Thèse de doctorat de l'Université du Maine

présentée par

Dominique PRÉVIT

pour l'obtention du titre de

Docteur de l'Université du Maine

Spécialité: Informatique

Génération d'exercices et analyse multicritère automatique de réponses ouvertes

PépiGen, un système auteur en algèbre élémentaire

Thèse soutenue le vendredi 30 mai 2008

Jury :

Mme. Monique GRANDBASTIEN	Rapporteur
M. Jean-Pierre PEYRIN	Rapporteur
M. Pascal LEROUX	Directeur
Mme. Elisabeth DELOZANNE	Co-directrice
M. Eric BRUILLARD	Examinateur
M. Jean-François NICAUD	Examinateur
Mme. Brigitte GRUGEON	Invitée

Remerciements

Tout d'abord, un grand merci à Elisabeth Delozanne qui m'a entraînée dans cette aventure. Alors que je m'étais inscrite à l'université du Maine pour obtenir un D.E.A, elle n'a pas hésité à me proposer, malgré mon age, un sujet qui devait aboutir à cette thèse qu'elle a encadrée. Elle m'a toujours encouragée et accompagnée dans ce travail, me proposant de nombreux sujets de lecture, m'apportant une aide précieuse par sa grande connaissance du domaine, me faisant rencontrer d'autre chercheurs pour échanger et approfondir le sujet de recherche.

Je remercie Pierre Tchounikine d'avoir accepté de m'inscrire au LIUM pour préparer une thèse et de m'avoir fait confiance. Enseignante à l'IUFM de Bretagne, mes obligations professionnelles ne m'ont pas permis d'être souvent au LIUM, mais j'y ai toujours été bien accueillie.

Je remercie Pascal Leroux pour avoir accepté cette co-direction avec Elisabeth et pour la confiance qu'il m'a toujours accordée sur l'aboutissement de ce travail.

Je remercie vivement Monique Grandbastien et Jean-Pierre Peyrin d'avoir accepté d'être rapporteurs de cette thèse alors qu'ils avaient très peu de disponibilités.

Les travaux menés dans le projet Aplusix ont inspiré mes recherches, aussi je remercie Jean-François Nicaud d'avoir accepté de faire partie du jury.

Je remercie Eric Bruillard, professeur à l'IUFM de Créteil, d'avoir accepté de faire partie de ce jury.

Je remercie Brigitte Grugeon d'avoir accepté de participer au jury. Le travail de cette thèse s'appuie sur ses travaux de recherche en didactique des mathématiques. Je lui suis très reconnaissante pour sa collaboration tout au long de ce travail.

Mon travail de recherche s'est déroulé au sein de l'équipe du projet Lingot et j'ai beaucoup apprécié de pouvoir travailler dans une équipe pluridisciplinaire. Je tiens à remercier tous les membres de cette équipe, qui du point de vue de la recherche m'ont apporté leur expertise, leurs conseils, un terrain d'expériences et d'un point de vue humain m'ont encouragée et soutenue. Je remercie plus particulièrement, Françoise Chenevetot, Marie-France Delord, Brigitte Grugeon Pierre Jacoboni, Christian Vincent. Ils m'ont accompagné tout au long de cette thèse, participant de façon assidue aux travaux de l'équipe, m'apportant leur expérience pour l'écriture de

scénarios, testant toutes mes propositions avec des retours critiques et constructifs. Je remercie Marie-France et Brigitte pour leurs relectures minutieuses.

Je remercie Loïc Allys d'avoir accepté de relire tout ce qui concernait les grammaires et l'analyse syntaxique sur lesquels s'appuyait une partie de mon travail de thèse.

Je remercie les membres du groupe AIDA et plus particulièrement Monique Baron, Hélène Giroire, Françoise Le Calvez, Brigitte de la Passardière, Jean-Marc Labat. Leur écoute attentive, leurs questions pertinentes, leurs suggestions m'ont permis d'approfondir mon travail et de progresser.

Je remercie vivement Brigitte de la Passardière pour sa relecture très minutieuse et les conseils très précieux de rédaction qu'elle m'a donnés.

Je remercie Stéphanie Jean-Daubias qui a conçu et réalisé un premier logiciel Pépite dont les expérimentations et les retours d'usages ont étayé ce travail de thèse.

Je remercie toutes les personnes de l'IUFM de Bretagne qui m'ont soutenue et encouragée permettant à ce travail d'aboutir.

La liste des personnes que j'ai rencontrées au cours de cette thèse est longue et je vais certainement en oublier. Je remercie tous ceux qui m'ont aidée dans ce travail de recherche riche et passionnant où j'ai beaucoup appris et que j'ai pris plaisir à mener à son terme.

Enfin, je voudrais remercier mon mari qui m'a toujours incitée à poursuivre des études, encouragée, soutenue et a accepté toutes les contraintes que ce travail de thèse imposait. Je remercie mes enfants pour m'avoir encouragée dans ce projet qui ne facilitait pas la vie de famille. Démarrer une thèse quand on approche de la retraite était une entreprise un peu folle mais passionnante et ce travail va se poursuivre dans les projets Pépite et Lingot.

Table des matières

Chapitre 1 : Introduction Générale

1. Introduction **3**

2. Contexte de recherche **4**

3. Le cadre : le projet Lingot **5**

4. Problématique et questions de recherche **6**

5. Plan de la thèse **7**

Chapitre 2 : Le projet Lingot

1. Introduction **11**

2. Exemples **14**
2.1. Côté élève 14
2.2. Côté enseignant 16
2.2.1. Scénario d'utilisation 17
2.2.2. État actuel du projet 17

3. Fondements didactiques du projet (1990-1995) **18**
3.1. Un modèle de la compétence algébrique à la fin du collège 18
3.2. Un outil de diagnostic papier-crayon 18

4. Le premier logiciel Pépite (1996-2000) **19**
4.1. PépiTest, le logiciel élève 20
4.2. PépiDiag, le logiciel de diagnostic automatique 22
4.3. PépiProf, le logiciel enseignant 22

5. Tests de la première version de Pépite (2000-2002) **24**
5.1. Différents contextes d'usage 25
5.2. Le point de vue des enseignants 26
5.3. Retour sur les hypothèses et les questions de recherche 28
5.3.1. PépiTest 28
5.3.2. PépiDiag 29
5.3.3. PépiProf 29
5.4. Leçons et questions 30
5.4.1. Définir et caractériser des banques de tests 30
5.4.2. Diagnostic assisté ou diagnostic automatique ? 30

5.4.3. Test systématique ou adaptatif ? 31
5.4.4. Exploitation du diagnostic 31

6. Vers un diagnostic automatique plus fiable 32
6.1. Vers une analyse automatique des raisonnements algébriques 33
6.2. Diagnostic automatique des réponses à cet exercice 34

7. Vers un assistant à l'exploitation du diagnostic 36
7.1. Des profils aux stéréotypes 36
7.2. Un bilan avec l'élève 40
7.3. Des activités d'apprentissage paramétrées par les compétences 40
7.4. Tests diagnostics pour d'autres niveaux scolaires 42

8. Conclusion 43

Chapitre 3 : État de l'art

1. Introduction 47

2. Les tuteurs cognitifs 49
2.1. Les fondements du projet 49
2.1.1. Modèle de la cognition 49
2.1.2. Principes de conception des « Cognitive Tutors » 50
2.2. Les tuteurs cognitifs dans les classes 51
2.3. Les activités des élèves avec un tuteur Cognitif 51
2.4. Le diagnostic cognitif 53
2.5. Logiciel d'exploitation par les enseignants 54
2.6. Outils auteur pour les tuteurs cognitifs 55
2.7. Ce que je retiens pour ma thèse 57

3. Andes, un tuteur en Physique 58
3.1. Les fondements 59
3.2. Les activités des élèves avec Andes 60
3.3. Le diagnostic dans Andes 60
3.4. Les outils auteurs dans Andes 62
3.4.1. Le graphe des solutions correctes 62
3.4.2. Le système de calcul formel spécifique 62
3.5. Ce que j'en retiens pour ma thèse 63

4. Aplusix 64
4.1. Une théorie de l'algèbre 65
4.2. Les activités des élèves avec Aplusix 67
4.3. Le diagnostic dans Aplusix 68
4.4. Outil auteur 70
4.5. Ce que j'en retiens pour ma thèse 71

5. Des projets plus éloignés 72

6. Les langages de modélisation pédagogique 75
6.1. Le LOM : un standard pour la description des objets pédagogiques 76
6.2. Langage de modélisation pédagogique EML 77
6.3. IMS-LD 79
6.4. Question and Test Interopérability (QTI) 80
6.4.1. Différents cas d'utilisation 81

6.4.2. Modèle d'informations de QTI 82
6.4.3. Ce que je retiens pour ma thèse 85
6.5. Modèle conceptuel d'évaluation à l'Open University of NederLand (OUNL) 87
6.5.1. Description du modèle 87
6.5.2. Ce que je retiens pour ma thèse 90
6.6. Langages spécifiques aux mathématiques 90
6.6.1. MathML 90
6.6.2. OpenMath 90

7. Conclusion 91

Chapitre 4 : Contexte, objectifs, problématique et méthodologie

1. Introduction 95

2. Les objets métiers, définitions 95
2.1. Test (ou test diagnostic) 96
2.2. Diagnostic 96
2.3. Tâches diagnostiques (ou exercices diagnostiques) 96
2.4. Stratégies de diagnostic 97
2.5. Types de tests 97

3. Scénarios de conception 98
3.1. Les utilisateurs de SuperPépite 98
3.2. Scénario 1 : un enseignant découvre SuperPépite et s'approprie un test type 99
3.3. Scénario 2 : un enseignant, habitué, compose un test diagnostic 99
3.4. Scénario 3 : un enseignant veut étudier l'évolution des compétences de ses élèves depuis le dernier test 100
3.5. Scénario 4 : une didacticienne des mathématiques génère des tests diagnostics 100
3.6. Scénario 5 : des élèves passent un test diagnostic 100
3.7. Scénario 6 : un élève dialogue avec un enseignant pour réfléchir sur son bilan de compétence 101
3.8. Scénario 7 : un enseignant constitue des groupes de travail dans sa classe 101
3.9. Scénario 8 : un auteur enrichit la banque d'exercices 102
3.10. Scénario 9 : un(e) développeur(e) informatique ajoute un modèle d'exercices 102
3.11. Conclusion 103

4. Architecte fonctionnelle de SuperPepite 103

5. Problématique 105

6. Méthodologie 107

Chapitre 5 : Modèle Conceptuel des classes d'exercices

1. Introduction 111

2. Méthodologie 111

3. Classe d'exercices « Reconnaître des égalités numériques vraies » 114
3.1. Indexation didactique 114
3.2. Génération de l'exercice 115
3.2.1. Interface élève 115

3.2.2. Génération des énoncés ... 115
3.3. Génération du diagnostic ... 116
3.4. Type de génération ... 116

4. Classe d'exercices « Reconnaître des sommes et des produits » ... 118
4.1. Génération de l'exercice ... 119
4.2. Génération du diagnostic ... 119

5. Discussion ... 120

6. Classe d'exercices « Correspondance entre aire et expression » ... 121
6.1. Indexation didactique ... 121
6.2. Génération de l'exercice ... 122
6.2.1. Interface élève ... 122
6.2.2. Génération de l'énoncé ... 123
6.3. Génération du diagnostic ... 124
6.4. Type de génération ... 124

7. Classe d'exercices « Expression littérale de l'aire d'un rectangle » ... 125
7.1. Génération de l'exercice ... 127
7.1.1. Interface élève ... 127
7.1.2. Génération des énoncés ... 128
7.1.3. Génération du diagnostic ... 129
7.1.4. Type de génération ... 130
7.2. Discussion ... 131

8. Classe d'exercices « Preuve et programme de calcul » ... 131
8.1. Génération de l'exercice ... 134
8.1.1. Interface élève ... 134
8.1.2. Génération de l'énoncé ... 134
8.2. Génération du diagnostic ... 135
8.3. Type de génération ... 136

9. Classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales » ... 139
9.1. Génération de l'exercice ... 140
9.1.1. Génération de l'interface ... 140
9.1.2. Génération des énoncés ... 140
9.2. Génération du diagnostic ... 142
9.3. Type de génération ... 142
9.4. Discussion ... 143

10. Classe d'exercices « Expressions algébriques d'un programme de calcul» ... 143
10.1. Génération de l'exercice ... 145
10.1.1. Interface Élève ... 145
10.1.2. Génération des énoncés ... 145
10.2. Génération du diagnostic ... 146
10.3. Type de génération ... 147

11. Élaboration d'un modèle conceptuel d'une classe d'exercices ... 148
11.1. Indexation didactique des exercices ... 149
11.2. Génération des énoncés ... 150
11.3. Description de l'interface ... 151
11.4. Génération du diagnostic ... 154

12. Conclusion..... 156

Chapitre 6 : Génération et analyse des expressions algébriques : Pépinière

1. Introduction..... 161

2. Scénarios d'utilisation de Pépinière..... 162
2.1. Scénario 1 : diagnostic d'une réponse algébrique..... 162
2.1.1. Construction d'un arbre d'expression..... 163
2.1.2. Comparaison d'expressions..... 164
2.2. Scénario 2 : génération automatique de la grille de codage d'un exercice..... 164
2.3. Architecture fonctionnelle de Pépinière..... 167

3. Analyse syntaxique des expressions algébriques dans Pépinière..... 167
3.1. Représentation des expressions algébriques dans Pépinière..... 168
3.1.1. Représentation usuelle..... 169
3.1.2. Représentation linéaire usuelle..... 169
3.1.3. Représentation linéaire..... 169
3.1.4. Représentation sous forme d'arbre binaire..... 169
3.1.5. Traduction d'une représentation linéaire en arbre d'expression..... 170
3.2. Analyse lexicale..... 171
3.3. Analyse syntaxique..... 171
3.3.1. Définition de la grammaire algébrique..... 172
3.3.2. Élimination de la récursivité à gauche..... 173
3.3.3. Conditions de déterminisme..... 174
3.3.4. Algorithme d'analyse..... 177
3.3.5. Étiquette des arbres d'expression obtenus par l'analyseur..... 178

4. Transformation des expressions algébriques..... 180
4.1. Transformation et règles de réécriture..... 182
4.2. Classes de transformations et terminaison..... 182
4.3. Transformations dans Pépinière..... 183
4.3.1. Types de règles..... 183
4.3.2. Heuristiques et terminaison des transformations..... 186
4.3.3. Discussion..... 187
4.4. Mécanisme d'inférence : unification..... 189
4.4.1. Principe de l'algorithme d'unification..... 190
4.4.2. Exemple d'unification..... 191
4.4.3. Mise en œuvre de l'unification..... 194
4.4.4. Modélisation des règles de réécriture..... 195
4.5. Traitement particulier des règles de réécriture de type 6 et 7..... 196
4.5.1. Poids d'un arbre..... 197
4.5.2. La procédure « AjoutTermes »..... 197
4.5.3. Exemples..... 199

5. Générateur des solutions anticipées..... 200
5.1. Construction de l'arbre des solutions anticipées..... 201
5.2. Modèle général de l'arbre des solutions..... 203

6. Comparaison des expressions algébriques..... 204
6.1. Arbres d'expression équivalents..... 204
6.1.1. Définitions..... 207

6.2. Procédures de transformation des arbres d'expression 208
6.2.1. Remplacement des soustractions 208
6.2.2. Nœuds dont les deux fils sont des feuilles 209
6.2.3. Nœuds dont les deux fils sont de poids différents 209
6.2.4. Sommes de plusieurs termes et produits de plusieurs facteurs 209
6.3. Utilisation de la comparaison d'expressions 210

7. Conclusion 210

Chapitre 7 : Le système PépiGen

1. Introduction 215

2. Diagrammes de cas d'utilisation 216
2.1. Créer un exercice 216
2.2. Composer un test diagnostic 217

3. Architecture fonctionnelle de PépiGen 218
3.1. Réalisation informatique 218
3.2. Architecture fonctionnelle 219
3.2.1. Le système auteur 219
3.2.2. Interpréteur et diagnostiqueur 220

4. Architecture logicielle 221
4.1. Couche domaine 221
4.2. Couche présentation 222
4.3. Couche application 223
4.4. Couche persistance 224

5. Génération des deux représentations d'un programme de calcul 225

6. Génération de la grille de codage 227
6.1. Codage associé à une branche de l'arbre des solutions 230
6.2. Génération de la grille de codage 230
6.2.1. Solutions correctes non optimales 233
6.2.2. Solutions partielles 233
6.2.3. Solutions incorrectes 233

7. Persistance des données 236
7.1. Structure du fichier XML « ModelesClasseExercices » 236
7.1.1. Racine du schéma XML du fichier « ModelesClasseExercices » 236
7.1.2. Schéma XML des données d'indexation didactique 237
7.1.3. Schéma XML de la classe GenerationEnonce 238
7.1.4. Schéma XML de l'interface abstraite 239
7.1.5. Schéma XML de la grille de codage 241
7.2. Structure du fichier XML « BanqueExercices » 244

8. Interpréteur d'exercice : PépiTest 246
8.1. Principales fonctionnalités 246
8.2. Exemple 247

9. Diagnostiqueur : PepiDiag 249
9.1. Principales fonctionnalités 249
9.2. Algorithme de diagnostic de la classe d'exercices « Preuve et programme de calcul » 250

9.3. Tests 251
10. Conclusion 255

Chapitre 8 : Conclusion

1. Introduction 259
2. Vers un métamodèle d'évaluation multicritère automatique 260
3. Évaluation multicritère des raisonnements algébriques 262
4. Conception d'une chaîne logicielle pour le diagnostic cognitif 264
5. Conclusion 267

Bibliographie

1. Bibliographie 269
2. Références sur le WEB. 286

Annexes

Annexes A : Modèle conceptuel

A.1. Dimensions et critères d'évaluation locaux aux réponses d'un exercice 293
A.2. Liste des capacités par composantes de la compétence algébrique 294
A.3. Description des classes d'exercices 295

Annexes B : Fichiers de tests Pépinière

B.1. Interface de tests de Pépinière 299
B.1.1. Réduction de l'expression (x+6)3 -3x
B.1.2. Réduction de l'expression (4x+6)/2-2x
B.1.3. Réduction de l'expression ((x+8)*3-4+x)/4+2-x
B.1.4. Réduction de l'expression ((3x-6)*2-3x)/3-x

B.2. Arbre des solutions anticipées pour la réduction de l'expression (x+6)*3-3x 301
B.2.1. Représentation « fils-aîné-frère-droit » de l'arbre des solutions anticipées
B.2.2. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (x+6)*3-3x

B.3. Arbre des solutions anticipées pour la réduction de l'expression (4x+6)/2-2x 302
B.3.1. Représentation « fils-aîné-frère-droit » de l'arbre des solutions anticipées
B.3.2. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (4x+6)/2-2x

B.4. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression ((3*x*-6)*2-3*x*)/3-*x* (exercice créé par une enseignante) 305

B.5. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (*x*+3x*+4)/4-1 (exercice créé par un enseignant) 306**

B.6. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression ((*x*+8)*3-4+*x*)/4+2-*x* (Pépite originel) 308

B.7. Transformation de deux arbres en arbres équivalents 316

B.7.1. Exemple d'un exercice de la classe « Preuve et programme de calcul »

B.7.2. Exemple d'un exercice de la classe « Déterminer si des expressions algébriques du second degré sont égales »

Annexes C : Modèles et tests de PépiGen

C.1. Grammaire de la palette de l'exercice « Preuve et programme de calcul » 323

C.1.1. description des états 323

C.1.2. Représentation de l'automate 325

C.2. Règles de réécriture correctes et erronées 326

C.2.1. Représentation de la structure du fichier XML des régles de réécriture 326

C.2.2. Schéma du fichier XML des régles de réécriture 326

C.2.3. Tableau des régles de réécriture utilisées dans le processus d'unification 327

C.2.4. Fichier XML des règles 329

C.3. Copies d'écrans de PépiGen – Programme de calcul : (x+6)3 -3x 333

C.3.1. Saisie d'un énoncé 333

C.3.2. Analyse des réponses : « Solutions correctes » 333

C.3.3. Analyse des réponses : « Solutions correctes non attendues » 334

C.3.4. Analyse des réponses : « Solutions partielles » 334

C.3.5. Analyse des réponses : « Solutions incorrectes » 335

C.4. Copies d'écrans de PépiGen – Programme de calcul ((3*x*-6)*2-3*x*)/3-*x* 335

C.4.1. Saisie d'un énoncé 335

C.4.2. Analyse des réponses : « Solutions correctes » 336

C.4.3. Analyse des réponses : « Solutions correctes non attendues » 336

C.4.4. Analyse des réponses : « Solutions incorrectes » 337

C.5. Copies d'écrans de PépiGen – Programme de calcul (*x*+3x*+4)/4-1 ..... 337**

C.5.1. Saisie d'un énoncé 337

C.5.2. Analyse des réponses : « Solutions incorrectes » 338

C.6. Copies d'écrans de PépiGen – Programme de calcul ((*x*+8)*3-4+*x*)/4+2-*x* 338

C.6.1. Saisie d'un énoncé 338

C.6.2. Analyse des réponses : « Solutions Correctes » 339

C.6.3. Analyse des réponses : « Solutions correctes non attendues » 339

C.6.4. Analyse des réponses : « Solutions incorrectes » 340

C.7. Schéma XML d'une classe d'exercice 340

C.7.1. Structure de « ClasseExercice » 340

C.7.2. Schéma XML de « ClasseExercice » 341

C.7.3. Structure de « IndexationDidactique » 341

C.7.4. Schéma XML de « IndexationDidactique » 342

C.7.5. Schéma XML de « GenerationEnonce » 342
C.7.5.1. Schéma XML correspondant au type « ContenuType » 342
C.7.5.2. Structure de « GenerationEnonce » 344
C.7.5.3. Schéma XML de « GenerationEnonce » 345
C.7.6. Structure de « InterfaceAbstraite » 346
C.7.7. Schéma XML de « InterfaceAbstraite » 346
C.7.7.1. Structure de « QuestionSimple » 348
C.7.7.2. Schéma XML de « QuestionSimple » 349
C.7.7.3. Schéma XML de « ChoixMultiple 350
C.7.7.4. Schéma XML de « ChoixExclusif » 350
C.7.8. Classe « GenerationGrilleCodage » 351
C.7.8.1. Structure de «GenerationGrilleCodage » 351
C.7.8.2. Schéma XML de « GenerationGrilleCodage » 352
C.7.8.3. Schéma XML de « ReponseQuestionFermee » 352
C.7.8.4. Structure de « Solutions » 353
C.7.8.5. Schéma XML de « Solutions » 354
C.7.8.6. Structure de « SolutionType » 355
C.7.8.7. Schema XML de « SolutionType » 355

C.8. Structure de « ModelesClasseExercices » 355

C.9. Fichier XML contenant deux modèles d'exercices 356

C.10. Base d'exercices 362
C.10.1. Structure de la base d'exercices 362
C.10.2. Schéma XMLde la base d'exercices 362
C.10.3. Structure de « Exercices » 362
C.10.4. Schéma XMLde « Exercice » 363
C.10.5. Structure de Solutions 364

C.11. Clones générés par PépiGen -Programme de calcul : (x+6)3 -3x 365
C.11.1. Programme de calcul : ((x*3 - 6)*2 - 3*x)/3 - x 368
C.11.2. Programme de calcul : (x + 3 * x + 4)/ 4 –1 373
C.11.3. Programme de calcul : (x * 4 + 6)/ 2 - 2 * x 381

C.12. Le diagnostiqueur : Tests 388

Annexes D : Implémentation du module auteur PépiGen

D.1. Mise en œuvre des cas d'utilisation 399

D.2. Diagrammes d'interaction : cas d'utilisation « Créer un exercice » 400
D.2.1. Action « Choisir une classe d'exercices » 400
D.2.2. Action « Saisir les paramètres d'un exercice » 401
D.2.3. Action « Voir l'exercice » 402
D.2.4. Action « Voir la grille de codage des réponses » 403
D.2.5. Action « Enregistrer un exercice » 405

D.3. Cas d'utilisation « Composer un test diagnostic » : PépiGenTest 406
D.3.1. Premier niveau d'utilisation : « Choisir un test existant » 407
D.3.2. Deuxième niveau d'utilisation : « Modifier un test existant » 407
D.3.3. Étude des actions « Enregistrer un test » et « Administrer un test » 408

Chapitre 1

Introduction Générale

1. Introduction 3
2. Contexte de recherche 4
3. Le cadre : le projet Lingot 5
4. Problématique et questions de recherche 6
5. Plan de la thèse 7

1. Introduction

Dans les systèmes éducatifs tant français qu'étrangers, un fort courant se dessine visant à personnaliser les enseignements et à différencier les parcours d'apprentissage en s'appuyant sur des évaluations diagnostiques [I.G.E.N. 2007], [PISA 2006]. Cette problématique de différenciation pose aux chercheurs deux questions fondamentales. La première concerne la compréhension des mécanismes d'apprentissage en tenant compte de la diversité cognitive des apprenants. La deuxième est de concevoir des outils permettant de gérer la complexité des facteurs à prendre en compte, complexité qui ne peut être appréhendée sans bénéficier de la puissance des technologies de l'information et de la communication. Ces questions très générales ne peuvent être abordées que dans un cadre pluridisciplinaire. S'agissant d'apprentissage humain, elles concernent bien évidemment les chercheurs en sciences humaines et sociales. Elles concernent aussi les chercheurs en informatique qui doivent produire des modélisations pour construire des outils supportant les démarches mais aussi des outils permettant de recueillir des données pour mettre au point ces démarches. Notre travail en informatique se situe dans le cadre pluridisciplinaire des recherches en EIAH [Balacheff 1994] et a pour but à moyen terme la modélisation et la mise en œuvre de systèmes informatiques supportant la différenciation des parcours d'apprentissage. De notre point de vue, une telle différenciation ne peut s'effectuer qu'avec une compréhension fine des connaissances et savoir-faire des élèves qui ne peut se faire que par l'analyse de raisonnements complexes. Des travaux ultérieurs nous ont montré la nécessité de disposer de banques d'exercices de diagnostic permettant l'analyse automatique et multicritère de réponses ouvertes exprimant des raisonnements complexes à différents niveaux scolaires.

L'objectif de notre travail de thèse est de concevoir et réaliser PépiGen, un outil auteur pour créer des banques d'exercices de diagnostic, c'est-à-dire des exercices et les grilles d'analyse des réponses des élèves à ces exercices. Pour cela nous nous appuyons sur des travaux de didactique pour modéliser d'une part des classes d'exercices et les grilles d'analyse associées et, d'autre part, pour mettre au point un logiciel capable d'analyser automatiquement non seulement les réponses simples, mais aussi les raisonnements algébriques complexes.

Dans ce chapitre d'introduction, nous commençons par présenter le contexte scientifique de notre recherche et la genèse de notre travail. Puis, nous esquissons notre problématique et dégageons les questions de recherche qui nous ont préoccupées dans cette thèse ainsi que les objectifs que nous nous sommes fixés. Nous décrivons ensuite notre approche méthodologique

avant d'exposer le plan de ce mémoire. Chacun de ces points est repris plus largement dans les chapitres suivants.

2. Contexte de recherche

Les manuels d'Interactions Homme Machine distinguent les logiciels adaptatifs et les logiciels adaptables [Simonin et Carbonell 2006]. Dans notre contexte, un logiciel adaptatif propose un parcours différencié et un logiciel adaptable permet à un utilisateur (enseignant, tuteur, apprenant) de composer un parcours selon ses besoins. Dans le premier cas, le logiciel construit un parcours, dans le deuxième cas il peut être plus facile pour l'utilisateur d'adapter un parcours élaboré automatiquement que de l'inventer de toute pièce. Dans les deux cas de figure, nous faisons l'hypothèse que la proposition d'un parcours pertinent s'appuie sur un raisonnement prenant en compte la situation d'apprentissage et des informations sur l'apprenant. Cette hypothèse est étayée par un certain nombre de recherches concernant, par exemple, les apprentissages scolaires où il a été démontré que les élèves se sont forgés des connaissances qui peuvent faire obstacle à des apprentissages plus élaborés. A titre d'exemple, dans le domaine qui nous intéresse, l'algèbre élémentaire, de nombreux élèves considèrent le signe égal comme une annonce de résultat et, de ce fait, ne savent pas manipuler des expressions algébriques équivalentes.

Cette hypothèse sous-tend de nombreux travaux menés sur la modélisation de l'apprenant dans les Tuteurs Intelligents (e.g. [Wenger 1987], [Balacheff 1994], [Hibou et Py 2006]), plus particulièrement pour la conception de systèmes de diagnostic cognitif des apprenants. Sur ce thème de recherche, la principale difficulté est d'analyser les productions des apprenants sur des tâches complexes. Si les réponses préformatées sont assez facilement analysables automatiquement, de telles réponses contraignent les productions de l'apprenant et peuvent empêcher l'expression et la détection de ses conceptions personnelles. L'analyse des réponses ouvertes n'a pas de solution générale et des recherches ont été menées pour étudier ce problème de façon spécifique à différents domaines (e.g. [VanLehn et al. 2005a, Shapiro 2005], [Hakem et al. 2005], [Chaachoua et al. 2007], [Prévit et al. 2004]). Notre travail se situe principalement sur ce premier axe de recherche qui consiste à établir un diagnostic cognitif à partir de l'analyse automatique des réponses d'apprenants à un ensemble de questions qui sont soit des questions ouvertes, soit des questions aux réponses préformatées.

Le développement de tels systèmes est très coûteux en temps et en expertise. Après une première génération de travaux visant à prouver la faisabilité de tels diagnostics, des travaux

actuels cherchent à capitaliser les expériences pour concevoir des systèmes auteurs permettant de produire plus facilement des outils de diagnostic. En effet, pour envisager la dissémination des travaux de recherche dans le système éducatif, les développements doivent avoir une certaine ampleur. Les projets de recherche qui ont comme objectif d'être utilisés dans des contextes écologiques (ibid.) ont été confrontés au développement d'outils pour faciliter le passage à l'échelle. Des systèmes auteurs génériques sont conçus pour élaborer des évaluations à base de réponses préformatées et des standards sont en cours d'élaboration pour mettre en œuvre de tels systèmes. Par contre l'analyse de réponses ouvertes nécessite des outils spécifiques au domaine et aux objectifs de modélisation. Notre deuxième axe de recherche concerne les systèmes auteurs pour produire des outils de diagnostic cognitifs.

3. Le cadre : le projet Lingot

Cette recherche se situe dans le cadre du projet Lingot, projet pluridisciplinaire regroupant des informaticiens, des didacticiens des mathématiques, des psychologues ergonomes, des enseignants de mathématiques et des formateurs d'enseignants [Delozanne et al 2002a]. L'objectif du projet Lingot est de concevoir et mettre à la disposition des enseignants de mathématiques des outils qui leur permettent de prendre en compte la diversité cognitive de leurs élèves afin de gérer à la fois leur classe et les apprentissages.

Le point de départ du projet est une recherche en didactique des mathématiques menée par B. Grugeon [Grugeon 1995] qui a proposé un modèle multidimensionnel de la compétence en algèbre élémentaire ainsi qu'un outil de diagnostic papier-crayon permettant d'analyser les difficultés des élèves non pas en terme d'erreurs mais en terme de conceptions, correctes ou non, construites par les élèves et en terme de cohérences de fonctionnement. Dans le cadre du projet Pépite, S. Jean a conçu et réalisé un premier prototype, lui aussi appelé Pépite [Jean 2000], qui montre qu'il est possible d'automatiser cet outil de diagnostic. Or les expérimentations et les retours d'usage de ce logiciel [Delozanne et al. 2002b] ainsi que des études ergonomiques [Rogalski 2005] ont montré d'une part un intérêt des enseignants pour une approche par compétences et, d'autre part, qu'il fallait concevoir des outils de diagnostic qui puissent être utilisés à différentes étapes de la construction des compétences et à différents niveaux de classe pour qu'ils puissent s'intégrer dans l'activité des enseignants. Notre objectif est donc de concevoir un système qui supporte différentes situations de diagnostic. Il ne s'agit pas seulement d'améliorer le premier logiciel selon une méthode de conception itérative permettant de répondre progressivement aux attentes des utilisateurs, mais d'étudier dans quelle mesure nous pouvons

proposer des modélisations pour concevoir un outil générique en ce sens qu'il permette de générer des exercices de diagnostic .

4. Problématique et questions de recherche

L'objectif du travail présenté dans cette thèse est la conception et la réalisation d'un système, PépiGen, qui permet à un auteur non informaticien (enseignant ou didacticien des mathématiques), de créer des exercices de diagnostic de compétence pour constituer des banques d'exercices de diagnostic.

Notre thèse consiste à proposer de partir de tests diagnostics validés pour définir des modèles paramétrés permettant à un logiciel de cloner les exercices de ces tests. Le clonage consiste à générer des questions équivalentes du point de vue du diagnostic, mais aussi à générer l'analyse automatique des réponses à ces questions, que ces réponses soient préformatées ou libres.

En tant que concepteur de PépiGen nous sommes donc confrontés à trois problèmes :

1. Caractériser des exercices équivalents du point de vue du diagnostic pour définir les classes d'exercices permettant de générer des clones.
2. Produire une banque d'exercices en générant des instances de ces classes d'exercices.
3. Analyser automatiquement les réponses ouvertes quand la technologie courante actuelle se limite à l'analyse de réponses préformatées ou simples.

Plus précisément notre travail vise à explorer les questions de recherche suivantes :

1. Comment caractériser des exercices équivalents du point de vue du diagnostic pour mettre en évidence des classes d'exercices ? Comment modéliser ces classes ?
2. Sur quels modèles de calcul formel s'appuyer pour générer puis analyser automatiquement des raisonnements algébriques complexes, qu'ils soient corrects ou inadaptés ? Comment les mettre en œuvre pour une évaluation multicritère automatique de réponses ouvertes complexes ?
3. Comment mettre en œuvre une chaîne logicielle permettant à des non informaticiens d'alimenter ces banques d'exercices ?

5. Plan de la thèse

Dans une première partie, nous situons notre travail par rapport aux travaux dans le domaine. Nous commençons dans le chapitre 2 par présenter le contexte de notre recherche : le projet Lingot. Ensuite, dans le chapitre 3, nous présentons des travaux sur la génération d'exercices et sur le diagnostic cognitif dans des problématiques voisines des nôtres. Dans les plates-formes de e-learning certains travaux s'intéressent à la génération de tests d'évaluation : nous étudions les modélisations proposées par les équipes travaillant autour de la spécification IMS-LD, QTI. Enfin nous étudions en quoi les langages de modélisation pédagogique nous ont été utiles pour résoudre nos problèmes.

Dans la seconde partie, nous présentons de façon approfondie notre travail en nous appuyant sur l'étude précédente. Nous commençons dans le chapitre 4 par approfondir les objectifs et la problématique de notre travail à la lumière du contexte scientifique présenté dans les chapitres 2 et 3. Le chapitre 5 décrit le modèle conceptuel des classes paramétrées d'exercices qui permettent de générer les exercices et le diagnostic associé. Le chapitre 6 détaille la représentation des expressions algébriques et Pépinière, un composant logiciel de calcul formel dédié au diagnostic de compétence, réutilisable, que nous avons mis au point. Le chapitre 7 présente PépiGen, le logiciel qui met en œuvre ce modèle conceptuel.

Le chapitre 8 résume les apports de cette thèse, revient sur les questions de recherche et dresse les perspectives de ce travail.

Les annexes A, B, C et D illustrent ou complètent respectivement les chapitres 5, 6 et 7. Ils présentent des tests et des copies d'écrans et la documentation nécessaire à la reproduction des modèles que nous avons mis en œuvre.

Chapitre 2

Le projet Lingot

1. Introduction 11

2. Exemples 14
2.1. Côté élève 14
2.2. Côté enseignant 16
2.2.1. Scénario d'utilisation 17
2.2.2. État actuel du projet 17

3. Fondements didactiques du projet (1990-1995) 18
3.1. Un modèle de la compétence algébrique à la fin du collège 18
3.2. Un outil de diagnostic papier-crayon 18

4. Le premier logiciel Pépite (1996-2000) 19
4.1. PépiTest, le logiciel élève 20
4.2. PépiDiag, le logiciel de diagnostic automatique 22
4.3. PépiProf, le logiciel enseignant 22

5. Tests de la première version de Pépite (2000-2002) 24
5.1. Différents contextes d'usage 25
5.2. Le point de vue des enseignants 26
5.3. Retour sur les hypothèses et les questions de recherche 28
5.3.1. PépiTest 28
5.3.2. PépiDiag 29
5.3.3. PépiProf 29
5.4. Leçons et questions 30
5.4.1. Définir et caractériser des banques de tests 30
5.4.2. Diagnostic assisté ou diagnostic automatique ? 30
5.4.3. Test systématique ou adaptatif ? 31
5.4.4. Exploitation du diagnostic 31

6. Vers un diagnostic automatique plus fiable 32
6.1. Vers une analyse automatique des raisonnements algébriques 33
6.2. Diagnostic automatique des réponses à cet exercice 34

7. Vers un assistant à l'exploitation du diagnostic 36
7.1. Des profils aux stéréotypes 36
7.2. Un bilan avec l'élève 40
7.3. Des activités d'apprentissage paramétrées par les compétences 40
7.4. Tests diagnostics pour d'autres niveaux scolaires 42

8. Conclusion 43

1. Introduction

Le projet Lingot est un projet qui se situe dans le domaine de recherche sur les EIAH (Environnements Informatiques pour l'Apprentissage Humain). Les objectifs sont doubles. Le premier objectif est de concevoir et de mettre en œuvre des situations d'apprentissage de l'algèbre incluant l'utilisation d'environnements informatiques pour permettre aux enseignants de prendre en compte la diversité cognitive des élèves pour réguler les apprentissages dans le cadre de la scolarité obligatoire. Le deuxième objectif est de fournir aux chercheurs des outils d'observation systématique permettant d'étudier, sur le long terme, les effets sur l'apprentissage des enseignements dispensés. L'approche du projet consiste à fonder la conception et les modélisations informatiques sur un modèle multidimensionnel de la compétence algébrique issu d'une analyse épistémologique, cognitive, sémiotique et didactique de l'enseignement et de l'apprentissage de l'algèbre à ce niveau scolaire [Grugeon 1995]. C'est donc fondamentalement un projet pluridisciplinaire qui regroupe des chercheurs en informatique (du LIUM et du LIP6) et en didactique des mathématiques (de Didirem) ainsi que des enseignants et des formateurs d'enseignants (des IUFM de Rennes, Créteil et Amiens). Dans le cadre d'un projet financé par le programme Cognitique du Ministère de la Recherche (appel d'offres « L'apprentissage et ses dysfonctionnements »), l'équipe a bénéficié du concours d'ergonomes (Jeanine Rogalski, CNRS, et son équipe) et d'une linguiste (Sylvie Normand alors à Dialang).

Nous avons choisi le domaine de l'algèbre car c'est le verrou d'accès à l'enseignement supérieur scientifique, et, pour beaucoup d'élèves un obstacle insurmontable. Notre projet repose sur l'hypothèse que les réponses des élèves à un ensemble de problèmes bien choisis révèlent des cohérences dans leurs raisonnements. La compréhension de ces cohérences devrait aider les enseignants à réguler les situations d'apprentissage. Dans notre approche, les réponses des élèves ne sont pas analysées seulement en terme d'erreurs ou d'absence de savoir (ou de savoir-faire) mais comme des indicateurs de conceptions qui témoignent du développement de leur pensée algébrique, et qui, parfois, sont inadaptées et font obstacle à l'apprentissage. Détecter ces cohérences est une tâche très difficile pour laquelle les enseignants ne sont pas formés et pour laquelle, surtout, ils ne sont pas « outillés ». De nombreux enseignants ont mis au point des grilles de compétences qu'ils gèrent à la main pour suivre individuellement la progression de leurs élèves. Mais la lourdeur de cette gestion manuelle rend cette pratique difficile à généraliser et à maintenir dans le temps. Nous faisons l'hypothèse que des logiciels pourraient « outiller » ou « instrumenter » l'activité des enseignants. D'une part, ils permettraient de rendre ces

pratiques « d'enseignement sur mesure » écologiquement viables et, d'autre part, ils permettraient de diffuser dans le corps enseignant les résultats de recherche, tant au plan français qu'international, qui mettent en évidence des obstacles et des leviers pour l'apprentissage de l'algèbre. La métaphore est ainsi de rechercher dans le fonctionnement des élèves les granules de connaissances (« les pépites ») sur lesquels s'appuyer pour leur permettre de construire des connaissances nouvelles (« les lingots »).

Le fondement didactique du projet est un modèle multidimensionnel de la compétence en algèbre élémentaire mis au point pour étudier l'enseignement dispensé en fin de collège ou début de lycée [Grugeon 1995]. Les différentes compétences (adéquates ou non) qu'un élève a construites sont mises en relation avec les compétences attendues par l'institution. Ceci permet d'obtenir un « profil de l'élève en algèbre » qui met en évidence les difficultés des élèves mais aussi des leviers pour l'apprentissage. À partir de ces analyses, l'équipe du projet Lingot construit des situations d'apprentissage permettant de présenter de nouveaux concepts ou de faire évoluer les conceptions des élèves vers les compétences attendues. Le point clé est de permettre au professeur de mettre en œuvre, dans sa classe, des stratégies d'apprentissage différenciées, ce qui est très important, spécialement pour des élèves en difficulté en algèbre.

Du point de vue informatique, le premier problème consiste à mettre au point des modèles formels pour systématiser et rendre interprétables par des machines, les modèles discursifs et descriptifs mis au point par les didacticiens. Ceci pose à la fois des problèmes de représentations des connaissances, sujet de recherche actif en EIAH (connaissances sur le domaine, sur les conceptions des élèves, sur les stratégies pédagogiques) et des problèmes de génie logiciel (interopérabilité, généricité, maintenance). Le second problème, dual du premier, est de faire en sorte que ces modèles exécutables, ne soient pas des boîtes noires mais puissent être interprétés, transformés, complétés, modifiés par des humains qu'ils soient élèves, enseignants ou chercheurs. Ce second problème relève des méthodes et des techniques de l'Interaction Humains-Machines.

Pour résoudre ces problèmes, l'équipe du projet Lingot a travaillé sur trois axes de recherche : un axe diagnostic des compétences, un axe apprentissage et un axe instrumentation de l'activité des enseignants de mathématiques. L'axe diagnostic constitue le projet Pépite, c'est l'axe qui a été développé prioritairement et sur lequel se situe notre travail de thèse. Il s'agit de mettre au point une chaîne logicielle permettant de faire passer des tests à des élèves pour obtenir un profil cognitif pour chaque élève de la classe mais aussi pour la classe dans son ensemble. En ce qui concerne l'apprentissage, notre objectif est d'associer à chaque type de

profil cognitif des situations d'apprentissage susceptibles de les faire évoluer. En ce qui concerne l'axe instrumentation de l'activité de l'enseignant, notre objectif est de concevoir des logiciels permettant aux enseignants d'intégrer le diagnostic dans leur pratique de classe.

