 – Comparaison des résultats des expérimentations pour le calcul d'aire en géométrie par les sujets 1 et 2

16 fonctions en plus, intervenant dans la définition des règles et des techniques de résolution ont été définies âr les sujets :

- **aire_triangle** :permettant de calculer l'aire d'un triangle
- **aire_rectangle** :permettant de calculer l'aire d'un rectangle
- **aire_carre** :permettant de calculer l'aire d'un carre
- **aire_trapeze** :permettant de calculer l'aire d'un trapèze
- **aire_losange** :permettant de calculer l'aire d'un losange
- **aire_pentagone** :permettant de calculer l'aire d'un pentagone régulier
- **aire_tpentagone_irregulier** :permettant de calculer l'aire d'un pentagone irrégulier
- **demi_perimetre_triangle** : permettant de calculer le demi périmètre d'un triangle

— **addition** : permettant de calculer la somme de deux nombres
— **soustraction** : permettant de calculer la soustraction de deux nombres
— **division** : permettant de calculer la division d'un nombre
— **racine_carre** : permettant de calculer la racine carré d'un nombre
— **tangente** : permettant de calculer la tangente
— **valeur_apotheme** : permettant de calculer l'apothème
— **cosinus** : permettant de calculer le cosinus
— **sinus** : permettant de calculer le sinus

Vocabulaires proposés pour le domaine

Vocabulaire proposé par le sujet 1.

Types objets	**Caractéristiques objets**
figure	identifiant aire_figure –> nombre segment_1 –> segment (obligatoire) segment_2 –> segment (obligatoire) segment_3 –> segment (obligatoire) segment_4 –> segment segment_5 –> segment angle_droit –> booléen
angle	identifiant segment_1 –> segment segment_2 –> segment valeur_angle –> nombre entier
segment	identifiant longueur_segment –> nombre entier
segments_paralleles	identifiant segment_1 –> segment segment_2 –> segment
segments_opposes	identifiant segment_1 –> segment segment_2 –> segment

Tableau 23 – vocabulaire pour le calcul d'aire proposé par le sujet 1

Vocabulaire proposé par le sujet 2.

Types objets	**Caractéristiques objets**
figure	identifiant aire_figure –> nombre liste_cote_figure –> type énuméré trois_cotes, quatre_cotes,cinq_cotes possede_angle_droit –> booléen
cote	identifiant longueur_cote –> nombre entier
trois$_c$*otes*	identifiant segment_1 –> segment segment_2 –> segment segment_3 –> segment
quatre$_c$*otes*	identifiant segment_1 –> segment segment_2 –> segment segment_3 –> segment segment_4 –> segment
cinq$_c$*otes*	identifiant segment_1 –> segment segment_2 –> segment segment_3 –> segment segment_4 –> segment segment_5 –> segment
segment	identifiant longueur_segment –> nombre entier
cote_paralleles	identifiant cote_1 –> cote cote_2 –> cote
cote_opposes	identifiant cote_1 –> cote cote_2 –> cote
cote_perpendiculaires	identifiant cote_1 –> cote cote_2 –> cote

Tableau 24 – vocabulaire pour le calcul d'aire proposé par le sujet 2

Problèmes à résoudre

Exemples de problèmes définis par les sujets :

Pentagone_régulier_1 : « Trouver le périmètre et l'aire d'un polygone ABCDE dont chaque côté mesure 3cm ».

Carre_1 : « On veut calculer l'aire et le périmètre d'un polygone ABCD dont chaque côté mesure 5cm et dont l'angle abc est droit ».

trapèze_1 : « Quelle est le périmètre et l'aire d'un champ qui a la forme d'un quadrilatère dont les dimensions des côtés opposés sont : AB =4 CD = 7».

Triangle_1 : « on cherche à calculer le périmètre et l'aire d'un polygone ABC dont l'angle ABC est droit, et les dimensions sont : AB=3 AC=5 ».

Parallélogramme_1 : « Calculer le périmètre et l'aire d'un polygone MNOP ».

Arbre de classification

Une partie de l'arbre de classification défini par le sujet 1.

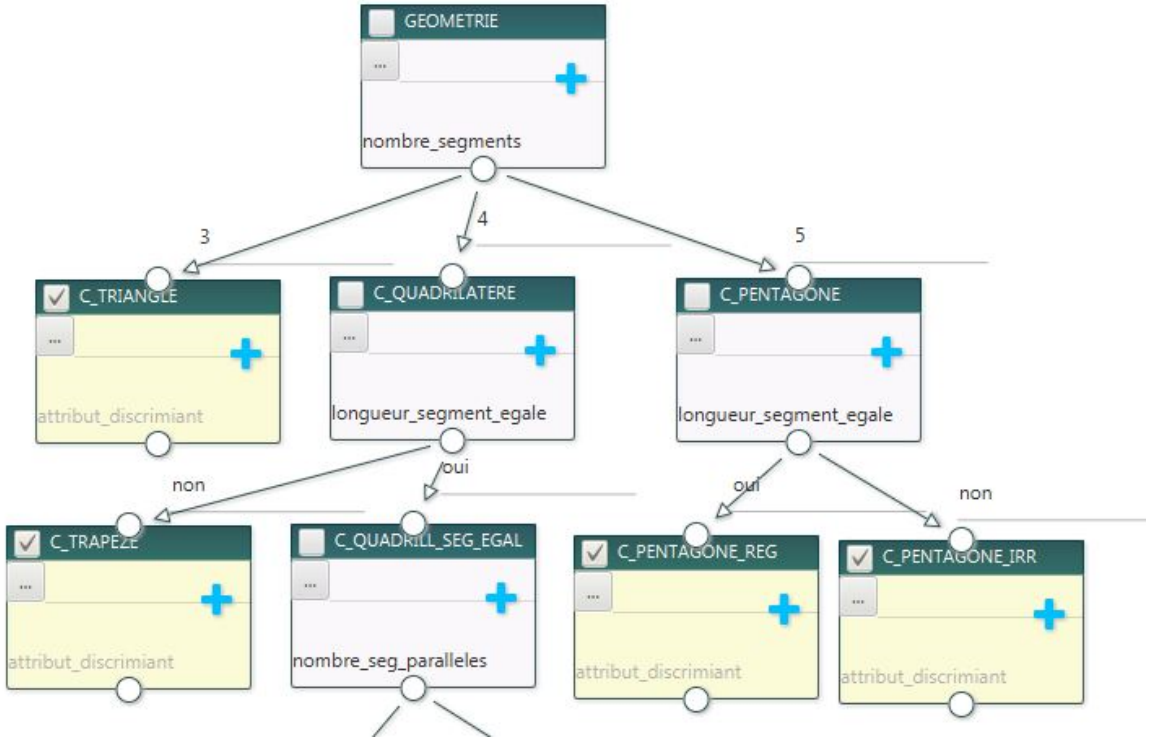

FIGURE 11 – Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie

Une partie de l'arbre de classification défini par le sujet 1.

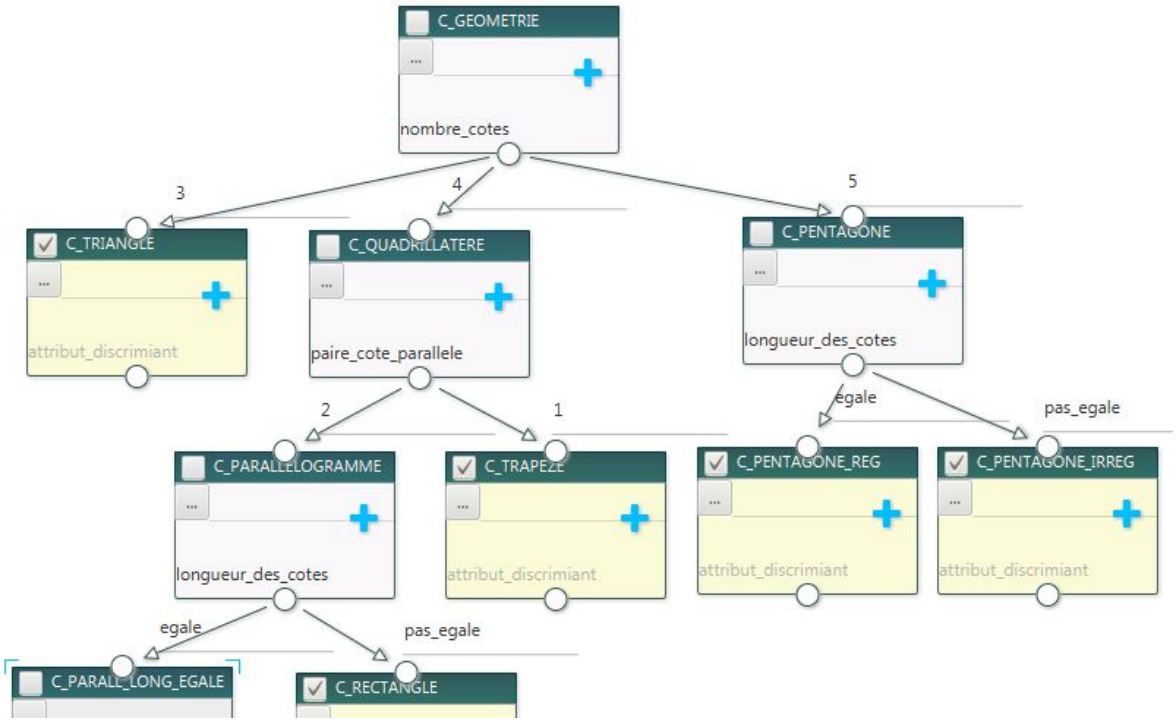

FIGURE 12 – Une partie de l'arbre de classification définie par le sujet 1 pour le calcul d'aire en Géométrie

Règles de reformulation

Nous présentons quelques règles de reformulation définies par le sujet 1.

/* Règle permettant de conclure sur la valeur de l'attribut *nombre_segments* ***/**
Si dans un problème il existe :
- une figure géométrique F
- S1 le segment_1 de F
- S2 le segment_2 de F
- S3 le segment_3 de F
- S4 le segment_4 de F
- S5 le segment_5 de F
- La longueur d'un des segments est différents des autres

Alors la valeur de l'attribut *nombre_segments* est égale à *5*
/**Règle permettant de conclure sur la valeur de l'attribut longueur_segment_egale****/**
Si dans un problème, il existe :
- une figure géométrique F
- la longueur C1 du segment_1 S1 de F
  est égale à la longueur C2 du segment_2 S2 de F
- la longueur C2 du segment_2 S2 de F
  est égale à la longueur C3 du segment_3 S3 de F
- la longueur C3 du segment_3 S3 de F
  est égale à la longueur C4 du segment_4 S4 de F
- la longueur C4 du segment_4 S4 de F
  est égale à la longueur C5 du segment_5 S5 de F
- la longueur C5 du segment_5 S5 de F
  est égale à la longueur C1 du segment_1 S1 de F

Alors la valeur de l'attribut *longueur_segment_egale* est égale à *oui* height

Tableau 25 – Exemple de règles de reformulation définies pour le calcul d'aire en géométrie

Techniques de résolution

Exemples de techniques de résolution :

/* technique permettant de calculer l'aire d'un triangle ***/**
- Soit F une figure
- soit C1 la longueur du segment_1 de F
- soit C2 la longueur du segment_1 de F
- soit C3 la longueur du segment_3 de F
- soit Sol le résultat de la fonction aire_triangle
- Solution égale Sol

Tableau 26 – Exemple de techniques de résolution définies pour le calcul d'aire en géométrie

Formation des adverbes en "ment"

Profil du sujet : Infographiste (non programmeur).