Les sections qui suivent retracent les grandes étapes du projet. Elles reprennent largement des publications de l'équipe, en particulier [Delozanne et Grugeon 2003, Delozanne et al. 2005] et un rapport interne que nous avons co-rédigé E. Delozanne et moi-même [Delozanne et Prévit 2007]. Attardons nous d'abord, dans la section 2, sur des exemples pour mieux comprendre la démarche adoptée, puis, survolons l'ensemble du projet. Tout d'abord, dans la section 3 nous présentons le point de départ du projet qui a démarré au début des années 1990 par une étude didactique, rapidement relayée par un travail en commun avec des informaticiens du LIUM (E. Delozanne et P. Jacoboni). Il a conduit à la réalisation d'un premier logiciel qui porte le même nom que le projet : le logiciel Pépite (Jean 2000) présenté dans la section 4. De 2000 à 2002 de nombreuses expérimentations ont été menées et nous en tirons les leçons dans la section 5. Par la suite, l'équipe a cherché à fiabiliser et à étendre le diagnostique automatique (section 6) et a mené une analyse de l'exploitation par des enseignants ou par des élèves des profils établis par Pépite (section 7). En conclusion nous situons notre thèse dans les recherches actuelles de l'équipe.

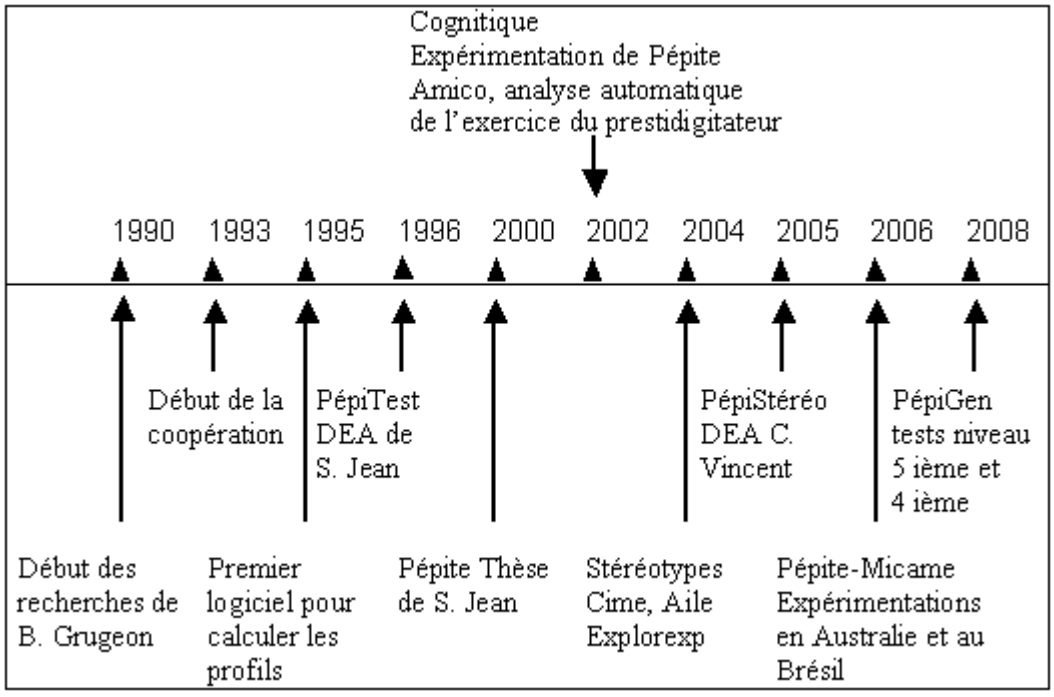


Figure 1 : Chronologie des projets Pépite et Lingot

2. Exemples

Un logiciel destiné à être utilisé dans les classes vise au moins deux catégories d'utilisateurs : les élèves et les enseignants. Nous commençons par illustrer les fonctionnalités du logiciel côté élève. Nous présentons ensuite un scénario de conception (c'est-à-dire les utilisations que nous envisageons) pour les logiciels sur lesquels nous travaillons.

2.1. Côté élève

Le tableau 1.a résume les réponses de plusieurs élèves de troisième à un même exercice. La figure 2 présente une copie de l'écran qui pose cet exercice aux élèves. Le tableau 1.b présente, sur ces réponses, l'analyse multidimensionnelle menée par le logiciel de diagnostic Pépite actuellement opérationnel et téléchargeable gratuitement sur le site du projet [site Pépite]. C'est ce que nous appelons le diagnostic local ou encore le codage des réponses des élèves. En effet le logiciel attribue à chaque réponse et pour chaque dimension[1] d'évaluation, un code, indiqué entre parenthèses.

Énoncé			
Un prestidigitateur est sûr de lui en réalisant le tour suivant. Il dit à un joueur : " Tu penses un nombre, tu ajoutes 8, tu multiplies par 3, tu retranches 4, tu ajoutes ton nombre, tu divises par 4, tu ajoutes 2, tu soustrais ton nombre : tu as trouvé 7 ". L'affirmation est-elle vraie ? Justifie ta réponse.			
Réponses de 4 élèves			
Khemarak	Nicolas	Karine	Laurent
Soit 5 un nombre ((5+8)×3-4+5)/4+2-5=7 ? ((13)×3-4+5)/4+2-5=7 ? (39-4+5)/4+2-5=7 ? 10+2-5=7 ? 10-3=7 ? 7=7 ? Oui donc cela marche	3 + 8 = 11 11 × 3 = 33 33 - 4 = 29 29 + 3 = 32 32/4 = 8 8 + 2 = 10 10 - 3 = 7	x + 8 = 8x 8x 3 × 8x = 24+3x= 27x 27x-4 = 23x 23x+x=24x 24x/4=6x 6x+2=8x 8x-x=7	=[(x+8)×3-4+x]/4+2-x =(3x+24-4+x)/4+2-x =4x+20/4 + 2-x =x+5 +2-x =7

Tableau 1.a : Analyse automatique par le logiciel de diagnostic Pépite des réponses de 4 élèves

[1] Rappelons que suite à différents travaux (avec les ergonomes mais aussi pour adapter le test de seconde à d'autres niveaux scolaires) les dimensions ont évolué. Dans ce tableau, nous adoptons la dénomination actuelle ; sur les écrans du logiciel Pépite, figure la dénomination originelle issue des travaux de B. Grugeon en 1995.

Analyse dimensionnelle des réponses des élèves			
Justification par l'exemple (J2)	Justification par l'exemple (J2)	Justification de type formel scolaire avec application de règles fausses (J31)	Justification par l'algèbre (J1)
Pas **d'utilisation des lettres** (L5)	Pas d'utilisation des lettres (L5)	Utilisation des lettres avec utilisation de règles fausses (L3)	Utilisation correcte des lettres (L1)
Traduction par une expression globale parenthésée (T1)	Traduction par expression partielle (T2)	Traduction par expression partielle enchaînée en succession d'opérations (T4)	Traduction par expression globale parenthésée (T1)
Signe = relation d'équivalence (E1)	= annonce un résultat (E2)	= annonce un résultat (E2)	= relation d'équivalence (E1)
Ecritures algébriques : correctes (A1)	Ecritures correctes (EA1)	Identification incorrecte de + et × (assemble les termes)(EA42) règles fausses : $x + a \rightarrow x\ a$ $a\ x \pm b \rightarrow (a \pm b)\ x$ $a\ x - x \rightarrow a - 1$	erreur de parenthèse avec mémoire de l'énoncé (EA31)
Validité : Réponse invalide (V3)	Réponse invalide (V3)	Réponse invalide (V3)	Réponse invalide (V3)

Tableau 1.b : Analyse automatique par le logiciel de diagnostic Pépite des réponses de 4 élèves

Remarquons tout d'abord qu'aucune de ces réponses n'est correcte. Elles témoignent cependant de différences très importantes dans la façon dont ces élèves abordent la pensée algébrique. Remarquons ensuite que Pépite ne construit pas un profil cognitif à partir d' un seul exercice mais sur un ensemble d'exercices représentatifs de la compétence algébrique à ce niveau scolaire. Cependant, d'après les réponses à cet exercice, l'analyse didactique permet d'émettre certaines hypothèses.

- Khemarak n'est pas encore entré dans une démarche d'utilisation de l'algèbre pour prouver puisqu'il se limite à prendre un exemple ; toutefois, sa maîtrise des calculs numériques et sa façon d'appréhender de façon globale l'expression numérique et de la transformer par équivalence est un levier d'apprentissage qui devrait lui permettre d'entrer rapidement dans une démarche algébrique. Après avoir vérifié cette prédiction sur d'autres exercices du test, le conseil sera de le faire travailler avec le logiciel CIME développé par l'équipe (section 7.3). Ce logiciel fait travailler les élèves sur la mise en

équation en leur faisant « boucher des trous » dans un texte de problème en français ou dans sa représentation sous forme d'équations.

- Nicolas utilise lui aussi une preuve par l'exemple et maîtrise ces calculs simples. Par contre il utilise une démarche arithmétique en indiquant une suite de calculs où le signe égal « annonce » un résultat. Là encore il faudrait vérifier sur d'autres exercices, mais il semble nécessaire de le faire travailler sur les différentes écritures d'un même nombre. Par exemple, avec la calculatrice défectueuse, logiciel développé par l'équipe de l'université de Montréal [site GRICEA].
- Karine[2] utilise la lettre x, mais sa résolution est classée : justification par le formel scolaire car elle utilise des règles fausses bien connues des enseignants et des chercheurs en didactique. Par cette expression, nous désignons les élèves pour qui faire des mathématiques c'est appliquer des règles vides de sens. Ce type d'élèves (nous disons les élèves qui correspondent à ce stéréotype) rencontre beaucoup de difficultés. Pour les faire progresser il ne suffit pas de leur répéter les règles correctes. Un long travail est nécessaire, d'une part pour déstabiliser les règles incorrectes, et, d'autre part, pour donner du sens à l'usage des lettres, par exemple, en leur proposant de résoudre des exercices où l'algèbre est un outil de généralisation et de preuve, ou bien des exercices où l'égalité est un symbole d'équivalence et non une annonce de résultat.
- Laurent est bien rentré dans une démarche algébrique et son raisonnement prouve qu'il sait traiter des expressions équivalentes. Comme beaucoup d'élèves à ce niveau ($3^{ième}$) il ne maîtrise pas encore parfaitement l'utilisation de parenthèses mais il garde le sens des opérations. L'utilisation d'un logiciel comme Aplusix développé par des chercheurs de Grenoble [site Aplusix] lui sera certainement profitable.

2.2. Côté enseignant

Illustrons maintenant l'utilisation envisagée des outils développés dans le projet Lingot et leur intégration dans les classes. Une méthode de conception utilisée depuis quelques années pour centrer les projets sur les besoins des utilisateurs et leur activité consiste à mettre au point des scénarios. Ces scénarios ont également l'avantage de permettre de mieux faire comprendre les objectifs du logiciel développé. Ces scénarios ont été établis au sein de l'équipe après de nombreuses observations d'enseignants en classe ou en formation. Ils constituent une

[2] Rassurez vous quelques années après elle a eu son bac (tertiaire pas scientifique mais quand même) !

compilation des analyses et des observations. Nous présentons un scénario d'utilisation de Pépite qui conduit l'enseignant à organiser des activités différenciées pour ses élèves en les faisant travailler sur des logiciels conçus dans le projet Lingot ou conçus par d'autres équipes.

2.2.1. Scénario d'utilisation

Avant de commencer sa première séquence d'algèbre avec sa classe de seconde, Clémence veut savoir où en sont ses élèves et jusqu'où sont allés ses collègues de 3ième. Elle fait passer à ses élèves un test avec Pépite en le paramétrant pour un début de seconde. Elle imprime un bilan individuel pour chaque élève. Elle imprime également un bilan de l'ensemble de la classe mettant en évidence trois groupes d'élèves et proposant une liste de compétences à faire travailler. Pour les trois groupes d'élèves, le logiciel conseille de travailler sur la « reconnaissance d'expressions algébriques » (i.e. repérer les opérations et les opérandes qui les constituent). Elle décide de travailler cette compétence avec un des logiciels CIME ou AILE[3] : CIME permet de travailler sur la relation entre un énoncé de problème et sa mise en équation, AILE permet d'associer une expression algébrique à son expression en français.

De plus, elle veut différencier plus finement les exercices proposés aux élèves en jouant sur les expressions algébriques[4]. Le logiciel lui propose de constituer des sous-groupes d'élèves en fonction de leurs profils de compétence en algèbre pour organiser la différenciation.

Pendant la séance de travaux dirigés, les élèves se connectent au site, font les exercices qui leurs sont proposés par le logiciel. Celui-ci met à jour l'historique de leur travail et prépare un bilan pour leur professeur. Clémence consulte rapidement le bilan pour chacun des élèves et s'attarde sur le bilan de la classe pour préparer la synthèse qu'elle fera en cours le lendemain.

2.2.2. État actuel du projet

Dans l'état de développement actuel du projet, la première partie du scénario est réaliste et a déjà été mise en œuvre dans des classes de secondes. Mon travail de thèse a pour objectif de permettre à des enseignants d'autres niveaux (5, 4, 3) de disposer de tels outils, ou bien de permettre à Clémence de tester l'évolution des connaissances de ses élèves en fin de trimestre avec un test du même type que celui qu'elle a fait passer en début d'année. En effet, pour l'instant, l'équipe ne dispose que d'un seul test à faire passer sur machine. Mon travail de thèse

[3] Logiciels que l'équipe a développés au sein du projet (section 7.3).

[4] Les élèves réalisent le même exercice mais avec des expressions différentes (par leur nature et par la structure des expressions algébriques, aides adaptées, etc.) : la différenciation est réalisée par le logiciel en fonction des caractéristiques des élèves (nature et structure des expressions algébriques, aides adaptées, etc.)

consiste ainsi à développer des outils auteurs pour disposer de banques d'exercices de diagnostic à différents niveaux scolaires.

En ce qui concerne les deux dernières étapes du scénario, elles sont encore largement prospectives mais l'équipe travaille à les rendre réalisables. Les sections suivantes décrivent les travaux mis en œuvre dans l'équipe du projet Lingot pour concevoir et réaliser les logiciels permettant de mettre en œuvre ce scénario complet.

3. Fondements didactiques du projet (1990-1995)

Ce projet s'appuie sur un travail préliminaire de recherche en didactique des mathématiques. Nous ne présentons ici que ce qui est utile à la compréhension de notre travail sans aborder les références théoriques qui sont présentées dans [Grugeon 1995, 1997].

3.1. Un modèle de la compétence algébrique à la fin du collège

S'appuyant sur des travaux théoriques et expérimentaux menés dans le domaine et aussi sur une étude de l'activité en algèbre élémentaire d'une cohorte d'élèves sur une période de plusieurs années, B. Grugeon a établi un modèle multidimensionnel des compétences algébriques attendues des élèves en fin de collège. Les différentes dimensions considérées sont : l'utilisation des lettres (inconnue, variable, nombre généralisé, abréviation ou étiquette), la pratique du calcul algébrique, la traduction entre différents registres sémiotiques (graphique, géométrique, algébrique, langue naturelle), les types de rationalité.

3.2. Un outil de diagnostic papier-crayon

Afin de situer les élèves par rapport à ce modèle, B. Grugeon a proposé un outil de diagnostic papier-crayon. Les compétences et les difficultés des élèves en algèbre y sont analysées selon trois entrées : le type de problème (l'algèbre comme un outil de résolution de problèmes arithmétiques, de généralisation, de preuve, de modélisation de situations), les objets de l'algèbre (en particulier l'utilisation des lettres, la production et l'interprétation des expressions algébriques), le calcul formel (aspects syntaxiques, techniques, sémantiques et sémiotiques). Elle propose un test : un ensemble d'une vingtaine d'exercices avec des questions fermées et des questions ouvertes. Une grille d'analyse, établie à partir du modèle multidimensionnel des compétences, est proposée à l'enseignant (ou au chercheur) pour coder les réponses des élèves aux différentes questions du test. Ensuite, par une analyse transversale du codage obtenu, l'enseignant (ou le chercheur) construit le profil cognitif de l'élève en l'algèbre. Dans ces

premiers travaux, un profil est une description en trois niveaux de la compétence algébrique des élèves (les figures 3, 4 et 5 montrent des écrans qui ont été conçus par la suite pour afficher des extraits du profil cognitif d'un élève) :

- Une description quantitative exprimée en termes de taux de réussite et de traitements algébriques maîtrisés,
- Une description qualitative exprimée en termes de modes de fonctionnement : utilisation des lettres, calcul algébrique, conversion c'est-à-dire l'articulation entre le registre algébrique et d'autres registres sémiotiques, le type de preuve,
- Une description de l'articulation entre les différents cadres et registres (graphique, algébrique, géométrique, langage naturel) sous la forme d'un diagramme.

Ces travaux de didactique ont suscité l'intérêt d'une équipe de chercheurs du LIUM, Élisabeth Delozanne et Pierre Jacoboni, travaillant au Mans sous la direction de Martial Vivet. En effet cette modélisation, assez précise, paraissait un point de départ pour une collaboration visant à relever un défi : produire des modèles des connaissances des élèves calculables par une machine et exploitables par des enseignants. Si le problème de recherche était en lui-même intéressant, l'automatisation de l'outil de diagnostic papier est vite apparu indispensable pour permettre la dissémination des résultats de cette recherche en didactique. En effet l'outil de diagnostic papier-crayon s'est révélé trop complexe pour être utilisé par les enseignants dans les classes [Lenfant 1997].

4. Le premier logiciel Pépite (1996-2000)

Ainsi, le premier travail de recherche du projet Pépite a consisté à automatiser l'outil de diagnostic papier-crayon. Ce travail a principalement été mené par Stéphanie Jean dans le cadre de sa thèse [Jean 2000]. Il avait pour objectif de tester trois hypothèses :

- (H1) il est possible à l'aide d'un ordinateur de collecter des données riches sur les compétences des élèves,
- (H2) il est possible, à partir de ces données, d'obtenir automatiquement (ou presque), un profil cognitif des élèves,
- (H3) les profils cognitifs élaborés aident les enseignants à prendre des décisions pour leurs élèves.

En effet, les membres de l'équipe se posaient principalement trois questions de recherche :

- L'utilisation d'un clavier et d'une souris limiterait-elle l'expression des élèves, que ce soit en langue naturelle ou en algèbre ?

- Comment analyser automatiquement les réponses des élèves en leur laissant la possibilité de produire eux-mêmes des réponses, l'étude des productions étant indispensable au diagnostic ?
- Comment les enseignants accepteraient-ils un tel outil qui leur donne accès à des informations inhabituelles sur les élèves ?

Au niveau informatique, ce premier travail a abouti à la réalisation du logiciel Pépite dont la mise en œuvre permettait de tester ces hypothèses et d'étudier ces questions. Ce logiciel est composé de trois modules : PépiTest, PépiDiag et PépiProf (Tableau 2).

L'OUTIL DE DIAGNOSTIC PAPIER-CRAYON ÉLABORATION DU PROFIL COGNITIF DES ÉLÈVES	**ARCHITECTURE DE PÉPITE** PROJET PÉPITE SYSTÈME DE CONSTRUCTION DU PROFIL COGNITIF DES ÉLÈVES
TEST Acteur : élève Action : effectue un test	PÉPITEST Acteur : élève Action : effectue le test
observable papier-crayon ↓	*trace de la session* ↓
DIAGNOSTIC Acteur : enseignant Action : application de la grille d'analyse nécessitant une expertise didactique	PÉPIDIAG Acteur : système Action : interprétation heuristique par application de la grille d'analyse
valeurs des critères pour chaque tâche ↓	*valeurs des critères pour chaque tâche* ↓
PROFILS Acteur : enseignant Action : analyse transversale nécessitant une interprétation Produit : taux de réussite modalités de fonctionnement diagramme d'articulation	PÉPIPROF Acteur : système Action : interprétation algorithmique par analyse transversale Produit : taux de réussite modalités de fonctionnement diagramme d'articulation

Tableau 2 : Outil de diagnostic papier-crayon et architecture de Pépite

4.1. PépiTest, le logiciel élève

PepiTest est le logiciel destiné aux élèves. Il leur propose de résoudre sur ordinateur 22 exercices inspirés de ceux de l'outil de diagnostic papier-crayon. Il recueille les réponses des élèves aux exercices. La Figure 2 montre une réponse d'un élève à un exercice proposé par PépiTest. D'autres exemples sont étudiés au chapitre 5.

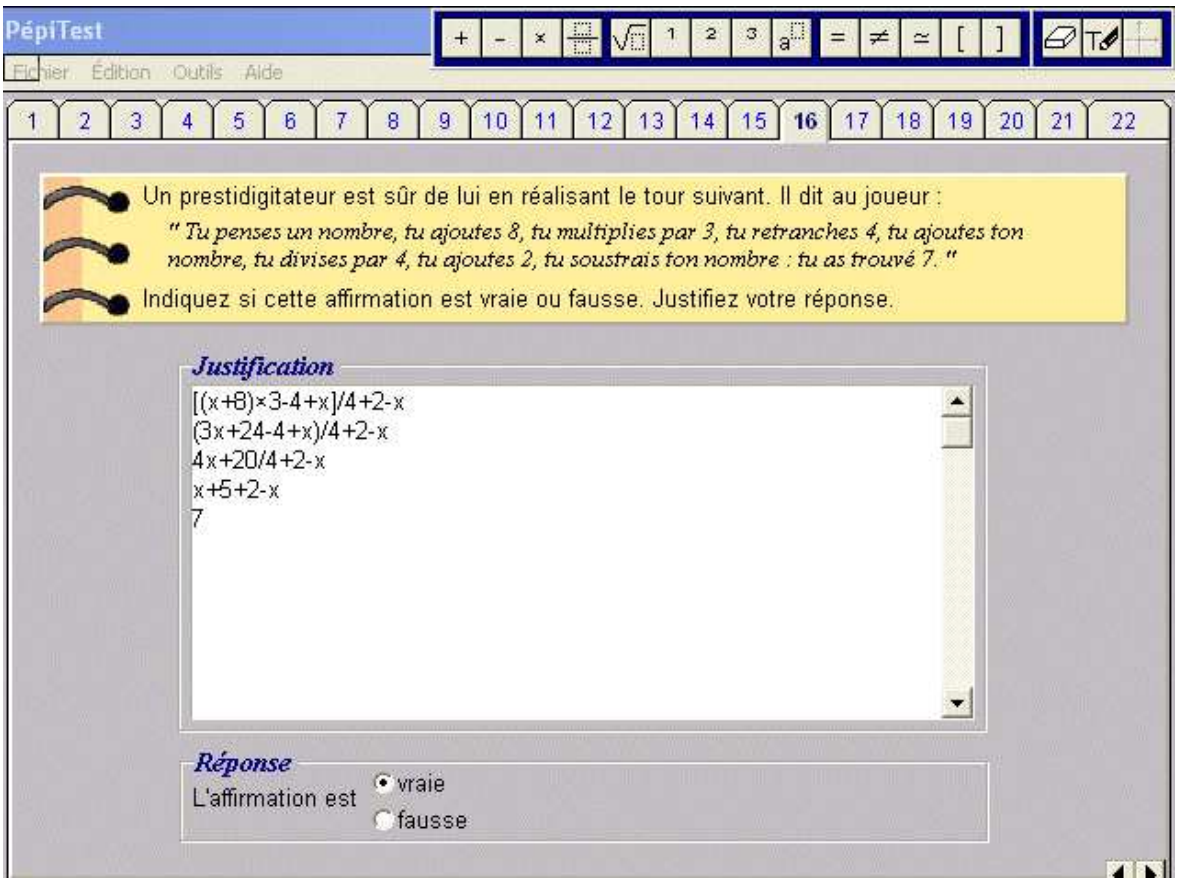


Figure 2 : Réponses de Laurent à un exercice de PépiTest

Ceux-ci sont constitués de questions fermées (réponses à choix multiples ou de réponses interactives par exemple en cliquant des zones sur un graphique) mais également de questions ouvertes exigeant des élèves la production d'expressions algébriques, de réponses en langage naturel ou de réponses combinant ces deux registres d'expression (dans le projet Lingot, nous appelons ce langage le langage mathurel). Pour les chercheurs en didactique, il est important que les élèves puissent formuler eux-mêmes leurs réponses dans leurs propres termes pour pouvoir établir un diagnostic conséquent du point de vue didactique, même si cela rend l'analyse automatique très complexe. L'interface du logiciel a été particulièrement soignée. Il est bien entendu crucial pour le diagnostic que les données recueillies permettent de décrire les compétences des élèves et non les problèmes d'utilisabilité de l'interface. En particulier, dès le début, les chercheurs en didactique des mathématiques s'interrogeaient sur les modifications des tâches mathématiques introduites par les difficultés de l'écriture des expressions algébriques sous forme linéaire avec un clavier et une souris.

4.2. PépiDiag, le logiciel de diagnostic automatique

PépiDiag est le module d'analyse des réponses. Il interprète les réponses des élèves à chaque exercice de PepiTest en appliquant des heuristiques dérivées de la grille d'analyse issue de l'analyse didactique. Comme l'outil de diagnostic papier-crayon, il associe chaque réponse d'élève à un code permettant de qualifier la réponse sur plusieurs dimensions. Nous appelons cette opération le codage des réponses des élèves. Ainsi, pour chaque élève, cette première version de PépiDiag remplit automatiquement une matrice de diagnostic de 55 lignes correspondant au nombre de questions dans PépiTest et de 36 colonnes correspondant aux différentes composantes décrites dans le modèle multidimensionnel des compétences algébriques. En fait, PépiDiag ne remplit que partiellement cette matrice car nous ne savons pas encore comment coder automatiquement toutes les réponses des élèves[5]. Les réponses fermées et les expressions algébriques simples sont analysées par cette première version. Les réponses en langage naturel et les réponses mixtes sont très partiellement analysées par recherche de mots clés. Les raisonnements algébriques (par exemple du type présentés au tableau 1) ne sont pas analysés par cette première version.

4.3. PépiProf, le logiciel enseignant

PépiProf est le logiciel destiné aux enseignants. Il établit le profil de l'élève par une analyse transversale de la matrice et le présente au professeur. Il fournit également un module pour modifier le codage des réponses de l'élève (i.e. pour modifier la matrice de diagnostic sans qu'elle apparaisse sous cette forme à l'enseignant) afin de permettre à l'enseignant de vérifier le codage effectué par le logiciel et de le corriger ou le compléter, si nécessaire. La figure 4 montre l'interface qui permet au professeur de vérifier et éventuellement corriger le diagnostic du logiciel. Dans la réponse de Laurent représentée sur la Figure 3, PépiDiag a codé : traitement incorrect, utilisation correcte des lettres, utilisation incorrecte des parenthèses menant à un résultat correct, traduction correcte du langage naturel en expression algébrique, justification par l'algèbre. S'il n'est pas d'accord avec le codage automatique, l'enseignant peut le modifier. Les figures 4, 5 et 6 montrent les profils affichés par PépiProf avec les trois descriptions du profil de Laurent.

[5] Notre travail de DEA, puis de thèse a permis d'avancer sur ce point comme nous le montrons plus loin.

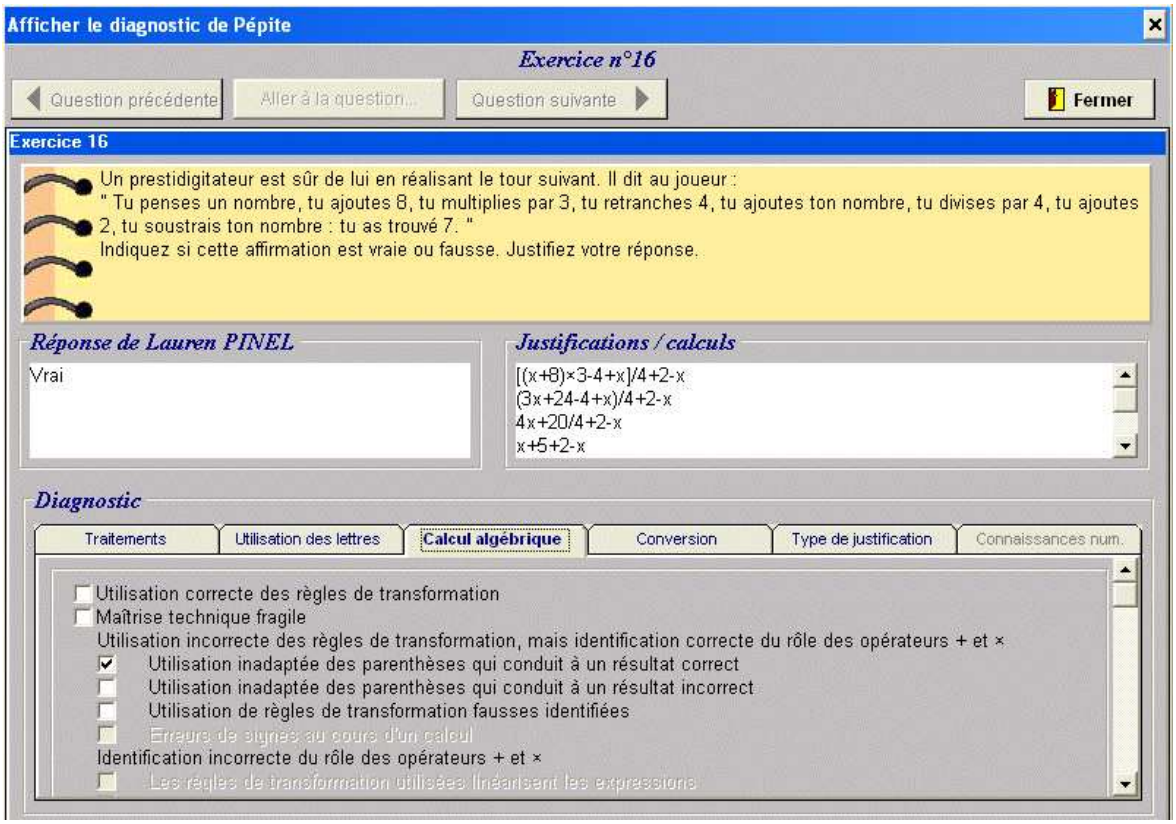


Figure 3 : PépiProf, vérification par l'enseignant du codage des réponses de Laurent

Figure 4 : PépiProf, Description quantitative du profil cognitif de Laurent

C'est sur cette version 1 de Pépite qu'ont été menées toutes les expérimentations qui sont décrites par la suite. Elles nous permettent de revenir sur les premières hypothèses et les questions de recherche.

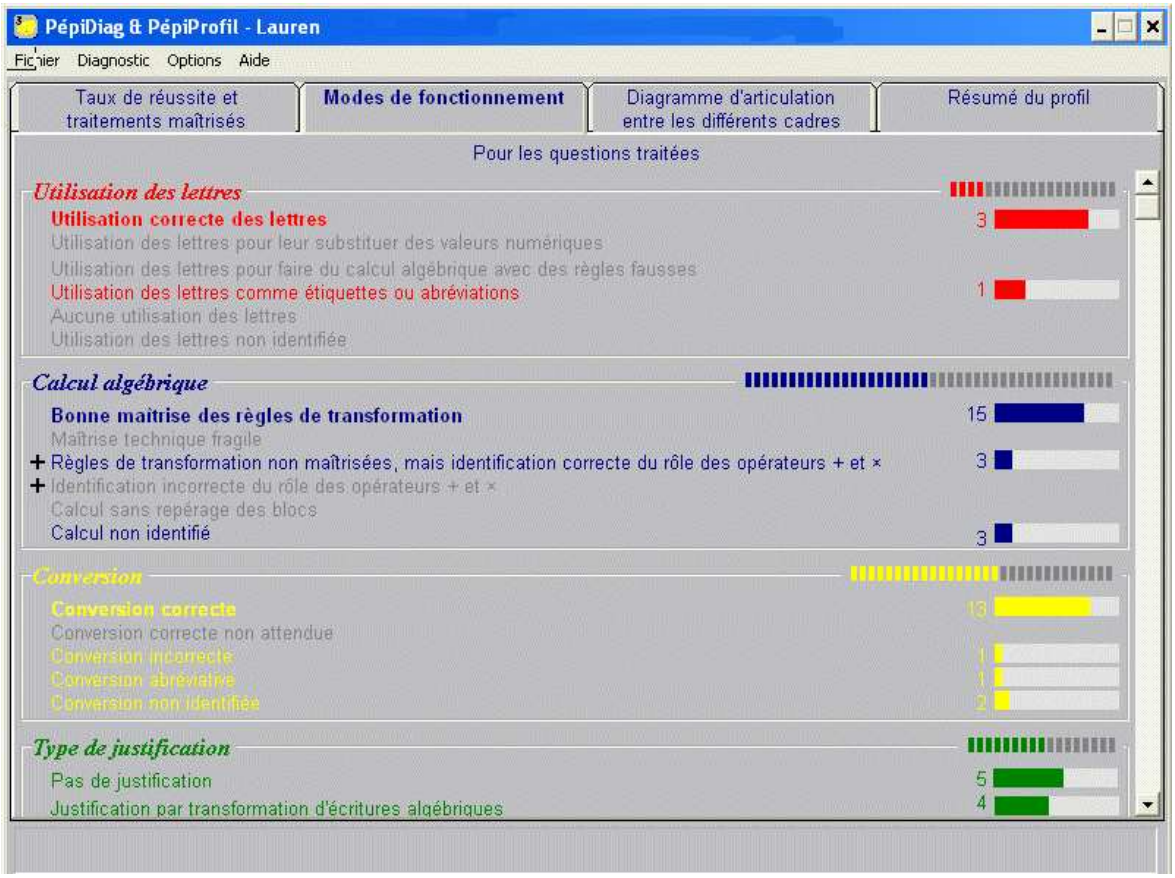


Figure 5 : PépiProf, Description qualitative du profil cognitif de Laurent

5. Tests de la première version de Pépite (2000-2002)

L'idée de départ du projet Pépite était de tenter d'automatiser (au moins partiellement) un outil de diagnostic papier-crayon construit par des chercheurs en didactique, [Grugeon 1995, Artigue et al. 2001]. Par rapport aux logiciels d'évaluation traditionnels, cet outil a deux caractéristiques principales. D'une part, il donne une description qualitative des compétences des élèves et ne se réduit pas à une analyse en terme de réussite/échec ou de listes d'erreurs, d'autre part, si l'analyse de réponses porte en partie sur des questions fermées (QCM) elle traite un grand nombre de réponses à des questions ouvertes. La version originelle de Pépite avait donc pour objectif premier de montrer la faisabilité informatique du projet [Jean et al. 1999]. L'objectif second était de disposer d'un prototype qui permette d'observer dans un contexte de classe, si les enseignants pouvaient s'approprier l'outil et comment. Dans cette section nous

résumons d'abord les utilisations de Pépite1 que nous avons étudiées (entre janvier 2000 et avril 2002) et les leçons que nous en avons tirées.

Rappelons que dans la méthodologie de conception centrée utilisateur, ces tests en cours de conception ont une valeur informative indispensable à l'équipe de conception.

5.1. Différents contextes d'usage

Nous avons observé l'utilisation de Pépite dans différents contextes [Delozanne et al. 2002b]. D'abord nous avons examiné PepiTest en laboratoire, enregistrant une élève en vidéo. Puis environ 200 élèves en fin de collège ou de seconde ont passé le test dans le cadre de leur enseignement normal de mathématiques. Pépite a été utilisé en atelier par des chercheurs ou des formateurs des enseignants. Il a aussi été utilisé dans la formation initiale ou continue des professeurs (IUFM de Créteil, Amiens, Paris, Rennes, Montpellier). Des sessions pilotes ont été menées avec des enseignants expérimentés volontaires. Enfin quelques enseignants nous ont rapporté avoir utilisé Pépite en dehors de notre présence. Le Tableau 3 synthétise ces contextes.

Contexte	Situation	Utilisateur	Nombre	Données collectées
Test d'élèves	Classe	élèves	200	Réponses d'élèves Questionnaires Observations Rapports
Recherche didactique	Recherche de régularités dans les profils d'élèves	chercheurs	3	Liste de problèmes d'utilisation ou des bogues Définition de classes de profils
Formation de formateurs d'enseignants	Etude d'un élève en algèbre, étude des compétences en algèbre	formateurs	40	Questionnaires
Formation d'enseignants	Etude d'un élève en algèbre, étude des compétences en algèbre	Stagiaires et professeur en responsabilité	100	Questionnaires Observations
Session pilote	Classes (Aide individualisée et évaluation)	Enseignants	4	Observations Rapports Cassettes audio
Utilisations spontanées	Classes	Enseignants	9	Rapports oral ou courriel

Tableau 3 : Les différents contextes d'expérimentations (1999-2002) de la version 1 de Pépite

5.2. Le point de vue des enseignants

Nous avons observé plusieurs utilisations spontanées de PépiTest non en tant qu'activité de diagnostic, mais pour mettre en place une activité d'apprentissage, en particulier pour provoquer des débats en binôme ou en classe entière. Nous avons recueillis des questionnaires auprès des enseignants. Enfin des collègues des IUFM de Rennes, Montpellier qui ont utilisé Pépite en dehors de l'équipe de conception nous ont apporté d'autres informations. Nous synthétisons ici les différents points évoqués.

Les points positifs relevés par les enseignants :

- le logiciel de test propose un large éventail d'exercices ; il donne des idées d'exercices à travailler et de compétences à faire acquérir,
- l'interface du logiciel de test est soignée et esthétique,
- le logiciel « marche » du premier coup comme indiqué sur la notice,
- l'ensemble du logiciel révèle des compétences que les enseignants n'avaient pas remarquées chez des élèves en difficulté et augmente la confiance de l'enseignant dans la réussite de ces élèves (enseignantes confirmées),
- le logiciel aide à comprendre l'esprit des nouveaux programmes de collège en algèbre (1 enseignante confirmée),
- le logiciel introduit une médiation qui permet d'instaurer une relation nouvelle avec l'élève sur ses erreurs (1 enseignante confirmée) et 6 mois après l'utilisation du logiciel : « la relation mise en place avec l'utilisation de Pépite a changé l'ambiance en aide individualisée et le regard que les élèves portaient sur ma compétence d'enseignante »,
- le logiciel apporte un éclairage neuf sur les compétences et « aide à prendre conscience des lacunes des élèves » (enseignant débutant).

Les points négatifs :

- des difficultés techniques multiples doivent être surmontées : par exemple pour accéder aux salles informatiques dans certains établissements, pour télécharger et installer le logiciel dans des salles qui ne sont pas reliées au réseau,
- des difficultés techniques supplémentaires sont liées à l'utilisation du système pour récupérer les réponses des élèves : bogues, problèmes d'utilisabilité,
- le logiciel permet de naviguer mais pour faire un retour à l'élève il faudrait pouvoir imprimer facilement,
- la complexité des profils et du vocabulaire utilisé pour les décrire est rebutante, voire rédhibitoire,

- le test est bien adapté au niveau lycée pour lequel il a été conçu, mais inadapté en collège,
- l'utilisation de Pépite par les élèves, mais aussi par l'enseignant prend trop de temps,
- le logiciel ne fournit pas encore de pistes de stratégies d'enseignement pour faire évoluer les élèves une fois que l'on a trouvé ce qui ne va pas,
- les diagnostics automatiques incomplets ou erronés obligent à un travail personnel conséquent de l'enseignant : appropriation du logiciel et de l'analyse didactique, reprise des codages, analyse des profils,
- certains enseignants disent se sentir pris en défaut parce qu'ils ne comprennent pas la description des profils, qu'ils n'ont pas réussi à manipuler le logiciel ou bien enfin parce que le logiciel leur retourne des réponses d'élèves qui les conduit à s'interroger sur leur enseignement (professeurs de collège).

Ils font également des suggestions. Comme outil d'évaluation, les enseignants pensent que Pépite est beaucoup trop coûteux en temps. Ils voudraient pouvoir choisir les exercices. Ils demandent à disposer de plusieurs tests avec différents niveaux ($4^{ième}$, $3^{ième}$, 2^{nde}) afin de pouvoir évaluer l'évolution des apprentissages. La plupart d'entre eux demandent que le logiciel prenne en charge un bilan de compétences à destination de l'élève. En effet, il est impossible pour eux de fournir une rétroaction personnelle à chaque élève et il n'est pas envisageable non plus de faire passer un test sans donner le résultat aux élèves. Certains d'entre eux demandent une « géographie de la classe » en plus des profils personnels pour situer et repérer des profils analogues et organiser les apprentissages en début d'année ou pour créer des groupes de travail. La plupart d'entre eux souhaitent qu'on leur propose des stratégies d'enseignement pour faire évoluer les compétences ou pour remédier aux difficultés qui ont été diagnostiquées. Certains ont rapporté que Pépite fait apparaître pour quelques élèves en grande difficulté des compétences qu'ils n'avaient pas perçues auparavant et cela augmente alors la confiance de l'enseignant dans les chances de succès de l'élève. Pour une enseignante, l'usage de Pépite augmente aussi la confiance des élèves envers sa compétence professionnelle : cette enseignante rapporte que selon ses élèves, travailler avec Pépite, montre qu'elle s'intéresse à eux individuellement et cherche à les comprendre avec des moyens modernes issus de la recherche.

De l'utilisation de Pépite par des chercheurs notons un fait très intéressant : les experts ne pratiquent pas le diagnostic comme ils le décrivent dans la méthode qu'ils ont proposée et qui est mise en application dans Pépite. Ils procèdent à un « diagnostic adaptatif ». Ils regardent les réponses d'élèves à un exercice significatif et, selon la réponse, ils énoncent une hypothèse générale et vont la confirmer et l'approfondir sur quelques exercices complémentaires. Ainsi lorsqu'ils cherchent à proposer une stratégie de remédiation pour un élève particulier, ils forment

leur diagnostic seulement à partir de quelques exercices et non sur le parcours systématique de l'ensemble des exercices du test.

5.3. Retour sur les hypothèses et les questions de recherche

Tous ces tests ont mis en évidence plusieurs résultats au niveau de la conception de chacun des trois modules du logiciel et aussi au niveau de l'activité de l'enseignant avec Pépite.

5.3.1. PépiTest

PepiTest recueille des données, les réponses des élèves, qui peuvent être ensuite utilisées pour le diagnostic. Premièrement, à partir de ces données, les chercheurs en didactique peuvent appliquer la grille d'analyse et construire « manuellement » le profil cognitif de l'élève. Deuxièmement, si nous considérons les réponses aux exercices de PepiTest, nous obtenons le spectre des réponses prévues par l'analyse didactique a priori. Cela signifie que le logiciel ne diminue pas l'éventail de réponses repérées par l'analyse a priori papier-crayon. Troisièmement, comme nous l'avions anticipé, les élèves ont rencontré des problèmes pour écrire les expressions algébriques « linéaires » avec clavier-souris mais ces difficultés ne les empêchent pas d'en produire. Quatrièmement, malgré des différences locales, des enseignants expérimentés identifient des cohérences de fonctionnement de leurs élèves comparables à celles identifiées avec l'outil papier-crayon [Lenfant 1997]. Ces résultats constituent, à notre sens, une première validation de la faisabilité du projet en ce qui concerne le recueil des données sur l'élève [Jean 2000, Jean-Daubias 2002]. Ils confortent en particulier nos hypothèses 1 et 2.

Du côté des enseignants, dans les sessions pilotes et les utilisations spontanées, les enseignants nous ont indiqué avoir repéré chez certains élèves des compétences ou encore des fragilités qu'ils n'avaient pas remarquées auparavant chez leurs élèves. La raison principale est que PepiTest propose des exercices sur des tâches inhabituelles dans les classes de mathématiques. C'est une des raisons pour laquelle les formateurs des enseignants de mathématiques pensent que c'est un outil utile pour aider les enseignants à prendre en compte les différents aspects de la compétence algébrique. Cependant, de nombreux enseignants trouvent le test trop long. L'équipe s'interroge sur la pertinence d'un diagnostic adaptatif ou bien sur la mise en évidence d'exercices prédictifs qui permettraient de réduire le test à une dizaine d'exercices maximum voire moins.

5.3.2. PépiDiag

Les utilisations de PépiDiag ont démontré quelques inconsistances dans la grille d'analyse didactique lorsqu'elle est appliquée de façon systématique et sans discernement par le logiciel. Par ailleurs, les utilisateurs doivent compléter 25 % du codage des réponses qui ne sont pas analysées et doivent corriger environ 10 % du codage des réponses effectuées par PépiDiag.

Le module PépiDiag de la version 1 ne permet pas un diagnostic complètement automatique et fiable. Il permet un diagnostic assisté qui apparaît fastidieux aux enseignants.

5.3.3. PépiProf

PépiProf assiste l'enseignant dans deux types de tâches : l'analyse des réponses des élèves au test (le codage) et l'étude des profils des élèves. Le module facilitant l'analyse des réponses des élèves (tâche de codage) s'est révélé adapté et est utilisé sans difficultés majeures par chacune des catégories d'utilisateurs. Il est apprécié par les enseignants stagiaires et par les formateurs. Il fournit un cadre pour interpréter les réponses des élèves. Il leur permet de comprendre les items du diagnostic quand ils sont présentés dans le contexte d'un exercice et avec les réponses des élèves. Pour les enseignants confirmés, il est ennuyeux et ils souhaitent un diagnostic automatique.

Le module pour travailler sur le profil de l'élève n'a été utilisé que par les chercheurs en didactique des mathématiques. Ils apprécient particulièrement d'accéder aux items du diagnostic de différentes manières : à partir des réponses d'élèves, à partir des items de diagnostic, à partir de la liste de questions liées à ces items. Mais ce module soulève des difficultés pour les enseignants, particulièrement pour les enseignants expérimentés. Ils ont des difficultés pour comprendre les items du diagnostic quand ils ne sont pas présentés en contexte (dans un exercice avec les réponses des élèves) lorsqu'ils ne connaissent pas le travail didactique sous-jacent. Ce module met en œuvre une expertise didactique qui peut être en contradiction avec des pratiques spontanées de diagnostic.

Dans la version 1 de Pépite, aucune stratégie d'enseignement n'est proposée pour faire évoluer le profil cognitif. Les enseignants ne sont donc pas motivés pour s'approprier un profil complexe qu'ils ne peuvent pas exploiter. Notre troisième hypothèse est donc loin d'être confortée, mais ce premier travail a permis de mieux comprendre le problème et de reformuler les questions de recherche.

5.4. Leçons et questions

Ce travail auprès des enseignants a montré la nécessité de travailler sur deux axes :

- concevoir un logiciel adaptable à différentes situations,
- étudier et instrumenter l'exploitation d'un tel outil de diagnostic dans la classe.

5.4.1. Définir et caractériser des banques de tests

En ce qui concerne l'activité des élèves, le modèle de la compétence algébrique a permis d'implémenter un logiciel qui recueille des données permettant d'aider les enseignants à identifier les difficultés des élèves en l'algèbre. La principale limitation de ce modèle vient de ce qu'il est prédéfini et spécifique à un niveau d'étude (fin de la scolarité obligatoire). Ceci nous amène à de nouvelles questions :

- Est-il possible de définir un modèle des compétences pour chaque niveau scolaire ?
- Comment permettre à des enseignants d'adapter le test à leur pratique de classe ? Comment déterminer les paramètres ?
- Est-il possible de définir des modèles d'exercices à partir desquels les enseignants pourraient construire leurs propres tests ?
- Est-il possible de définir les modèles diagnostiques liés aux modèles d'exercices pour produire le diagnostic quand un professeur a défini un test ?

Notre travail de thèse a consisté à approfondir ces questions et à y apporter des éléments de réponses. Nous y revenons dans le chapitre 4.

5.4.2. Diagnostic assisté ou diagnostic automatique ?

Nous avons montré qu'il était possible d'automatiser partiellement le diagnostic en mettant en application le modèle de diagnostic de B. Grugeon. Cependant, le diagnostic dans la version 1, étant partiel et non complètement fiable, l'enseignant doit le compléter et le vérifier.

Certaines situations nécessitent un diagnostic automatique : par exemple pour étudier de grands corpus ou fournir un bilan de compétences aux élèves. De plus, contrairement à ce que nous pensions au départ, où il nous semblait que les enseignants n'accepteraient pas qu'un logiciel évalue leurs élèves, ils sont plutôt demandeurs d'un diagnostic automatique à partir du moment où ils ont compris la façon de coder du système. En effet un diagnostic automatique leur permettrait de gagner du temps pour se consacrer à ce qui les intéresse : organiser les apprentissages.

- Les méthodes linguistiques ou statistiques peuvent-elles améliorer et fiabiliser le diagnostic sur les réponses ouvertes en particulier quand les réponses sont en langue naturelle ?

- La mise au point d'un logiciel de calcul formel dédié peut-elle résoudre le problème de l'évaluation des réponses ouvertes représentant des raisonnements algébriques ?

Nous nous sommes attelées à ce dernier problème dans notre travail de DEA et surtout dans notre thèse (chapitre 6) tandis que d'autres membres de l'équipe se sont penchés sur l'analyse statistique [Waymel 2005] et linguistique [Normand et al. 2005].

5.4.3. Test systématique ou adaptatif ?

Pour ce qui concerne le système dans son ensemble, le modèle de diagnostic proposé par les travaux de B. Grugeon est trop complexe. Il a été élaboré pour comprendre et décrire le fonctionnement des élèves. Ceci explique peut-être qu'il soit bien accepté par des débutants mais pas par des experts. Les différences entre les débutants et les experts peuvent être observées aussi dans les termes employés pour décrire le profil de l'élève.

Nous avons aussi noté que B. Grugeon, elle-même, emploie un diagnostic adaptatif qui concerne un nombre restreint d'exercices dont le choix dépend de la réponse des élèves mais également de l'objectif du diagnostic afin de définir une stratégie d'enseignement. Des stratégies de diagnostic adaptatif (i.e. qui posent des exercices différents selon les réponses des élèves) permettraient-elles de construire des profils cognitifs ?

Un doctorant, Aso Darwesh a commencé en Novembre 2007, une thèse à l'Université Paris 6 sous la direction de Jean-Marc Labat et Elisabeth Delozanne, pour mettre au point un diagnostic adaptatif.

Par ailleurs, B. Grugeon et F. Chenevotot testent l'idée d'exercices de diagnostics prédictifs qui permettraient de limiter le nombre d'exercices du test diagnostic.

5.4.4. Exploitation du diagnostic

Pour ce qui concerne le logiciel destiné aux enseignants, les seules fonctionnalités utilisées, en formation, le module de codage des réponses des élèves et la description quantitative du profil (imprimée pour être donnée aux élèves). Nous avons observé beaucoup d'incompréhension sur la description qualitative. Il semble que ce logiciel soit adapté aux chercheurs en didactique et adapté pour des sessions de formation, moins bien adapté aux professeurs qui, dans la gestion courante de la classe, ont besoin d'un profil plus opérationnel, c'est-à-dire associé à des activités pour le faire évoluer.

- Est-il possible de définir des profils types qui donnent un moyen d'associer des stratégies d'apprentissage de l'algèbre à chacun de ces profils ?

- Ces profils peuvent-ils être présentés de façon compréhensible aux enseignants, voire aux élèves ?
- Comment déterminer des stratégies d'apprentissage associées aux profils ?
- Comment concevoir un logiciel pour aider les enseignants à mettre en œuvre ces stratégies ?

Dans le cadre du projet Cognitique, une équipe d'ergonomes dirigée par Jeanine Roglaski a mené une étude pour cerner l'activité de diagnostic des enseignants et pour proposer un profil plus facilement accessible.

Dans les deux sections suivantes, nous présentons les travaux pour, d'une part, étendre et fiabiliser le diagnostic automatique de Pépite et, d'autre part, faciliter l'exploitation du diagnostic par les enseignants. Nous ne détaillons que les travaux qui sont en relation étroite avec l'objectif de notre thèse.

6. Vers un diagnostic automatique plus fiable

L'objectif était de fiabiliser le diagnostic automatique en améliorant les réponses aux questions ouvertes. Deux études exploratoires ont été menées. Dans la première, dans le cadre de notre DEA, nous avons mis en œuvre une analyse automatique des raisonnements algébriques des élèves sur un exercice de preuve (celui présenté dans la section 2). La seconde étude concerne l'analyse des justifications que les élèves expriment en utilisant la "langue naturelle" ou plutôt une langue où ils incorporent des expressions mathématiques à des mots d'usage courant. Par ailleurs, et toujours dans l'objectif de fiabiliser le système, le portage du système dans le langage Java a été initié dans le cadre d'un stage de maîtrise au LIUM [Vaseux 2003]. La version Java de PépiTest est téléchargeable sur le site du projet [site Pépite]. Dans cette section nous présentons plus particulièrement la première étude qui a initialisé nos travaux de thèse et nous évoquerons brièvement la seconde étude.

VERSIONS	AUTEURS	CARACTÉRISTIQUES
Pépite 1	S. Jean (2000)	Delphi 1 Diagnostic des questions fermées, des réponses algébriques sur une ligne Très peu de langage naturel
Pepite 1.2 (téléchargeable sur ([http://pepite.univ-lemans.fr])	D. Prévit (2002)	Delphi 1 Amélioration du diagnostic pour un type de raisonnement algébrique
Pepitest-Java (téléchargeable sur ([http://pepite.univ-lemans.fr])	M. Vaseux (2003)	Java, - impression du test et des réponses des élèves - version client serveur
PépiStéréo	C. Vincent (2004)	XML et XSL, Java Intégration des stéréotypes Nouvelle présentation des profils cognitifs
PepiGenExo	Thèse en cours de D. Prévit (2008)	Java Génère des clones des exercices de diagnostic
SuperPépite	En cours de spécification	Chaîne logicielle pour • Créer des exercices, des tests • Administrer les tests • Exploiter les résultats

Tableau 4 : Les différentes versions du logiciel Pépite

6.1. Vers une analyse automatique des raisonnements algébriques

Ce travail, mené par deux informaticiennes (D. Prévit, E. Delozanne) et une didacticienne (B. Grugeon), a donné lieu à un mémoire de DEA et à deux publications [Prévit 2002, Delozanne et al. 2003, Prévit et al. 2004]. Il est à l'origine de notre travail de thèse. Il concerne l'analyse du raisonnement algébrique des élèves dans un exercice particulier de modélisation pour conduire une preuve, l'exercice dit du prestidigitateur (section 4.1 et Figure 2). Très important dans le diagnostic humain établi par les didacticiens de l'équipe, il n'était pas du tout analysé dans la version 1 de Pépite (version 2000). D'un point de vue didactique, c'est un des exercices prédictifs du profil de l'élève. D'un point de vue informatique, il est intéressant car c'est le plus complexe en raison des réponses des élèves à traiter et de la grille d'analyse à mettre en œuvre. Enfin, réussir l'analyse automatique de cet exercice autorise l'analyse automatique des autres exercices de Pépite comportant des raisonnements algébriques.

A partir des exemples présentés dans la section 2, nous précisons d'abord l'analyse didactique a priori de cet exercice qui définit la grille de codage, puis nous décrivons la méthode que nous avons mise au point pour obtenir un codage automatique de cet exercice. Ce premier

travail pour notre DEA, nous a permis d'explorer les différents aspects du problème et d'en proposer une solution générique dans notre thèse (chapitre 5).

6.2. Diagnostic automatique des réponses à cet exercice

Afin de pouvoir coder les réponses à cet exercice, le logiciel doit détecter la production d'expressions équivalentes à celles prévues dans la grille d'analyse des réponses anticipées. Pour cela nous construisons d'abord un arbre représentant les expressions algébriques entrées par les élèves en généralisant et systématisant l'approche que Jean [Jean 2000] avait initié dans la version 1, sur un exercice simple (exercice 3). Ensuite, nous comparons cet arbre avec les arbres construits à partir des expressions algébriques anticipées spécifiées par l'analyse didactique a priori. Nous détectons ainsi les règles de transformations erronées spécifiées dans l'analyse didactique a priori de l'exercice (Tableau 5).

Identification incorrecte du rôle des opérateurs + et ×	Utilisation inadaptée des parenthèses qui conduit à un résultat correct	Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect
x + a → x a a x ± b → (a ± b) x a x - x → a – 1	(a + b) × c → a + b × c puis a + b × c → résultat de (a + b) × c Il y a mémoire de l'énoncé (a + b)/d → a+b /d puis a + b/d → résultat de (a/d + b/d) Il y a mémoire de l'énoncé (exemple de Laurent)	(a + b) × c → a + b × c puis a + b × c → résultat de (a + b × c) Il n'y a pas mémoire de l'énoncé (a + b)/d → a+b /d puis a + b/d → résultat de (a + b/d) Il n'y a pas mémoire de l'énoncé

Tableau 5 : Critères d'évaluation sur la dimension des écritures algébriques et règles erronées associées

Reprenons l'exemple de Laurent (section 2) pour illustrer le fonctionnement du diagnostic automatique que nous avons mis en place. Du point de vue de l'analyse a priori, l'élève est dans une démarche algébrique (code J1 : « Type de justification », justification par l'algèbre). L'analyse de l'expression écrite à la première ligne fait apparaître une expression complètement parenthésée qui traduit globalement l'énoncé du problème (code L1 : « Utilisation des lettres », utilisation correcte des lettres, code T1 : « Traduction », traduction correcte). Les lignes 3 et 4 de la réponse de l'élève témoignent de l'« utilisation des règles d'écriture et de réécriture algébrique» avec une utilisation inadaptée des parenthèses qui conduit cependant à une réponse correcte (code EA31).

L'algorithme que nous avons mis en place commence par construire un arbre représentant l'expression algébrique de la première ligne. Il repère d'abord la présence de l'utilisation de lettres (J1), puis que l'expression produite par l'élève est l'expression globale complètement parenthésée attendue (L1, V1). Il construit ensuite les arbres correspondant aux lignes suivantes et, par comparaison avec les arbres obtenus à partir de l'analyse a priori, repère et code les erreurs éventuelles, ici EA31. Enfin l'erreur décelée amène à évaluer le traitement algébrique comme étant incorrect (V3). La Figure 5 présente l'énoncé de l'exercice du prestidigitateur, la réponse d'un élève et une partie du codage établi par le logiciel de diagnostic (affichage en clair du codage EA31 : utilisation inadaptée de parenthèses).

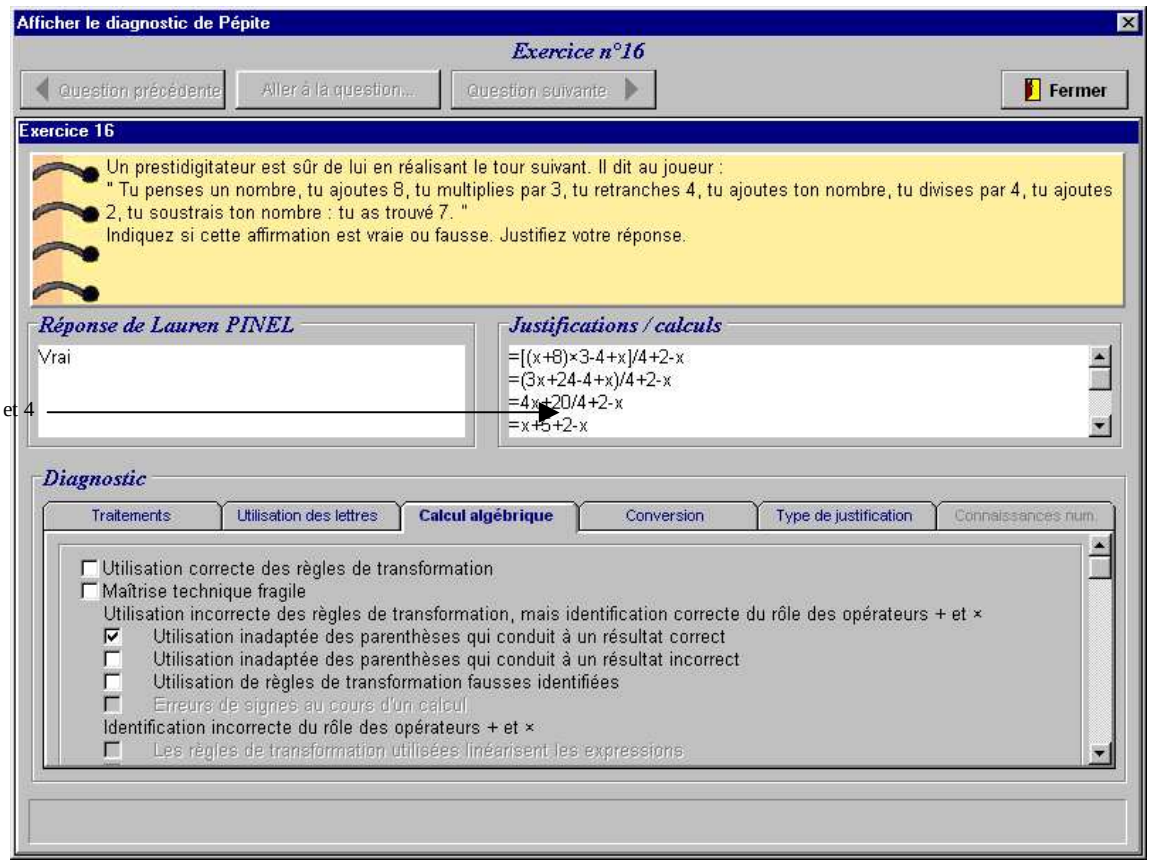


Figure 6 : Résultat du codage automatique de la réponse de Laurent[6]

Le codage automatique de cet exercice est une amélioration capitale par rapport à la version 1. D'une part, analyser les justifications de cet exercice permet d'envisager l'analyse des réponses à des exercices mettant en œuvre des tâches de résolution d'équations, de systèmes d'équations, de mathématisation, de résolution de situations mathématiques appliquées à

[6] Les dimensions qui figurent sur l'écran sont celles qui étaient définies dans la thèse de B. Grugeon. Rappelons que suite à différents travaux (avec les ergonomes mais aussi pour adapter le test de seconde à d'autres niveaux scolaires) les dimensions ont évolué. Dans le texte, nous adoptons la dénomination actuelle.

d'autres domaines que le domaine algébrique (géométrique, numérique, fonctionnel). D'autre part, comme nous l'avons dit, cet exercice peut être repéré comme un exercice clé en vue d'un diagnostic adaptatif. Il permet d'émettre des hypothèses sur des cohérences locales dans le fonctionnement des élèves en algèbre telles qu'elles ont été définies par B. Grugeon dans son modèle multidimensionnel de la compétence algébrique : au niveau du type de traitement algébrique, du statut des objets et des lettres mis en œuvre dans la résolution, des traductions entre différents registres sémiotiques, de la gestion dans le registre des écritures algébriques, du niveau de rationalité. Enfin l'étude et la mise en œuvre informatique du diagnostic sur cet exercice nous a permis d'envisager une approche du diagnostic plus générale que celle qui avait été mise en œuvre dans la version 1, nouvelle approche qui constitue le cœur de notre travail de thèse et que nous précisons dans les chapitres suivants.

7. Vers un assistant à l'exploitation du diagnostic

Les expérimentations ont montré que le profil cognitif était difficile à comprendre par les enseignants et beaucoup trop complexe pour être montré aux élèves. L'équipe du projet Lingot a mené plusieurs travaux pour faciliter l'exploitation du diagnostic par les enseignants mais aussi par les élèves. Les premiers concernent une refonte totale du profil cognitif pour le proposer aux enseignants. Nous les présentons car ils permettent de comprendre comment est utilisé le diagnostic local sur lequel nous travaillons dans notre thèse pour construire un profil de plus haut niveau. Les seconds, que nous évoquons sans les détailler car ils ne s'apparentent pas vraiment à notre travail, se préoccupent de la présentation du profil à l'élève. Les troisièmes s'intéressent à la définition de parcours d'apprentissage adaptés au profil cognitif diagnostiqué par Pépite et s'appuient sur des activités d'apprentissage paramétrées par les différents éléments du modèle de la compétence algébrique. Enfin les derniers travaux que nous présentons ont pour objectif de mettre au point des tests diagnostics pour tous les niveaux scolaires : fin de cinquième et fin de quatrième.

7.1. Des profils aux stéréotypes

Le plus gros travail et le plus abouti, a été mené auprès des enseignants pour faciliter l'exploitation du diagnostic en particulier en classe où ils ont à gérer non pas un individu mais tout le groupe classe. Le projet Cognitique a permis des études ergonomiques pour restructurer le profil cognitif avec un double objectif : faciliter son appropriation et faciliter la gestion de la classe en fournissant une « géographie cognitive » de la classe (Figure 8). De plus, un important

effort a été mené pour simplifier la terminologie utilisée et pour transposer les termes se référant à des concepts issus de la recherche en didactique en des termes utilisés par les enseignants. À partir de l'étude des profils cognitifs calculés par Pépite, l'équipe a mis au point une échelle de compétences pour situer les élèves sur cette échelle. Les trois composantes retenues sont : l'usage de l'algèbre, la pratique du calcul algébrique et la traduction d'une représentation dans une autre. Ainsi, chaque élève appartient à un stéréotype qui correspond à son niveau de compétence et qui permet de définir l'axe prioritaire sur lequel le faire travailler (Tableau 6).

Composante	Notation	Objectif	Niveaux de compétence
Usage de l'algèbre	UA	Etudier la disponibilité de l'outil algébrique et la capacité à le mobiliser dans des situations de modélisation (production de formules ou mise en équation) et de preuve.	*Niveau 1* : Disponibilité de l'outil algébrique et mobilisation adaptée.
			Niveau 2 : Mobilisation de l'outil algébrique et traduction algébrique non adaptée
			Niveau 3 : Mobilisation de l'outil algébrique sans cohérence entre le modèle et la situation
			Niveau 4 : Non disponibilité de l'outil algébrique pour généraliser, prouver ou modéliser et démarches arithmétiques persistantes.
Traduction d'une représentation à une autre	T	Etudier la capacité à traduire une expression d'un registre à un autre et la flexibilité à interpréter une représentation d'un registre à un autre.	*Niveau 1* : Traduction correcte.
			Niveau 2 : Traduction pas toujours adaptée.
			Niveau 3 : Au moins une traduction sans cohérence entre le modèle et la situation.
Calcul algébrique	CA	Etudier la capacité à calculer algébriquement.	*Niveau 1* : L'élève traite un nombre suffisant d'exercices. Il propose des interprétations appropriées des expressions.
			Niveau 2 : L'élève parvient à traiter certains exercices mais il commet encore trop d'erreurs dans les calculs algébriques avec un appui insuffisant sur l'interprétation des expressions.
			Niveau 3 : L'élève ne réussit pas à mener des calculs : soit il ne maîtrise pas le rôle des opérateurs, soit il s'est construit des règles de calcul fausses

Tableau 6 : Caractérisation d'un stéréotype.

Pour chacune des composantes, les caractéristiques personnelles de l'élève sont alors présentées sous forme de taux de réussite, de points forts (des leviers pour l'apprentissage) et points faibles (des obstacles à l'apprentissage ou des conceptions inadaptées) qui permettent d'affiner les objectifs de travail (Figure 7).

21	Date du test : 29/09/2004	Classe : 2nd 10
Mickael	Questions traitées : **61 %** Taux de réussite aux questions traitées : **61 %**	Imprimer ce profil

Stéréotype et commentaires		Caractéristiques personnelles
UA3	**Usage de l'algèbre Niveau 3** L'élève ne résout pas assez d'exercices avec la démarche algébrique. Ses justifications sont mal assurées : - soit elles sont trop rarement correctes - soit elles comportent des arguments non algébriques ou incomplets	Exercices de mathématisation : Taux de réussite : 18 % Leviers Début d'utilisation de l'algèbre pour prouver Fragilités L'outil algébrique n'est pas bien maîtrisé et justification par l'algèbre dominante dans un contexte trop faible En particulier… Justification de type scolaire reposant sur l'application de règles incorrectes Exercice 4c Justification en langage naturel Preuve avec utilisation de propriétés énoncées en langage naturel à l'exercice 4a
T3	**Traduction Niveau 3** L'élève a des difficultés : - soit à exprimer algébriquement les relations entre variables - soit à associer une expression algébrique à une autre représentation (ou vice versa). Dans au moins un cas, l'écriture symbolique est utilisée pour « sténographier » ou abréger la situation, c'est à dire sans retrouver les relations entre les différentes variables en jeu.	Exercices de reconnaissance : Taux de réussite : 48 % Fragilités Traduction abréviative En particulier… Traduction incorrecte Expression non parenthésée ou confusion aire – périmètre à l'exercice 3 Confusion produit – somme à l'exercice 5a Traduction incorrecte pour : retrancher du résultat, à l'exercice 11p1b Traduction abréviative Mauvais calcul avec les coordonnées des points d'intersection avec les axes dans l'exercice 7 6xE=P traduit terme à terme par la relation : il y a 6 fois plus d'élèves que de professeurs à l'exercice 10
CA3	**Calcul Algébrique Niveau 3** L'élève ne réussit pas à mener des calculs : - soit il ne maîtrise pas le rôle des opérateurs - soit il s'est construit des règles de calcul fausses	Exercices techniques : Taux de réussite : 14 % Fragilités Rôle des opérateurs non maîtrisé En particulier… Utilisation inadaptée des parenthèses mais qui conduit toutefois à un résultat correct Expression non parenthésée. a+3(a+b) pour (a+3)(a+b) à l'exercice 3p2 Utilisation inadaptée des parenthèses et qui conduit à un résultat incorrect 3+5a=8a à l'exercice 2 Utilisation de règles de transformation fausses identifiées Erreur dans l'identité remarquable $a^2-b^2=(a-b)^2$ à l'exercice 9a Règle incorrecte ax=b donne x=-b/a à l'exercice 9c Identification incorrecte de x et + : Linéarisation des expressions Confusion dans le rôle de x et + à l'exercice 9d Identification incorrecte de x et + : Assemblage des termes Mauvais regroupement $(x+2)^2-5(x+2)=(x+2)(2-5)$ à l'exercice 9b

Figure 7: Profil de Mickael construit par PépiStéréo

Pour chacun des stéréotypes, des recommandations de travail sont établies pour les faire évoluer. L'enseignant dispose d'une liste des élèves classés par stéréotypes afin de pouvoir constituer des groupes de travail homogènes ou hétérogènes. Lors de tests pilotes menés dans un lycée de la région parisienne, les enseignants ont apprécié cette nouvelle présentation. Et certains continuent de l'utiliser en début de seconde. Des problèmes de stabilité du diagnostic de PépiDiag, empêchent une expérimentation à grande échelle.

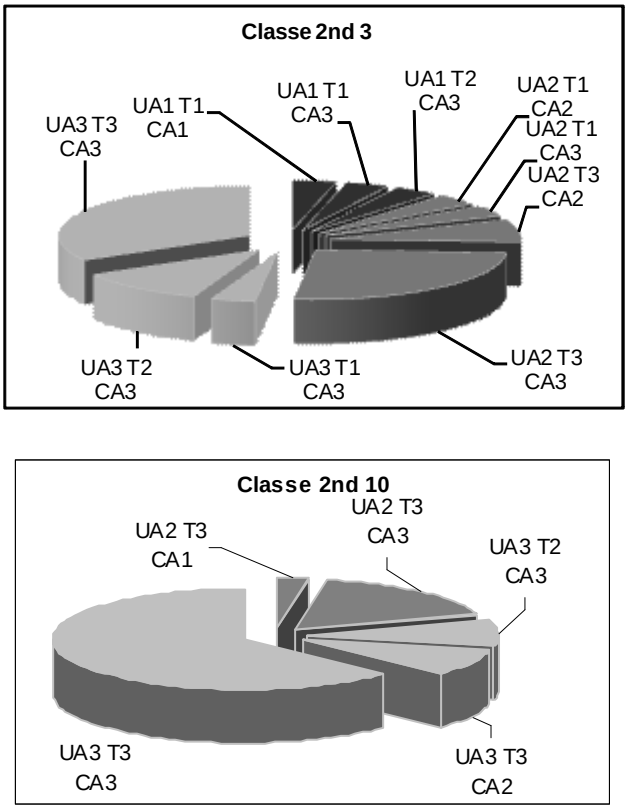


Figure 8 : Géographie cognitive de deux classes de seconde

Cette nouvelle façon d'exprimer le profil cognitif de l'élève nous a conduit à compléter PépiDiag pour faire apparaître les règles de calcul erronées utilisées par l'élève. De plus, l'équipe a mis au point un algorithme pour la construction du profil à partir du codage des réponses par PépiDiag. Enfin, C. Vincent a mis au point un prototype, PépiStéréo, pour tester cette nouvelle approche [Vincent et al. 2005]. La Figure 7 présente le profil calculé par PépiStéréo pour un élève, Mickael. Le Tableau 7 résume les différents niveaux de modélisation de la compétence algébrique dans Pépite.

Logiciels	Modèle individuel de l'élève	Modèles des tâches et compétence algébrique
PépiTest	Réponses de l'élève	Type d'exercice : technique, reconnaissance et mathématisation
PépiDiag	Codage des réponses de l'élève à chaque exercice Matrice de diagnostic	Pour chaque exercice, une grille d'analyse multidimensionnelle : justesse, utilisation des lettres, calcul algébrique, traduction d'une représentation à une autre, type de justification
PépiProf	Profil global : - taux de réussite (global, questions traitées, type d'exercices, type de traitements privilégié) - description qualitative (cohérences de fonctionnement) - articulation entre représentations	Analyse transversale des réponses à l'ensemble du test Types de traitements Seuils paramétrables de réussite Modalités de fonctionnement
PépiStéréo	- Situation de l'élève sur une échelle de compétence multidimentionnelle pour définir son stéréotype - Sur chaque dimension, caractéristiques personnelles de l'élève : taux de réussite, points forts, points faibles, règles fausses	Agrégation des codages Marqueurs de conceptions inadaptées

Tableau 7 : Niveaux de modélisation mis en œuvre par Pépite

7.2. Un bilan avec l'élève

Lors des expérimentations de 2002, des enseignants ont demandé à des membres de l'équipe de faire un retour individuel à leurs élèves. Ces séances ont montré l'intérêt des adolescents pour ce type de diagnostic mais aussi la difficulté de compréhension de ce bilan. Des études ont été menées pour envisager un bilan à destination de l'élève, en particulier, avec l'équipe du LINC dans l'objectif de définir un agent conversationnel animé [Farouk et al. 2007].

7.3. Des activités d'apprentissage paramétrées par les compétences

Dans le but de définir des parcours d'apprentissage, Grugeon, Coulanges et Gélis ont mis au point des prototypes permettant de générer des exercices donnés en fonction de variables didactiques caractérisant des niveaux de compétence. Trois prototypes ont ainsi été développés pour proposer différentes tâches pour faire travailler les élèves sur les traductions entre plusieurs représentations [Grugeon et al. 2003, 2005].

CIME (Figure 9) s'appuie sur un environnement informatique déjà existant, nommé « Bouchons les trous », conçu à partir d'une idée originale de René de Cotret [Lemoyne et al.

2002]. Il s'agit de compléter une mise en équations de problèmes (avec des équations ou des énoncés écrits lacunaires). Dans CIME, conçu par Lalina Coulanges, les exercices sont générés automatiquement à partir des variables didactiques suivantes : la structure du problème donné (ici problème du premier degré de type rapport et différence), les conditions numériques liées aux relations en jeu, la forme écrite de l'énoncé (plus ou moins congruente avec les équations), le nombre d'équations et d'inconnues, les équations de forme plus ou moins congruente avec l'énoncé, le nombre et la place des trous, le contenu des trous à compléter (opérateurs, données numériques, etc.).

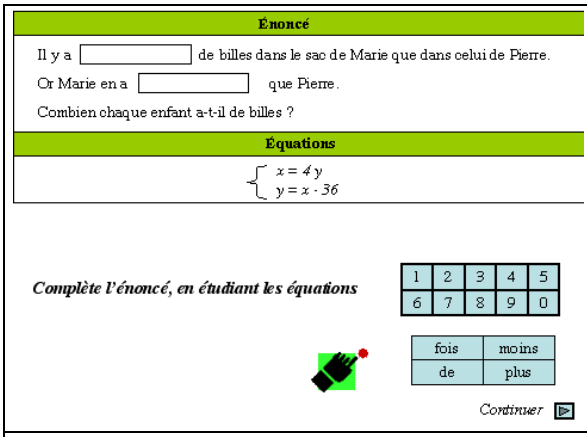


Figure 11a : Consigne initiale dans CIME

Le système met à disposition une palette de mots et de chiffres

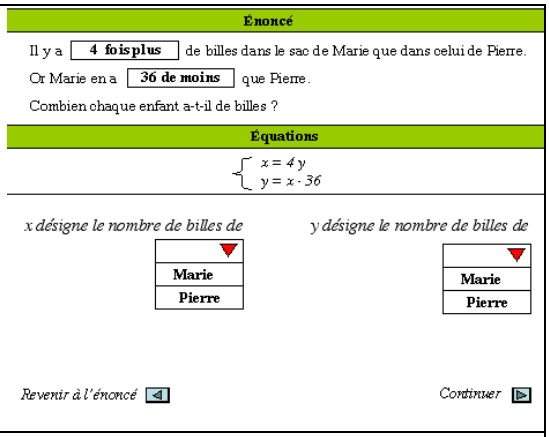


Figure 11b : Identification des inconnues

Suite à une erreur, le système demande d'identifier les inconnues

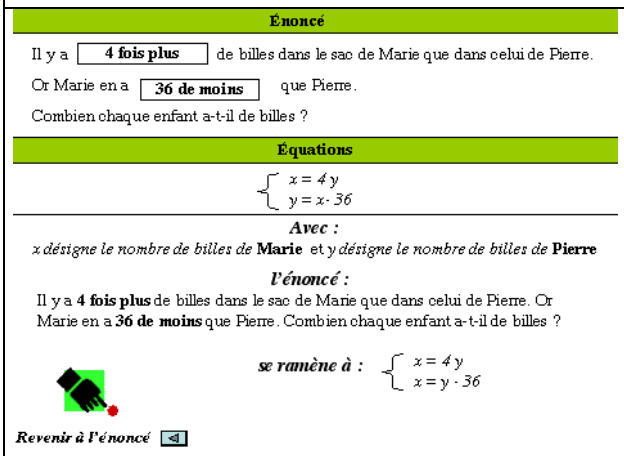


Figure 11c : Mise en contradiction de la réponse de l'élève

Le système renvoie les équations obtenues à partir de l'énoncé, tel qu'il a été complété par l'élève

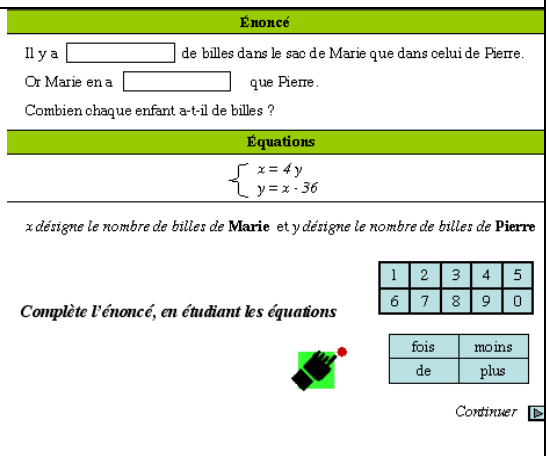


Figure 11d : Retour à l'énoncé

Le système met à disposition une palette de mots et de chiffres pour compléter l'énoncé, et les inconnues désignées

Figure 9 : Exemple de situation d'interaction CIME [Grugeon et al. 2003]

Le principe général d'AILE, conçu par J.-M. Gélis, consiste à associer une liste d'expressions algébriques et leurs traductions en langage naturel (Figure 10). Sa conception est fondée sur la thèse de Caroline Bardini [Bardini 2003]. Les expressions sont générées automatiquement à partir de patterns.

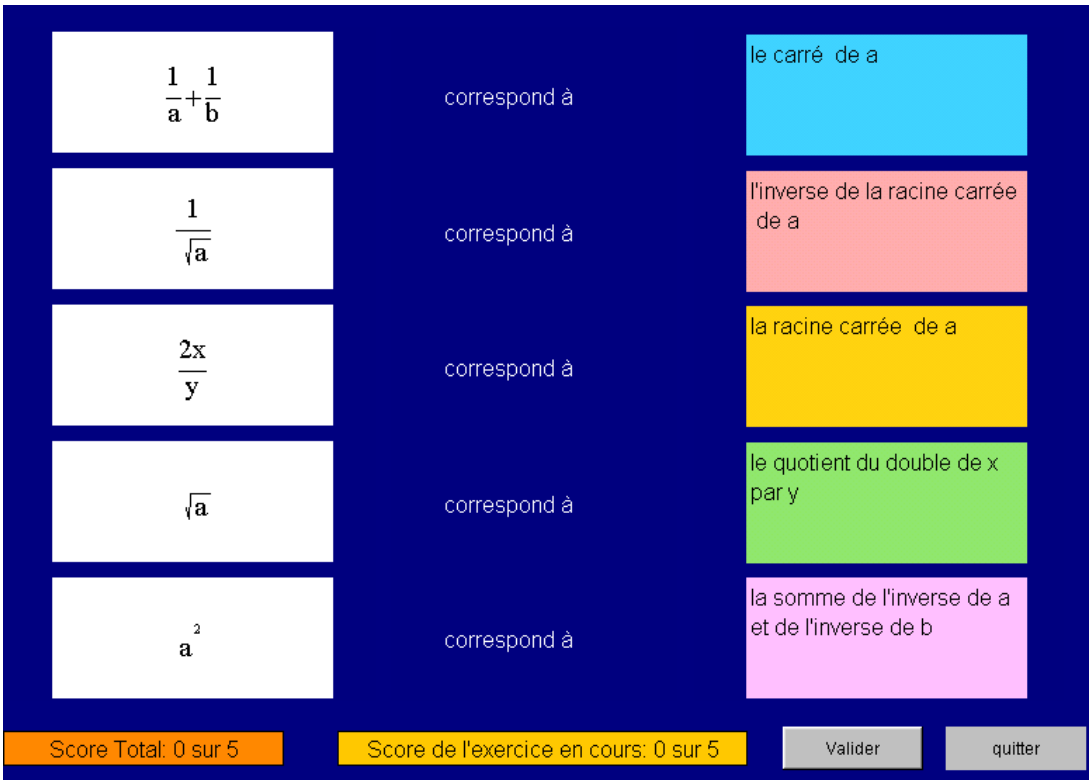


Figure 10 : Interface élève de AILE

Le troisième logiciel ExplorExp, conçu et réalisé par Jean-Michel Gélis, permet d'associer trois représentations : la représentation algébrique, en langue naturelle et arborescente (Figure 11).

7.4. Tests diagnostics pour d'autres niveaux scolaires

Françoise Chenevotot et Brigitte Grugeon ont mis au point et expérimenté des tests papier-crayon pour construire des profils cognitifs en fin de cinquième et en fin de quatrième [Chenevotot et al. 2008]. Ce travail en cours est très important. Avec celui conduit sur les situations d'apprentissage paramétrées, il a engagé une réflexion sur la caractérisation du niveau de difficulté des exercices. Par exemple, l'exercice du prestidigitateur est un exercice de fin de troisième s'il comporte une division et trois niveaux de parenthèses mais il peut être posé en cinquième avec une expression ne comportant que des additions et des multiplications et

seulement un niveau de parenthèses. Ces caractérisations nous ont inspiré pour la partie indexation didactique de notre modèle (Chapitre 5).

Enfin, Caroline Bardini et moi-même avons traduit le logiciel Pépite en anglais et en brésilien. C. Bardini a fait passer le test à des élèves Australiens et Brésiliens. Les réponses des élèves qui ont été recueillies (une quarantaine dans chacun de ces deux pays) sont tout à fait similaires à celles des étudiants français en dépit de certaines différences dans le curriculum.