Méthode à enseigner. Le sujet a choisi d'enseigner la formation des adverbes en français terminés par « ment ».

L'expert a défini :

- 17 classes dont 11 sont opérationnelles
- 16 règles
- 11 techniques de résolution
- 40 problèmes
- 4 Fonctions intervenant dans la définition des règles et des techniques de résolution ajouter_chiane : permet de concaténer deux chaines de caractères enlever_chaine : enlever une sous chaine à une chaine de caractères dernière_lettre_mot : qui renvoie la dernière lettre d'un mot liste_de$_v$*oyelles* : $fonctionquirenvoielalistedesvoyelles$

Vocabulaire proposé pour le domaine

Types objets	**Caractéristiques objets**
mot	identifiant–>chaine de caractères adverbe–>chaine de caractères

Tableau 27 – vocabulaire pour la formation des adverbes français en "ment"

Problèmes à résoudre

Exemples de problèmes définis par l'expert :

Adverbe_faible : Donner l'adverbe du mot (faible) ».
Adverbe_apparent : Donner l'adverbe du mot (apparent) ».
Adverbe_suffisant : Donner l'adverbe du mot (suffisant) ».
Adverbe_vrai : Donner l'adverbe du mot (vrai) ».

Adverbe_assidu : Donner l'adverbe du mot (assidu) ».

Arbre de classification

La **Figure 13** présente une partie de l'arbre de classification proposé pour le domaine.

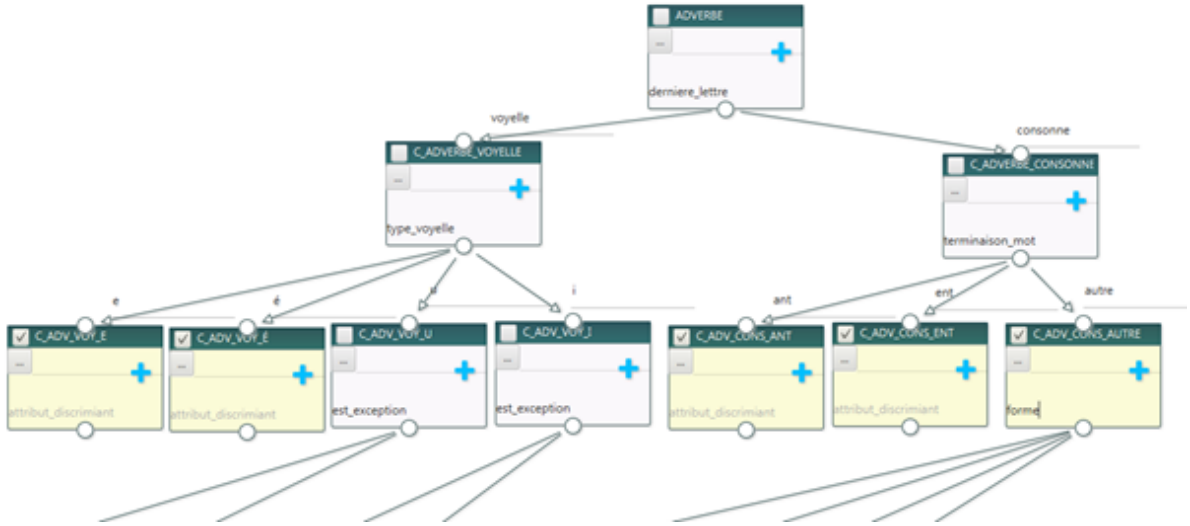

FIGURE 13 – Une partie de l'arbre de classification pour la formation des adverbes en "ment"

Règles de reformulation

/* Exemples de règle permettant de conclure sur la valeur de l'attribut derniere_lettre ***/**

Si dans un problème, il existe
- un mot M
- la dernière lettre du mot appartient à la liste des voyelles

Alors la valeur de l'attribut *derniere_lettre* est égale à *voyelle*

Si dans un problème, il existe
- un mot M
- la dernière lettre du mot n'appartient pas à la liste de s voyelles
Alors la valeur de l'attribut *derniere_lettre* est égale à *consonne*

Techniques de résolution

/* technique permettant de former l'adverbe des mots terminés par "ant" ***/**
- Soit M un mot
- soit M1 le radical du mot en enlevant la terminaison 'ant'
- Soit M2 le mot obtenu en ajoutant 'amment' au radical de M
- Solution est égale à M2

/* technique permettant de former l'adverbe des mots terminés par "ent" ***/**
- Soit M un mot
- soit M1 le radical du mot en enlevant la terminaison 'ent'
- Soit M2 le mot obtenu en ajoutant 'emment' au radical de M
- Solution est égale à M2

Critères retenus pour évaluer le processus d'acquisition et sa mise en œuvre dans AMBRE-KB

Dans le chapitre 6, nous avons identifié différentes facettes pour évaluer si AMBRE-KB permet d'éliciter les différentes connaissances à la conception d'un EIAH AMBRE. Nous détaillons ici les critères retenus pour évaluer chacune de ces facettes.

La première section détaille les critères retenus pour évaluer si AMBRE-KB permet d'éliciter les connaissances nécessaires à l'EIAH pour résoudre des problèmes. Dans la deuxième section, nous présentons les critères retenus pour évaluer si les interfaces proposées dans AMBRE-KB sont assez intuitives. La dernière section détaille les critères retenus pour évaluer l'assistance fournie AMBRE-KB.

Critères retenus pour évaluer l'élicitation des connaissances avec AMBRE-KB

Facette à évaluer	Définition du vocabulaire pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi ?
Méthodologie envisagée	Définition du vocabulaire de son domaine par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'expliciter les connaissances nécessaires pour définir le vocabulaire du domaine choisi

Tableau 28 – Critères retenus pour évaluer le vocabulaire d'un domaine

Facette à évaluer	Description des problèmes à résoudre pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de décrire les problèmes du domaine ?
Méthodologie envisagée	Définition des problèmes du domaine par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet de décrire l'ensemble des problèmes du domaine

Tableau 29 – Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné

Facette à évaluer	Élicitation des connaissances de classification pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de définir les connaissances de classification du domaine choisi ?
Méthodologie envisagée	Élicitation des connaissances de classification par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'éliciter les connaissances de classification du domaine

Tableau 30 – Critères retenus pour évaluer la description des problèmes à résoudre pour un domaine donné

Facette à évaluer	Élicitation des connaissances de reformulation pour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de définir les connaissances de reformulation du domaine choisi ?
Méthodologie envisagée	Élicitation des connaissances de reformulation par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'éliciter les connaissances de reformulation du domaine

Tableau 31 – Critères retenus pour évaluer l'élicitation des connaissances de reformulation pour un domaine donné

Facette à évaluer	Élicitation des connaissances de résolutionpour le domaine choisi
Point à analyser	Est-ce que AMBRE-KB permet de définir les connaissances de résolution du domaine choisi ?
Méthodologie envisagée	Élicitation des connaissances de résolution par l'auteur en utilisant AMBRE-KB + réponse à la question correspondante dans le questionnaire
Condition de validation	AMBRE-KB permet d'éliciter les connaissances de résolution du domaine

Tableau 32 – Critères retenus pour évaluer l'élicitation des connaissances de résolution pour un domaine donné

Critères retenus pour évaluer les interfaces de AMBRE-KB

Facette à évaluer	Définition du vocabulaire pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour la définition du vocabulaire ?
Méthodologie envisagée	Définition du vocabulaire par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir le vocabulaire sans aide extérieure.

Tableau 33 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la définition du vocabulaire pour un domaine donné

Facette à évaluer	Description des problèmes à résoudre pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour la Description des problèmes à résoudre ?
Méthodologie envisagée	Description des problèmes à résoudre par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les problèmes du domaine sans aide extérieure.

Tableau 34 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour la Description des problèmes à résoudre pour un domaine donné

Facette à évaluer	Élicitation des connaissances de classification pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour Élicitation des connaissances de classification à résoudre ?
Méthodologie envisagée	Élicitation des connaissances de classification par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les connaissances de classification sans aide extérieure.

Tableau 35 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de classification pour un domaine donné

Facette à évaluer	Élicitation des connaissances de reformulation pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour Élicitation des connaissances de reformulation
Méthodologie envisagée	Élicitation des connaissances de reformulation par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les connaissances de reformulation sans aide extérieure.

Tableau 36 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de reformulation pour un domaine donné

Facette à évaluer	Élicitation des connaissances de résolution pour le domaine choisi
Point à analyser	Ergonomie de l'interface proposée pour Élicitation des connaissances de résolution
Méthodologie envisagée	Élicitation des connaissances de résolution par l'auteur en utilisant AMBRE-KB, puis jugement de l'auteur (réponses aux questions correspondantes dans le questionnaire)
Condition de validation	L'auteur a pu définir les connaissances de résolution sans aide extérieure.

Tableau 37 – Critères retenus pour évaluer l'ergonomie de l'interface proposée pour définir les connaissances de résolution pour un domaine donné

Critères retenus pour évaluer l'assistance fournie avec AMBRE-KB

Facette à évaluer	Flexibilité offerte avec AMBRE-KB
Point à analyser	Est-ce que la flexibilité fournie à l'auteur est utile et permet d'aider l'auteur durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	La flexibilité proposée dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation

Tableau 38 – Critères retenus pour évaluer la flexibilité proposée dans AMBRE-KB

Facette à évaluer	Guidage offerte avec AMBRE-KB
Point à analyser	Est-ce que le guidage fourni à l'auteur est utile et permet d'aider l'auteur durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	Le guidage proposé dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation

Tableau 39 – Critères retenus pour évaluer les fonctionnalités de guidage proposées dans AMBRE-KB

Facette à évaluer	Contrôle de connaissances offerte avec AMBRE-KB
Point à analyser	Est-ce que le Contrôle des connaissances proposé à l'auteur est utile et permet d'aider l'auteur durant le processus d'élicitation ?
Méthodologie envisagée	Utilisation des fonctionnalités d'assistance + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	Le guidage proposé dans AMBRE-KB est utile et permet d'aider l'auteur durant le processus d'élicitation

Tableau 40 – Critères retenus pour évaluer les fonctionnalités de contrôle de connaissances proposées dans AMBRE-KB

Critères retenus pour évaluer le feedback proposé avec AMBRE-KB

Facette à évaluer	Feedback offert avec AMBRE-KB
Point à analyser	Test de la résolution des problèmes à résoudre ?
Méthodologie envisagée	Test de la résolution des problèmes à résoudre + réponses aux questions posées sur cette partie dans le questionnaire
Condition de validation	Le feedback proposé permet à l'auteur de comprendre ses erreurs lorsque la résolution échoue et de pouvoir les corriger

Tableau 41 – Critères retenus pour évaluer les fonctionnalités de feedback proposées dans AMBRE-KB

Questionnaire distribué aux sujets

1. **Date :** _____

2. **Domaine testé :** _____

3. **Profil :** _____

Définition du vocabulaire

4. **Avez-vous rencontré des problèmes pour définir le vocabulaire ?**
   - ☐ J'ai compris dès le début. 5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

5. **Est-ce que les interfaces proposées sont assez intuitives ?**
   - ☐ J'ai compris dès le début.5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

6. **Est-ce que vous avez pu définir toutes les connaissances nécessaires au vocabulaire du domaine que vous avez choisi ?**
   - ☐ Oui 5/5

- ☐ Sinon, merci de préciser les types de connaissances que vous souhaitiez pouvoir définir.