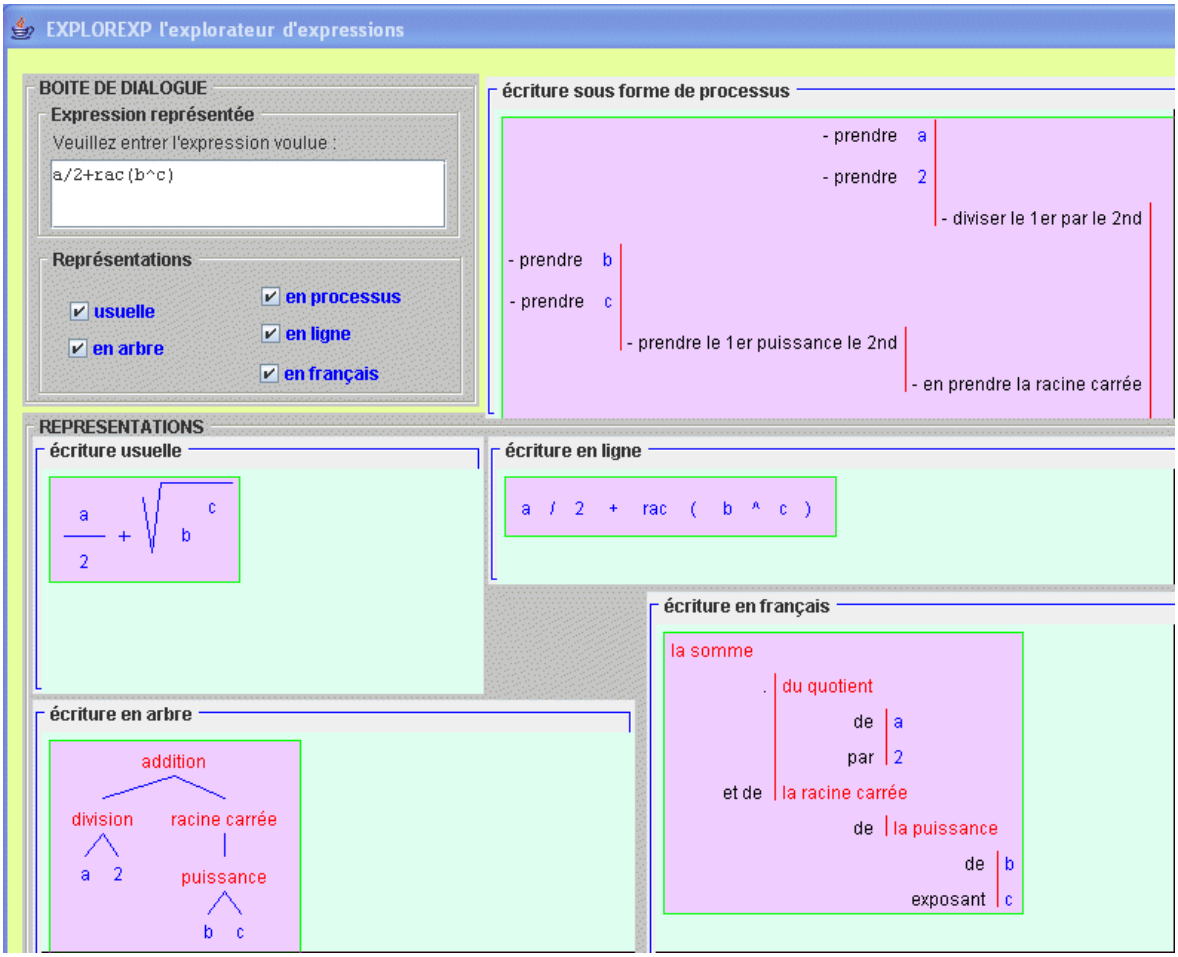


Figure 11 : Interface élève de ExploExp

8. Conclusion

Notre travail se situe donc dans la problématique de génération automatique (ou assistée) de banque d'exercices de diagnostic. En effet, c'est un des problèmes clé des projets Pépite et Lingot. Nous devons construire un outil permettant le développement aisé de nombreux exercices et la génération automatique du codage des réponses des élèves à ces exercices. Ce

codage constitue la fondation sur laquelle repose tout l'édifice : construction des profils d'un élève, d'une classe, proposition d'activités d'apprentissage, suivi de l'évolution des profils.

Par rapport au premier logiciel Pépite, il s'agit, à partir des exercices originels de proposer des modélisations qui permettent de générer des clones des exercices originels afin de pouvoir poser des exercices de mêmes types mais avec des énoncés différents, et ce, à différents moments de la construction de la compétence algébrique.

En nous appuyant sur les prototypes AILE et CIME, nous pouvons envisager le type de paramètres qui permettent de passer d'un clone à un autre. Cependant, dans notre thèse nous devons être capables de plus d'analyser des réponses ouvertes plus complexes que celles de ces deux logiciels.

Par rapport à ces trois logiciels et à notre travail de DEA, nous devons définir une architecture et une modélisation générique qui permettent d'intégrer ces prototypes de type « preuves de concepts » construits pour tester des questions de recherche par des programmeurs différents sans souci d'intégration ou d'interopérabilité.

C'est donc l'objectif de notre travail de thèse sur lequel nous revenons dans les chapitres suivants. Mais auparavant, nous allons étudier l'état de l'art concernant les problèmes que nous avons à traiter.

Chapitre 3

État de l'art

1. Introduction 47

2. Les tuteurs cognitifs 49
2.1. Les fondements du projet 49
2.1.1. Modèle de la cognition 49
2.1.2. Principes de conception des « Cognitive Tutors » 50
2.2. Les tuteurs cognitifs dans les classes 51
2.3. Les activités des élèves avec un tuteur Cognitif 51
2.4. Le diagnostic cognitif 53
2.5. Logiciel d'exploitation par les enseignants 54
2.6. Outils auteur pour les tuteurs cognitifs 55
2.7. Ce que je retiens pour ma thèse 57

3. Andes, un tuteur en Physique 58
3.1. Les fondements 59
3.2. Les activités des élèves avec Andes 60
3.3. Le diagnostic dans Andes 60
3.4. Les outils auteurs dans Andes 62
3.4.1. Le graphe des solutions correctes 62
3.4.2. Le système de calcul formel spécifique 62
3.5. Ce que j'en retiens pour ma thèse 63

4. Aplusix 64
4.1. Une théorie de l'algèbre 65
4.2. Les activités des élèves avec Aplusix 67
4.3. Le diagnostic dans Aplusix 68
4.4. Outil auteur 70
4.5. Ce que j'en retiens pour ma thèse 71

5. Des projets plus éloignés 72

6. Les langages de modélisation pédagogique 75
6.1. Le LOM : un standard pour la description des objets pédagogiques 76
6.2. Langage de modélisation pédagogique EML 77
6.3. IMS-LD 79
6.4. Question and Test Interopérability (QTI) 80
6.4.1. Différents cas d'utilisation 81
6.4.2. Modèle d'informations de QTI 82
6.4.3. Ce que je retiens pour ma thèse 85
6.5. Modèle conceptuel d'évaluation à l'Open University of NederLand (OUNL) 87
6.5.1. Description du modèle 87
6.5.2. Ce que je retiens pour ma thèse 90

6.6. Langages spécifiques aux mathématiques 90
6.6.1. MathML 90
6.6.2. OpenMath 90
7. Conclusion 91

1. Introduction

L'objectif de notre travail de thèse étant de concevoir un outil auteur utilisé par des enseignants, des formateurs d'enseignants ou des chercheurs en didactique des mathématiques pour créer des banques d'exercices de diagnostic, nous étudions dans quelle mesure le processus de conception peut être automatique ou assisté. Plus précisément, nous envisageons de créer un système permettant, d'une part, de générer automatiquement ou avec l'aide d'un auteur non informaticien des exercices et, d'autre part, de générer une analyse multidimensionnelle automatique des réponses à ces exercices dans le cas où ces réponses sont libres ou contraintes. Avant de présenter notre approche pour résoudre ce problème, nous étudions les travaux antérieurs sur les problèmes soulevés par notre approche. Nous nous limitons aux travaux qui nous sont apparus les plus significatifs pour situer notre travail.

Nous retenons d'abord des travaux dont l'objectif est le diagnostic cognitif des apprenants en mathématiques. Les pionniers dans ce domaine sont les travaux de l'équipe de Pittsburg autour des tuteurs cognitifs et en particulier d'Algebra Tutor [Anderson 1983, Anderson et al. 1985, Koedinger et al. 1997, Razzaq et al. 2005]. Toujours à Pittsburg, les travaux de l'équipe de VanLehn autour du tuteur Andes se rapportent à l'apprentissage de la physique [VanLehn et al 2005] ; pour étudier les réponses ouvertes des élèves, ils ont été amenés (entre autres) à définir un module de calcul formel spécifique pour étudier les raisonnements mathématiques des élèves nécessaires à la résolution des problèmes de physique (essentiellement la résolution de systèmes d'équations) [Shapiro 2005]. Les travaux de l'équipe de Jean-François Nicaud autour du micro-monde Aplusix sont parmi les plus aboutis et les plus documentés dans le domaine des logiciels pour l'apprentissage du calcul algébrique [Bouhineau et Nicaud 2006, Chaachoua et al. 2007]. Enfin, plus récemment, les travaux autour de DIANE sont proches des nôtres mais à un autre niveau scolaire [Hakem et al. 2005]. MIM [Zapata Rivera et al. 2007] est un environnement de travail en ligne qui cherche à ouvrir le modèle de l'élève aux élèves et aux parents et Wear [Virvou et al. 2001] propose d'intégrer un modèle du professeur à l'environnement de travail en ligne. Ces deux logiciels ne semblent pas se préoccuper de l'analyse de réponses ouvertes. Wims et ActiveMath sont plutôt centrés sur des formations en ligne, PERLEA travaille sur le diagnostic cognitif pour développer et généraliser la phase de présentation des profils obtenus [Leroux et Jean-Daubias 2006]. Ces systèmes sont fondés sur un modèle des connaissances du domaine étudié et se rattachent ainsi au champ de recherche des tuteurs intelligents même si Aplusix a évolué vers un micromonde [Wenger 1987, Nicaud et Vivet 1988, Bruillard 1997, Bruillard et al. 2000, Leroux 2006]. Pour chacun de ces systèmes nous commençons par présenter le projet de

recherche, puis les principes fondateurs, leur utilisation, l'interface élève, le diagnostic cognitif, les outils pour l'enseignant et les outils auteurs.

Ces recherches ont produit des systèmes qui sont utilisés dans les classes à large échelle, mais les études montrent que leur développement est très coûteux en temps et mobilise d'importants moyens humains [e. g. Murray 2003]. De plus, leur conception exige une étroite collaboration entre les chercheurs de disciplines différentes et les enseignants. Les outils auteurs visent à réduire les efforts nécessaires à la production de tels systèmes en assumant certains aspects de la conception, en guidant l'auteur, en lui proposant des éléments prédéfinis qu'il peut adapter [Bell 2003, Blessing 2003, Ritter et al. 2007].

Notre deuxième axe de recherche concerne ainsi les outils auteurs. Murray a mené une étude très complète des outils auteurs élaborés pour faciliter le développement de tuteurs intelligents [Murray 1999, 2003a]. Il décèle quatre niveaux de complexité dans les tâches à réaliser par un auteur. Au premier niveau l'auteur utilise des modèles qu'il instancie en complétant des formulaires ; au deuxième niveau il définit les relations entre les objets ou les concepts ; au troisième niveau il élabore des modélisations ; au dernier niveau, selon le type de questions ou les contenus qu'il propose, l'auteur doit comprendre le fonctionnement du système et est amené à le programmer. D'après Murray, et notre propre expérience le confirme, si les tâches de premier niveau sont réalisables par des enseignants, les tâches de deuxième niveau demandent une formation à l'utilisation du système auteur, et les dernières ne leur sont pas accessibles sans formation à la modélisation et à la programmation.

Parallèlement à ces travaux très liés aux recherches en Intelligence Artificielle, d'autres travaux sont apparus avec le développement de la formation en ligne. Ces derniers se préoccupent de l'évaluation des apprenants mais, contrairement aux premiers, ils ne visent pas une modélisation cognitive précise dans un domaine déterminé. Leur objectif est plus souvent de mettre en place des systèmes interopérables et réutilisables qui facilitent la création d'outil d'évaluation en ligne. Dans cette perspective, nous détaillons la spécification QTI (*Question and Test Interoperability*) qui permet de créer des exercices et des tests dont les réponses sont préformatées afin qu'ils puissent être interprétés sur toute plate-forme d'enseignement en ligne qui se conformerait à cette spécification. Nous nous intéressons aussi au modèle conceptuel élaboré par l'Open University of NederLand (OUNL) pour d'une part élargir la vision de l'évaluation sous-tendue par QTI à l'évaluation multicritère de productions individuelles ou collectives d'apprenants et, d'autre part intégrer l'évaluation au modèle IMS-LD afin d'associer l'enseignement, l'apprentissage et l'évaluation.

2. Les tuteurs cognitifs

Initiés au début des années 1980, ces travaux, menés au Pittsburgh Science of Learning Center s'appuient sur la théorie cognitive ACT (Adaptive Control of Thought) puis ACT-R (Advanced Computer Tutoring theory) de John Anderson. Le principe fondateur est que l'instruction doit être conçue en référence à un modèle cognitif de la compétence que l'élève doit acquérir. Un tuteur cognitif possède un modèle computationnel qui lui permet de résoudre les problèmes posés à l'élève. Le premier objectif de ces recherches était de tester ces modèles cognitifs en montrant que de tels tuteurs pouvaient favoriser les apprentissages. Des tuteurs ont ainsi été développés et testés en laboratoire et dans des classes dans plusieurs domaines principalement la programmation (Lisp Tutor), la géométrie (Geometry Tutor) et l'algèbre élémentaire (Algebra Tutor). Devant le succès de ces prototypes, à partir des années 1990, l'équipe a décidé de déployer ses efforts pour disséminer ces résultats de recherche et faire de l'usage des tuteurs cognitifs une pratique courante dans les écoles américaines. Cette décision de changer d'échelle, a obligé l'équipe à faire face à de nouvelles questions de recherche concernant l'intégration des technologies dans le système scolaire et les méthodes de développement des tuteurs cognitifs. En 2004, un produit commercial est diffusé et utilisé dans plus de 1700 collèges et lycées aux États-Unis [Koedinger et al. 2004]. Dans un article de 2005 [Roll et al. 2005], cette équipe affirme que leurs tuteurs sont utilisés pendant une année complète dans 5% des écoles des États-Unis. Parallèlement au développement de ce logiciel commercial de nombreux prototypes ont été mis au point pour tester des questions de recherche. Certains sont accessibles sur le site du projet (http://www.learnlab.org).

2.1. Les fondements du projet

Nous présentons de façon très succincte la théorie ACT et le principe de fonctionnement des tuteurs cognitifs.

2.1.1. Modèle de la cognition

La théorie ACT (*Active Control of Thought*) [Anderson 1983] décrit les mécanismes d'apprentissage dans un contexte de résolution de problèmes. Elle distingue deux types de connaissances en étroite interaction : les *connaissances déclaratives* (les savoirs) et les *connaissances procédurales* (les savoir-faire). Les connaissances procédurales s'acquièrent puis sont renforcées par la pratique. En fonction du but à atteindre, les apprenants utilisent des connaissances déclaratives, sous la forme d'informations, et, lorsque ces connaissances ont été utilisées plusieurs fois dans un même contexte, ils créent de nouvelles procédures associées à ce contexte. Anderson appelle le processus qui permet de passer des connaissances déclaratives aux

connaissances procédurales, la *compilation des connaissances*. Les règles de production sont utilisées pour modéliser les connaissances procédurales.

2.1.2. Principes de conception des « Cognitive Tutors »

Conçus pour explorer la théorie ACT, les premiers tuteurs cognitifs construits par Anderson et son équipe sont Lisp-Tutor et Geometry-Tutor [Anderson et al. 1985a, 1985b], puis Algebra-Tutor. Le modèle cognitif est constitué d'un ensemble de règles de production qui décrivent la compétence de référence, c'est-à-dire, la façon dont un élève « *idéal* » résout un problème. Ce modèle est complété par des règles de production erronées qui correspondent aux erreurs récurrentes relevées auprès des élèves. Le Tableau 1 donne des exemples de règles de production dans le domaine de la résolution d'équations du premier degré. Ces règles modèlisent une solution correcte avec deux stratégies de résolution et une solution incorrecte (*Misconception*) [Koedinger et al. 2004].

Cognitive or Behavior Model: A system that can solve problems in the various ways students can	
Strategy 1:	IF the goal is to solve a(bx+c) = d THEN rewrite this as abx + ac = d
Strategy 2:	IF the goal is to solve a(bx+c) = d THEN rewrite this as bx + c = d/a
Misconception:	IF the goal is to solve a(bx+c) = d THEN rewrite this as abx + c = d

Tableau 1 : Exemple de règles de production (extrait de [Koedinger et al. 2004])

Le modèle cognitif est complété par des principes pédagogiques qui s'appuient sur une méthode dite du « traçage de modèle » (*model-tracing*) et du « traçage des connaissances » (*knowledge-tracing*) [Anderson et al. 1995]. Le « traçage de modèle » permet d'assurer le suivi de l'élève, pas-à-pas, en comparant ses réponses pour résoudre un problème à celles qui ont été prévues par le modèle cognitif. En cas d'erreur, et dès que la réponse de l'élève est différente de celle prévue, ce dernier reçoit une rétroaction du système qui vise à le ramener dans le droit chemin de la solution correcte. L'idée est d'encourager l'élève à produire des solutions correctes pour augmenter sa motivation (pédagogie de la réussite) et aussi pour ne pas laisser s'installer des erreurs. Le « traçage des connaissances » est utilisé pour évaluer l'évolution des connaissances de l'élève au cours de la résolution de plusieurs problèmes.

2.2. Les tuteurs cognitifs dans les classes

À partir des années 1990, une collaboration s'engage entre les chercheurs du groupe ACT de l'université de Carnegie Mellon et des enseignants des écoles de Pittsburgh pour développer un nouveau curriculum pour l'enseignement des mathématiques : le projet PUMP (Pittsburgh Urban Mathematics Project). Pour résoudre une situation de crise dans l'enseignement des mathématiques aux États-Unis, le conseil national des professeurs de mathématiques préconise le développement d'activités de résolution de problèmes utilisant différents registres (algébrique, géométrique et langage naturel) s'appuyant sur l'utilisation par les élèves des technologies de l'information.

Dans ce cadre, un projet dirigé par Koedinger [Koedinger et al. 1997] vise à concevoir un tuteur cognitif pour l'algèbre PAT (PUMP Algebra Tutor) dont l'utilisation est un élément important dans la mise en œuvre du projet PUMP. Ce projet permet une expérimentation de PAT sur une grande échelle et sur plusieurs années dans un contexte de classe et non plus en laboratoire. Selon les auteurs, une étude comparative entre des classes qui suivaient un enseignement traditionnel et des classes qui utilisaient PAT montre que les seconds ont significativement amélioré leurs résultats aux tests. D'autres études montrent que les élèves apprennent plus vite. Les critères d'évaluation retenus sont : la réussite aux tests standardisés et la durée d'apprentissage pour réussir ces tests.

2.3. Les activités des élèves avec un tuteur Cognitif

Les élèves utilisent les tuteurs cognitifs pour résoudre des problèmes. Lorsque l'élève le résout sans erreur le tuteur n'intervient pas. Les rétroactions revêtent plusieurs formes : un drapeau (rouge/vert) qui indique que la réponse est correcte ou non (*error flaging*), une rétro-action à la demande (demand-feedback) ou pas de rétroaction (*no-feedback*). Plusieurs niveaux d'aides sont prévus pour laisser la possibilité aux étudiants de corriger par eux-mêmes leurs erreurs avant d'obtenir la solution (*bottom out hint*).

Algebra Tutor comporte un tableur, un grapheur et un résolveur d'équations. De plus, par un suivi des connaissances en jeu dans la résolution du problème, le tuteur identifie les difficultés. Les Figure 1et 2 présentent des copies d'écran du logiciel commercialisé.

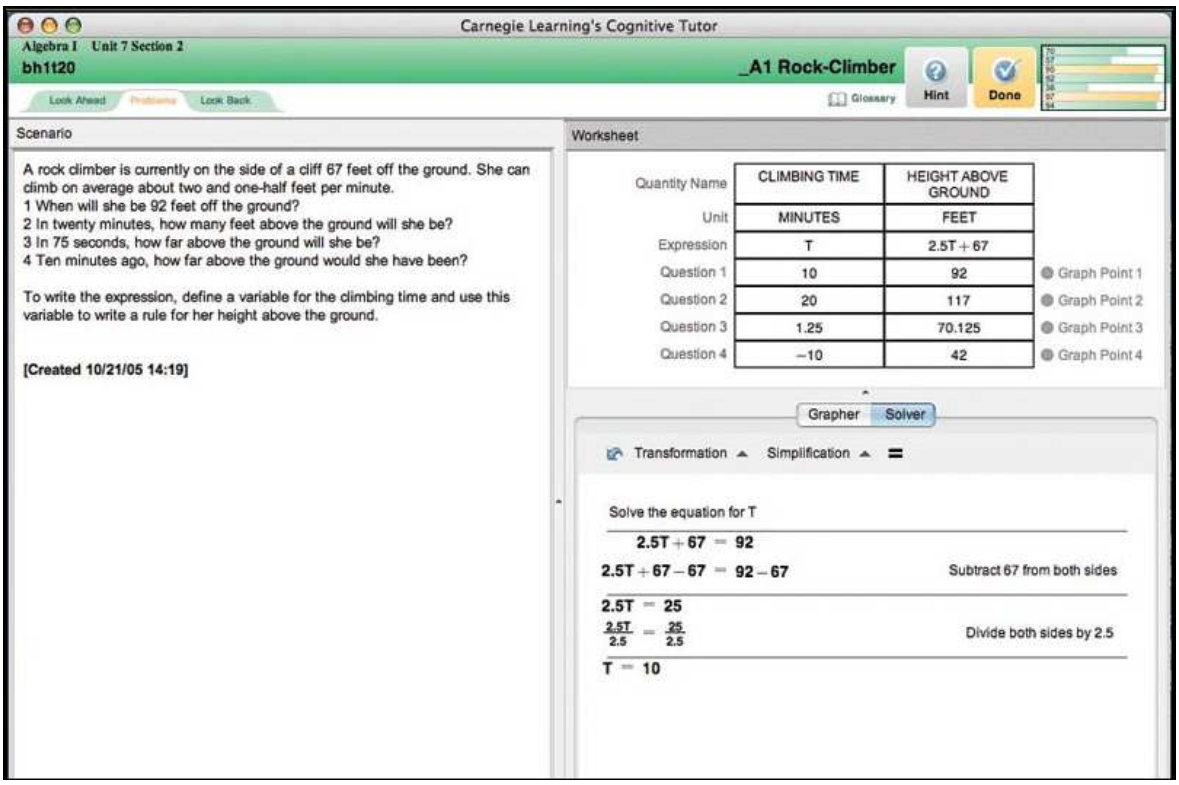


Figure 1 : une copie d'écran du logiciel Algebra 1 Cognitive Tutor [Carnegie Learning].

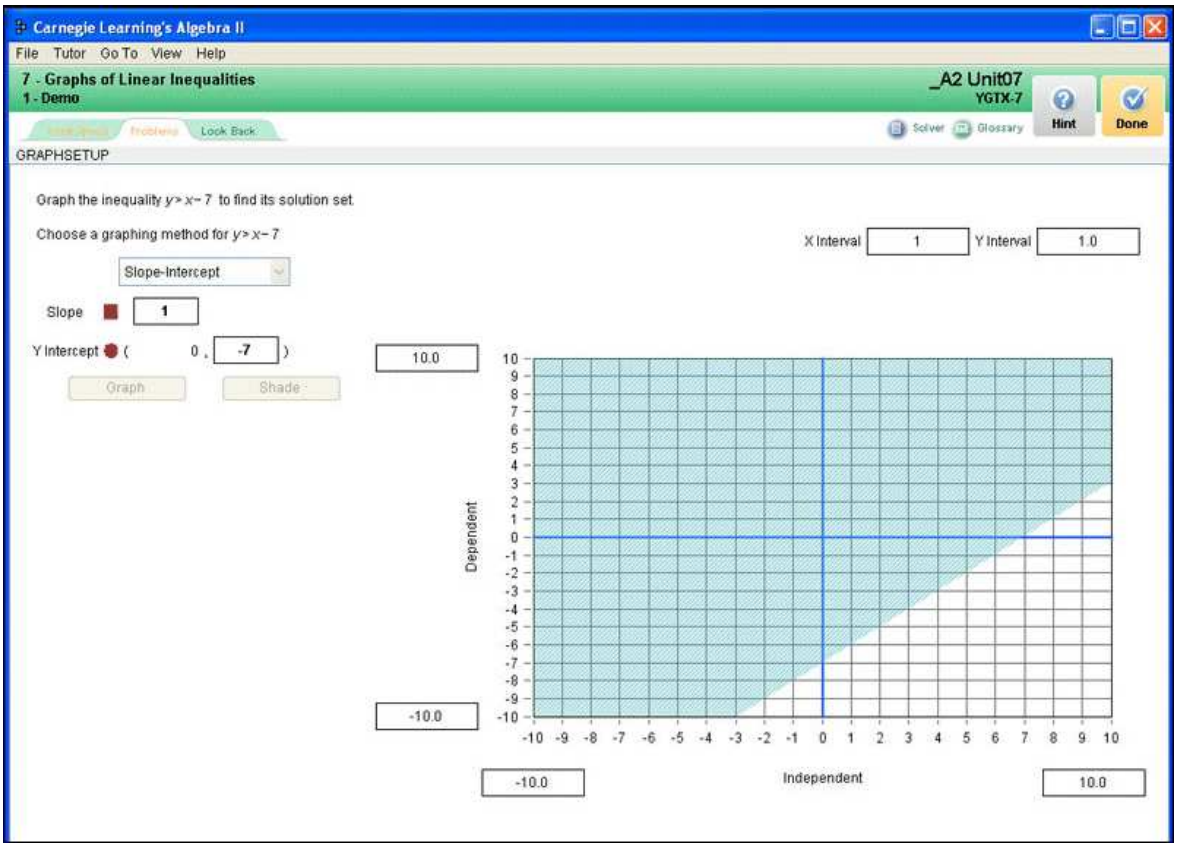


Figure 2 : Copie d'écran de Algebra 2 Cognitive Tutor [Carnegie Learning]

2.4. Le diagnostic cognitif

Dans Algebra Tutor, les réponses de l'élève sont préformatées ou sont des expressions numériques ou algébriques simples. Le principe du « traçage de modèle », revient à étudier pas-à-pas les réponses et à associer la réponse de l'élève à une règle correcte ou erronée qui aurait produit cette étape. Le diagnostic local à un exercice consiste à mettre en évidence les règles de production appliquées. Le « traçage des connaissances » repère quant à lui, l'évolution des connaissances de l'élève dans la résolution de plusieurs exercices en utilisant un réseau bayésien.

[Hefferman et Koedinger 2002] décrivent comment un tuteur cognitif qu'ils ont développé (Mrs Linquist) analyse la réponse des élèves sur des problèmes qui nécessitent la combinaison de plusieurs règles de production. Selon le niveau de compétences de l'élève, son raisonnement est modélisé par une seule règle composant les capacités simples ou plusieurs règles modélisant chaque capacité élémentaire.

Dans le logiciel Assistment, un des prototypes développés [Feng et Heffernan 2005], [Razzaq et al. 2005], les aides servent aussi à affiner le modèle de l'élève (Figure 3.a). La Figure 3.a présente une question et la Figure 3.b les aides (*hint*), le message d'erreur (*Buggy mess*age) et les questions et les messages qui apparaissent au cours de la résolution du problème par un élève. La première question apparaît seulement si l'étudiant tape « 23 » (la réponse erronée rencontrée le plus souvent pour cet item). Après une erreur, les élèves doivent répondre à une série de questions (e.g. les deux premières questions qui suivent la question d'origine). Ces questions ont un double objectif : (i) fournir à l'élève l'étayage lui permettant de réussir et (ii) affiner le diagnostic en permettant de mieux cerner la conception erronée de l'élève.

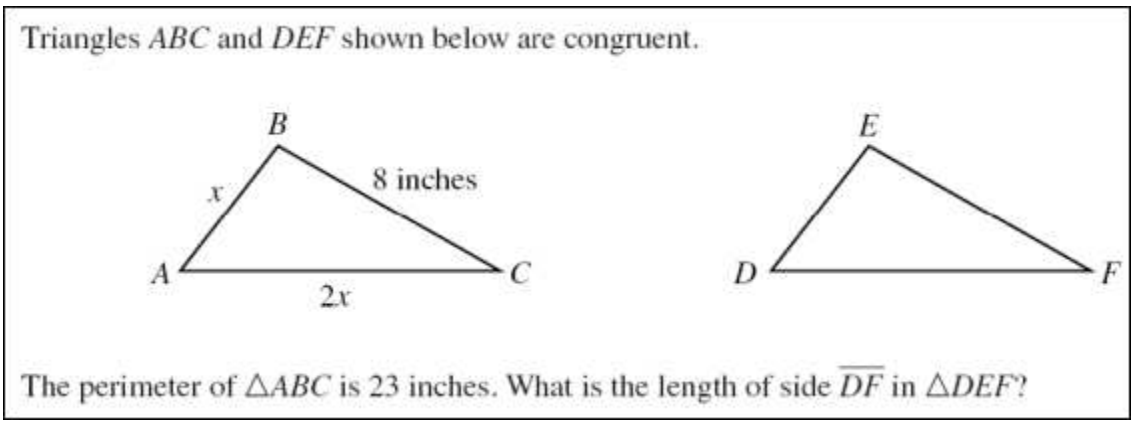


Figure 3.a : Une question dans Assistment [RAZZAQ et al. 2005]

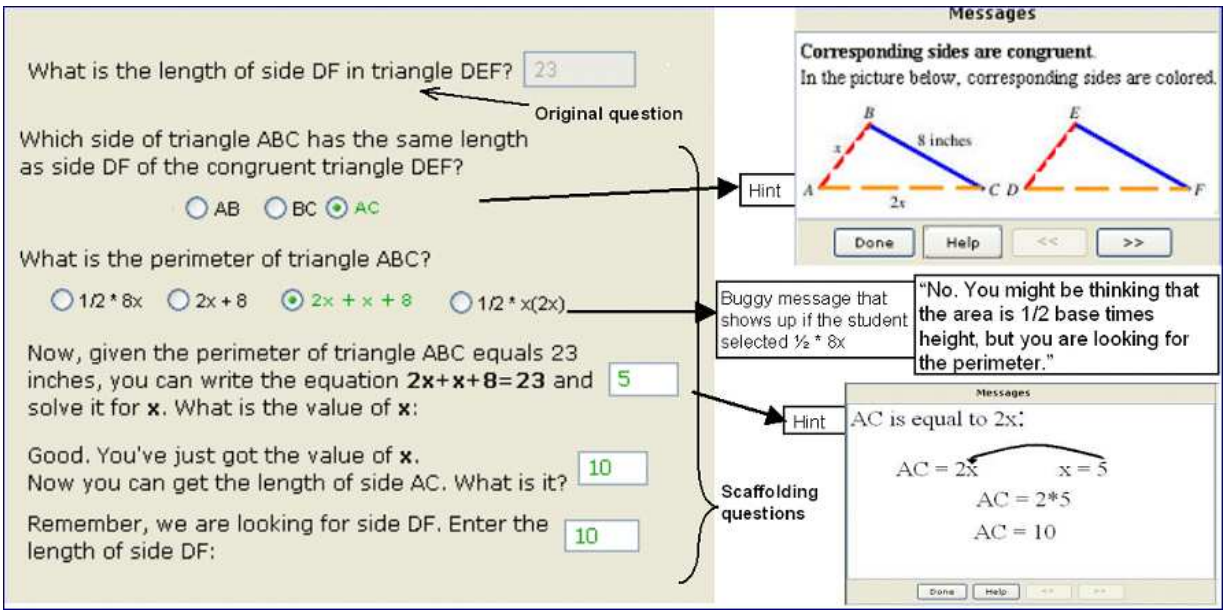


Figure 3.b : Détail des rétroactions prévues pour la question précédente [Razzaq et al. 2005]

2.5. Logiciel d'exploitation par les enseignants

Différents outils ont été développés pour aider les enseignants dans l'évaluation des compétences de leurs élèves. Le carnet de notes donne des éléments sur le travail réalisé par chaque élève : durée de réalisation d'un exercice, temps de travail dans la journée, nombre d'exercices faits, pourcentage de réponses correctes, etc. Les enseignants peuvent aussi accéder à des tableaux qui mémorisent les actions de l'étudiant, les questions qui lui ont été posées, les réponses correctes ou erronées qu'il a données, le nombre de fois où il a demandé un conseil. La Figure 4 montre un écran qui résume pour l'enseignant les résultats du diagnostic pour l'ensemble des élèves d'une classe.

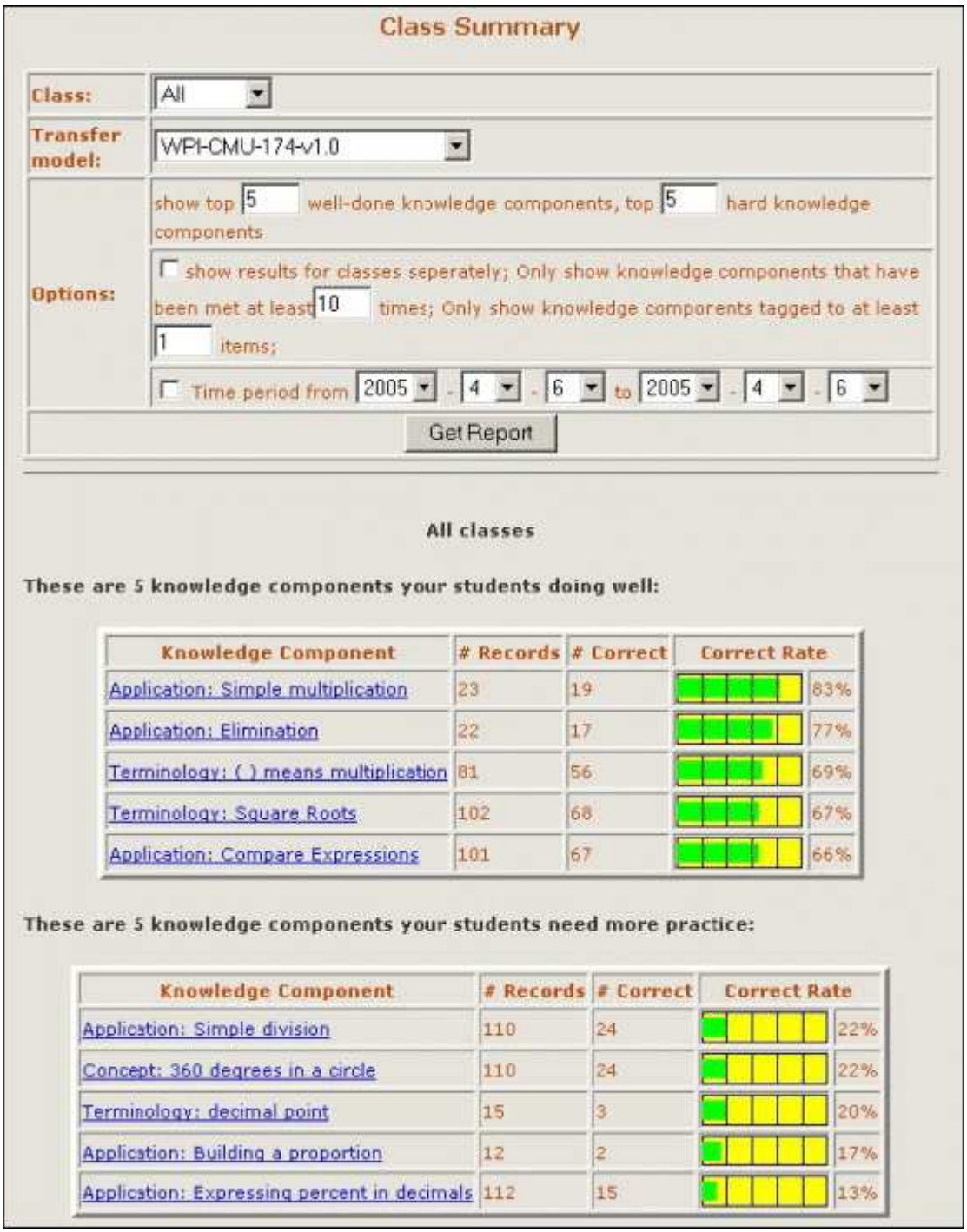


Figure 4 : Profil d'une classe affiché par Assistment [Feng et Heffernan 2006]

2.6. Outils auteurs pour les tuteurs cognitifs

Dès les années 1990, l'objectif de disséminer les tuteurs cognitifs dans les écoles américaines a poussé l'équipe à concevoir des outils de développement, d'abord pour les chercheurs, TDK, (Tutor Development Kit, [Anderson et Pelletier 1991]) puis aussi pour les enseignants, CTAT (Cognitive Tutor Authoring Tools) [Ritter et al. 1998, Blessing 2003, Aleven et al. 2006]. CTAT

permet de développer de vrais tuteurs cognitifs mais aussi des pseudo tuteurs qui sont obtenus en produisant des exemples (Example-Tracing Tutors). L'objectif de départ était de permettre aux enseignants de définir eux-mêmes des modèles cognitifs, mais, quel que soit la qualité des outils auteurs, cette tâche de modélisation est extrêmement complexe et s'est révélée hors de portée de personnes qui n'avaient pas été spécialement formées. L'équipe s'est donc orientée vers deux directions.

La première consiste à faire créer un exercice à un enseignant. L'auteur simule les différentes solutions correctes ou erronées de l'exercice qu'il crée et le système auteur construit les règles de production. L'environnement de création d'un pseudo-tuteur comporte :

- un constructeur d'interface graphique qui propose à l'auteur une palette d'outils pour créer une interface élève,
- un module d'enregistrement des actions (*Behavior Recorder*) qui mémorise sous la forme d'un parcours dans un graphe les différentes actions réalisées pour accomplir la tâche proposée.

À l'aide du constructeur d'interface graphique, l'auteur crée les énoncés, les zones de saisies, les boutons, tous les éléments de l'interface élève qui permettent les interactions entre l'élève et le pseudo-tuteur. Puis, il réalise l'exercice, selon différentes stratégies, correctes ou erronées et l'enregistreur crée au fur et à mesure un graphe des solutions correctes ou erronées (Figure 5). L'auteur a la possibilité d'associer aux différents nœuds du graphe des rétroactions (e.g. aides, messages d'erreur).

Dans l'exemple représenté par la Figure 5, l'auteur crée un exercice qui consiste à faire la somme des deux fractions. Pour réduire au même dénominateur, l'auteur a envisagé deux démarches : choisir le plus petit dénominateur commun (12), ou choisir le produit des deux dénominateurs (24). En complétant et validant les numérateurs et les dénominateurs des différentes fractions, le système créé automatiquement les liens et les nœuds du graphe. L'auteur peut aussi envisager des calculs erronés. L'auteur peut attacher à ces liens différents types de messages (rétroactions, aides, description des connaissances mobilisées pour réaliser une tâche). Ce tuteur cognitif est créé « par démonstration » sans programmation.

Ce logiciel est facile à utiliser [Aleven et al. 2006], mais l'auteur est limité par le nombre de solutions possibles car son travail peut devenir vite fastidieux si ce nombre augmente.

La deuxième direction envisagée est un travail de collaboration avec les enseignants. À partir d'exemples donnés par les enseignants, des ingénieurs cogniticiens utilisent CTAT pour créer des modèles cognitifs génériques qui sont affinés, vérifiés et critiqués par les enseignants qui en observent le fonctionnement. Les enseignants formulent également les aides et les rétroactions. C'est l'exemple du développement de Assistment [Razzaq et al. 2005].

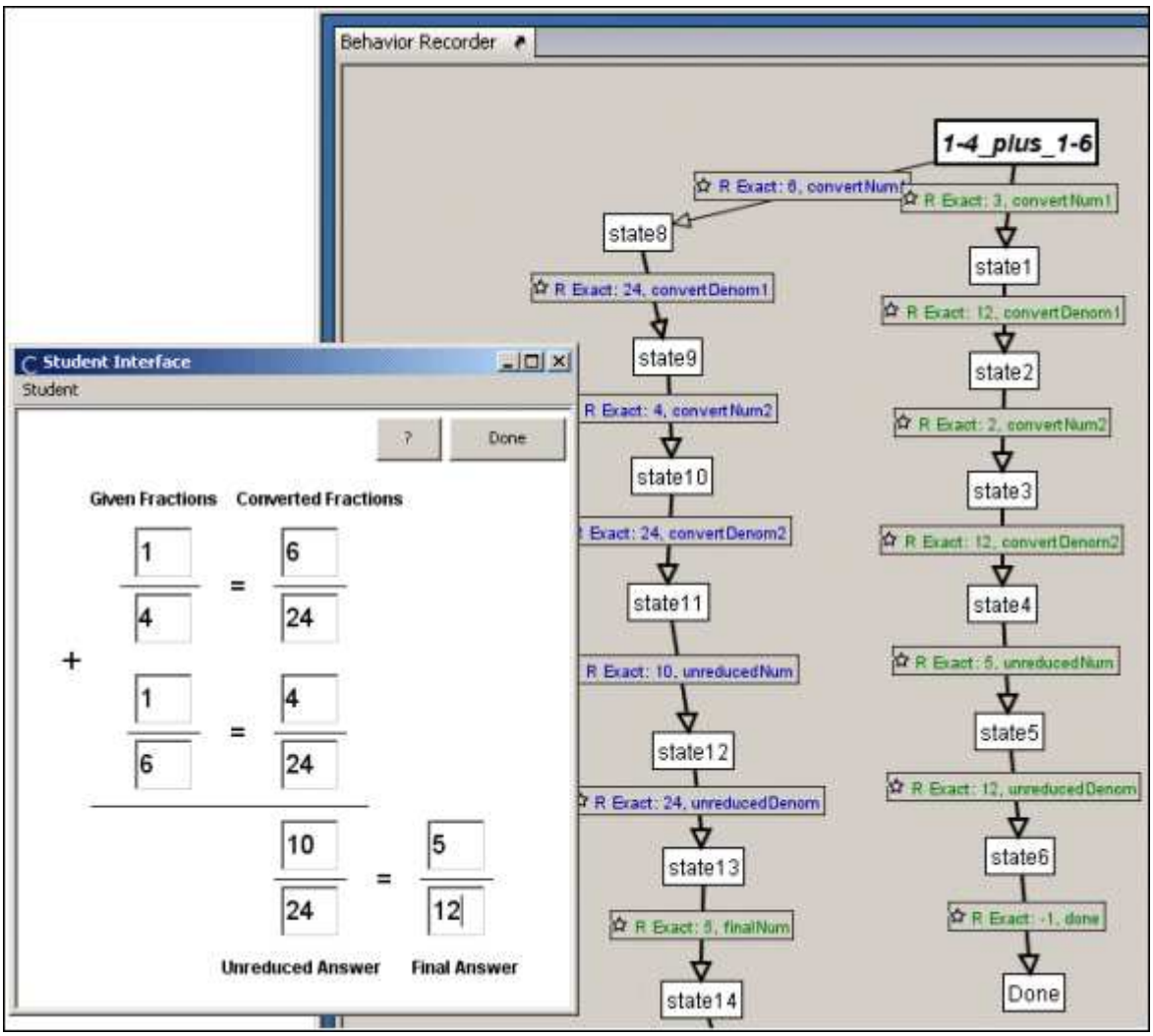


Figure 5 : Exemple de création d'un « Pseudo-Tutor » avec CTAT

2.7. Ce que je retiens pour ma thèse

Cette étude résume les travaux de vingt-cinq années de recherches de tout un laboratoire si l'on prend comme point de départ l'année 1983 où Anderson publie « The Architecture of Cognition ». De notre point de vue, l'originalité de ces travaux est triple. Premièrement, ils s'appuient sur un modèle simple de l'apprentissage, qui a cependant permis de modéliser la résolution d'un grand nombre de problèmes scolaires assez complexes [Anderson et al. 1995]. Deuxièmement, ces chercheurs ont affronté les problèmes de l'intégration de produits issus de la recherche dans le système éducatif qu'ils contribuent ainsi à faire évoluer. Troisièmement, l'utilisation à grande échelle de ces tuteurs permet des études quantitatives riches d'enseignement.