  _____

  _____

  _____

7. **Est-ce que vous avez des suggestions concernant le vocabulaire ou sa définition ?**

   _____

   _____

   _____

Description des problèmes à résoudre

8. **Avez-vous rencontré des problèmes pour décrire les problèmes à résoudre ?**
   - ☐ J'ai compris dès le début. 5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

9. **Est-ce que les interfaces proposées sont assez intuitives ?**
   - ☐ J'ai compris dès le début.5/5
   - ☐ J'ai compris vers la fin.
   - ☐ Je n'ai pas compris.

10. **Est-ce que vous avez pu définir toutes les connaissances nécessaires pour décrire les problèmes du domaine que vous avez choisi ?**
    - ☐ Oui 5/5
    - ☐ Sinon, merci de préciser les types de connaissances que vous souhaitiez pouvoir définir.

11. **Est-ce que vous avez des suggestions concernant la description des problèmes ?**

Élicitation des connaissances de classification

12. **Avez-vous rencontré des problèmes pour définir l'arbre de classification du domaine que vous avez choisi ?**
    - ☐ J'ai compris dès le début. 5/5
    - ☐ J'ai compris vers la fin.
    - ☐ Je n'ai pas compris.

13. **Est-ce que les interfaces proposées sont assez intuitives ?**
    - ☐ J'ai compris dès le début.5/5
    - ☐ J'ai compris vers la fin.
    - ☐ Je n'ai pas compris.

14. **Est-ce que vous avez des suggestions concernant la définition de l'arbre de classification ?**

Élicitation des connaissances de reformulation

15. **Avez-vous rencontré des problèmes pour définir les règles de reformulation du domaine que vous avez choisi ?**
    - ☐ J'ai compris dès le début. 1/5
    - ☐ J'ai compris vers la fin. 4/5
    - ☐ Je n'ai pas compris.

16. **Est-ce que les interfaces proposées sont assez intuitives ?**
    - ☐ J'ai compris dès le début.1/5
    - ☐ J'ai compris vers la fin. 4/5
    - ☐ Je n'ai pas compris.

17. **Est-ce que vous avez des suggestions concernant la définition des règles de reformulation ?**

    _____

    _____

    _____

Élicitation des connaissances de résolution

18. **Avez-vous rencontré des problèmes pour définir les techniques de résolution du domaine que vous avez choisi ?**
    - ☐ J'ai compris dès le début. 1/5
    - ☐ J'ai compris vers la fin. 4/5
    - ☐ Je n'ai pas compris.

19. **Est-ce que les interfaces proposées sont assez intuitives ?**
    - ☐ J'ai compris dès le début.1/5

- ☐ J'ai compris vers la fin. 4/5
- ☐ Je n'ai pas compris.

20. Est-ce que vous avez des suggestions concernant la définition des techniques de résolution ?

_____

_____

_____

Assistance fournie avec AMBRE-KB

21. Que pensez-vous de la flexibilité offerte par AMBRE-KB lors de la définition des connaissances ?

- ☐ Je n'ai pas remarqué.
- ☐ J'ai compris vers la fin. 5/5

22. Que pensez-vous des fonctionnalités de guidage proposées par AMBRE-KB ?

- ☐ Je n'ai pas remarqué.1/5
- ☐ Elles sont utiles mais peuvent être améliorées. 2/5
- ☐ Elles sont utiles et m'ont beaucoup aidé. 3/5

23. Que pensez-vous des fonctionnalités de contrôle des connaissances offertes par AMBRE-KB ?

- ☐ Elles sont insuffisantes et ne m'ont pas vraiment aidé.
- ☐ Elles sont utiles mais ne sont pas assez suffisantes.
- ☐ Elles sont utiles et m'ont beaucoup aidé. 5/5

24. Que pensez-vous des fonctionnalités de guidage proposées par AMBRE-KB ?

- □ Je n'ai pas compris les messages d ?erreur.
- □ J'ai compris les messages d'erreur mais ils ne sont pas utiles.
- □ Les messages d'erreur sont utiles et m ?ont permis de comprendre et de corriger mes erreurs. 2/5

Fonctionnement général de AMBRE-KB

25. Est-ce que vous avez compris le fonctionnement général de AMBRE-KB ?

- □ Oui.5/5
- □ Non.

Bibliographie

[1] Shaaron Ainsworth, Nigel Major, Shirley Grimshaw, Mary Hayes, Jean Underwood, Ben Williams, and David Wood. Redeem : Simple intelligent tutoring systems from usable tools. In *Authoring Tools for Advanced Technology Learning Environments*, pages 205–232. Springer, 2003.

[2] Vincent Aleven. Using background knowledge in case-based legal reasoning : a computational model and an intelligent learning environment. *Artificial Intelligence*, 150(1-2) :183–237, 2003.

[3] Vincent Aleven. Rule-based cognitive modeling for intelligent tutoring systems. In *Advances in intelligent tutoring systems*, pages 33–62. Springer, 2010.

[4] Vincent Aleven, Bruce M Mclaren, Jonathan Sewall, and Kenneth R Koedinger. A new paradigm for intelligent tutoring systems : Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2) :105–154, 2009.

[5] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, Martin Van Velsen, Octav Popescu, Sandra Demi, Michael Ringenberg, and Kenneth R Koedinger. Example-tracing tutors : Intelligent tutor development for non-programmers. *International Journal of Artificial Intelligence in Education*, 26(1) :224–269, 2016.

[6] Nathalie Aussenac-Gilles. *Conception d'une methodologie et d'un outil d'acquisition de connaissances expertes*. PhD thesis, Toulouse 3, 1989.