Notre travail ne peut se comparer à ces recherches, ne serait-ce que par leur ampleur ; nous nous limitons, dans notre thèse, au diagnostic alors que Algebra Tutor couvre les apprentissages et le diagnostic soit une année d'enseignement. Néanmoins, sur le thème des outils auteurs de diagnostic, nous pouvons préciser la spécificité de notre approche.

Nous nous fondons, nous aussi, sur un modèle de la compétence attendue à un niveau scolaire donné. Ce modèle comporte à la fois les capacités attendues et un certain nombre de conceptions erronées. Le modèle de Grugeon n'utilise pas le formalisme des règles de production des tuteurs cognitifs. Tout d'abord, il n'est pas une application d'un modèle général de la cognition mais s'appuie sur une analyse épistémologique et didactique du domaine considéré. Ainsi, la nature du diagnostic est différente.

« In general we do not attempt to provide any deep diagnosis of the cognitive origins of the error. Rather, we simply try to explain why it is an error » [Anderson et al. 1995: 199].

Le modèle qui fonde notre travail ne repère pas simplement des erreurs mais des cohérences dans le fonctionnement des élèves pour expliquer la récurrence et la résistance de ces erreurs. Le diagnostic que nous devons mettre en place ne se limite pas à détecter une seule règle mais doit conduire une analyse de la réponse selon plusieurs dimensions. De plus, ce modèle fait une place plus grande aux productions des élèves dont les réponses ouvertes sont plus complexes que celles qui sont étudiées par Algebra Tutor.

En ce qui concerne le système auteur, si nous développons des outils pour faciliter le développement par des informaticiens de modèles d'exercices (par exemple en développant un logiciel de calcul formel pour l'analyse des raisonnements algébriques), notre objectif est de minimiser la charge de travail des enseignants en automatisant au maximum la création d'exercices et l'analyse des réponses. Notre méthode correspond plus à celle mise en œuvre dans Assistment à savoir développer des modèles génériques à partir d'exemples mis au point par des enseignants et des didacticiens, plutôt que de les faire développer par les enseignants qui ont bien d'autres charges.

3. Andes, un tuteur en Physique

Le système Andes, développé par l'équipe de Kurt VanLehn à l'université de Pittsburg et à l'United States Naval Academy, est aussi un tuteur intelligent. À l'instar des recherches sur les Tuteurs Cognitifs, l'équipe de Andes se préoccupe également de l'intégration des Tuteurs Intelligents dans le système universitaire. Mais, contrairement aux travaux de l'équipe d'Anderson qui sont associés à une vaste réforme des programmes de mathématiques au lycée,

Andes vise à s'intégrer en douceur dans les pratiques usuelles d'enseignement de la physique [VanLehn et al. 2005b]. L'objectif est de favoriser les apprentissages en aidant les étudiants à rendre leur travail personnel plus efficace sans modifier ni les cours, ni les livres, ni les exercices, ni les évaluations, ni la pédagogie. Les auteurs comparent Andes aux logiciels de soutien scolaire qui permettent aux étudiants de s'entraîner à la résolution de problèmes. La différence essentielle avec ces logiciels est qu'Andes ne s'intéresse pas seulement à la réponse au problème, mais à l'ensemble de la solution élaborée par l'étudiant en le laissant libre de produire sa solution comme il l'entend, ce qui est très important pour l'apprentissage de connaissances complexes.

Ce logiciel est utilisé régulièrement depuis 1999 dans le cadre des cours de physique à l'United States Naval Academy et, plus récemment, dans les LearnLab Courses du Pittsburg Science of Learning Center que nous avons déjà mentionné (http://www.learnlab.org). Il couvre une part importante du programme de première année en mécanique, en électricité et en magnétisme. Une version d'Andes est téléchargeable et fonctionne aussi en ligne (http://www.andes.pitt.edu). Ce système a donné lieu à de multiples expérimentations et sert de support à des études permettant de tester des hypothèses sur l'apprentissage et sur les techniques permettant de mieux comprendre et d'améliorer l'apprentissage, en physique mais aussi dans d'autres domaines, par exemple en statistique [Chi et VanLehn 2007] ou par le dialogue [Katz et al. 2003]. Un travail important a été mené autour d'Andes pour analyser les réponses à des questions ouvertes comportant des résolutions de systèmes d'équations [Shapiro 2005] et des réponses en langage naturel [Rose et al. 2003, Jordan et al. 2006].

3.1. Les fondements

La conception d'Andes repose sur une analyse didactique et cognitive très précise du domaine à enseigner [VanLehn et al. 2005]. Les auteurs considèrent la résolution d'un problème de physique comme une preuve où les inférences sont justifiées par des principes de physique (e.g. la seconde loi de Newton, la relation entre le poids et la masse) et aussi d'autres principes secondaires pour l'apprentissage de la physique mais indispensables pour la résolution de problèmes comme les principes mathématiques ou les relations qui montrent une bonne compréhension des phénomènes physiques. Andes possède une représentation de chacun de ces principes pour aider les étudiants à les maîtriser pour résoudre les problèmes. Le premier objectif pédagogique d'Andes est d'amener les étudiants à résoudre des problèmes en leur enseignant explicitement tous les principes qui sous-tendent leur résolution. Le second objectif est de conduire les étudiants à envisager les solutions à partir des principes de physique et non à partir

de traits de surface (e.g. un plan incliné). Il s'agit pour les auteurs de développer une compréhension du problème au niveau conceptuel.

La résolution des exercices dans Andes demande de la part de l'élève une analyse de la situation physique, une connaissance des relations entres les grandeurs physiques mises en jeu dans la situation, l'affectation de variables algébriques pour représenter ces grandeurs, la traduction des relations entres ces grandeurs sous formes d'équations algébriques et, enfin, l'utilisation de l'outil algébrique pour résoudre ces équations et calculer ainsi les quantités inconnues. La plupart de ces étapes n'ont donc pas une réponse unique. Ainsi la modélisation des connaissances des étudiants est beaucoup plus complexe que dans les tuteurs cognitifs. Les solutions acceptables ou erronées ne peuvent pas être décrites comme une suite d'étapes successives et prédéterminées mais se traduisent par un ensemble de contraintes sous la forme d'équations algébriques.

3.2. Les activités des élèves avec Andes

Pour résoudre un problème avec Andes (Figure 6), un étudiant commence par lire l'énoncé, puis il doit dessiner les vecteurs et définir les variables en jeu. Ensuite, il entre des équations dans l'ordre de son choix tant que les éléments saisis sont corrects. Cependant, Andes encourage l'étudiant à commencer par les équations traduisant d'abord les principes généraux avant de les instancier (objectif n° 2). À chaque saisie de l'étudiant, le système indique si la saisie est correcte (vert) ou incorrecte (rouge), lui laissant la possibilité de corriger spontanément ses erreurs ou de demander ce qui est faux (*What's wrong with that ?*). L'étudiant peut aussi demander une aide sur l'étape de résolution suivante (*What's next* ?) ou pour résoudre les équations qu'il a écrites. Comme dans les tuteurs cognitifs, une séquence d'aides est prévue pour faire réfléchir l'étudiant avant de donner la solution.

3.3. Le diagnostic dans Andes

Andes évalue la validité de la réponse d'un étudiant à chaque entrée afin d'afficher la réponse en vert (correcte) ou rouge (incorrecte). Pour ce *flagging feedback*, le principe est de situer la réponse de l'élève dans un graphe des solutions correctes. Andes distingue deux types d'entrées : les équations et les non-équations (principalement : dessins de vecteurs, définitions des variables). Pour ces dernières, la validité est simple à vérifier : une étape de ce type est correcte si elle correspond exactement à un élément du graphe des solutions. Pour les premières, la validité est plus complexe à établir. Quand un étudiant saisit une équation, Andes vérifie d'abord sa syntaxe à l'aide d'une grammaire *context-free* mise au point à partir d'un corpus de plusieurs

milliers d'équations. Puis il vérifie que les dimensions sont consistantes. Ensuite il vérifie l'équation de l'étudiant par une technique numérique (*color by number*) : Andes remplace par leur valeur dans la solution finale du problème, les variables présentes dans l'équation saisie par l'étudiant. Shapiro [Shapiro 2005] présente une démonstration qui prouve qu'une équation est correcte (coloriée en vert) par cette méthode si et seulement si elle peut être obtenue par un raisonnement mathématiquement correct à partir des données du problème et des principes de physique. D'après les auteurs, cette technique s'est avérée très efficace en dépit d'une limitation : dans certains cas, une sous-expression incorrecte peut être coloriée en vert.

Pour générer le *What's wrong? feedback,* Andes fait appel à une base de connaissances de gestionnaires (*handlers*) spécifiques des erreurs répertoriées. Un gestionnaire d'erreurs prend en entrée la réponse de l'étudiant et retourne *nil* s'il ne la reconnaît pas ou une séquence d'aides et une priorité s'il la reconnaît. Ainsi pour générer cette rétroaction, Andes s'appuie sur la solution finale du problème, sur l'ensemble des quantités intervenant dans le problème et sur la base de connaissances des gestionnaires d'erreurs.

Pour déterminer l'aide sur le prochain pas de résolution (*Next step? help*), Andes commence par déterminer le pas de résolution à traiter. Du fait de la relative liberté de l'étudiant dans l'organisation de sa résolution (les actions sont faiblement ordonnées) la reconnaissance du plan de l'étudiant est délicate. D'autant que de nombreux étudiants débutants ne semblent pas avoir de plan. Pour déterminer le pas suivant, Andes choisit dans la branche qui contient le plus d'étapes déjà traitées, le premier pas de solution que l'étudiant n'a pas traité. Ensuite, il s'assure que l'étudiant a effectué l'étape principale qui consiste, par exemple, à énoncer le principe de physique en jeu. Si oui, Andes produit la séquence d'aides attachée au pas de résolution, sinon il produit celle qui est lié à l'étape principale. Dans cette approche la difficulté consiste à marquer les étapes de résolution qui sont effectuées. Shapiro a mis au point un algorithme (*indy check algorithm,* [Shapiro 1995]) qui permet d'éviter de calculer toutes les combinaisons algébriques possibles des étapes.

Cette équipe a mené de nombreux travaux pour construire un modèle de l'élève en utilisant des réseaux bayésiens [VanLehn et Martin 1997, VanLehn et Niu 2001]. Cependant cette technique, si elle s'est montrée efficace pour l'évaluation des principes maîtrisés ou non par les étudiants, elle s'est avérée inutile voir problématique pour produire les rétroactions et les aides et a été abandonnée au profit de l'approche heuristique que nous venons de présenter rapidement. Le réseau bayésien pourrait par contre être utilisé pour définir des parcours d'apprentissage.

3.4. Les outils auteurs dans Andes

C'est un ingénieur cogniticien qui ajoute un exercice dans Andes. Nous n'avons pas trouvé de description précise de cette activité. Elle s'appuie cependant sur deux modules fondamentaux : le module qui génère le graphe des solutions de l'exercice et le module de calcul formel qui permet le diagnostic des entrées des étudiants.

3.4.1. Le graphe des solutions correctes

Le résolveur de problèmes de physique s'appuie sur des bases de connaissances composées de règles de production modélisant les principes de la physique et les objets sur lesquels les appliquer. Pour chaque problème, ce résolveur crée le graphe des solutions qui est sauvegardé avec l'énoncé de l'exercice. Cette technique a l'avantage d'accélérer le diagnostic (puisque le graphe est calculé une fois pour toutes) mais surtout de faciliter la maintenance et l'évolution de la base de connaissances. Quand une nouvelle version est produite, les fichiers des solutions des exercices sont automatiquement comparés pour se prémunir contre les effets indésirables des nouveautés introduites.

3.4.2. Le système de calcul formel spécifique

Shapiro [Shapiro 2005] a développé un module de calcul formel qui gère tous les problèmes mathématiques liés soit à la génération du graphe des solutions, soit au diagnostic des entrées des étudiants, soit à la mise à disposition d'outils pour supporter la résolution des systèmes d'équations par les étudiants.

La résolution d'un système d'équations est ce que l'on peut attendre d'un système de calcul formel. Cependant les concepteurs de Andes ont préféré créer leur propre système de résolution d'équations pour mettre en œuvre des méthodes spécifiques de résolution des équations en physique (par exemple pour vérifier la consistance des dimensions), mais surtout pour répondre aux besoins du diagnostic.

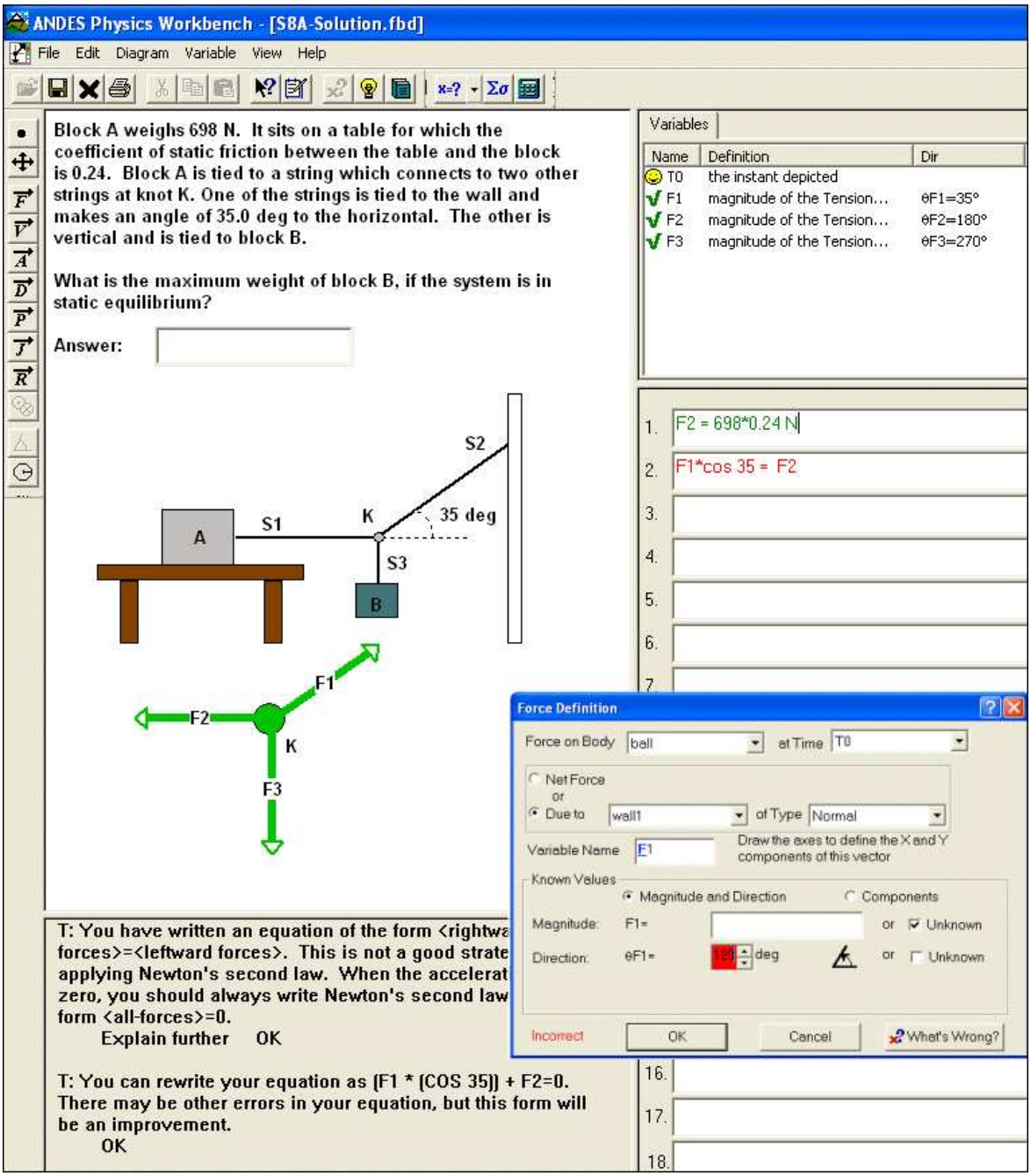


Figure 6 : Un exercice en cours de résolution avec Andes [http://www.andes.pitt.edu/]

3.5. Ce que j'en retiens pour ma thèse

Andes est un tuteur intelligent opérationnel qui est utilisé depuis de nombreuses années dans des cours et par de gros effectifs. Il a donné lieu à de nombreux prototypes pour tester des hypothèses de recherche. De plus, cette équipe a mené de nombreux travaux pionniers et mis au point des techniques innovantes sur les réseaux bayésiens, sur les dialogues tutoriaux et aussi sur l'analyse automatiques de textes saisis par les étudiants.

En ce qui concerne le diagnostic des réponses ouvertes, Andes analyse chaque étape du raisonnement des étudiants en s'appuyant sur un graphe des solutions correctes construit une fois pour toutes. Pour créer un exercice, dans PépiGen, nous générons et mémorisons aussi une grille d'analyse des solutions anticipées pour étudie chaque ligne du raisonnement de l'élève. Mais contrairement au graphe des solutions d'Andes, notre grille d'analyse présente aussi les solutions incorrectes car notre objectif est de repérer ces solutions incorrectes pour obtenir un diagnostic cognitif, alors que celui d'Andes est d'améliorer les connaissances des étudiants par la résolution de problèmes. Nous partageons le point de vue des concepteurs d'Andes sur l'intérêt de construire la représentation des solutions une fois pour toutes. Dans le projet Lingot, cette option présente un avantage supplémentaire. Ce n'est pas une grille abstraite comme une base de règles, mais cette grille réifie l'application concrète des dimensions et des critères d'évaluation sur un exercice, ce qui la rend beaucoup plus explicite pour les humains et leur permet ainsi des vérifications plus faciles pour la mise au point. Ce point est important car nous avions constaté la difficulté, pour les enseignants non familiers avec l'analyse didactique, de comprendre les critères d'analyse hors du contexte des réponses d'élèves illustrant l'application de ces critères.

De la même façon que Shapiro a développé un logiciel de calcul formel spécifique aux problèmes de physique d'Andes, une part importante de notre travail a consisté à développer un système de calcul formel spécifique à l'algèbre élémentaire car les systèmes de calcul formel usuels se révèlent inadaptés au diagnostic fin des réponses d'élèves que nous voulons produire. Notre logiciel est capable de produire les solutions correctes (comme celui de Shapiro) mais aussi les solutions incorrectes alors que Andes propose seulement des rétroactions si l'erreur est répertoriée dans le gestionnaire d'erreurs. Dans Andes, l'objectif du diagnostic est de vérifier la validité de la réponse et de fournir une rétroaction locale à l'étape de calcul et non d'identifier une démarche globale de résolution, alors que notre objectif est d'identifier les caractéristiques de la démarche de l'élève et non ses erreurs à une étape.

Un autre projet fondé sur un logiciel de calcul formel spécifique pour le domaine de l'algèbre élémentaire est le logiciel Aplusix.

4. Aplusix

Ce projet de recherche a pour objectif le développement d'environnements d'apprentissage de l'algèbre élémentaire et la modélisation des connaissances des élèves dans ce domaine. Ce projet, dirigé par Jean-François Nicaud, a vu le jour au laboratoire de Recherche en Informatique (LRI) à Orsay. C'est maintenant un projet pluridisciplinaire développé principalement à Grenoble et regroupant des chercheurs en Informatique du LIG, en Didactique des Mathématiques de

Did@tic et en Psychologie Cognitive de l'université Paris 8. Le travail de cette équipe s'est centré autour de la conception, du développement et d'expérimentation d'un logiciel portant le même nom que le projet, le logiciel Aplusix. Ce dernier a été conçu au début comme un tuteur intelligent qui guide un élève dans la résolution d'exercices d'algèbre ; l'élève libéré des calculs qui sont effectués par la machine, peut se concentrer sur les aspects stratégiques du raisonnement algébrique. Actuellement, Aplusix est un micromonde qui permet aux élèves de produire et de manipuler des expressions algébriques pour, en particulier, résoudre des équations, des inéquations, des systèmes d'équations et d'inéquations ,et des problèmes de factorisation, développement et de réduction de polynômes du niveau de l'enseignement secondaire [Bouhineau et Nicaud 2006]. Aplusix est commercialisé et maintenant disponible en plusieurs langues ; une version de démonstration peut être téléchargée [site Aplusix].

4.1. Une théorie de l'algèbre

La conception de l'environnement Aplusix repose sur un cadre théorique pour une modélisation cognitive et computationnelle de l'algèbre [Nicaud 1994, Gelis 1994, Nicaud et al. 2001]. Dans ce cadre théorique, l'objet central est l'expression algébrique ; un raisonnement algébrique, pour les types de problèmes considérés dans l'enseignement secondaire, consiste à appliquer des règles de réécriture qui conservent l'équivalence des expressions. Dans le chapitre 6, nous présentons des aspects formels de ce travail. Nous nous limitons ici à illustrer le fonctionnement du résolveur de problèmes d'Aplusix conçu pour mettre en œuvre les modèles proposés.

Aplusix résout des problèmes algébriques qui sont définis par un type (e.g. factoriser, résoudre) et une expression algébrique E. Le problème se résout en appliquant à E (ou à des sous-expressions propres et bien formées de E) des règles de transformation qui conservent l'équivalence des expressions jusqu'à l'obtention d'une solution. Le Tableau 2 présente un exemple de résolution d'un problème de factorisation.

Pour l'exemple présenté dans le Tableau 2, la première règle appliquée à l'expression x^2-9 s'écrit : $A^2 - B^2 \rightarrow (A - B)(A + B)$

(x+3) (x-5) + x^2-9 : énoncé
(x+3) (x-5) + (x+3) (x-3) : factorisation de x^2-9
(x+3) (x-5+x-3) : mise en facteur de x+3
(x+3) (2x-8) : réduction de x-5+x-3

Tableau 2 : exemple de factorisation

La Figure 7 montre un exemple d'écran illustrant le fonctionnement du premier logiciel Aplusix.

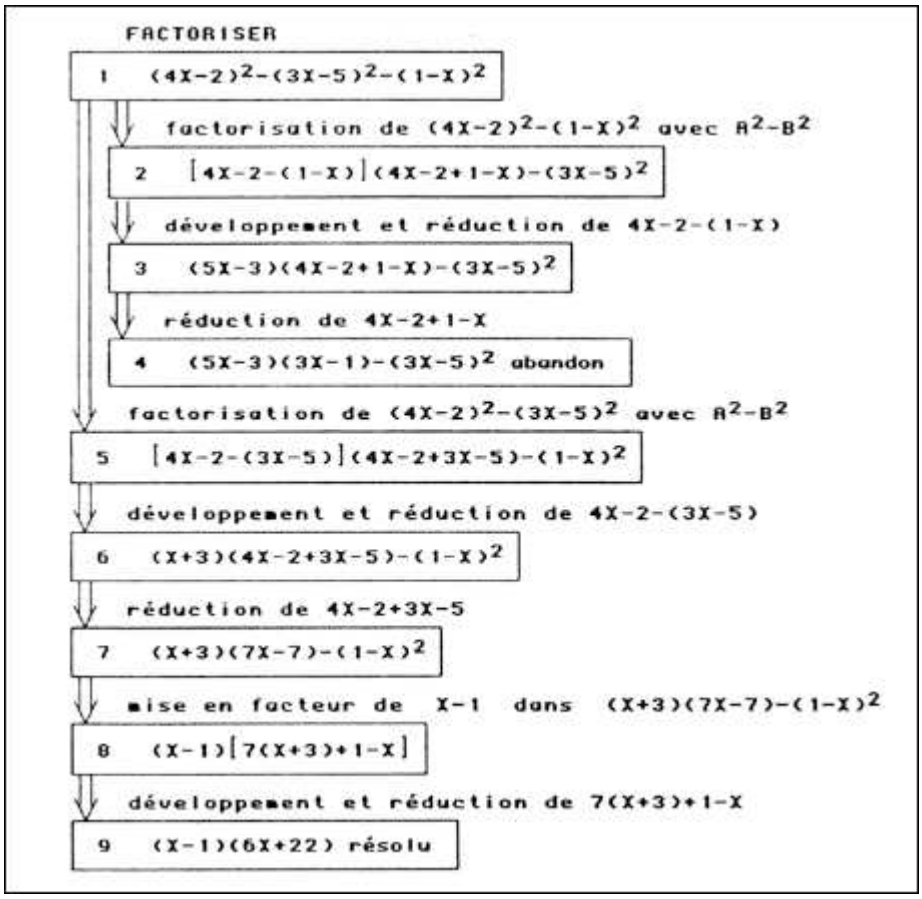


Figure 7 : Représentation à l'écran d'un raisonnement (extrait de [NICAUD 1993])

Lorsque cela est possible, les variables de la règle sont substituées par des expressions (dans l'exemple A est remplacée par x et B par 3). Une *étape de calcul* est une expression algébrique présentée comme résultat de l'application d'une règle. La production d'une étape de calcul à partir d'une expression E peut se modéliser ainsi :

1. Détermination des règles applicables à des sous-expressions de E (cela inclus E),
2. Choix des règles applicables,
3. Application de la règle choisie.

Le choix de la règle de transformation est appelé *stratégie* et les éléments qui constituent la stratégie (qui sont des règles d'un type différent des règles de transformation) sont appelés *heuristiques.*

4.2. Les activités des élèves avec Aplusix

Parallèlement à cet effort pour définir les fondements mathématiques et cognitifs d'Aplusix, un travail très important a été mené pour créer un environnement pour l'élève qui soit à la fois facile à utiliser et pertinent du point de vue des apprentissages [Bouhineau et Nicaud 2006]. Ces auteurs relèvent ainsi une tension entre la qualité ergonomique de l'interface et ses qualités didactiques.

Dans Aplusix, un élève résout des problèmes en produisant ligne après ligne les différentes expressions algébriques correspondant à son raisonnement. Deux modes de fonctionnement sont proposés. Le mode « d'entraînement »fournit un étayage sous forme de rétroactions visuelles de type *flagging feedback*, indiquant si les expressions produites par l'élève sont équivalentes : noir pour les expressions valides, bleu pour les expressions incomplètes ou mal typées et rouge pour les expressions indéfinies (Figure 9). Dans le mode « test », aucune rétroaction n'est affichée. Du point de vue de l'élève, la grande originalité d'Aplusix consiste dans l'éditeur d'expressions, qui a été mis au point pour être le plus proche possible de l'environnement papier-crayon.

L'élève manipule les expressions à la souris et dispose aussi d'un clavier virtuel (Figure 8). Un menu « Observer/corriger mon travail » lui permet de revenir sur les exercices qu'il a réalisés et particulièrement sur tous les détails de ses actions en utilisant le « magnétoscope », un menu « Statistiques » affiche ses résultats. Les Figure 8 et Figure 9 présentent des copiées d'écran du logiciel.

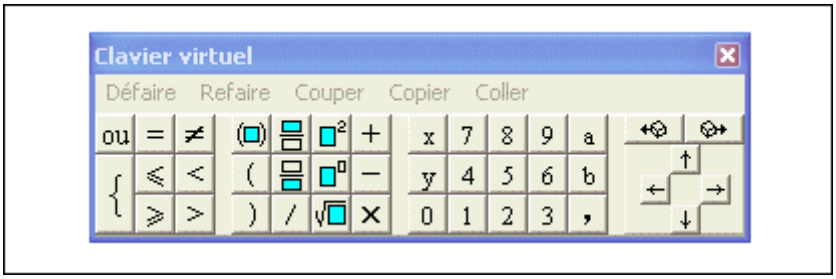


Figure 8 : Le clavier virtuel du logiciel Aplusix

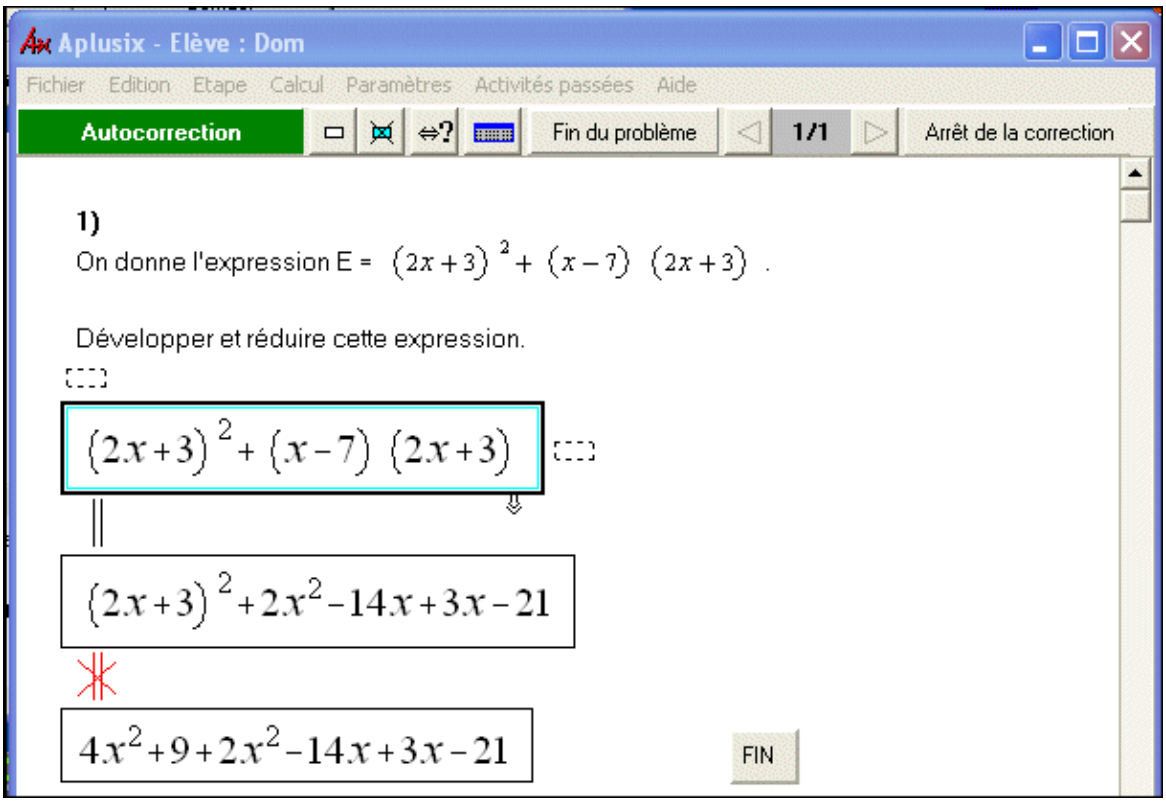


Figure 9 : Exemple d'utilisation de Aplusix en mode entraînement

4.3. Le diagnostic dans Aplusix

Dans Aplusix le diagnostic se fait en deux temps [Chaachoua et al. 2007]. Tout d'abord chaque pas de calcul est analysé en termes de règles de réécriture correctes ou erronées (diagnostic local). Puis l'ensemble des productions de l'élève est interprété en termes de théorèmes en actes [Vergnaud 1991] pour interpréter le fonctionnement de l'élève non plus au niveau des pas de calcul mais au niveau des tâches algébriques (diagnostic global).

Le diagnostic local s'appuie sur la théorie des règles de réécriture [Dershowitz et Jouannaud 1990] que nous ne détaillerons pas ici car elle est décrite largement dans le chapitre 6. Un gros travail a été fourni par cette équipe pour mettre au point une bibliothèque de règles correctes et erronées à partir de l'étude de nombreuses résolutions de problèmes menées avec Aplusix par plusieurs milliers d'élèves en France et au Brésil principalement.

Le diagnostic local est mis en œuvre par un logiciel, nommé Anaïs, qui interprète les productions de l'élève en termes de règles de réécriture. Pour chaque pas de calcul de l'élève comprenant une expression de départ et une expression cible, un moteur de recherche heuristique en chaînage avant développe un arbre en appliquant toutes les règles de la bibliothèque applicables à l'expression de départ. Des heuristiques permettent de contrôler l'expansion de

l'arbre en prenant en compte la distance à l'expression cible. Quand le processus réussit, si plusieurs chemins ont été détectés, ces chemins sont classés par ordre de coût décroissant. Anaïs produit une « trace enrichie sémantiquement » en associant à chaque pas de calcul les règles de transformations qui permettent d'interpréter les productions des élèves. Notons qu'Anaïs sert aussi à la mise au point et à la maintenance de la bibliothèque de règles de réécriture. Comme dans le projet Andes, des comparaisons automatiques entre les fichiers de diagnostic produits entre deux modifications de la base de règles permettent de contrôler les effets de bords de ces modifications.

Le diagnostic global consiste à regrouper les milliers de règles de réécriture en paquets de règles pour caractériser des règles d'actions [Vergnaud 1991], puis de regrouper les règles d'actions en théorèmes en actes (ibid.). Pour mettre en œuvre ce type de modélisation, Chaachoua et ses collègues ont mis au point une description des règles de réécriture en distinguant les « variables de contexte » se rapportant à l'expression traitée (e.g. équation ou inéquation) et les « traits » caractérisant la transformation (e.g. changement de signe de l'argument sur lequel porte la transformation). À une règle de réécriture mise en évidence par le diagnostic local, sont associés un ou plusieurs vecteurs appelés VCL (Vecteur de comportement local selon un trait). L'étude du corpus permet de détecter, pour chaque trait, des VCL stables qui correspondent aux règles d'actions, puis de regrouper ces règles d'actions. Cette méthodologie a été appliquée sur les règles de type mouvement dans la résolution d'équation ou d'inéquation pour laquelle les auteurs distinguent trois traits : le signe, le sens et l'opérateur ce qui est illustré par l'exemple suivant.

La stratégie principale de résolution d'une équation du premier degré est d'isoler la variable en utilisant des règles de récriture qui permettent d'effectuer des opérations sur les deux membres (e.g. Règle 1 : $A = B \rightarrow A - C = B - C$). Ces règles combinées avec des règles de réduction aboutissent à des règles dites « règles de mouvement » additif ou multiplicatif selon l'opération effectuée (e.g. Règle 2 : $A = B \rightarrow A - B = 0$, règle additive) [Nicaud et al. 2005a]. Ce mouvement qu'il soit additif ou multiplicatif porte sur une sous-expression appelée *Argument* : c'est l'objet qui passe d'un membre dans l'autre. Ces mécanismes sont modélisés par une règle appelée *Mouvement* à laquelle est associée un vecteur de sept variables qui décrivent le contexte (l'expression) et les traits ou les paramètres du mouvement (e.g. l'orientation horizontale du mouvement de la droite vers la gauche, le changement de signe de l'argument, le changement de sens). Le Tableau 3 donne un exemple de diagnostic produit avec cette méthode.

Diagnostiquer 2x+4 = 7x-6 → -5x = -2 Avec ces données, deux diagnostics sont produits.	
Le meilleur est : - mouvement additif correct de 7x - réduction correcte 2x-7x → -5x - mouvement additif erroné de 4 - calcul correct -6+4 → -2	L'autre diagnostic est : - mouvement additif correct de 7x - réduction correcte 2x-7x → -5x - mouvement additif correct de 4 - calcul erroné -6-4 → -2

Tableau 3 : Exemple de diagnostic obtenu avec l'algorithme [Nicaud et al. 2005b]

Pour chaque trait un « théorème en actes global (TeA global) » est un ensemble d'actions du même type quelque soit le contexte. Il peut donc être correct dans un contexte et incorrecte dans un autre. Par exemple, pour la tâche « mouvement », un TeA global est « conservation du signe » qui est correct pour les mouvements multiplicatifs, et incorrect pour les mouvements additifs. Pour cette tâche et ce trait les auteurs ont mis en évidence 5 théorèmes en actes globaux. Ceux-ci sont particularisés en TeA de profondeur 1, 2 ou 3 suivant que l'on fixe une, deux ou trois variables de contexte. Ceci donne un treillis comportant 22 nœuds pour les TeA relatifs au trait signe de la tâche mouvement. Notons que ces théorèmes en actes ont été déterminés automatiquement par l'analyse des données recueillies lors des expérimentations.

4.4. Outil auteur

Une première étude menée envisageait trois approches pour la génération d'exercices dans Aplusix [Baron et Simonet 1992] : (i) une approche qui consiste à définir « un prototype » caractérisé par un type de problème (e. g. factoriser, développer, résoudre), une expression générale F, et un ensemble de contraintes portant sur les variables de F, le générateur se contentant d'instancier les variables en respectant les contraintes, (ii) définir seulement des contraintes et le générateur génère les exercices et appelle le résolveur de problèmes pour vérifier qu'ils ont une solution en utilisant les connaissances de la base de connaissances, (iii) une génération obtenue à partir de cas en identifiant des différences et des ressemblances entre exercices.

Quelques années plus tard, la terminologie a changé, mais c'est la première approche qui a été retenue pour construire automatiquement des listes d'exercices engendrés à partir de patrons [Bouhineau et al. 2005a].

Ces patrons sont organisés en une hiérarchie dont les feuilles sont appelées des *patrons effectifs* et les nœuds des *patrons abstraits*. Un patron abstrait définit une famille paramétrée par des variables. Il est effectif quand il génère un exercice par instanciation des variables [Bouhineau et al. 2005b]. Le choix des variables prend en compte les résultats des recherches en

didactique, les programmes scolaires, les difficultés repérées par les enseignants. Au sommet de la hiérarchie des patrons abstraits, on trouve les cinq types de problèmes : calculer, factoriser, développer, réduire, résoudre. Chacun de ces patrons abstraits se décomposent de nouveau en patrons abstraits qui correspondent à des formes d'expressions algébriques avec lesquelles les exercices seront générés.

Des variables didactiques sont prises en compte pour instancier ces patrons : ce sont par exemple le type de nombres utilisés, la nature des opérations, le degré des polynômes, le nombre de termes, le nombre de parenthèses. Le Tableau 4 donne un aperçu des cinq grands types abstraits du type « Factoriser » qui se répartissent eux-mêmes en sous-catégories. Ainsi 436 patrons effectifs ont été retenus. Les patrons sont instanciés par tirage aléatoire avec respect des contraintes pour générer un exercice.

-Type Factoriser 1 : Polynôme somme de termes constants et de monômes avec un facteur commun, autrement dit polynômes de forme générale AB + AC avec A, B et C constants ou monômes de degré 1 ou plus.
-Type Factoriser 2 : Polynôme somme de produits formés d'un facteur commun de degré 1 et de facteurs constants ou monômes, autrement dit polynômes de forme générale AB + AC avec A du premier degré, B et C de degré 1 au maximum
- Type Factoriser 3 : Polynôme somme de produits du premier degré, autrement dit polynômes de forme générale AB + AC avec A du premier degré, B et C constant ou de degré 1.
- **Type Factoriser 4** : Polynômes développés liés aux identités remarquables
- **Type Factoriser 5** : Polynôme complexe, somme de polynômes de types Factoriser 3 ou Factoriser 4. Le facteur commun n'est pas toujours apparent, une transformation intermédiaire et/ou une factorisation du type correspondant aux formes Factoriser 4 (liées aux identités remarquables) sont nécessaires. Le facteur commun est de la forme ax + b.

Tableau 4 : Patrons abstraits du type « Factoriser » extrait de [Bouhineau et al. 2005a]

4.5. Ce que j'en retiens pour ma thèse

Aplusix est un logiciel utilisé dans un contexte de recherche pour mener des études didactiques très poussées qui seraient impossibles sans cette instrumentation. C'est aussi un logiciel utilisé dans des classes de collèges et de lycée, par des enseignants et des élèves « ordinaires » indépendamment de toute activité de recherche. C'est un logiciel robuste fondé sur

des modèles mathématiques rigoureux. Par rapport aux systèmes que nous venons d'étudier, Aplusix apporte une interface élève très sophistiquée d'un point de vue technique qui permet à l'élève une grande liberté dans la saisie de raisonnements complexes qui sont analysés automatiquement en temps réel.

En ce qui concerne le diagnostic cognitif, Aplusix s'intéresse au même domaine que le projet Pépite mais il se focalise sur la composante Calcul Algébrique de la compétence quand Pépite s'intéresse au diagnostic sur plusieurs autres dimensions. Sur cette composante, son diagnostic local se situe à un niveau de granularité beaucoup plus fin que celui de Pépite, et, même le diagnostic global d'Aplusix (les théorèmes en actes) est d'un niveau de granularité inférieur à celui du diagnostic local de Pépite. Par exemple, sur les équations, les théorèmes en actes de mouvement d'Aplusix sont codés par Pépite EA1 (utilisation correcte des règles de transformation) lorsqu'ils sont correctes et, sinon, EA4 (identification incorrecte des opérateurs). En plus de ce code, l'analyse didactique a priori repère les règles de calcul correctes ou erronées appliquées. Ces dernières sont explicitées dans l'annexe C2. Le diagnostic de Pépite est moins détaillé sur le calcul algébrique mais plus complet sur les types d'usage de l'algèbre qui sont envisagés par l'élève et sur son habileté à articuler les expressions algébriques et d'autres types de représentations pour traduire des situations. L'objectif du diagnostic de Pépite n'est pas tant de détecter des fonctionnements particuliers dans le calcul algébrique que de mettre en évidence des cohérences dans ces fonctionnements pour situer l'élève dans l'entrée dans l'algèbre et dans la rupture avec l'arithmétique.

En ce qui concerne la génération des exercices, il est difficile d'établir une comparaison car les documents sur ce sujet sont limités. Cependant, l'approche que nous avons retenue consiste à combiner l'approche automatique par instanciation de patrons d'expressions sous contraintes utilisée par Aplusix, et une approche de raisonnement à partir de cas pour déterminer des classes d'exercices équivalents du point de vue du diagnostic (Chapitre 5). Cependant ce raisonnement à partir de cas est effectué par un chercheur qui a la charge de créer ces classes d'exercices et il n'est pas automatisé.

5. Des projets plus éloignés

Un projet assez proche de Pépite, mais dans un autre domaine, est le projet DIANE mené par Emmanuel Sander et Khider Hakem de l'équipe AIDA [Hakem et al. 2005]. Il vise à établir un diagnostic cognitif des compétences de résolution de problèmes additifs des élèves de l'école primaire. Il s'appuie sur une analyse cognitive fine établie à partir d'un corpus de plusieurs milliers de réponses d'élèves. Une interface a été développée pour permettre aux élèves de saisir

leurs calculs librement sous une forme assez proche du papier crayon. Le logiciel DIANE conduit automatiquement une analyse multidimensionnelle de la réponse de l'élève et un générateur d'exercices isomorphes a été développé. Les démarches de Diane et Pépite sont très voisines. L'analyse des réponses dans Pépite est plus complexe car les problèmes sont plus complexes.

Le projet PERLEA développé au sein du LIRI est dans la continuité du projet Pépite dans la mesure où il développe en la généralisant la phase de présentation des bilans ou profils obtenus à partir du diagnostic cognitif. Mais il adopte une approche générique en déconnectant la phase de diagnostic qui dépend des disciplines et des activités proposées, de la phase d'utilisation des profils. Le projet PERLEA travaille sur cette deuxième phase en proposant un cadre pour gérer les profils sans y associer de contenus.