[7] Nathalie Aussenac-Gilles. *Méthodes ascendantes pour l'ingénierie des connaissances*. PhD thesis, Université Paul Sabatier-Toulouse III, 2005.

[8] Nathalie Aussenac-Gilles, Jean Charlet, and Chantal Reynaud. Les enjeux de l'Ingénierie des connaissances. In P. Marquis F. Sedes, J.M. Ogier, editor, *Information, Interaction, Intelligence - Le point sur le i(3)*. Cepadues, January 2012.

[9] N Balacheff. Contribution à la réflexion sur la recherche sur les environnements informatiques pour lâapprentissage humain. *Les technologies en éducation : Pers-*

pectives de recherche et questions vives. Paris : INRP–MSH-IUFM de Basse Normandie, pages 193–201, 2002.

[10] Nicolas Balacheff, Monique Baron, Cyrille Desmoulins, Monique Grandbastien, and Martial Vivet. Conception d'environnements interactifs dâapprentissage avec ordinateur. tendances et perspectives. *Actes des journées nationales du PRC IA*, pages 315–338, 1997.

[11] Emmanuel Bigand, François Madurell, Barbara Tillmann, and Marion Pineau. Effect of global structure and temporal organization on chord processing. *Journal of Experimental Psychology : Human Perception and Performance*, 25(1) :184, 1999.

[12] Stephen B Blessing. A programming by demonstration authoring tool for model-tracing tutors. In *Authoring Tools for Advanced Technology Learning Environments*, pages 93–119. Springer, 2003.

[13] John H Boose. A survey of knowledge acquisition techniques and tools. *Knowledge acquisition*, 1(1) :3–37, 1989.

[14] Jill Boucher. The theory of mind hypothesis of autism : Explanation, evidence and assessment. *British Journal of Disorders of Communication*, 24(2) :181–198, 1989.

[15] Alain Bronner, Thomas Huguet, Denis Bouhineau, and Jean-François Nicaud. Usages didactiques du logiciel aplusix pour l'enseignement de l'algebre. 2003.

[16] E Bruillard, Élisabeth Delozanne, Pascal Leroux, Paul Delannoy, Xavier Dubourg, Pierre Jacoboni, Jérôme Lehuen, Daniel Luzzati, and Philippe Teutsch. Quinze ans de recherche informatique sur les sciences et techniques éducatives au lium. *Revue Sciences et Techniques Educatives*, 7(1) :87–145, 2000.

[17] Jean Charlet. L'ingénierie des connaissances, entre science de l'information et science de gestion. In *Entre la connaissance et l'organisation, l'activité collective*. November 2004. Ce document est la version allongée du chapitre référencé.

[18] Jean Charlet, Manuel Zacklad, and Gilles Kassel. Ingénierie des connaissances. 2000.

[19] Michelene TH Chi, Paul J Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive science*, 5(2) :121–152, 1981.

[20] Christophe Choquet. *Ingénierie et réingénierie des EIAH-L'approche REDiM*. PhD thesis, Université du Maine, 2007.

[21] William J Clancey. Heuristic classification. *Artificial intelligence*, 27(3) :289–350, 1985.

[22] William J Clancey. From guidon to neomycin and heracles in twenty short lessons. *AI Magazine*, 7(3) :40, 1986.

[23] Albert T Corbett, Kenneth R Koedinger, and John R Anderson. Intelligent tutoring systems. *Handbook of human-computer interaction*, 5 :849–874, 1997.

[24] Stephane Crozat and Philippe Trigano. *An Ontological Approach for Design and Evaluation of Tutoring Systems*, pages 651–651. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[25] Denise Dellarosa Cummins. Role of analogical reasoning in the induction of problem categories. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 18(5) :1103, 1992.

[26] Jean-Marc David, Jean-Paul Krivine, and Reid Simmons. Second generation expert systems : a step forward in knowledge engineering. In *Second generation expert systems*, pages 3–23. Springer, 1993.

[27] André Didierjean. Apprendre à partir d'exemples : abstraction de règles et/ou mémoire d'exemplaires ? *L'année psychologique*, 101(2) :325–348, 2001.

[28] R Dieng, O Corby, F Gandon, A Giboin, J Golebiowska, N Matta, and M Ribière. Knowledge management : Méthodes et outils pour la gestion des connaissances, 2005.

[29] Rose Dieng. *Méthodes et outils d'acquisition des connaissances.* PhD thesis, INRIA, 1990.

[30] Rose Dieng. Comparison of conceptual graphs for modelling knowledge of multiple experts. *Foundations of Intelligent Systems*, pages 78–87, 1996.

[31] Nathalie Dubus. *Développement d'un système expert dans le processus de planification des ressources en eau au Burkina Faso.* PhD thesis, Université Joseph Fourier (Grenoble), 1994.

[32] Nathalie Duclosson, Stéphanie Jean-Daubias, and Stéphanie Riot. Ambre-enseignant : un module partenaire de l'enseignant pour créer des problèmes. 2005.

[33] M Duribreux, P Caulier, and B Houriez. Application industrielle d'une approche mixte de modélisation des connaissances. *), Ingénierie des connaissances : évolutions récentes et nouveaux défis, Eyrolles et France Télécom R&D, Paris, France*, 2000.

[34] Valérie Emin and Jean-Philippe Pernin. Un environnement graphique de scénarisation pédagogique dirigé par les intentions. In *actes de la conférence TICE 2010*, pages 16–pages. to be published, 2010.

[35] Michel Fayol. *L'enfant et le nombre : du comptage à la résolution de problèmes*. Delachaux et Niestlé Paris, 1990.

[36] Mary L Gick and Keith J Holyoak. Schema induction and analogical transfer. *Cognitive psychology*, 15(1) :1–38, 1983.