Des travaux plus éloignés de nos préoccupations sont menés autour de l'utilisation de plateformes d'enseignement en ligne.

Wear [Virvou et Moundridou 2001] est un projet pour développer un outil auteur pour un environnement d'apprentissage en ligne de l'algèbre utile dans des domaines autres que les mathématiques (e. g. physique, économie). La proposition intéressante de ces auteurs consiste à intégrer un modèle de l'enseignant à l'environnement de travail. Nous n'avons pas trouvé de publications décrivant le système qui ne semble pas avoir dépasser le stage de la maquette preuve de concept. De même, le Math Intervention Module (MIM) [Zapata-Rivera et al. 2007] s'appuie sur un modèle de compétence et un modèle d'argumentation pour faire des rapports sur l'activité de l'élève à destination des parents, des enseignants ou des élèves eux-mêmes. Nous n'avons pas trouvé aussi de description de ce prototype dont les évaluations semblent se fonder sur des réponses préformatées ou simples, mais qui est intéressant pour les travaux de l'équipe portant sur la présentation du diagnostic aux différentes parties prenantes de l'évaluation.

Un autre travail développé dans l'équipe AIDA par Fabrice Vandebrouck et Claire Cazes, [Vandebrouck et Cazes, 2005] apporte aux chercheurs des informations concernant l'utilisation de la plateforme Wims pour des TD de Mathématiques par des étudiants. Les expressions mathématiques entrées par les étudiants sont vérifiées par un logiciel de calcul formel tel que MUPAD, PARL ou Maxima. Il permet d'afficher à l'enseignant les notes moyennes, le temps moyen passé par type d'exercice et des indicateurs sur l'évolution des notes dans le temps. Ce système n'est pas centré sur le diagnostic cognitif mais il détecte des stratégies d'usage suivies par certains étudiants telles que se concentrer sur des exercices faciles ou préférer les exercices avancés. Il permet aussi une classification automatique des exercices en fonction de deux indicateurs, à savoir la difficulté et le rendement.

ActiveMath est un projet européen coordonné par Erica Melis au German Research Institute for Artificial Intelligence dans le laboratoire Deutsche Forschungszentrum für Künstliche Intelligenz mit Sitz in Kaiserslautern (DFKI) [Melis et al. 2001]. Un objectif est de tirer bénéfice du meilleur des deux mondes des tuteurs intelligents et de la formation en ligne [Mellis et al. 2006]. C'est un environnement adaptatif d'apprentissage des mathématiques qui génère dynamiquement des cours de mathématiques et des exercices interactifs adaptés aux profils de l'élève. Actuellement ActiveMath fait l'objet d'évaluations à grande échelle dans les écoles en Allemagne et dans les Universités d'Espagne et d'Ecosse [Melis et al. 2007].

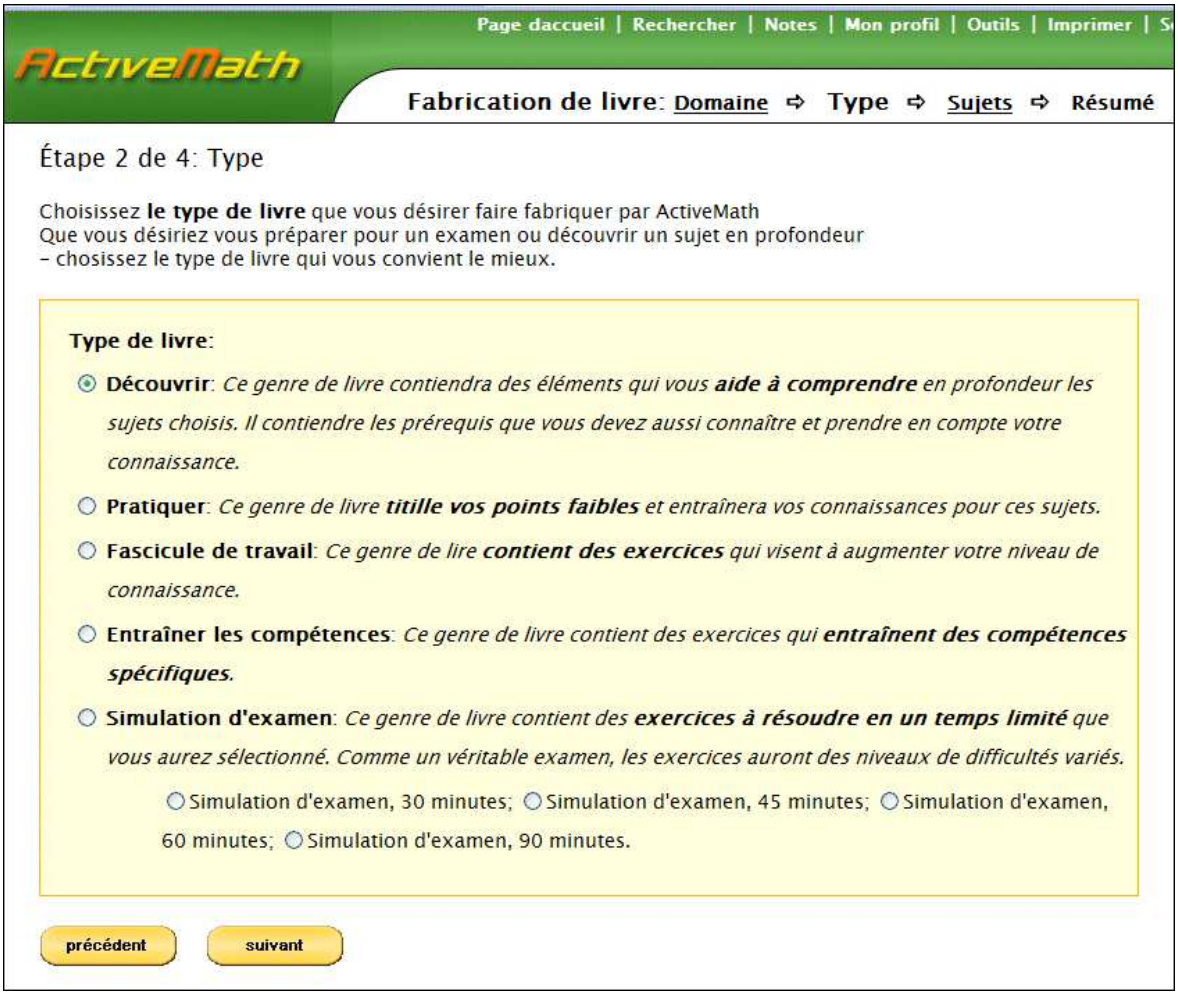


Figure 10 : Interface élève de ActiveMath

Cet environnement a été conçu pour donner à l'élève la possibilité d'explorer un large éventail d'exercices et de situations de résolution de problèmes, aussi ActiveMath offre différentes sortes d'exercices interactifs avec un ensemble de rétroactions pour guider, soutenir et stimuler l'utilisateur. En outre l'utilisateur dispose d'outils autonomes (e.g. calculatrice, traceurs, logiciels de géométrie) intégrés dans le logiciel. Les points forts d'ActiveMath sont d'utiliser les

technologies Web récentes, de s'appuyer sur des ontologies et des métadonnées pour sélectionner les documents pertinents à présenter aux élèves, d'avoir mis au point un langage OMDoc pour décrire les documents mathématiques et développé récemment un éditeur d'expressions algébriques performant.

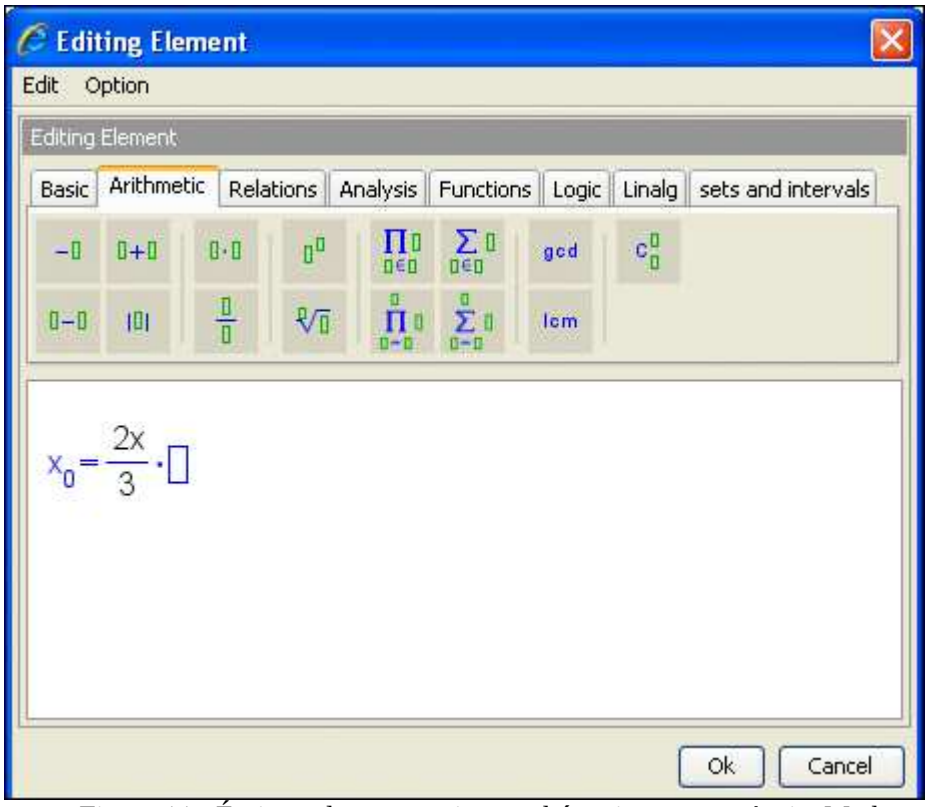


Figure 11 : Écriture des expression mathématiques avec ActiveMath (http://www.leactivemath.org/, décembre 2007)

6. Les langages de modélisation pédagogique

D'une part, le développement de l'usage des systèmes d'apprentissage et du e-leaning multiplie les besoins de ressources numériques et les solutions logicielles. D'autre part, les recherches ont montré que la mise à disposition de ressources sur une plate-forme de formation à distance ne suffisait pas pour assurer la réussite d'un dispositif d'enseignement [Koper 2001]. Enfin, en ce qui concerne l'évaluation des apprenants, les études montrent que les systèmes d'évaluation sur ordinateurs sont de plus en plus utilisés [Bennett 2002]. Pour répondre à ces besoins, des processus sont engagés pour décrire les ressources numériques et créer des langages de modélisation pédagogique. Après une brève présentation du langage de description pédagogique LOM (Learning Object Metadata), du langage de modélisation pédagogique EML (Education Modelling language) développé à l'OUNL (Open University of the NetherLands) et des spécifications du consortium IMS, IMS-LD (Learning Design Specification), élaborées à

partir de ce langage, nous décrivons la spécification IMS-QTI (Question and Test Interoperability). Nous énonçons les raisons qui incitent les chercheurs à élaborer des outils logiciels qui facilitent l'utilisation de ces spécifications ou à proposer d'autres langages comme c'est le cas de chercheurs de l'OUNL qui ont mis un point un langage de modélisation pour l'évaluation que nous présentons.

6.1. Le LOM : un standard pour la description des objets pédagogiques

Le terme d'objet pédagogique est utilisé dans la communauté francophone pour traduire l'appellation américaine de *Learning Object*. Les objets pédagogiques suscitent de nombreux travaux visant à la standardisation et à l'indexation. Il existe plusieurs métaphores [Bourda 2001] pour en donner une représentation dont celle du jeu de Lego, mais ces définitions sont dans l'ensemble imprécises.

Le groupe de travail LTSC (*Learning Technology Standards Committee*) de l'IEEE (*Institut of Electrical and Electronics Engineers*) donne en 2001 une définition d'un objet pédagogique. « Learning Objects are defined here as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. » [IEEE 2002]. La traduction proposée par l'AILF (Association des Informaticiens de Langue Française) est *« toute entité numérique ou non, qui peut être utilisée, réutilisée ou référencée lors d'une formation dispensée à partir d'un support technologique.* » Ainsi un objet pédagogique peut être un document imprimé, un cours, une image graphique, un applet Java. Dans la pratique un objet pédagogique peut être utilisé de façon indépendante d'un contexte précis et peut être réutilisé [Koper 2001].

La réutilisation des objets pédagogiques nécessite des standards de description. Le 12 juin 2002, le LTSC de l'IEEE a adopté le LOM, premier standard concernant la description des Objets Pédagogiques. Le LOM définit un schéma conceptuel de données ou plus simplement un schéma de métadonnées pour décrire les objets pédagogiques. Ce schéma est constitué d'un ensemble de données qui donnent les principales caractéristiques d'un objet pédagogique selon différents aspects. La version 6 du LOM définit une soixantaine de descripteurs regroupés en 9 catégories. La nomenclature des éléments du LOM version 1.0 peut être consultée dans [La Passardiere et Grandbastien 2003].

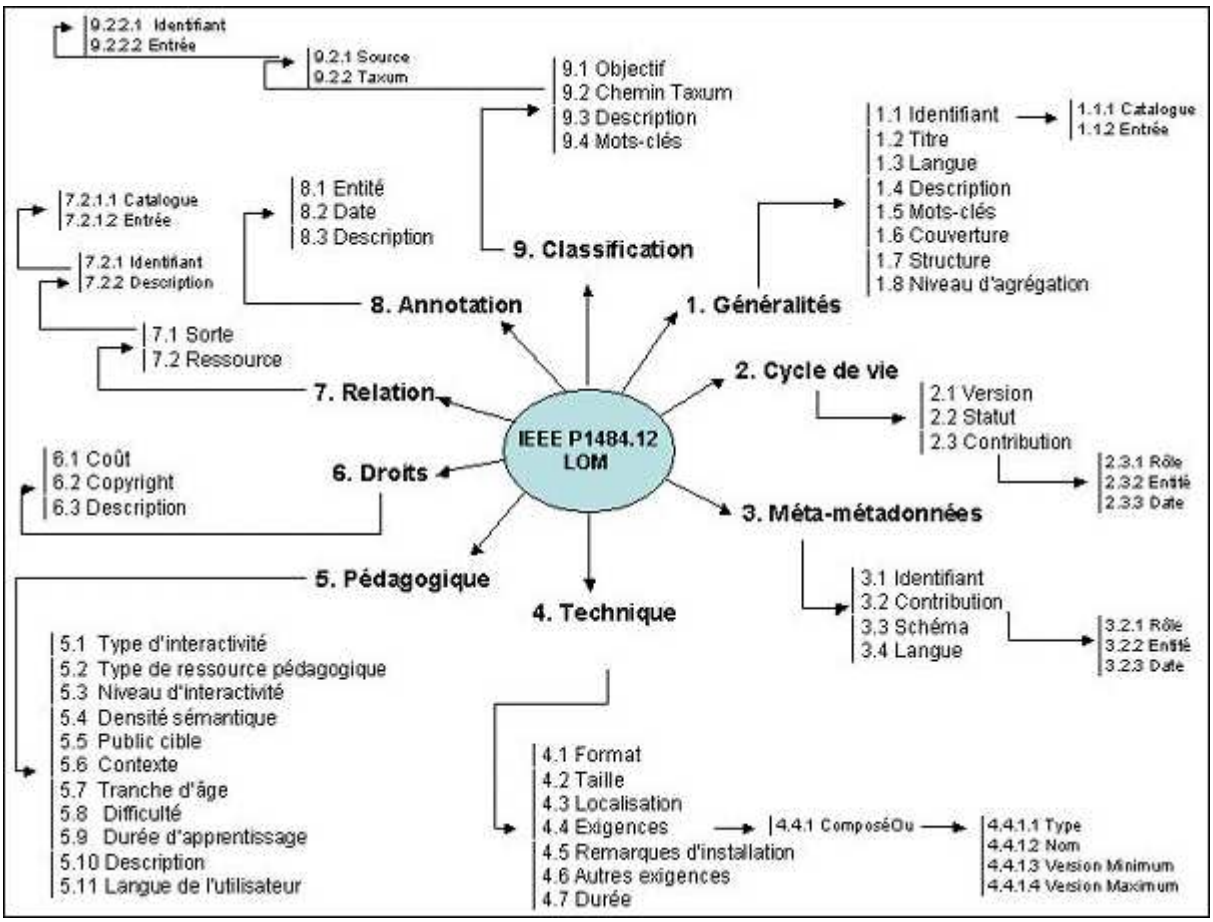


Figure 12 : Organisation du schéma des métadonnées [La Passardière et Jarraud 2004]

6.2. Langage de modélisation pédagogique EML

Les travaux de recherche ont montré que la mise à disposition de ressources pédagogiques sur une plate-forme de formation à distance ne suffisait par pour assurer la réussite d'un dispositif d'enseignement. Koper, chercheur à l'OUNL propose une approche qui place les activités d'apprentissage et non les ressources au centre du dispositif [Koper 2001]. S'appuyant sur un ensemble de propriétés qui doivent être vérifiées par le dispositif, il propose de décrire les situations d'apprentissage à l'aide d'un langage de modélisation pédagogique, EML. C'est une spécification qui décrit la structure et les processus mis en oeuvre dans une unité d'apprentissage. Selon Koper les critères de qualité d'un langage de modélisation pédagogique sont les suivants :

1. ***Formalisation*** : décrire des unités d'apprentissage de manière formelle pour pouvoir automatiser le processus,
2. ***Flexibilité pédagogique*** : décrire des unités d'apprentissage qui s'appuient sur des théories et des modèles d'apprentissage et d'enseignement différents,

3. ***Typage explicite des objets d'apprentissage*** : exprimer explicitement la signification sémantique des différents objets d'apprentissage et fournir la structure sémantique des contenus ou les fonctionnalités des objets d'apprentissage utilisés dans le contexte de l'unité d'apprentissage,
4. ***Complétude*** : décrire complètement l'unité d'apprentissage, y compris tous les objets d'apprentissage typés, les interactions entre les objets, les activités et le *workflow* entre les étudiants et les enseignants, indépendamment de la représentation numérique ou non de ces objets,
5. ***Reproductibilité*** : donner une description des unités d'apprentissage qui permette la répétition de son exécution,
6. ***Personnalisation*** : décrire les aspects de personnalisation d'une unité d'apprentissage afin que le contenu et les activités puissent être adaptés en s'appuyant sur les préférences, les connaissances antérieures, la situation personnelle des utilisateurs. En outre, le contrôle du processus doit pouvoir être donné selon le cas à l'apprenant, à un membre de l'équipe d'encadrement, à un ordinateur ou à un concepteur,
7. ***Indépendance du média avec le contexte d'utilisation*** : décrire le contenu des ressources indépendamment du média utilisé de telle sorte qu'il puisse être utilisé dans différents formats de publication ainsi que dans différentes situations d'enseignement,
8. ***Interopérabilité et durabilité*** : veiller à l'indépendance entre les standards utilisés pour la notation des unités d'apprentissage et les techniques utilisées pour interpréter ces notations par les systèmes de gestion d'apprentissage telles que les plates-formes de formation à distance,
9. ***Réutilisabilité*** : identifier, isoler, décontextualiser, échanger et réutiliser les objets d'apprentissage dans d'autres contextes,
10. ***Compatibilité*** : utiliser si possible les standards et les spécifications existants,
11. ***Cycle de vie*** : permettre la production, la modification, la conservation, la distribution et l'archivage de tous les objets d'apprentissage et des unités d'apprentissage.

Parallèlement, un groupe d'étude se met en place en 2001 au sein du WS/LT (*WorkShop of Learning Technologies*) du CEN/ISSS avec pour objectif de faire l'inventaire des différents modèles d'EML existants, de les comparer et de créer un premier modèle commun [Rawlings et al. 2002]. Ce modèle s'appuie sur une définition où le concept d'unité d'apprentissage est central : une unité d'apprentissage décrit la démarche d'apprentissage, les ressources et les services nécessaires pour atteindre des objectifs d'apprentissage. Ainsi, elle ne peut pas être décomposée sans perdre du sens et l'efficacité recherchée pour atteindre ces objectifs. Cette étude conclut que le langage EML correspond le mieux à cette définition. Ce langage est intégré dans

les travaux de standardisation engagés par le consortium IMS qui ont abouti à la spécification IMS-LD.

6.3. IMS-LD

Le méta-modèle d'information d'IMS-LD se décompose en trois composants principaux :

- Le modèle conceptuel représente le vocabulaire, les relations entre les concepts et les relations avec la spécification IMS Content Packaging utilisée pour la diffusion des contenus et des activités,
- Le modèle d'information décrit les éléments de IMS-LD sur trois niveaux :
    - le niveau A décrit le vocabulaire nécessaire pour concevoir des unités d'apprentissage supportant diverses pédagogies,
    - le niveau B ajoute au niveau A des propriétés qui permettent une personnalisation,
    - le niveau C ajoute un mécanisme de notifications entre les acteurs impliqués dans l'unité d'apprentissage pour supporter la collaboration et le tutorat.
- Le modèle de comportement décrit un ensemble de comportements d'exécution que les systèmes délivrant les unités d'apprentissages devront implémenter.

Une personne (*person*) joue des rôles (*role*) dans un processus d'apprentissage/enseignement (e.g. le rôle d'apprenant, *learner*, de personnel, *staff*). Dans ce rôle, elle réalise des activités d'apprentissage (*learning activity*) ou de soutien (*support activity*) pour obtenir des résultats (*outcome*). Les activités sont réalisées dans un environnement (*environment*) constitué d'objets d'apprentissage *(learning object*) et de services utilisés pour la réalisation des activités. La méthode (*method*) a des objectifs d'apprentissage *(learning objectives*) qui s'appuient sur des prérequis pour les apprenants (*prerequisite*s). Ce modèle repose sur la métaphore de la pièce de théâtre (*play*). Plusieurs pièces peuvent se jouer en parallèle et une pièce est constituée d'un ou plusieurs actes (*act*) qui font référence à un ou plusieurs rôles (*role-part*). Ainsi, le modèle associe un rôle à une activité.

Au niveau B, la méthode peut avoir des conditions dont les propriétés (*property*) peuvent être regroupées (*global elements*) pour définir des propriétés spécifiques à un groupe. Ce niveau apporte un mécanisme pour conserver des informations sur les apprenants, sur les groupes d'apprenants et individualiser les parcours.

Au niveau C, un notification est générée par un résultat et peut créer un nouveau rôle associé à une activité. La Figure 13 présente le modèle conceptuel de la structure d'un *Learning design* décrit précédemment.

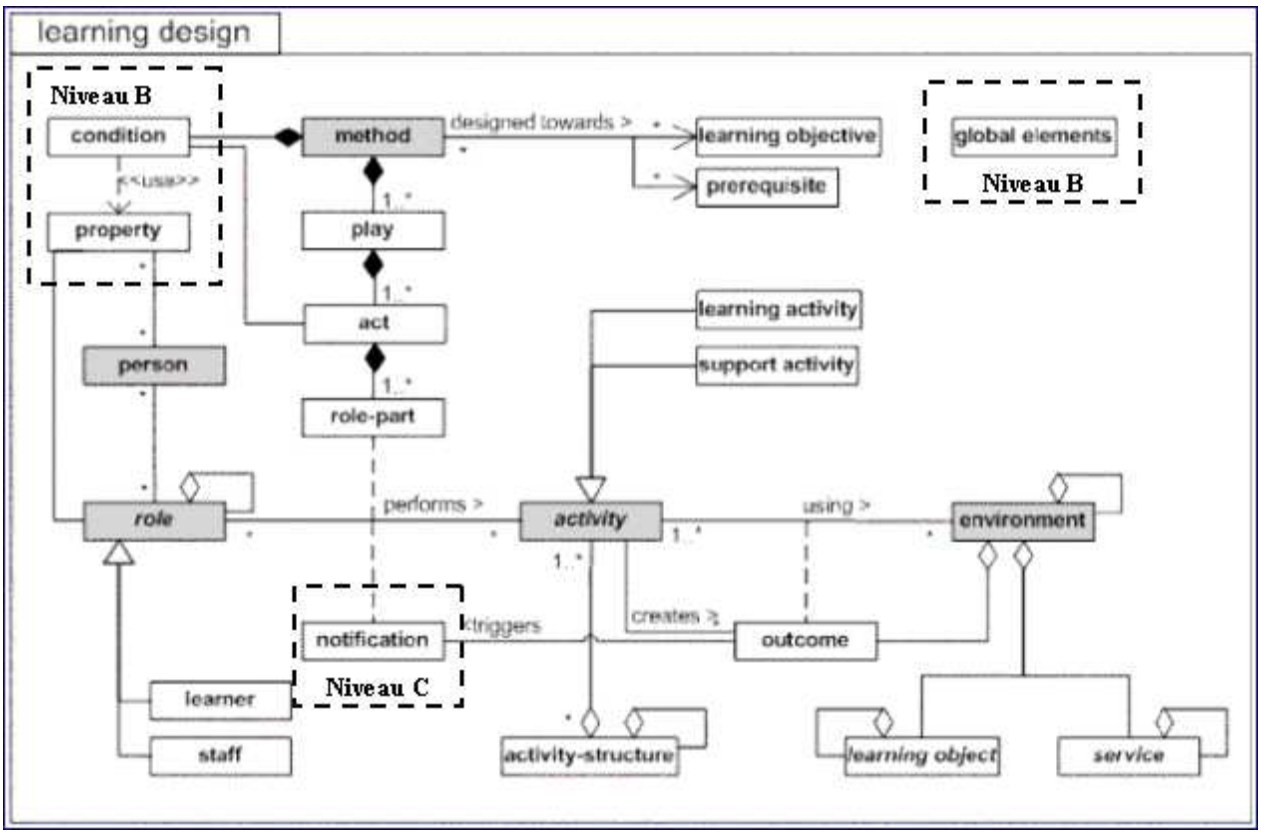


Figure 13 : Modèle conceptuel de l'ensemble IMS-LD, [IMS-LD 2003].

La spécification IMS-LD a pour objectif de décrire les situations d'apprentissage ou d'enseignements, mais n'intègre pas les tests et les évaluations. Nous étudions maintenant la spécification IMS-QTI qui permet de décrire les exercices et les tests.

6.4. Question and Test Interopérability (QTI)

Les spécifications QTI fournissent un modèle d'informations pour l'évaluation par questionnaires de réponses préformatées ou simples (numérique ou expression simple). Elles ont été élaborées pour faciliter l'interopérabilité entre différents systèmes (e.g. outils auteurs, éditeurs de plates-formes d'enseignement en ligne, banques d'exercices). L'objectif est de :

- proposer pour des questions et pour des tests un contenu dont le format de stockage et d'échange est indépendant de l'outil auteur utilisé pour les créer,
- faciliter l'utilisation de banques d'exercices et de tests dans différents systèmes d'apprentissage et d'évaluation, et plus particulièrement, sur différentes plates-formes de formation à distance.

6.4.1. Différents cas d'utilisation

La résolution d'un exercice ou le passage d'un test se fait par le biais des interactions avec un système d'enseignement en ligne. Les spécifications QTI sont exprimées sous la forme d'un métamodèle UML. Le diagramme des composants présenté dans les spécifications [IMS-QTI 2006] décrit des cas d'utilisation de QTI mettant en jeu cinq modules et sept types d'acteurs potentiels qui utilisent les questions (**assessmentItem**) ou les tests d'évaluation (**assesmentTest**). Ces modules sont : (i) un système auteur (**authoringTool**) utilisé pour créer ou modifier une exercice, (ii) un système de gestion d'exercices qui constituent une banque d'exercices (**itemBank**), (iii) un outil de construction de tests (**testConstructionTool**), (iv) un système de diffusion des tests auprès des candidats à évaluer (**assessmentDeliverySystem**), et (v) un système d'apprentissage (**learningSystem**). Le système d'apprentissage gère les activités d'apprentissage avec ou sans tuteur.

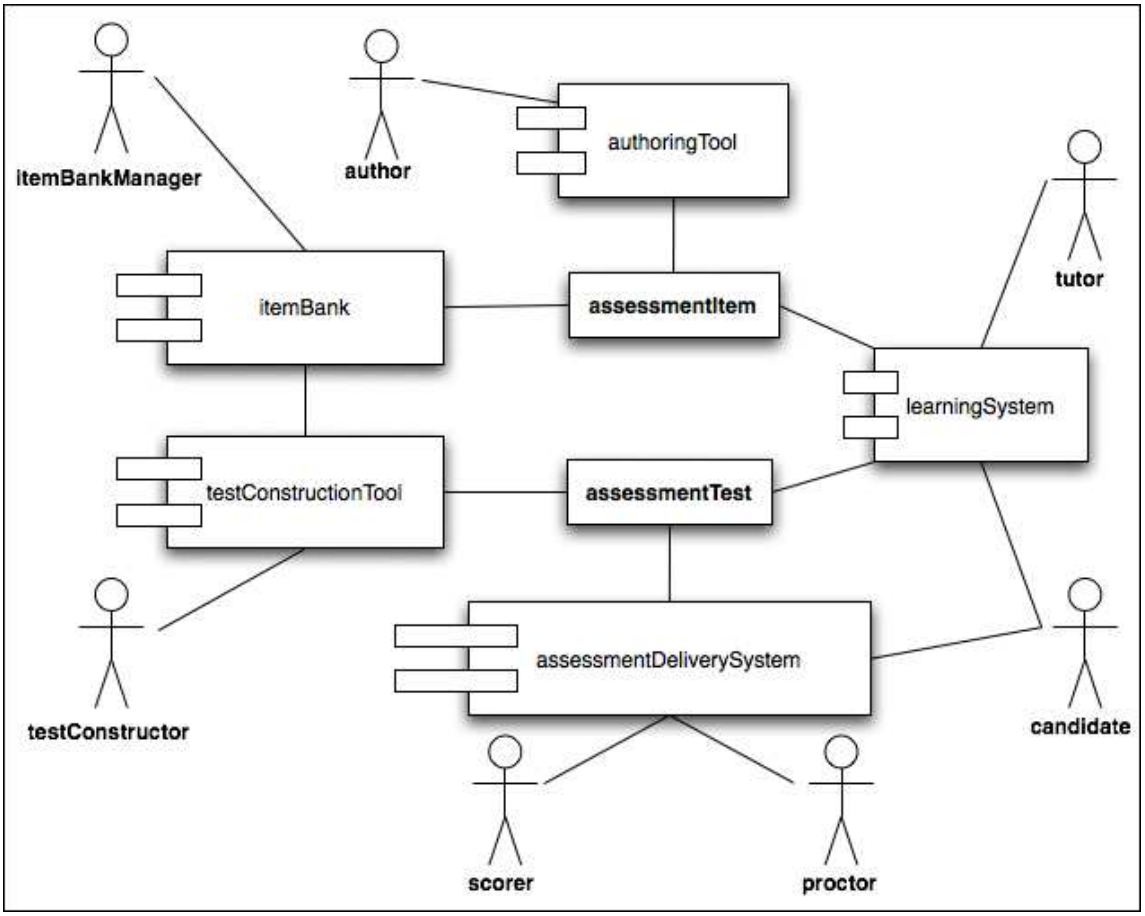


Figure 14 : Acteurs et modules [IMS-QTI 2006]

Les spécifications décrivent les principaux rôles des acteurs humains ou logiciels qui interviennent dans ces différents modules. L'auteur (**autho**r) crée les contenus d'un exercice avec

un système auteur et se distingue du gestionnaire de la base d'exercices (**ItemBankManager**). Le concepteur d'un test (**TestConstructor**) crée un test, généralement à partir de la base d'exercices, selon différents critères. Le surveillant (**Proctor**) est chargé de la diffusion et de la mise en œuvre du test mais il ne joue pas de rôle dans l'évaluation du candidat. Un évaluateur (**Scorer**) est chargé d'évaluer les réponses des candidats ou des apprenants (**candidate**). L'évaluateur peut être un humain ou un logiciel car certaines questions sont évaluées automatiquement en utilisant des règles définies par l'auteur de la question. Le tuteur (**tutor**) est impliqué dans la gestion des apprentissages et ne joue pas nécessairement un rôle dans l'évaluation.

6.4.2. Modèle d'informations de QTI

Les deux principales structures de la spécification QTI sont les questions représentées par la classe UML **assesmentItem**, et les tests représentés par la classe **assessmentTest**. Les spécifications QTI permettent aussi de générer des questions semblables, c'est-à-dire des clones à partir de « variables de modèle » déclarées dans la question (**templateDeclaration**).

6.4.2.1. Les questions

La classe **assessmentItem** contient un ensemble de classes qui décrivent le processus de notation de la question (**responseProcessing**), l'interface de l'apprenant (**stylesheet** et **itemBody**), les rétroactions à l'apprenant (**modalFeedback**) et les déclarations de trois types de variables (**responseDeclaration**, **outcomeDeclaration**, **templateDeclaration**). Une « variable de réponse » (**responseDeclaration**) est associée à un type d'interaction (e.g. choix simple, choix multiples, saisie d'un texte), et selon la cardinalité, précise les valeurs d'une ou plusieurs réponses correctes. Une « variable de sortie » (**outcomeDeclaration**) est associée à un résultat (*score*) et à une rétroaction (*feedback*) : la valeur du résultat est donnée dans la déclaration ou instanciée au cours du processus de notation (**responseProcessing**). Une variable de modèle (**templateDeclaration**) est utilisée pour le clonage, et sa valeur est instanciée au moment du processus de clonage (**templateProcessing**). La Figure 15 représente les classes qui composent la classes **assessmentItem**.

La présentation de l'exercice est décrite par la classe **stylesheet** qui renvoie à une feuille de style et par la classe **itemBody** qui caractérise le type d'interactions (e.g. choix simple, choix multiple, associations) et tous les éléments qui contribuent à la présentation de l'exercice à l'apprenant (e.g. texte des questions, médias).

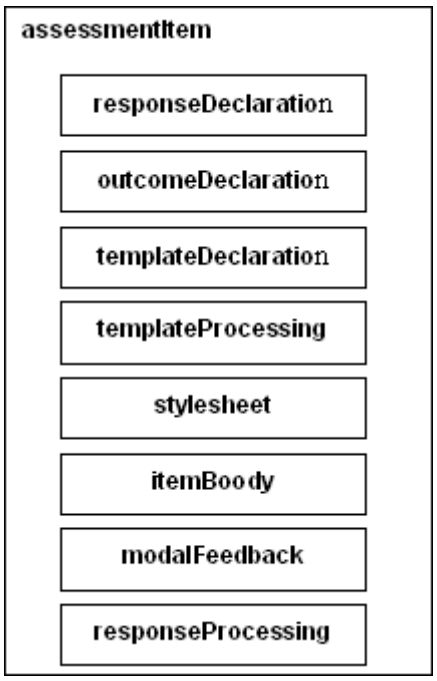


Figure 15 : Composition de la classe assessmentItem

La Figure 16 présente la première question de l'exercice de PépiTest «Reconnaître des égalités vraies ». Nous avons décrit cette question en utilisation la modélisation QTI afin de générer des clones de cette question [Leblanc 2007]. Le Tableau 5 donne un extrait du fichier QTI ainsi obtenu.

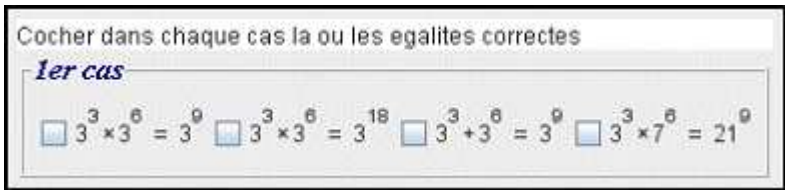


Figure 16 : Extrait d'un exercice de Pépitest

Cet extrait donne la déclaration des variables de réponses, des variables de sortie et des variables de modèle, le processus de clonage et une partie de la présentation (**itemBody**). Nous n'avons pas associé de feuille de style (**stylesheet**) car, à l'heure actuelle PépiTest n'est pas une application Web. La question comporte quatre choix, la réponse correcte correspondant au dernier choix. Les variables de modèles sont a, n et p : la variable a correspond à la base de la puissance (instanciée par la valeur 5 dans l'exemple) et les variables n et p aux exposants (instanciées par les valeurs 2 et 3). Ainsi l'expression du premier choix est $a^n * a^p$. Ces expressions sont décrites avec le langage MathML (classe **itemBody)**. L'attribut *mathVariable* (classe **templateDeclaration**) a la valeur *true* pour indiquer que la variable de modèle *a* qui se trouve dans l'expression $\boldsymbol{a^n * a^p}$ exprimée en MathML doit être remplacée par la valeur numérique

obtenue dans le processus de gestion du clone. La valeur *true* de l'attribut *shuffle* (classe **choiceInteraction**) indique que l'ordre des choix est obtenu par tirage aléatoire. Ainsi nous pouvons utiliser les spécifications QTI pour générer des clones de cette classe d'exercices. Cependant, actuellement les spécifications ne permettent pas d'assurer une analyse multidimensionnelle des réponses.

```
<responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
    <correctResponse>
        <value>Choix4</value>
    </correctResponse>
</responseDeclaration>
<outcomeDeclaration identifier="SCORE" cardinality="single" baseType="integer">
    <defaultValue>
        <value>0</value>
    </defaultValue>
</outcomeDeclaration>
<templateDeclaration identifier="a" cardinality="single" baseType="integer"
mathVariable="true" paramVariable="false" />
<templateDeclaration identifier="n" cardinality="single" baseType="integer"
mathVariable="true" paramVariable="false" />
<templateDeclaration identifier="p" cardinality="single" baseType="integer"
mathVariable="true" paramVariable="false" />
<templateProcessing>
    <setTemplateValue identifier="a">
            <randomInteger min="2" max="10" />
     </setTemplateValue>
    …/….
</templateProcessing>
<itemBody>
    <choiceInteraction responseIdentifier="RESPONSE" shuffle="true" maxChoices="1">
    <simpleChoice identifier="Choix1">
    écriture de a^n*a^p exprimée en MathML.
     </simpleChoice>
    ../….
</itemBody>
```

Tableau 5 : Extrait du fichier QTI de la question présentée à la Figure 16

6.4.2.2. Les tests

La classe **assessmentTest** contient un ensemble de classes qui décrivent les résultats associés au test (**outcomeDeclaration**), le processus de gestion des résultats (**outcomeProcessing**), éventuellement la durée du test (**timeLimits**), le retour à l'apprenant (**testFeedback**) et la structure du test (**testPart**). La classe **testPart** donne la composition du test en section et sous-sections (Figure 17), la façon dont les questions sont présentées à l'apprenant et sont évaluées.

L'apprenant peut répondre aux questions dans l'ordre sans possibilité de revenir en arrière ou choisir les questions. Les questions sont évaluées l'une après l'autre ou seulement à la fin du test. Un ensemble de conditions et de règles (**precondition**, **branchRule**) définissent ces différentes possibilités. Le processus de gestion des résultats est régi par un ensemble de règles (e.g. **outcomeIf**, **outcomeElseIf**).

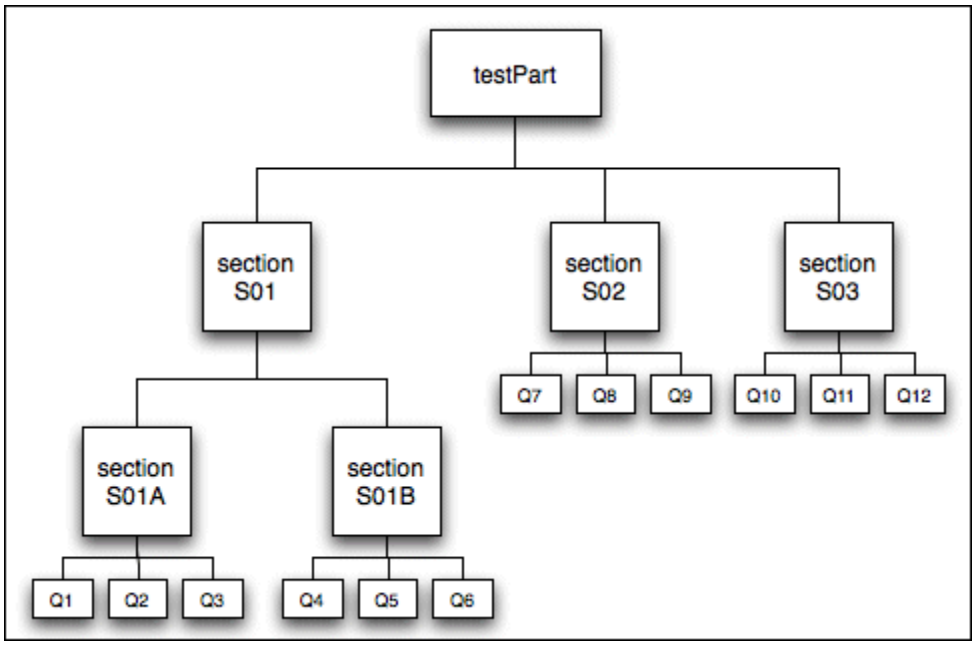


Figure 17 : Structure d'un test [IMS 2006]

6.4.3. Ce que je retiens pour ma thèse

Les spécifications de QTI offrent de nombreuses possibilités pour les questions fermées à choix prédéfinis. Les interactions auxquelles sont associées une ou plusieurs réponses peuvent se faire par le biais de tous les objets graphiques habituels (e.g. case à cocher, menu déroulant) et offrent la possibilité de diversifier les types de questions (e.g. exercices d'appariements, textes à trou, zones cliquables sur une figure). La génération automatique de clones de questions favorise la création de tests différents. Les possibilités offertes par les spécifications QTI pour construire des tests et proposer des rétroactions en fonction des résultats de l'apprenant permettent de différencier les parcours des apprenants. Mais, selon [Michel et Rouissi 2003], la richesse et la complexité des spécifications de QTI imposent une aide logicielle capable de générer les balises suffisantes et d'aider l'auteur. [Blat et al. 2007] soulignent l'importance de disposer d'outils pour utiliser les spécifications IMS-QTI.

Selon les auteurs des spécifications [IMS-QTI 2006], le processus de gestion des réponses offre seulement la possibilité d'évaluer des réponses simples. Par exemple, il n'est pas possible d'évaluer une réponse textuelle. Pour évaluer certains types de réponses, la classe qui décrit le

processus de gestion des réponses offre la possibilité d'utiliser des programmes externes en donnant une adresse URL par le biais d'un attribut. En ce qui concerne plus particulièrement l'utilisation des variables, QTI permet d'exprimer une variable en fonction d'autres variables mais ne permet pas de définir des contraintes entre variables. Reprenons l'exemple proposé par [Auzende et al. 2007].

> « *a et b sont des entiers tels que a est compris entre 2 et 10, b est compris entre 2 et 40, b est impair et a+b est multiple de 10.* »

Cet exemple caractérisé par l'interdépendance entre les deux variables a et b ne peut pas être décrit avec QTI. Pour y remédier, [Auzende et al. 2007] ont proposé une extension des spécifications. Dans la présentation de leur modèle d'évaluation [Joosten-ten Brinke et al. 2007] précisent que la spécification QTI offre des possibilités pour échanger des tests comportant des questions fermées à choix prédéfinis. Cependant, ils constatent que la mise en œuvre de la spécification QTI s'avère difficile. De plus, dans un récent test des applications logicielles (players) qui utilisent QTI, Gorissen [Gorissen 2006] a constaté qu'aucune ne supporte tous les formats définis dans QTI. Ainsi, l'interopérabilité est limitée aux questions à choix multiples et à leurs variantes.

En ce qui concerne notre travail de thèse, les spécifications QTI permettent de générer des clones d'exercices comportant des questions à choix prédéfinis (Figure 18). Cependant nous rencontrons plusieurs difficultés : le processus de gestion des réponses calcule un résultat mais n'offre pas la possibilité de faire une évaluation multicritère. Pour les exercices dont les questions nécessitent la génération d'expressions algébriques complexes, le processus de clonage de QTI ne permet pas la production de ces expressions qui nécessite un calcul formel. QTI ne permet pas non plus la génération automatique des réponses anticipées sous forme de raisonnements algébriques complexes, ni leur analyse. Il est également impossible de définir avec QTI des exercices où les énoncés ont des figures variables. De notre étude, il ressort que, concernant la génération de clones des exercices de PépiTest, seuls 2 exercices pourraient être décrits à l'aide de la spécification QTI. Enfin, comme le souligne de nombreux auteurs, cette spécification met en œuvre un modèle extrêmement limité d'évaluation ; elle ne prend pas en charge l'évaluation multicritère qui est essentielle pour un diagnostic cognitif.

Après cette étude en tenant compte d'une part du manque d'interopérabilité dans la pratique de la spécification QTI et, d'autre part, de son inadéquation aux besoins du diagnostic cognitif, nous avons donc décidé de développer un modèle adapté à nos besoins qui soit à la fois assez

proche de l'analyse didactique fondatrice du projet pour permettre le diagnostic cognitif dans notre cas spécifique, mais assez général pour être utilisable pour d'autres types de diagnostic.

6.5. Modèle conceptuel d'évaluation à l'Open University of NederLand (OUNL)

Ayant fait le constat que la mise en œuvre des spécifications QTI s'avérait difficile et qu'elle se limitait aux question à choix multiples et à leurs variantes [Joosten-ten Brinke et al. 2007], les chercheurs de l'OUNL proposent un modèle conceptuel pour l'évaluation. Il se situe dans une nouvelle approche où les évaluations sont au centre du processus éducatif et ont un impact direct sur les processus d'apprentissage des élèves. D'une part, des évaluations sommatives aident les enseignants à savoir si les élèves ont atteint les objectifs qu'ils leur ont fixé, d'autre part des évaluations formatives aident les élèves à atteindre leurs objectifs, par le biais des rétroactions. Ce modèle conceptuel supporte ces nouveaux types d'évaluation ainsi que les évaluations de type traditionnel. Pour combiner ces nouvelles exigences avec des ressources limitées en temps et en personnes, il est nécessaire de concevoir des évaluations qui puissent être partagées et réutilisées dans d'autres contextes. Ainsi le modèle proposé reprend les critères de qualité énoncés dans la section 6.2. [Koper 2001]. Actuellement une première version du modèle répond aux critères de souplesse, réutilisabilité, interopérabilité et durabilité, complétude et typage des objets. La description du modèle sous la forme de diagrammes de classes UML est une première étape pour une formalisation.

6.5.1. Description du modèle

Le modèle est décomposé en plusieurs paquetage UML (Figure 19).

6.5.1.1. Assessment design

La première étape de la conception du modèle d'évaluation est de situer le processus d'évaluation dans le contexte institutionnel de l'université. L'objectif du module est donc de décrire un ou plusieurs plans d'évaluation (**Assessmentplan**) qui s'appuient sur un ensemble d'idées directrices (**Assessmentpolicy**), de règles de décisions (**Decisionrule**), d'objectifs (e.g. positionnement, diagnostic, examen) qui sont décrits par la classe (**Assessmentfunction**). Un plan d'évaluation est composé de scénarios d'évaluation (**AssessemntScenario**) en relation avec des sessions (**AssessemntSession**). Ce module comporte les classes qui décrivent les éléments à évaluer (**Trait, ComplexTrait** et **ElementaryTrait**). Le trait (**Trait**) décrit ce qui est évalué : une compétence, une capacité, une aptitude, une construction mentale. Un trait complexe est composé de traits élémentaires qui sont indivisibles. Avec ce modèle, l'élément évalué est le trait : c'est

une autre façon d'évaluer un apprenant et les spécifications QTI, qui permettent d'évaluer des réponses, ne prennent pas en compte ce type d'évaluation.

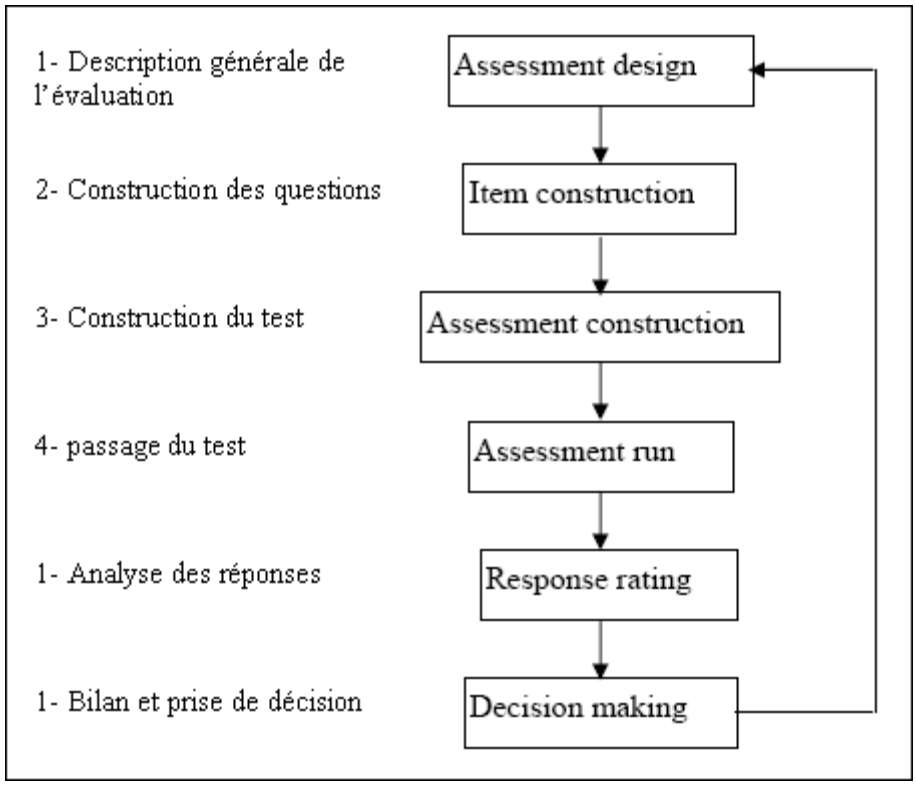


Figure 19 : Les étapes du processus d'évaluation [Joosten-ten Brinke et al. 2007]

6.5.1.2. Itemconstruction

La description de la classe **Item** s'appuie sur la notion de **Trait** définie dans l'étape précédente. Cette classe est le plus petit élément qui permet d'évaluer (**indicator**) un trait d'un candidat. Elle est la généralisation de trois sous-classes (**ConstructionItem, SelectionItem, DemonstrationItem**) qui décrivent plus finement le type de question (questions ouverte, question fermée, démonstration). A un item sont associées des rétroactions (**Hint, Prompt, Feedback**) et un item . La classe (**ratingInstruction**) précise pour chaque item les caractéristiques d'un bonne réponse pour attribuer une note ou une appréciation.

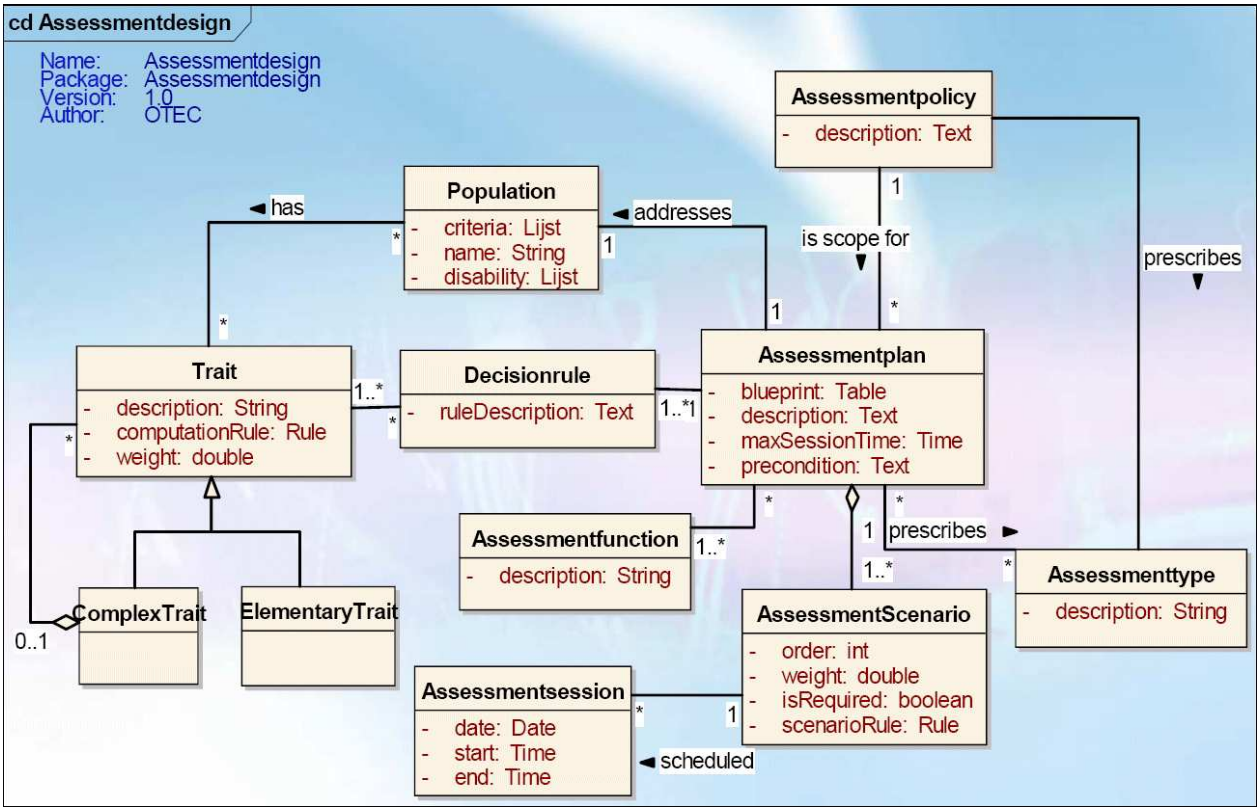


Figure 20 : Schéma conceptuel de l'ensemble des classes de Assessmentdesign

6.5.1.3. Assessmentconstruction

Ce paquetage décrit la constitution d'une unité d'évaluation (**UnitOfAssessment**) qui est basée sur une classe (**UnitOfAssessmentDefinition**) qui décrit la session d'évaluation, les candidats qui sont évalués, la structure de l'évaluation (e.g. le nombre de questions prévues, l'ordre des questions), la façon d'évaluer, le barème, la durée.

6.5.1.4. Reponseprocessing

Le paquetage **Reponseprocessing** contient toutes les classes qui évaluent les réponses et les notent. Les items sont évalués par un évaluateur humain (enseignants, les pairs, le candidat) ou un logiciel.

6.5.1.5. Assessmentrun, Decisionmaking

Dés qu'une évaluation a été composée, elle peut être proposée aux candidats. Le paquetage **Assessmentrun** décrit le passage du test : les horaires, la durée, à quel moment les réponses sont évaluées (e.g. après chaque item, en fin de test). Le paquetage **Decisionmaking** est basé sur les résultats du candidat et décrit les décisions qui sont prises, ces décisions étant précisées dans le plan d'évaluation (**AssessmentPlan**) et fondées sur des règles de décision (**Decisionrule**).

6.5.2. Ce que je retiens pour ma thèse

Ce modèle permet d'évaluer des réponses correctes ou incorrectes mais aussi des éléments tels que les compétences, les capacités, différentes traits de caractères, ce que ne permettent pas les spécifications QTI. Le modèle propose des items avec des réponses ouvertes (**ConstructionItem**) et des items où l'activité du candidat dans la réalisation d'une tâche, d'une procédure, d'une activité est observée (**DemonstrationItem**). Il prend en compte le contexte dans la construction de l'évaluation et dans son organisation. Cependant , actuellement, c'est un modèle descriptif et non exécutable. Tout ce qui concerne l'interface élève et les interactions n'apparaissent pas dans le modèle.

Une idée à retenir dans ce modèle, c'est de l'avoir construit avec un nombre limité de classes ce qui donne une description facile à utiliser, et, d'avoir pris en compte lorsque cela était possible les spécifications QTI.

6.6. Langages spécifiques aux mathématiques

La génération d'exercices de mathématiques pose des problèmes spécifiques liés à l'expression et l'affichage des expressions mathématiques plus particulièrement pour les expressions comportant des fractions et des racines carrées qui sont utilisés dans tous les niveaux de classe. Pour y remédier certains standards ont été définit.

6.6.1. MathML

MathMl est une spécification qui défini le Mathematical Markup Language qui décrit la structure et le contenu des notations habituellement utilisées en mathématiques. L'objectif est de pouvoir afficher des formules mathématiques sur le Web comme cela est possible pour du texte avec le langage HTML. La spécification du langage de balisage MathML contient une trentaine de balises qui décrivent les notations des structures abstraites et environ cent cinquante balises qui précisent sans ambiguïté le sens d'une expression. Bien que les balises MathML soient compréhensibles, il est prévu que, dans tous les cas les auteurs utiliseront des éditeurs d'équation, des programmes de conversion et d'autres outils logiciels spécialisés pour générer MathML.

6.6.2. OpenMath

OpenMath vise à intégrer des expressions mathématiques dans des pages sur le Web. Il est aussi conçu pour être un format d'échange entre différents logiciels ou systèmes. Le standard OpenMath a été approuvé par le groupe OpenMath Society. Les objets OpenMaths représentent des entités mathématiques qui peuvent être échangées entre différents systèmes. Ainsi ces objets sont codés en XML ou en code binaire. Des dictionnaires sont utilisés pour définir les symboles

utilisés pour représenter les concepts mathématiques et les connaissances mises en commun par OpenMath.

7. Conclusion

Cet état de l'art et l'historique du projet Lingot du chapitre précédent, montrent que peu de systèmes diagnostiquent des raisonnements complexes exprimés librement. Les systèmes qui le font, mettent en œuvre un résolveur de problèmes qui permet d'évaluer la validité de chaque étape du raisonnement et de détecter des erreurs préalablement répertoriées par une analyse humaine (didactique ou psychologique) ou automatique. Ces résolveurs de problèmes/ diagnostiqueurs sont en fait des logiciels de calcul formel spécialement conçus pour répondre aux besoins de l'EIAH considéré. Ils sont fondés sur la construction d'un arbre des solutions possibles : Andes et PAT ne considèrent que les solutions correctes tandis que Aplusix et PepiGen envisagent les solutions correctes et incorrectes. Chacun de ces systèmes utilisent des heuristiques pour limiter l'explosion combinatoire dans l'espace de recherche.

Concernant la création d'exercices, seul Aplusix et Diane mettent en œuvre une création automatique de listes d'exercices indexées par des variables didactiques/cognitives. Andes et Algebra Tutor mobilisent des ingénieurs cogniticiens pour créer des exercices. Nous adoptons une solution mixte qui consiste à générer automatiquement les exercices contraints et à proposer des patrons (pas dans le sens des patrons d'Aplusix dans le sens de gabarit ou de formulaire à remplir) à compléter quand les énoncés laissent une marge de liberté importante. Chaque projet de recherche utilise ces propres critères pour décrire les exercices posés à l'élève. Nous avons défini de façon précise le modèle de description des exercices de façon à ce que ce modèle puisse être réutilisé.

Enfin une dernière remarque concerne la méthodologie de développement de tous ces projets de recherche. On constate qu'un temps de maturation très important a été nécessaire avant d'obtenir des systèmes opérationnels. Des prototypes plus ou moins aboutis ont été construits pour tester des questions de recherche et des solutions. Ils ont recueillit des corpus importants de travaux d'élèves qui ont permis d'améliorer considérablement la compréhension des problèmes. Les systèmes qui sont maintenant opérationnels ont tous été construits à partir d'études didactiques et/ou cognitiques rigoureuses, à partir d'une large collaboration entre chercheurs de plusieurs disciplines et avec des enseignants pour se confronter aux problèmes de l'intégration des ces outils dans les pratiques. Ces approches itératives et participatives sont au cœur des démarches actuelles en EIAH.

Chapitre 4

Contexte, objectifs, problématique et méthodologie

1. Introduction 95

2. Les objets métiers, définitions 95
2.1. Test (ou test diagnostic) 96
2.2. Diagnostic 96
2.3. Tâches diagnostiques (ou exercices diagnostiques) 96
2.4. Stratégies de diagnostic 97
2.5. Types de tests 97

3. Scénarios de conception 98
3.1. Les utilisateurs de SuperPépite 98
3.2. Scénario 1 : un enseignant découvre SuperPépite et s'approprie un test type 99
3.3. Scénario 2 : un enseignant, habitué, compose un test diagnostic 99
3.4. Scénario 3 : un enseignant veut étudier l'évolution des compétences de ses élèves depuis le dernier test 100
3.5. Scénario 4 : une didacticienne des mathématiques génère des tests diagnostics 100
3.6. Scénario 5 : des élèves passent un test diagnostic 100
3.7. Scénario 6 : un élève dialogue avec un enseignant pour réfléchir sur son bilan de compétence 101
3.8. Scénario 7 : un enseignant constitue des groupes de travail dans sa classe 101
3.9. Scénario 8 : un auteur enrichit la banque d'exercices 102
3.10. Scénario 9 : un(e) développeur(e) informatique ajoute un modèle d'exercices 102
3.11. Conclusion 103

4. Architecte fonctionnelle de SuperPepite 103

5. Problématique 105

6. Méthodologie 107

1. Introduction

Notre travail de thèse se centre sur la conception et la réalisation du logiciel PépiGen qui permet à un auteur d'alimenter une banque d'exercices de diagnostic. Un travail avec les utilisateurs potentiels et une étude de l'état de l'art [Murray 2003a] nous ont amené à automatiser le plus possible la génération d'exercices et l'analyse des réponses en laissant cependant une marge de manœuvre à l'auteur. Pour cela nous avons identifié des classes paramétrées d'exercices ; un exercice d'une classe est généré par la détermination automatique ou assistée des valeurs des paramètres et le diagnostic est généré automatiquement.

Dans les premières sections de ce chapitre, afin de montrer comment notre travail s'insère dans l'ensemble du projet, nous commençons par spécifier le logiciel dans lequel s'insèrera notre travail. En effet l'objectif à moyen terme du projet Pépite est la conception d'un système, que nous avons appelé SuperPépite, qui permette à des enseignants de concevoir des tests diagnostics pour une évaluation des compétences de leurs élèves en algèbre élémentaire à différents moments de la scolarité et dans différents contextes. Ce projet comporte plusieurs volets, traités par différents membres de l'équipe. Nous définissons les principaux objets métiers de notre application et caractérisons les différentes catégories d'utilisateurs de SuperPépite. Nous décrivons ensuite des scénarios de conception que nous avons mis au point avec des utilisateurs potentiels. Nous terminons cette partie en situant les modules développés dans le cadre de cette thèse dans l'architecture fonctionnelle d'ensemble de SuperPépite.

Ensuite, nous précisons notre problématique puis nous formulons les questions de recherche qui ont guidé notre travail et la méthodologie que nous avons mise en œuvre pour explorer ces questions.

2. Les objets métiers, définitions

Avant de présenter différents scénarios qui aident à comprendre les contraintes de notre travail, nous allons définir les principaux objets métiers qui interviennent dans ces scénarios. D'autres objets sont définis de façon plus formelle dans le chapitre 5 présentant le modèle conceptuel. Nous définissons ici principalement ce que nous entendons par test, diagnostic et par tâche diagnostique.

2.1. Test (ou test diagnostic)

Un test est un ensemble d'exercices liés au niveau de la classe. Il permet à un enseignant et au logiciel de construire une vue des différents aspects de la compétence algébrique d'un élève, d'un groupe d'élèves ou d'une classe. Généralement, un test d'évaluation précise les capacités maîtrisées et quelques fois signale les erreurs commises. Un test diagnostic est un ensemble d'exercices spécialement conçus pour détecter non seulement les capacités maîtrisées mais aussi les conceptions que se sont formées les apprenants et qui peuvent s'avérer des obstacles à l'apprentissage.

2.2. Diagnostic

Ce terme est polysémique. Il désigne à fois la procédure qui permet d'établir une vue sur la compétence et le résultat de cette procédure (la vue). A partir des réponses des élèves au test, nous distinguons deux types (de procédure) de diagnostic : le diagnostic local à un exercice et le diagnostic global au test. Le *diagnostic local* porte sur un seul exercice. Dans le projet Pépite, ce diagnostic local consiste à coder les réponses de l'élève en appliquant la grille d'analyse multidimensionnelle prévue dans l'analyse didactique a priori de l'exercice (Chapitre 2). Le résultat de ce diagnostic local est donc un *code* qui caractérise la réponse de l'élève en fonction de critères d'évaluation sur plusieurs dimensions. Le *diagnostic global* porte sur l'ensemble des exercices du test. Une analyse transversale des codes attribués aux réponses aux différents exercices d'un test conduit à établir une vue générale sur les aspects de la compétence algébrique mis en jeu dans les exercices du test ; le résultat de ce diagnostic est le *profil cognitif* de l'élève si le test couvre l'ensemble des différents aspects de la compétence attendus à un niveau scolaire donné. Ce profil cognitif est composé d'une part d'un *stéréotype* qui situe l'élève par rapport à des niveaux de référence et, d'autre part, des *caractéristiques personnelles* de l'élève qui font apparaître ses *points forts* (capacités maîtrisées ou en cours d'acquisition) et ses *points faibles* (ses erreurs) [Vincent et al. 2005].

2.3. Tâches diagnostiques (ou exercices diagnostiques)

Une tâche diagnostique est un exercice et une analyse didactique a priori de cet exercice qui précise les connaissances, les conceptions correctes ou erronées qui pourront être inférées à partir des réponses des élèves à cet exercice. De façon plus précise, une tâche diagnostique est composée d'un exercice, d'informations qui permettent de situer cet exercice dans le modèle de

la compétence algébrique et d'une grille d'analyse des réponses anticipées des élèves à cet exercice. Pour simplifier nous parlons souvent d'exercices de diagnostic.

2.4. Stratégies de diagnostic

Nous distinguons deux stratégies de diagnostic[1] : un diagnostic prédéfini et un diagnostic adaptatif. Le diagnostic prédéfini consiste à faire passer à tous les élèves d'une même classe le même ensemble d'exercices. Il permet de faire des comparaisons entre élèves. Le diagnostic adaptatif consiste à poser des exercices qui sont choisis en fonction des réponses de l'élève aux exercices précédents. Il permet d'obtenir des bilans plus rapidement en minimisant le nombre d'exercices.

2.5. Types de tests

Nous envisageons d'administrer trois types de tests : le test prédéfini, le test adaptable et le test adaptatif. Le premier correspond à un diagnostic prédéfini ; il comporte un nombre fixe d'exercices déterminés et permet de tester l'ensemble des aspects de la compétence attendus à un niveau donné. Le deuxième est adaptable[2] par l'enseignant. Il correspond à un diagnostic prédéfini ; l'enseignant choisit le nombre et le type d'exercices proposés, le logiciel lui indiquant l'ensemble des aspects de la compétence testés ou bien l'enseignant choisit un niveau, un ensemble d'aspects de la compétence à tester et une durée de test et le logiciel lui propose un test si les contraintes ne sont pas trop fortes. Dans ce dernier cas, le logiciel suggère de diminuer les contraintes. Le troisième type correspond à un diagnostic adaptatif ; l'enseignant choisit un niveau et un ensemble initial d'exercices (ou de compétences), au moment du passage du test le système détermine les exercices à poser en fonction des choix de l'enseignant et des réponses de l'élève. Ce dernier type de diagnostic n'est pas étudié dans le cadre de cette thèse.

Le système Super-Pépite a pour objectifs de permettre à des enseignants de choisir un des trois types de tests, de le faire passer aux élèves, de recueillir et analyser les résultats et de faire un retour aux élèves sous la forme de profils cognitifs et de conseils pour faire évoluer ce profil. L'objectif de PépiGen est de faciliter la création de banques d'exercices de diagnostic.

---

[1] Un autre type de diagnostic que nous pouvons qualifier de dynamique consiste à suivre l'évolution des compétences lors d'activités d'apprentissage et non lors d'activités spécifiques d'évaluation. Nous ne traitons pas (encore) ce cas dans ce mémoire.

[2] Cette terminologie reprend celle qui est utilisée en IHM pour définir les paramétrages choisis par l'usager (logiciels adaptables) ou calculés par le système (logiciels adaptatifs)

3. Scénarios de conception

En ingénierie des Interfaces Homme Machine (IHM) une des techniques préconisées de conception utilisateur est la rédaction de scénarios [Beaudoin-Lafon et Mackay 2002, Caroll et al. 2001b]. Les *scénarios d'utilisation* partent d'observations de l'activité des utilisateurs et consistent à écrire une histoire en mêlant plusieurs observations ; puis, des *scénarios de conception* sont créés en introduisant le système à concevoir. L'objectif de cette technique de conception est de garder à l'esprit la logique d'utilisation du système à élaborer pendant toute la durée de sa conception et de considérer les fonctionnalités à mettre en œuvre dans le contexte de leur utilisation. En effet, les informaticiens ont la charge de faire fonctionner le système et ont tendance à se focaliser sur la logique de fonctionnement. Les scénarios sont un des moyens pour éviter ce biais. Les premiers scénarios servent à donner une idée des principales utilisations du logiciel et à en créer une vision commune. Ces premiers scénarios sont ensuite affinés et précisés tout au long de la conception.

Nous décrivons d'abord les principales catégories d'utilisateurs de SuperPépite puis plusieurs scénarios de conception sur lesquels nous nous appuyons pour présenter les principales utilisations envisagées de ce système. Dans le chapitre 7, nous définissons les fonctionnalités de PepiGen, l'outil auteur de SuperPépite, et décrivons les cas d'utilisation expliquant son fonctionnement en nous appuyant sur ces scénarios.

3.1. Les utilisateurs de SuperPépite

SuperPépite vise quatre grandes catégories d'utilisateurs : les élèves, les enseignants, les auteurs d'exercices, et les didacticiens. Les *élèves* passent un test en résolvant les exercices, après quoi ils obtiennent un bilan de leurs compétences algébriques établi en collaboration avec le système ou avec leur enseignant. Les *enseignants* choisissent dans une banque de tests, en fonction de leurs objectifs, un test qu'ils font passer à leurs élèves. Éventuellement ils adaptent des tests prédéfinis. Le logiciel les aide à produire un bilan de la compétence algébrique de leurs élèves individuellement et/ou en groupe. Ultérieurement, le logiciel proposera des parcours d'apprentissage adaptés au niveau de compétence des élèves testés. Les *auteurs* construisent des exercices à partir de modèles d'exercices. Certains exercices sont générés automatiquement, mais, pour d'autres, il est intéressant de laisser un humain définir l'énoncé en fonction de ses préférences ou de ses objectifs. Les *didacticiens* étudient les différentes productions des utilisateurs avec le système (réponses d'élèves, bilans de compétences, génération des exercices et des tests). Les *informaticiens*, en partant d'un exemple (un modèle) développent des classes

d'exercices qui permettent de générer automatiquement des clones de l'exercice originel, éventuellement, à partir de paramètres entrés par un auteur.

À ces quatre principales catégories, il faut ajouter les *informaticiens* qui ont la charge de développer le système, d'en assurer la maintenance et, en particulier, de créer de nouvelles classes d'exercices en fonction des besoins exprimés par les utilisateurs, besoins qui, bien entendu, ne sont pas figés une fois pour toutes mais évoluent avec le temps. La nécessité d'obtenir des banques d'exercices de taille critique implique de penser aussi les outils auteurs pour leur faciliter la tâche.

Notre travail dans cette thèse, se centre sur la génération d'exercices afin de faciliter la création des banques d'exercices et donc sur le travail des auteurs. Cependant, nous présentons les autres scénarios de conception pour illustrer le cadre de notre travail. Les quatre premiers scénarios envisagent la création d'un test par un enseignant ou un didacticien, le suivant la passation d'un test par un groupe d'élèves, les suivants envisagent trois cas d'utilisation du bilan construit par Pépite. Enfin, les deux derniers sont ceux sur lesquels nous avons travaillé dans le cadre de cette thèse.

3.2. Scénario 1 : un enseignant découvre SuperPépite et s'approprie un test type

Jean, enseignant stagiaire, souhaite tester ses élèves de troisième en début d'année. Un collègue lui a envoyé l'adresse du site SuperPépite. Il clique sur « tests diagnostics ». Il choisit un test prédéfini pour un début de troisième et une durée de 45 minutes. Il parcourt les exercices qui sont proposés dans ce test : l'exercice numéro 16, « Preuve et programme de calcul » lui paraît trop complexe. Pour se familiariser avec le logiciel, Jean demande au système de remplacer cet exercice par un autre du même type mais plus simple. Le système lui propose plusieurs clones. Jean choisit un de ces clones. Le test lui semblant trop long, il supprime les exercices 19 et 20. Pour voir quel type de bilan est construit, il traite les exercices en faisant des erreurs. Puis il regarde le profil construit par le système à partir de ses réponses.

3.3. Scénario 2 : un enseignant, habitué, compose un test diagnostic

En prévision d'une séance d'aide individualisée qui a lieu la semaine suivante, Christian, enseignant en classe de seconde A, veut proposer à un petit groupe d'élèves un test pour vérifier, dans un premier temps, leurs capacités en calcul algébrique, puis, dans un deuxième temps pour étudier s'ils sont capables de mobiliser l'algèbre pour résoudre un problème. Il se connecte sur le site et clique sur « Nouveau test ». Il choisit le niveau classe de seconde, puis, la composante à

tester : « Effectuer du calcul algébrique ». Le logiciel propose une liste de plusieurs types d'exercices pour tester cette composante dont : « Reconnaître des égalités vraies », « Associer des expressions du second degré équivalentes », « Développer et factoriser une expression du second degré », « Preuve et programme de calcul ». Christian sélectionne le premier ; le logiciel affiche l'interface élève d'un représentant de cette classe d'exercices (Chapitre 7, Figure 10.a). Christian le dépose dans son test. Puis, il demande à voir des exercices du type « Preuve et programme de calcul ». Il en sélectionne deux qu'il ajoute à son test. Il vérifie que le temps estimé pour la passation du test composé de 3 exercices sélectionnés est inférieur à la demi-heure et il enregistre le test sous le nom « test-2A-12oct07 » dans sa base de tests personnelle.

3.4. Scénario 3 : un enseignant veut étudier l'évolution des compétences de ses élèves depuis le dernier test

Christian, un mois après un premier test veut vérifier que son groupe en aide individualisée a progressé depuis le 12 octobre. Pour cela, il ouvre le test du 12 octobre et en demande un clone. Le logiciel lui présente un clone de son test, c'est-à-dire un test dont les exercices ont les mêmes caractéristiques par rapport au modèle de référence de la compétence algébrique. Christian l'enregistre dans sa base de tests personnelle.

3.5. Scénario 4 : une didacticienne des mathématiques génère des tests diagnostics

Françoise, didacticienne des mathématiques, veut créer plusieurs tests diagnostics du niveau quatrième pour les mettre en œuvre avec des enseignants dans leur classe à différents moments de l'année scolaire : un premier test en début d'année scolaire pour vérifier les acquis de cinquième, un test en cours d'année pour repérer les difficultés rencontrées et un dernier test pour faire un bilan en fin de quatrième. Elle définit le niveau caractérisant le test, un commentaire sur le contexte du test, ajoute des exercices dans le test, vérifie les capacités testées et le nombre de fois où elles sont testées, éventuellement réordonne les exercices et enregistre le test dans la base de tests. Elle vérifie que la durée prévue de passation ne dépasse pas 45 minutes. Elle crée ainsi pour les trois périodes de l'année envisagées plusieurs tests pour proposer un choix aux enseignants de quatrième avec lesquels elle travaille.

3.6. Scénario 5 : des élèves passent un test diagnostic

Huit élèves de la seconde A viennent en salle informatique pour l'aide individualisée et se connectent à leur Environnement Numérique de Travail. Leur enseignant y a déposé le test qu'il

a fabriqué (scénario 2). Les élèves résolvent les trois exercices pendant vingt minutes en interagissant avec l'interpréteur d'exercices. Ils réalisent les exercices en commençant par ceux qui leur paraissent plus faciles puis enregistrent leur travail dans le dossier qui leur est proposé par le logiciel et qui se trouve sur le serveur du réseau local de l'établissement. A la fin, Christian leur demande de faire des exercices interactifs d'application du cours. Pendant ce temps, il se rend auprès de chacun des élèves pour faire avec eux, un bilan en commentant le diagnostic établi par SuperPépite à partir de leurs réponses aux trois exercices du test. Il leur fait comparer avec les réponses au test précédent.

3.7. Scénario 6 : un élève dialogue avec un enseignant pour réfléchir sur son bilan de compétence

Robin est un élève de troisième qui a des résultats scolaires faibles. Dans le cadre du soutien scolaire organisé par une association de son quartier, il a passé un test de compétence avec SuperPépite. Marie-France, une enseignante a étudié le profil construit par le logiciel et en discute avec Robin. Elle lui présente son profil en commençant par relever les points positifs ; puis elle corrige les exercices qui révèlent des conceptions inadaptées à déstabiliser[1]. Elle lui propose ensuite une liste d'exercices adaptés à son profil.

3.8. Scénario 7 : un enseignant constitue des groupes de travail dans sa classe

En début de seconde avant d'entamer le chapitre sur les fonctions, Pauline veut homogénéiser sa classe et s'assurer des bases de ses élèves en algèbre. Elle a fait passer un bilan de compétence en algèbre à toute sa classe de 2nde. SuperPépite lui montre une géographie de sa classe qui fait apparaître qu'un groupe de douze élèves est en niveau 3 sur chacune des composantes de la compétence algébrique. Elle décide de prendre ces élèves en aide individualisée sur deux semaines :

- La première semaine, elle prend ceux qui ont le type d'erreur « Identification incorrecte de x et + : Linéarisation des expressions » et leur propose des exercices où il s'agira de déstabiliser les conceptions inadéquates, par exemple en substituant des valeurs numériques dans des relations algébriques.

[1] Certaines conceptions que se sont construites les élèves (par exemple le signe égal annonce un résultat, une lettre désigne une unité), qui peuvent avoir un domaine de validité (par exemple en arithmétique de l'école primaire) peuvent s'avérer des obstacles pour les apprentissages ultérieures. Pour permettre à l'élève de s'adapter à un nouveau contexte, il est nécessaire de « déstabiliser » ses conceptions antérieures.

- La deuxième semaine, elle fera travailler ceux qui ne maîtrisent pas les priorités des opérations (ce peut être les mêmes).

3.9. Scénario 8 : un auteur enrichit la banque d'exercices

Claude, formatrice d'enseignants de mathématiques, veut présenter à des stagiaires des exercices de diagnostic à différents niveaux scolaires. Elle lance PépiGen et choisit de générer un clone de l'exercice « Expressions d'un programme de calcul ». Pour cela, elle utilise la palette de mots pour saisir en français les trois étapes d'un programme de calcul. Au fur et à mesure, PépiGen affiche le programme sous forme d'une expression algébrique et d'une phrase en français. Quand elle valide l'énoncé, PépiGen génère l'ensemble des solutions anticipées correctes ou erronées et leur grille de codage. Claude demande à voir cette grille de codage pour s'assurer qu'elle est correcte, puis, elle enregistre cet exercice avec le niveau quatrième dans la banque d'exercices. Elle en crée de plus complexes pour les niveaux supérieurs. Elle envisage de demander à ses stagiaires de continuer à enrichir ainsi la banque d'exercices par niveaux.

3.10. Scénario 9 : un(e) développeur(e) informatique ajoute un modèle d'exercices

Aminata, ingénieure informaticienne, reçoit par courriel une tâche diagnostique (un exercice et l'analyse a priori des capacités en jeu, des réponses prévisibles des élèves etc.) crée par une didacticienne, Françoise. Dans un premier temps, elle étudie l'exercice et repère les invariants et les paramètres qui, de son point de vue, permettent d'en générer des clones. Elle propose plusieurs maquettes pour spécifier l'interface élève et l'interface auteur.

Dans un deuxième temps, afin de stabiliser la spécification de la classe d'exercices sous la forme d'un fichier XML dont le format est décrit par un Schéma XML (Chapitre 7, section 8.1), elle prend rendez-vous avec Françoise pour en discuter et préciser les contraintes sur les paramètres. Ensuite, elle cherche dans la banque de classes d'exercices, une classe dont l'interface élève est voisine et dont elle pourrait réutiliser une partie du code. Sinon elle programme l'interface-élève. Elle fait de même pour l'interface auteur. Ensuite, elle programme la génération des réponses anticipées aux questions ouvertes en s'appuyant, d'une part sur l'analyse didactique a priori, et, d'autre part sur le logiciel de calcul formel dédié au diagnostic cognitif de SuperPépite.

Puis, elle teste son programme et le fait tester par Françoise et les autres membres de l'équipe avant de l'ajouter dans la banque de classes d'exercices.

3.11. Conclusion

Les scénarios 1 à 8 ont fait l'objet d'études dans le cadre du programme Cognitique (2002-2004) [Delozanne et al. 2005, Rogalski 2005]. Pour mettre en œuvre SuperPépite, il faut disposer de la brique de base : celle qui permet de générer des banques d'exercices et l'analyse automatique des réponses associées à ces exercices. C'est l'objectif de PépiGen et le défi de notre thèse.

4. Architecte fonctionnelle de SuperPepite

L'évaluation multicritère des réponses ouvertes est un problème complexe qui comprend de nombreuses facettes comme les scénarios nous le font pressentir et comme en témoignent les recherches que nous avons étudiées au chapitre précédent. Nous en distinguons principalement 6. Pour chacun de ces problèmes, nous avons spécifié un logiciel pour le résoudre. Notre travail de thèse c'est centré sur la résolution des deux premiers. D'autres membres de l'équipe ont exploré la résolution des autres problèmes qui a donné lieu à des prototypes réalisés dans le cadre de stage de DEA ou de master. Nous décrivons, chacun de ces problèmes, le nom du logiciel spécifié pour résoudre ce problème et l'état de son développement.

1. La *génération d'exercices de diagnostic* : réalisée par le module *PépiGen* (conçu et réalisé dans le cadre de cette thèse). *PépiGen* permet à un auteur de créer des exercices de diagnostic à partir de classes paramétrées d'exercices de diagnostic.
2. Le *diagnostic automatique* : réalisé à partir des réponses des élèves à un test par le module *PépiDiag* qui élabore le diagnostic local (réalisé dans cette thèse) et global (le diagnostic automatique a été mis en oeuvre dans des prototypes destinés à étudier l'acceptation par les enseignants [Jean 2000, Vincent et al. 2005], [Bernard 2006]). L'analyse automatique des réponses ouvertes des élèves est réalisée en s'appuyant sur *Pépinière* (conçu et réalisé dans le cadre de cette thèse) qui est le module de calcul formel qui gère l'analyse des expressions algébriques. D'autres chercheurs de l'équipe se sont penchés sur l'analyse des réponses en langage naturel (ou mathurel) [Normand et al. 2005] mais ces études n'ont pas encore débouché.
3. La *génération de tests* adaptés à un niveau scolaire et proposant un éventail de tâches suffisamment larges et inhabituelles pour permettre un diagnostic cognitif : *PepiGenTest* sera le module permettant de choisir des tâches diagnostiques pour élaborer un test pour la construction du diagnostic (prédéfini, adaptable, adaptatif). Ce module n'est pas réalisé actuellement car il s'appuie sur PépiGen.

4. *L'exploitation des profils par les enseignants* : *PepiProf* assiste l'enseignant pour établir un profil cognitif de ses élèves en algèbre et pour élaborer une stratégie d'enseignement pour faire évoluer ces profils. Ce module a fait l'objet de nombreux travaux en particulier avec le concours d'ergonomes [Rogalski 2005]. Le problème est particulièrement complexe puisqu'il s'agit d'anticiper des usages qui n'existent pas : aucun enseignant ne peut à la main faire des diagnostics aussi fins que ceux construits par SuperPépite. De plus, la mise au point de parcours d'apprentissage est également complexe à instrumenter. Des prototypes ont là encore été construits pour mettre à l'épreuve des hypothèses et aider l'équipe à faire des choix.
5. *L'exploitation des profils par les élèves* : *PepiEleve* proposera à un élève le bilan de compétence en algèbre et une stratégie d'enseignement pour progresser. Pour l'instant, ce module n'existe pas. Des études sont menées actuellement à partir d'analyses d'entretiens entre un enseignant et un élève. Un objectif est de construire un agent virtuel pour mener ce type d'entretien [Farouk et al. 2007].
6. *L'interface d'analyse des données pour les chercheurs* : *PepiChercheur* permettra au chercheur de travailler sur les productions des élèves ou des enseignants et sur les différents diagnostics produits. Des outils ad hoc ont été construits dans ce but dans le cadre d'un projet de master 1.