[37] JE Gilbert and CY Han. Case-based reasoning applied to instruction method selection for intelligent tutoring systems. In *Rin Workshop proceedings of ITS*, pages 11–15, 2000.

[38] Monique Grandbastien. Teaching expertise is at the core of its research. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10 :335–349, 1999.

[39] Viviane Guéraud and Anne Lejeune. Une approche auteur pour la scénarisation et le suivi de situations d'apprentissage. In *EIAH 2011-Conférence Environnements Informatiques pour l'Apprentissage Humain*, pages 11–17, 2011.

[40] Nathalie Guin. *Reformuler et classer un problème pour le résoudre. L'architecture SYRCLAD et son application à quatre domaines*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 1997.

[41] Nathalie Guin. *Des meÌta-modeÌles pour guider lâeÌlicitation des connaissances en EIAH : contributions aÌ lâenseignement de meÌthodes et aÌ la personnalisation des activiteÌs*. PhD thesis, Université Claude Bernard-Lyon 1, 2014.

[42] N Guin-Duclosson. Syrclad : une architecture de résolveurs de problèmes permettant d'expliciter des connaissances de classification, reformulation et résolution. *Revue d'Intelligence Artificielle*, 13(2) :225–282, 1999.

[43] N Guin-Duclosson, S Jean-Daubias, and S Nogry. Le projet ambre : utiliser un paradigme d'apprentissage fondé sur le ràpc pour faire acquérir à des élèves une méthode de résolution de problèmes. *Raisonnement à partir de cas*, 2007.

[44] Pitrat Jacques. Métaconnaissance, futur de l'intelligence artificielle. *Hermès, Paris*, 1990.

[45] TM Khan. Case-based evaluation for student modelling. In *Workshop*, volume 5, pages 16–22, 2000.

[46] Kenneth R Koedinger and Vincent Aleven. An interview reflection on âintelligent tutoring goes to school in the big cityâ. *International Journal of Artificial Intelligence in Education*, 26(1) :13–24, 2016.

[47] Kenneth R Koedinger, Vincent Aleven, Neil Heffernan, Bruce McLaren, and Matthew Hockenberry. Opening the door to non-programmers : Authoring intelligent

tutor behavior by demonstration. In *Intelligent tutoring systems*, volume 3220, pages 7–10. Springer, 2004.

[48] Pierre Laforcade, Thierry Nodenot, and Christian Sallaberry. Un langage de modélisation pédagogique basé sur uml. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation (STICEF)*, pages 89–116, 2005.

[49] Sébastien Lallé. *Assistance to build and compare knowledge diagnostic techniques*. Theses, Université de Grenoble, December 2013.

[50] Pascal Leroux. *Machines partenaires des apprenants et des enseignants-Étude dans le cadre d'environnements supports de projets pédagogiques*. PhD thesis, Université du Maine, 2002.

[51] Noboru Matsuda, Victoria Keiser, Rohan Raizada, Arthur Tu, Gabriel Stylianides, William W Cohen, and Kenneth R Koedinger. Learning by teaching simstudent : Technical accomplishments and an initial use with students. In *International Conference on Intelligent Tutoring Systems*, pages 317–326. Springer, 2010.

[52] Antonija Mitrovic, Brent Martin, Pramuditha Suraweera, Konstantin Zakharov, Nancy Milik, Jay Holland, and Nicholas McGuigan. Aspire : an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2) :155–188, 2009.

[53] Antonija Mitrovic, Pramuditha Suraweera, Brent Martin, Konstantin Zakharov, Nancy Milik, and Jay Holland. Authoring constraint-based tutors in aspire. In *International Conference on Intelligent Tutoring Systems*, pages 41–50. Springer, 2006.

[54] Riichiro Mizoguchi and Jacqueline Bourdeau. Using ontological engineering to overcome ai-ed problems : Contribution, impact and perspectives. *International Journal of Artificial Intelligence in Education*, 26(1) :91–106, 2016.

[55] E Shiri A Mohammad, Esma Aïmeur, and Claude Frasson. Sara : A case-based student modelling system. In *European Workshop on Advances in Case-Based Reasoning*, pages 425–436. Springer, 1998.

[56] Allen Munro. Authoring simulation-centered learning environments with rides and vivids. In *Authoring Tools for Advanced Technology Learning Environments*, pages 61–91. Springer, 2003.

[57] Tom Murray. Authoring intelligent tutoring systems : An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10 :98–129, 1999.

[58] Tom Murray. An overview of intelligent tutoring system authoring tools : Updated analysis of the state of the art. In *Authoring tools for advanced technology learning environments*, pages 491–544. Springer, 2003.

[59] Tom Murray, Stephen Blessing, and Shaaron Ainsworth. *Authoring tools for advanced technology learning environments : Toward cost-effective adaptive, interactive and intelligent educational software*. Springer Science & Business Media, 2003.

[60] Roger Nkambou, Jacqueline Bourdeau, and Valéry Psyché. Building intelligent tutoring systems : An overview. *Advances in Intelligent Tutoring Systems*, pages 361–375, 2010.

[61] Thierry Nodenot. *Contribution à l'Ingénierie dirigée par les modèles en EIAH : le cas des situations-problèmes coopératives*. PhD thesis, Université de Pau et des Pays de l'Adour, 2005.

[62] Sandra Nogry. Faciliter l'apprentissage à partir d'exemples en situation de résolution de problèmes. 2005.

[63] Sandra Nogry, Nathalie Guin, and Stéphanie Jean-Daubias. Ambre-add : An its to teach solving arithmetic word problems. *Technology, Instruction, Cognition & Learning*, 6(1), 2008.

[64] Mary S Riley et al. Development of children's problem-solving ability in arithmetic. 1984.

[65] Aline Robert, Janine Rogalski, and Renan Samurcay. *Enseigner des méthodes*. Institut de Recherche pour L'Enseignement des Mathematiques, 1987.

[66] Marc Rogalski. Enseigner des méthodes des mathematiques. *Recherche em Didactiques des Mathématiques*, 1(1) :1–10, 1990.

[67] Marc Rogalski. Les concepts de l'eiao sont-ils indépendants du domaine. *L'exemple d'enseignement de méthodes en analyse. Recherches en Didactiques des Mathématiques*, 14(1.2) :43–66, 1994.

[68] Brian H Ross and Patrick T Kennedy. Generalizing from the use of earlier examples in problem solving. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 16(1) :42, 1990.

[69] Roger C Schank. Dynamic memory : A theory of learning in people and computers, 1982.

[70] A.H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, 1985.

[71] Alan H Schoenfeld. Metacognitive and epistemological issues in mathematical understanding. *Teaching and learning mathematical problem solving : Multiple research perspectives*, pages 361–380, 1985.

[72] Alan H Schoenfeld. Problem solving in context (s). *The teaching and assessing of mathematical problem solving*, 3 :82–92, 1989.

[73] Alan H Schoenfeld. Toward a theory of teaching-in-context. *Issues in education*, 4(1) :1–94, 1998.

[74] Guus Schreiber. *Knowledge engineering and management : the CommonKADS methodology*. MIT press, 2000.

[75] Guus Schreiber, Bob Wielinga, and Joost Breuker. *KADS : A principled approach to knowledge-based system development*, volume 11. Academic Press, 1993.

[76] Kay G Schulze, Robert N Shelby, Donald J Treacy, Mary C Wintersgill, Kurt Vanlehn, and Abigail Gertner. Andes : An intelligent tutor for classical physics. *Journal of Electronic Publishing*, 6(1), 2000.

[77] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering : principles and methods. *Data & knowledge engineering*, 25(1-2) :161–197, 1998.

[78] Pramuditha Suraweera, Antonija Mitrovic, and Brent Martin. The role of domain ontology in knowledge acquisition for itss. In *Intelligent Tutoring Systems*, pages 215–230. Springer, 2004.

[79] Pierre Tchounikine. Précis de recherche en ingénierie des eiah. 2009.

[80] Gheorghe Tecuci and Harry Keeling. Developing an intelligent educational agent with disciple. *International Journal of Artificial Intelligence in Education*, 10(3-4) :221–237, 1999.

[81] Wouter R Van Joolingen and Ton De Jong. Simquest. In *Authoring tools for advanced technology learning environments*, pages 1–31. Springer, 2003.

[82] Michael van Lent and John E Laird. Learning procedural knowledge through observation. In *Proceedings of the 1st international conference on Knowledge capture*, pages 179–186. ACM, 2001.

[83] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The andes physics tutoring system : Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3) :147–204, 2005.

[84] Maria Virvou and Maria Moundridou. A web-based authoring tool for algebra-related intelligent tutoring systems. *Educational Technology & Society*, 3(2) :61–70, 2000.

[85] C Vogel. Kod : une méthode pour le recueil et la modélisation de connaissances. *Journées internationales sur les systèmes experts et leurs applications, Avignon, France*, 1990.

[86] Bob J Wielinga. Reflections on 25+ years of knowledge acquisition. *International Journal of Human-Computer Studies*, 71(2) :211–215, 2013.

[87] Bob J Wielinga, A Th Schreiber, and Jost A Breuker. Kads : A modelling approach to knowledge engineering. *Knowledge acquisition*, 4(1) :5–53, 1992.

[88] Beverly Woolf. *Building Intelligent Interactive Tutors, Student-Centered Strategies for Revolutionizing E-Learning*. 01 2008.

[89] Beverly Park Woolf. *Building intelligent interactive tutors : Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.

[90] Ming Zhou, Jianwen Ren, Jianxun Qi, Dongxiao Niu, and Gengyin Li. Common-kads methodology for developing power grid switching orders systems. *Emerging Technologies in Knowledge Discovery and Data Mining*, pages 87–98, 2007.

Publications durant la thèse

Publications disponibles depuis https://liris.cnrs.fr/membres/?idn=adiattar&onglet=publis

Conférences internationales avec comité de lectures et actes

[1] A. Diattara, N. Guin, V. Luengo and A. Cordier (2017). « Preliminary evaluations of AMBRE-KB, An Authoring Tool to Elicit Knowledge to be Taught Without Programming ». [Extension de l'article publié à CSEDU : **soumis**]

[2] A. Diattara, N. Guin, V. Luengo and A. Cordier .: An Authoring Tool to Elicit Knowledge to be Taught without Programming. In: Paula Escudeiro, Gennaro Costagliola, Susan Zvacek, J.U. and B.M.M. (ed.) International Conference on Computer Supported Education. pp. 82–91. , Porto, Portugal (2017). **Article nominé pour le prix du meilleur article**.

[3] A. Diattara, N. Guin, V. Luengo and A. Cordier (2016). « Towards an Authoring Tool to Acquire Knowledge for ITS Teaching Problem Solving Methods ». European Conference on Technology Enhanced Learning (EC-TEL 2016), 16 septembre 2016, Lyon (France), pp. 575-578.

Conférences nationales avec comité de lectures et actes

[4] A. Diattara, "Vers un outil auteur pour des EIAH destinés à l'apprentissage de méthodes de résolution de problèmes : acquisition des connaissances de la méthode à enseigner," in RJCEIAH, 2016.

[5] A. Diattara, N. Guin, M. Lefevre, B. Kong Win Chang, and V. Luengo, "Vers un outil auteur pour des EIAH destinés à l'apprentissage de méthodes : formalisation des connaissances à acquérir." in TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION POUR L'ENSEIGNEMENT, 2014, pp. 180–185.