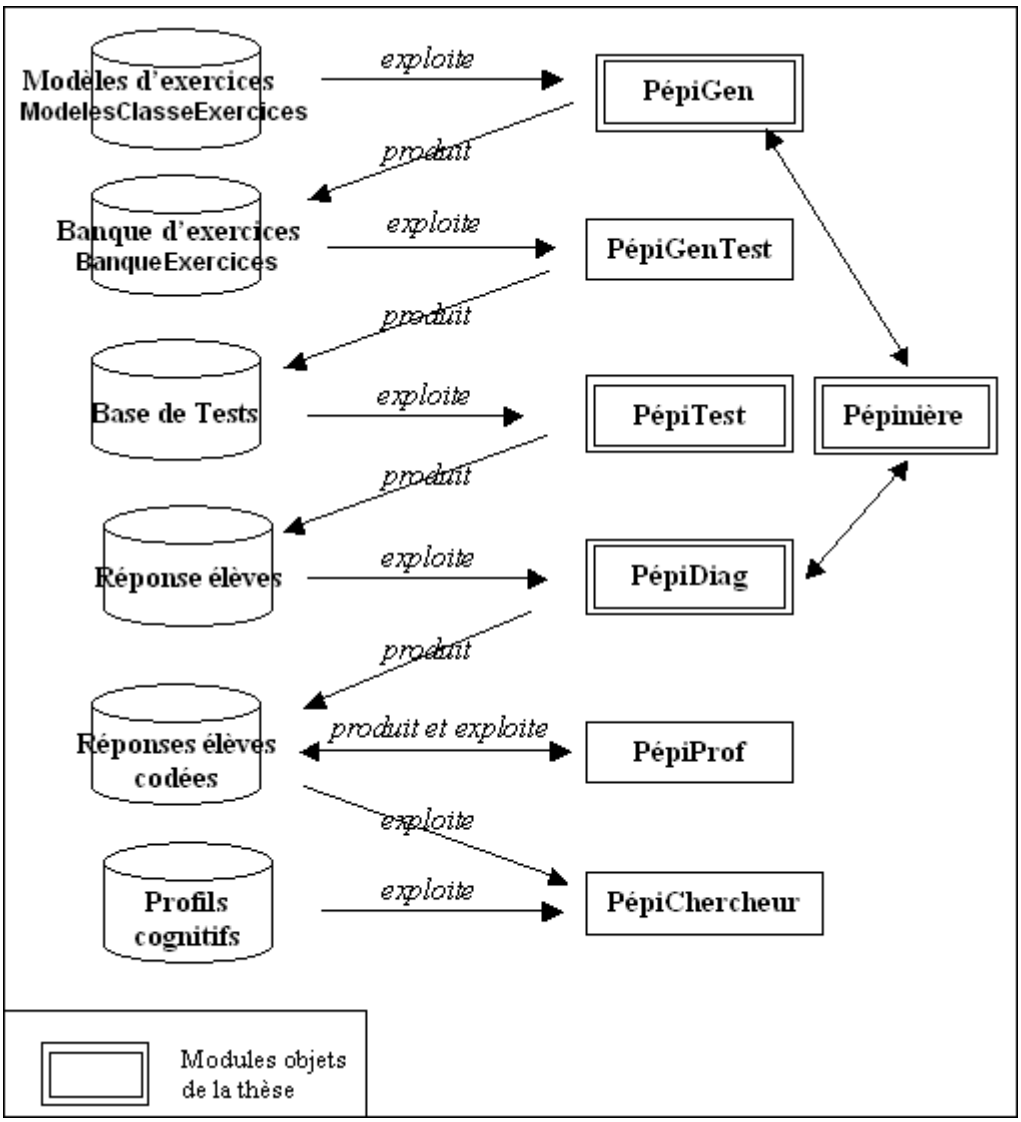


Figure 1 : Architecture fonctionnelle de SuperPépite

5. Problématique

L'objectif de notre travail en EIAH est de créer un outil pour faciliter le développement d'exercices de diagnostic. Notre problème est double : il s'agit de définir un système auteur mais aussi de permettre une analyse automatique et multicritère de réponses ouvertes. Notre problématique concerne donc deux thèmes de recherche, celui des systèmes auteurs et celui du diagnostic cognitif. Dans un état de l'art sur les systèmes auteurs pour les tuteurs intelligents que nous avons déjà cité au chapitre précédent [Murray 2003a], Murray distingue sept catégories de systèmes auteurs selon les caractéristiques des systèmes conçus : planification pédagogique, stratégies tutorielles, apprentissage sur simulateur, systèmes experts, différents types de

connaissances, objectifs spécifiques, hypermédias adaptatifs. PépiGen partage les caractéristiques de deux de ces catégories.

PepiGen a comme objectifs le diagnostic et une modélisation fine des connaissances de l'élève prenant en compte des conceptions naïves ou inadaptées, ce qui le rattache à la catégorie des systèmes auteurs fondés sur un *système expert*, i.e. un modèle exécutable du domaine d'expertise. *Notre problématique première est donc de concevoir un modèle exécutable de diagnostic cognitif en algèbre élémentaire*.

Ceci nous amène à résoudre le problème crucial de l'analyse de raisonnements d'élèves exprimés librement, problème qui est un thème de recherche récurrent en EIAH comme nous l'avons étudié au chapitre précédent et qui n'admet que des solutions spécifiques. D'après Murray [Murray 2003a p. 522] les évaluations des systèmes auteurs ont montré que seuls les systèmes auteurs à objectifs spécifiques étaient utilisables sans formation préalable. Ils sont caractérisés par le fait qu'ils embarquent une expertise pédagogique et réduisent ainsi la tâche de l'auteur à instancier des patrons en remplissant des formulaires qui le guident fortement. Murray souligne que l'un des problèmes soulevés par cette approche est qu'une fois que la tâche a été suffisamment codifiée pour devenir un patron, le système auteur est limité aux tâches modélisées ce qui en limite l'audience potentielle. Cependant, le fait d'embarquer des modèles de tâches spécifiques permet de construire des systèmes auteurs riches [Bell 2003] et de participer à la diffusion de résultats de recherches en didactique dans le système éducatif. Les études d'utilisabilité que nous avons menées (Chapitre 2) ont montré que la majorité des enseignants consultés souhaitent disposer d'outils de diagnostic prêts à l'emploi, éventuellement de pouvoir les adapter à leur contexte. [Zapta-Rivera et al. 2007] relèvent aussi ces préférences chez les enseignants. Une très petite minorité d'enseignants veut avoir la possibilité de créer ses propres énoncés. Comme le souligne [VanLehn et al. 2005], la majorité rêve d'un correcteur de copies automatique.

Notre problématique seconde est d'automatiser au maximum le processus de création d'exercices afin que le logiciel se charge des tâches fastidieuses et l'auteur de la partie créative.

Nous avons aussi fait le choix de fonder PépiGen sur l'utilisation de patrons instanciables qui s'appuient sur les résultats de la recherche en didactique des mathématiques de ces dernières années. Dans PépiGen, l'instanciation des patrons est, soit automatique quand elle est très contrainte, soit paramétrable par un humain quand une marge d'initiative est possible. Afin de rendre notre modèle plus réutilisable, l'expertise didactique n'est pas codée en dur dans nos patrons, mais elle est réifiée sous forme de fichiers XML respectant un schéma où le système

prend les informations concernant les dimensions et les critères d'évaluation, les connaissances et les capacités mathématiques à tester.

Ceci nous amène à explorer les questions de recherche suivantes :

1. Comment caractériser des exercices équivalents du point de vue du diagnostic pour définir des patrons instanciables d'exercices ?
2. Sur quels modèles de calcul formel s'appuyer pour générer puis analyser automatiquement des raisonnements algébriques corrects ou inadaptés ? Comment les mettre en œuvre pour une évaluation multicritère automatique de réponses ouvertes complexes ?
3. Comment mettre en œuvre une chaîne logicielle permettant à des auteurs non informaticiens d'alimenter des bases d'exercices diagnostiques ?

6. Méthodologie

La méthodologie que nous avons adoptée repose sur une démarche ascendante de généralisation à partir de premiers prototypes mis au point par des chercheurs en didactique [Grugeon 1995], puis informatisés [Jean 2000]. Cette méthodologie de prototypage est bien adaptée au contexte de conception participative qui est celui de notre projet.

Pour mettre en évidence les patrons d'exercices de diagnostic, nous avons dans un premier temps, étudié chaque exercice du test originel en le considérant comme un exemplaire d'une classe d'exercices équivalents du point de vue du diagnostic. Nous avons cherché les invariants et les paramètres permettant de générer des « clones » de cet exercice adaptés à différents niveaux scolaires. Les différentes étapes de construction de ce modèle ont été discutées avec les membres de l'équipe du projet Lingot (informaticiens, ergonome, chercheurs en didactique et praticiens). En travaillant avec des mathématiciens, ce que nous informaticiens appelons un patron instanciable du point de vue de la génération d'exercices, est apparu comme une classe d'exercices équivalents à un exercice donné, celui du test originel. Dans la suite de notre thèse, nous utilisons le terme de classe d'exercices de diagnostic (qui nous sert à communiquer avec nos partenaires) dans le sens de patron instanciable au sens de Murray. Pour définir le modèle conceptuel des classes d'exercices, nous avons ensuite formalisé la description des classes d'exercices obtenues. Dans le chapitre 5, nous présentons dans le détail la méthodologie mise en œuvre pour définir le modèle conceptuel d'une classe d'exercices.

Après la description des classes d'exercices de diagnostic, le principal problème à résoudre pour passer d'un modèle descriptif à un modèle exécutable est celui du traitement des

expressions algébriques. Afin de pouvoir générer automatiquement les réponses plausibles, correctes ou incorrectes, nous avons du concevoir un système de calcul formel adapté à notre contexte. Ce module, nommé Pépinière, est indépendant du reste de l'application. Il commence par transformer une expression algébrique saisie sous la forme d'une chaîne de caractères en arbre d'expression. Puis, lorsqu'il est appelé par PépiGen, il transforme cet arbre pour générer les solutions anticipées en tenant compte de règles fausses ou erronées. Lorsqu'il est appelé par pépiDiag, il compare cet arbre aux solutions anticipées pour établir le diagnostic local. Pour concevoir pépinière, nous nous sommes appuyés sur les théories classiques d'analyse syntaxique ([Aho et al. 1993], [Jouannaud 2007], sur la théorie des réécritures de Dershovitz et Jouannaud [Dershovitz et al. 1990], sur les travaux de Gélis [Gélis 1994] et sur l'algorithme d'unification [Pitrat 1966], [Grandbastien 1974], [Laurière 1988]. Nous avons explicité les règles de réécriture incorrectes avec les didacticiennes de l'équipe, nous les avons organisées par paquets et nous avons défini des heuristiques afin d'éviter les bouclages et l'explosion combinatoire. Ce système a été testé et validé sur des problèmes de réduction et de développement d'expressions algébriques. Ce travail de conception et de développement particulièrement lourd est décrit dans le chapitre 6.

Enfin, pour valider notre proposition nous avons développé la chaîne logicielle qui permet à un auteur de définir un exercice en spécifiant les paramètres d'une classe d'exercice, puis à un élève de résoudre l'exercice et au logiciel de diagnostic de coder la réponse de l'élève. Le chapitre 7 décrit l'ensemble de cette chaîne et illustre son utilisation sur des exemples d'exercices créés avec PépiGen. Dans le chapitre 8, nous revenons sur nos résultats et nos questions de recherche.

Chapitre 5

Modèle Conceptuel des classes d'exercices

1. Introduction 111

2. Méthodologie 111

3. Classe d'exercices « Reconnaître des égalités numériques vraies » 114
3.1. Indexation didactique 114
3.2. Génération de l'exercice 115
3.2.1. Interface élève 115
3.2.2. Génération des énoncés 115
3.3. Génération du diagnostic 116
3.4. Type de génération 116

4. Classe d'exercices « Reconnaître des sommes et des produits » 118
4.1. Génération de l'exercice 119
4.2. Génération du diagnostic 119

5. Discussion 120

6. Classe d'exercices « Correspondance entre aire et expression » 121
6.1. Indexation didactique 121
6.2. Génération de l'exercice 122
6.2.1. Interface élève 122
6.2.2. Génération de l'énoncé 123
6.3. Génération du diagnostic 124
6.4. Type de génération 124

7. Classe d'exercices « Expression littérale de l'aire d'un rectangle » 125
7.1. Génération de l'exercice 127
7.1.1. Interface élève 127
7.1.2. Génération des énoncés 128
7.1.3. Génération du diagnostic 129
7.1.4. Type de génération 130
7.2. Discussion 131

8. Classe d'exercices « Preuve et programme de calcul » 131
8.1. Génération de l'exercice 134
8.1.1. Interface élève 134
8.1.2. Génération de l'énoncé 134
8.2. Génération du diagnostic 135
8.3. Type de génération 136

9. Classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales » 139
9.1. Génération de l'exercice 140
9.1.1. Génération de l'interface 140
9.1.2. Génération des énoncés 140
9.2. Génération du diagnostic 142
9.3. Type de génération 142
9.4. Discussion 143

10. Classe d'exercices « Expressions algébriques d'un programme de calcul» 143
10.1. Génération de l'exercice 145
10.1.1. Interface Élève 145
10.1.2. Génération des énoncés 145
10.2. Génération du diagnostic 146
10.3. Type de génération 147

11. Élaboration d'un modèle conceptuel d'une classe d'exercices 148
11.1. Indexation didactique des exercices 149
11.2. Génération des énoncés 150
11.3. Description de l'interface 151
11.4. Génération du diagnostic 154

12. Conclusion 156

1. Introduction

Le problème abordé dans ce chapitre est de savoir, comment proposer des modèles génériques pour une mise en œuvre informatique de banques de tests diagnostics, à partir d'une analyse didactique d'exercices particuliers et des réponses à ces exercices recueillies auprès des élèves. Notre objectif est de créer des « clones » des exercices de diagnostic de PépiTest, c'est-à-dire des exercices qui ont le même type d'énoncé et la même grille de codage que les exercices originels. En effet les enseignants veulent pouvoir tester l'ensemble des compétences et des cohérences de fonctionnement à différents moments de la scolarité et demandent à disposer de plusieurs tests diagnostics [Delozanne et al 2002b]. Deux questions sont soulevées par ce travail. La première question est l'objet de ce chapitre : à partir des tâches diagnostiques définies par [Grugeon 1995], comment mettre en évidence les caractéristiques qui permettent de générer des clones d'un exercice et le diagnostic local associé. En d'autres termes, il s'agit de définir un modèle conceptuel de classes d'exercices pour générer les clones et l'analyse automatique des réponses à ces exercices. Dans les chapitres suivants nous traitons la deuxième question : quels modèles informatiques élaborer pour mettre en œuvre ce modèle conceptuel ?

Dans ce chapitre, nous commençons par présenter la méthodologie utilisée pour définir des classes d'exercices. Puis nous analysons et caractérisons plusieurs classes d'exercices représentatives des principales difficultés à résoudre pour générer des « clones » et le diagnostic local sur ces clones. Nous terminons par la description du modèle conceptuel qui fonde la mise en œuvre informatique du logiciel PépiGen.

2. Méthodologie

La méthodologie adoptée est une méthodologie ascendante qui décrit puis généralise les exercices de PépiTest mis au point par les chercheurs en didactique. Pour ceci, chaque exercice a été étudié dans la perspective de générer des exercices équivalents du point du vue du diagnostic cognitif en nous appliquant à mettre en évidence des invariants, des paramètres et des contraintes sur ces paramètres. Un premier document caractérisant les classes d'exercices pour générer des clones a été ainsi produit. Par la suite, ce dernier a été complété et modifié dans une démarche itérative de validation. Ce travail nous a conduit à spécifier le modèle conceptuel que nous proposons dans ce chapitre. Pour notre étude nous avons distingué plusieurs points de vue des acteurs sur les exercices.

- Tout d'abord, les didacticiens de l'équipe caractérisent les exercices en fonction de quatre éléments principaux. Premièrement, un exercice est situé par rapport aux trois grandes

composantes de la compétence algébrique mises en jeu (effectuer du calcul algébrique, utiliser l'outil algébrique pour résoudre des problèmes, traduire algébriquement dans différentes représentations (géométrique, graphique, langage naturel) et vice-versa). Deuxièmement, un exercice révèle des *capacités* (entre une et six parmi une trentaine) telles qu'elles sont énoncées dans les programmes officiels. Troisièmement, un exercice met en œuvre un *type de tâches* qui caractérise le problème posé à l'élève (e.g. reconnaître des égalités toujours vraies, résoudre une équation, associer à l'aire d'une surface une expression algébrique). Quatrièmement, dans un test diagnostic un exercice vise un *objectif* de diagnostic (e.g. identifier les compétences de l'élève en calcul numérique, identifier les règles de conversion utilisées pour passer d'une représentation géométrique à une représentation algébrique). Pour atteindre ces objectifs, [Grugeon 1995] a mis en place une *grille d'analyse multidimensionnelle* qui définit des *critères d'évaluation* repérés par un *code* et regroupés sur huit *dimensions d'évaluation*. Ces éléments sont détaillés dans les annexes A1.

- Pour les élèves, les exercices sont composés de *questions* auxquelles ils doivent répondre via l'interface. Les exercices comportent deux types de questions : des *questions ouvertes* dont les réponses nécessitent des démarches de calcul (suite d'expressions algébriques) ou des justifications (en langue naturelle ou « mathurelle ») et des *questions fermées* avec des réponses de type *choix multiples* ou *choix exclusifs* ou des réponses préformatées. A ces deux types de questions correspondent différents types d'interactions élève-logiciel. Pour les questions ouvertes, l'élève saisit, dans des zones de textes, les calculs, les réponses et les justifications ; pour les questions fermées, selon les exercices, différentes modalités se présentent : clics sur des cases ou des zones, choix dans des menus, utilisation de palettes de termes, associations.

- Un enseignant qui souhaite préparer un test pour une classe dispose d'une banque d'exercices et d'une banque de tests diagnostics. En fonction de ses objectifs, soit il compose un test en choisissant des exercices en fonction du niveau scolaire, de la durée du test et des capacités à tester, soit il choisit un test prédéfini et éventuellement le modifie.

- Un auteur d'exercices a pour objectif d'obtenir un exercice équivalent à un exercice originel. Si le champ des possibles est très contraint, l'auteur demande au système de générer automatiquement des clones qu'il peut accepter ou refuser. Dans le cas contraire, en fixant des paramètres, il peut exprimer des préférences sur les questions et charger ensuite le système de générer automatiquement l'interface et la grille d'analyse des réponses. Cette dernière associe à chaque question, l'ensemble des *réponses anticipées* correctes, partielles ou incorrectes et une *grille de codage* de ces réponses. Une *procédure de diagnostic* applique cette grille aux réponses des élèves, pour produire d'une part des *codes* qui correspondent à des *critères d'évaluation*, et,

d'autre part, une liste éventuellement vide des *règles correctes ou erronées* supposées produire ces réponses.

Pour prendre en compte les points de vue des acteurs, notre modèle distingue différents types d'informations qui concernent l'indexation des exercices d'un point de vue didactique, la génération d'un exercice et la génération du diagnostic. *L'indexation didactique* des exercices utilisés pour la création de tests se fait en référence aux *objectifs*, aux *composantes*, aux *capacités*, aux *types de tâches*, au *niveau scolaire* et à la durée du test. La génération d'un exercice est concernée par les caractéristiques de l'*interface* et les modalités d'interaction d'une part, par la *génération des énoncés* et des questions d'autre part. La *génération du diagnostic* local à l'exercice crée automatiquement l'ensemble des réponses anticipées à partir de l'analyse didactique a priori. Pour tous les exercices de PépiTest, nous nous sommes attachées à définir les *invariants* et les *paramètres*. Au regard de ces informations nous proposons le *type de génération* à mettre en œuvre pour l'exercice et le diagnostic associé : une *génération automatique* ou une *génération assistée*, et nous détaillons cette génération pour chaque classe d'exercices étudiée dans ce chapitre.

Pour définir le modèle conceptuel d'une classe d'exercices, nous avons retenu dans le cadre de cette thèse des classes d'exercices qui, d'une part sont représentatives des différents types d'exercices de diagnostic de PépiTest, et, d'autre part pointent les éléments spécifiques à prendre en compte et les difficultés à résoudre pour la génération du « clone » et du diagnostic.

Les deux premières classes d'exercices que nous étudions correspondent à des questions fermées, la première permettant d'explorer les possibilités et les limites de modélisation de la spécification QTI (Chapitre 3). Les deux classes d'exercices suivantes permettent de faire apparaître pour la génération de l'exercice et du diagnostic, des difficultés liées à la génération automatique d'une figure sur laquelle porte la question posée. Nous étudions ensuite deux classes d'exercices qui correspondent à des questions ouvertes et mettent en jeu des raisonnements algébriques. Ces deux classes illustrent les difficultés à produire, pour la génération du diagnostic, les différentes étapes d'un raisonnement algébrique plausible à ce niveau scolaire par application de règles correctes ou erronées. Enfin nous étudions une classe d'exercices qui porte sur la production de textes en langage naturel contraint et présente un exemple de traitement de réponses en langage naturel contraint.

Dans la suite de ce chapitre, nous détaillons chacune de ces classes d'exercices. Nous présentons les informations d'indexation didactique, les informations concernant la génération de l'exercice et la génération du diagnostic local à l'exercice, puis le type de génération envisagé.

Pour la génération de l'exercice, nous distinguons les informations qui concernent l'interface de celles qui concernent la génération des énoncés. Les autres exercices de PépiTest constituent des variantes de ces classes d'exercices.

3. Classe d'exercices « Reconnaître des égalités numériques vraies »

Cette classe d'exercices est la plus simple du point de vue de la génération de l'exercice et du diagnostic. La FFigure 1 présente l'interface élève de l'exercice originel de cette classe, la Figure 2 en présente un clone. Étudions les informations nécessaires pour en déduire le modèle conceptuel.

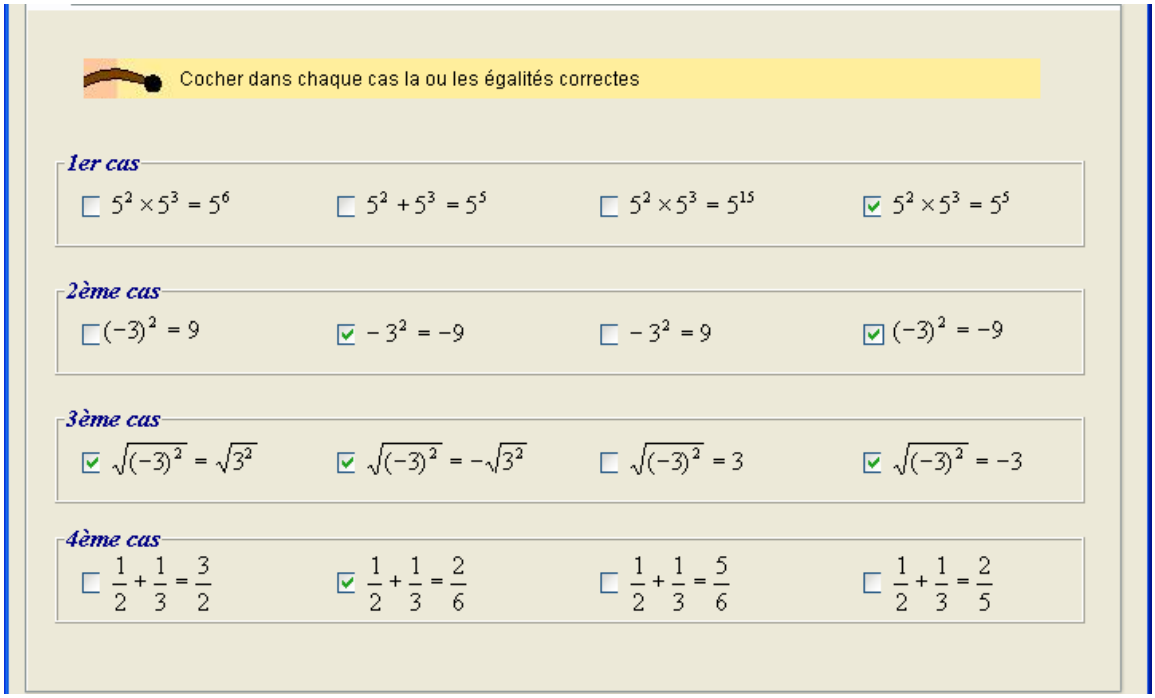


Figure 1 : Interface de l'exercice originel de la classe « Reconnaître des égalités vraies »

3.1. Indexation didactique

Les exercices de cette classe ont pour objectif de vérifier les compétences d'un élève dans le cadre numérique. Ils appartiennent à la composante « *Effectuer du calcul algébrique* » et mettent en jeu la capacité « *Reconnaître la structure d'une expression algébrique* ». Le type de tâche consiste à rechercher si les égalités portant sur des expressions numériques sont vraies ou fausses (Tableau 1.a).

Indexation didactique	
Objectif	Vérifier les compétences d'un élève dans le cadre numérique
Composante	Effectuer du calcul algébrique
Capacité	Reconnaître la structure d'une expression algébrique
Type de tâche	Rechercher si les égalités portant sur des expressions numériques sont vraies ou fausses.
Critères d'évaluation	V1, V2, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V2 : correct partiel ou non attendu V3 : incorrect
	EA1, EA31, EA33, EA42 : critères d'évaluation de la dimension « Utilisation des règles d'Écriture et de réécriture Algébrique » EA1 : Utilisation correcte des règles de transformation EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct EA33 : Utilisation de règles de transformation fausses identifiées EA42 : Les règles de transformation utilisées « assemblent » les termes

Tableau 1.a : Caractérisation de la classe « Reconnaître des égalités numériques vraies »

3.2. Génération de l'exercice

3.2.1. Interface élève

D'un point de vue abstrait, l'interface est composée d'un *énoncé* et de quatre *questions* (1er cas à 4ème cas). L'énoncé est un *texte* invariant. Chaque question à *choix multiples* conduit à une *réponse* sous la forme d'un ou plusieurs *choix* parmi les quatre réponses proposées (Tableau 1.b). Pour définir des clones de cet exercice, il suffit de faire varier les égalités numériques qui constituent ces choix et pour chaque question la position des choix correctes.

3.2.2. Génération des énoncés

En remplaçant les nombres proposés dans l'exercice originel par des variables, nous faisons apparaître des *invariants* qui caractérisent les égalités numériques proposées pour chaque clone. Par exemple, l'égalité numérique $5^2 \times 5^3 = 5^6$ est produite par l'égalité $a^{\,n} \times a^{\,p} = a^{\,n+p}$. Les expressions littérales obtenues sont les invariants et les lettres *a*, *b*, *n* et *p* sont les *paramètres* de l'exercice. Ces paramètres *a*, *b*, *n* et *p* sont soumis à des *contraintes* qui déterminent la complexité des égalités proposées : ce sont des nombres entiers positifs qui vérifient les inégalités suivantes : $\forall\, a, b, n, p \in \mathrm{N}$ $1 < a < 10$ $1 < b < 10$ $1 < n < 10$ $1 < p < 10$

Ces contraintes sont définies en accord avec les enseignants et les didacticiens. Le Tableau 1.b présente l'ensemble des invariants et des paramètres de la classe d'exercices et le Tableau 2 présente les valeurs des paramètres pour deux représentants de cette classe (Figure 1 et Figure 2).

Interface					
- Un énoncé dont le texte est invariant - Quatre questions à choix multiples (question fermée) avec quatre choix dont les contenus sont les invariants de l'exercice					
Génération des énoncés					
		Choix1	Choix2	Choix3	Choix4
Invariants	Q_1	$a^n \times a^p = a^{n \times p}$	$a^n + a^p = a^{n+p}$	$a^n \times a^p = a^{a \times p}$	$a^n \times a^p = a^{n+p}$
	Q_2	$(-a)^2 =$ *éval*(a^2)	$-a^2 =$ $-$ *éval*(a^2)	$-a^2 =$ *éval*(a^2)	$(-a)^2 =$ $-$ *éval*(a^2)
	Q_3	$\sqrt{(-a)^2}=\sqrt{a^2}$	$\sqrt{(-a)^2}=-\sqrt{a^2}$	$\sqrt{(-a)^2}=a$	$\sqrt{(-a)^2}=-a$
	Q_4	$\frac{1}{a}+\frac{1}{b}=\frac{b}{a}$	$\frac{1}{a}+\frac{1}{b}=\frac{2}{a\times b}$	$\frac{1}{a}+\frac{1}{b}=\frac{a+b}{a\times b}$	$\frac{1}{a}+\frac{1}{b}=\frac{2}{a+b}$
	Remarque : *éval*(a^2) signifie que le membre droit de l'égalité est le résultat du calcul de a^2				
Paramètres	- *a, b, n* et *p* - La position p_i des choix à l'interface (Choix1 à 4)				
Contraintes	$\forall$ *a, b, n, p* $\in$ N $1 < a < 10$ $1 < b < 10$ $1 < n < 10$ $1 < p < 10$				
Type de génération	Pour chaque question, génération automatique des valeurs de *a, b, n* et *p* et des positions P_i des choix (tirage aléatoire)				

Tableau 1.b : Caractérisation de la classe « Reconnaître des égalités numériques vraies »

3.3. Génération du diagnostic

Les *réponses anticipées* sont les quatre choix proposés pour chacune des quatre questions. Ainsi la première question a quatre réponses anticipées qui correspondent aux égalités numériques : $5^2 \times 5^3 = 5^6$, $5^2 + 5^3 = 5^5$, $5^2 \times 5^3 = 5^{15}$ et $5^2 \times 5^3 = 5^5$. La *grille de codage* élaborée par les didacticiens associe à chaque réponse un *code* qui caractérise la *règle correcte* ou *erronée* vraisemblablement mise en œuvre dans la réponse. Un *commentaire* explicite éventuellement le code. Par exemple, le choix de cocher l'égalité $5^2 + 5^3 = 5^5$ est associé au commentaire « Calcul incorrecte avec règles de regroupement » et a pour code V3, EA42. Le Tableau 1.c présente cette grille de codage. Pour cette classe d'exercices, la grille de codage est indépendante du clone car les égalités qui correspondent aux différents choix sont des invariants de la classe d'exercices.

3.4. Type de génération

Les choix proposés dans cette classe d'exercices sont des expressions littérales fixes où les paramètres *a, b, n* et *p* de l'exercice sont très contraints et prennent un nombre limité de valeurs (huit valeurs). Nous générons les paramètres et la position p_i des réponses correctes de façon aléatoire. Les réponses anticipées étant les invariants de la classe d'exercices, la grille de codage

est invariante quelque soit le clone de l'exercice. Ainsi, pour cette classe d'exercices, nous avons choisi de générer les exercices de façon automatique.

Génération du diagnostic local					
		Choix1	Choix2	Choix3	Choix4
Q_1	**Code**	V3, EA33	V3, EA42	V3, EA42	V1, EA1
	Commentaires	Règle erronée du type $a^n \times a^p = a^{n \times p}$	Calculs incorrects avec « règles de regroupement »	Calculs incorrects avec « règles de regroupement »	Réponse correcte
Q_2	**Code**	*V1 ou V2, EA1	*V1 ou V2, EA1	V3, EA31	V3, EA33
	Commentaires	Réponse correcte	Réponse correcte	Écriture sans parenthèse avec mémoire	Écriture incorrecte
Q_3	**Code**	V1, V3	V3, EA33	*V1 ou V2, EA1	V3, EA33
	Commentaires	V3 : Traitement incorrect	Représentation erronée de la racine carrée d'un nombre positif	Réponse correcte	Représentation erronée de la racine carrée d'un nombre positif
Q_4	**Code**	V3, EA42	V3, EA33	V1 EA1	V3, EA33
	Commentaires	Calcul incorrect basé sur le produit en croix	Calcul opératoire incorrect	Réponse correcte	Calcul opératoire incorrect
* Cas particuliers : pour les choix 1 et 2 de la question Q_2 et le choix 3 de la question Q_3 les codes V1 et V2 sont liés au nombre de réponses correctes : le code est V1 si les 2 réponses correctes sont cochées et code est V2 si une seule réponse a été cochée.					
Type de génération		Génération automatique du diagnostic : la grille de codage est invariante quelque soit le clone d'exercice généré			

Tableau 1.c : Caractérisation de la classe « Reconnaître des égalités numériques vraies »

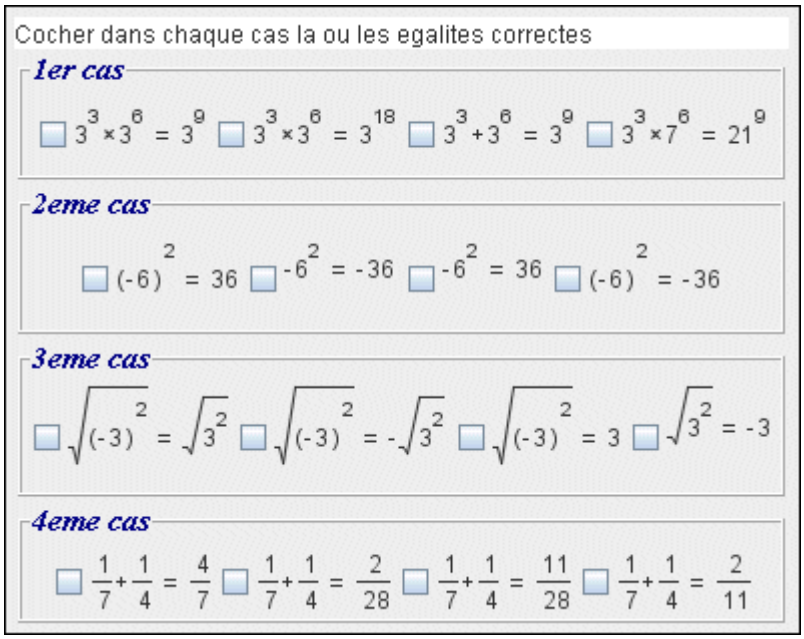


Figure 2 : Interface d'un clone de la classe « Reconnaître des égalités vraies »

	Question	*a*	*b*	*n*	*p*		Question	*a*	*b*	*n*	*p*
Exercice originel (Figure 1)	1	5		2	3	**Clone** (Figure 2	1	3		3	6
	2	3					2	6			
	3	3					3	3			
	4	2	3				4	7	4		

Tableau 2 : Valeur des paramètres *a*, *b*, *n*, *p*

4. Classe d'exercices « Reconnaître des sommes et des produits »

Cette classe d'exercices comporte huit questions à *choix exclusif* où les réponses proposées sont des expressions littérales composées de sommes de termes ou de produits de facteurs (Figure 3). C'est aussi une classe simple du point de vue de la génération de l'exercice et du diagnostic. Le Tableau 3.a présente les informations concernant l'indexation didactique.

Indexation didactique	
Objectifs	Vérifier si un élève reconnaît la structure d'une expression littérale (somme ou produit) et identifie le rôle attribué aux signes +, × et aux exposants
Composante	Traduire algébriquement dans différentes représentations (algébrique, arithmétique et langage naturel)
Capacité	Traduire une expression algébrique en langage naturel et vice-versa
Type de tâche	Associer une forme à une expression algébrique
Critères d'évaluation	V1, V2, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V3 : incorrect

Tableau 3.a : Caractérisation de la classe « Reconnaître des sommes et des produits »

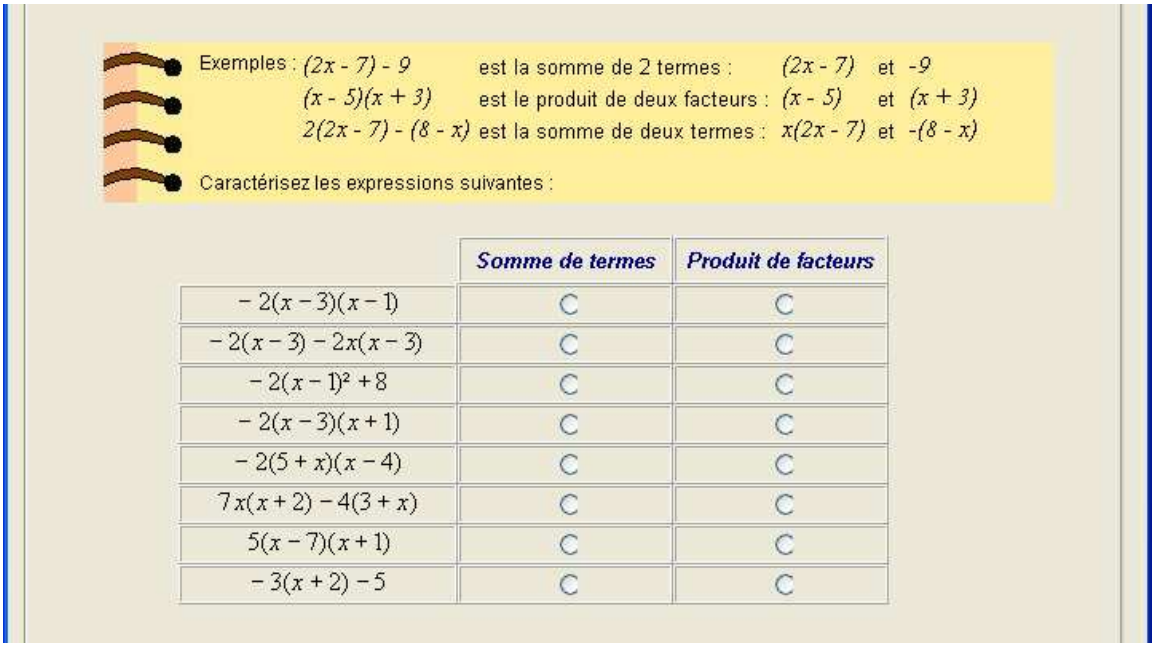


Figure 3 : Exercice originel de la classe « Reconnaître des sommes et des produits »

4.1. Génération de l'exercice

En remplaçant les valeurs numériques proposées dans l'exercice originel par des variables, nous obtenons des expressions paramétrées fixes où les paramètres prennent un nombre limité de valeurs. Par exemple, l'expression $-2(x - 3)(x - 1)$ est produite par l'expression $a(x + b)(x + c)$. Ces expressions sont les invariants de l'exercice et les lettres a, b et c sont les *paramètres*. Ainsi, pour définir des clones de cet exercice, il suffit de faire varier la valeur des paramètres, la position q_i des questions dans l'interface et la position p_i des deux colonnes « somme » et « produit ». Ces paramètres sont soumis à des *contraintes*. Les valeurs de *a*, *b*, *c*, *d* vérifient les conditions suivantes : $\forall\, a \in Z, b \in Z, c \in Z, d \in Z,$

$-9 \leq a \leq 9$ et $a \neq 0$, $-9 \leq b \leq 9$ et $b \neq 0$, $-9 \leq c \leq 9$ et $c \neq 0$, $-9 \leq d \leq 9$ et $d \neq 0$

Le Tableau 3.b présente les invariants et les paramètres de cette classe et le Tableau 4 présente les valeurs des paramètres pour deux représentants de cette classe qui illustrent notre discours.

Les paramètres *a*, *b*, *c* de l'exercice sont très contraints et prennent un nombre limité de valeurs (dix huit valeurs). Nous avons choisi de générer ces paramètres de façon aléatoire.

Interface		
- Un énoncé dont le texte est invariant - Huit questions avec deux choix exclusifs (question fermée) : somme ou produit		
Génération des énoncés		
Invariants	Q_1 : $a(x + b)(x + c)$	Q_5 : $a(b + x)(x + c)$
	Q_2 : $a(x + b) + c\,x(x + b)$	Q_6 : $a\,x(x + b) + c(d + x)$
	Q_3 : $a(x + b)^2 + c$	Q_7 : $a(x + b)(x + c)$
	Q_4 : $a(x + b)(x + c)$	Q_8 : $a(x + b) + c$
Paramètres	- Les variables *a*, *b*, *c*, *d* - La position P_i des questions et celle des 2 colonnes de choix	
Contraintes	Les valeurs de a, b, c, d vérifient les conditions suivantes : $a \in Z, b \in Z, c \in Z, d \in Z,$ $-9 \leq a \leq 9$ et $a \neq 0$, $-9 \leq b \leq 9$ et $b \neq 0$, $-9 \leq c \leq 9$ et $c \neq 0$, $-9 \leq d \leq 9$ et $d \neq 0$	
Type de génération	- Pour chaque question, génération automatique des valeurs de *a*, *b*, *n* et *p* - Génération automatique des positions q_i des questions et de la position p_i des deux colonnes correspondant au « Choix somme » et au « Choix produit » (aléatoire)	

Tableau 3.b : Caractérisation de la classe « Reconnaître des sommes et des produits »

4.2. Génération du diagnostic

Les *réponses anticipées* sont les deux choix proposés pour chacune des questions. La *grille de codage* élaborée par les didacticiens associe à chaque réponse un *code* qui caractérise la validité de la réponse. Par exemple, le choix de cocher l'option « Somme de termes » qui correspond à

l'expression $-2(x - 3)(x - 1)$ est associé au code V3. Pour cette classe d'exercices, la grille de codage est indépendante du clone car les expressions qui correspondent aux différents choix sont des invariants de la classe d'exercices.

Génération du diagnostic local					
	Choix somme	Choix produit		Choix somme	Choix produit
Q_1 : $a\,(x + b)\,(x + c)$	V3	V1	Q_5 : $a\,(b + x)\,(x + c)$	V3	V1
Q_2 : $a\,(x + b) + c\,x\,(x + b)$	V1	V3	Q_6 : $a\,x\,(x + b) + c\,(d + x)$	V1	V3
Q_3 : $a\,(x + b)^2 + c$	V1	V3	Q_7 : $a\,(x + b)\,(x + c)$	V3	V1
Q_4 : $a\,(x + b)\,(x + c)$	V3	V1	Q_8 : $a\,(x + b) + c$	V1	V3
Type de génération	Génération automatique du diagnostic : la grille de codage est invariante quelque soit l'exercice généré				

Tableau 3.c : Caractérisation de la classe « Reconnaître des sommes et des produits »

	Question	***a***	***b***	***c***	***d***
Exercice originel (Figure 3)	1	2	3	1	
	2	2	3		
	3	2	1	8	
	4	2	3	1	
	5	2	5	4	
	6	7	2	4	3
	7	5	7	1	
	8	3	2	5	

Tableau 4 : Paramètres *a*, *b*, *c*, *d*

5. Discussion

Les deux classes d'exercices que nous avons étudiées dans les sections 3 et 4 sont des questions fermées à choix multiples ou exclusifs dont les choix correspondent à des expressions numériques ou algébriques instanciées par des paramètres qui sont des variables sur lesquelles pèsent des contraintes. Ainsi, ces paramètres, fortement contraints, peuvent être générés automatiquement de manière aléatoire. Pour la génération de clones de ces deux classes d'exercices, nous nous sommes conformés au standard QTI. Un prototype PepiQTI a été développé dans le cadre du stage de master de recherche de Jérémy Leblanc. En ce qui concerne la génération du diagnostic, seule la validité de la réponse est analysée. Comme cette spécification n'autorise pas l'évaluation multidimensionnelle des réponses, le nombre de classes d'exercices de PépiTest qui peuvent être décrites complètement avec cette spécification est réduit. Ainsi, seul l'exercice « Reconnaître des sommes et des produits » peut être complètement

décrit avec les spécifications QTI car l'analyse de la réponse utilise seulement une dimension, la validité, avec deux valeurs (correcte ou incorrecte).

6. Classe d'exercices « Correspondance entre aire et expression »

Cette classe d'exercices s'appuie sur l'exploitation d'une figure géométrique. La Figure 4 présente l'interface élève de l'exercice originel de cette classe et la Figure 5 présente les deux réponses correctes.

6.1. Indexation didactique

Les exercices de cette classe ont pour objectif de savoir si un élève est capable d'associer une expression algébrique à un domaine plan qui a pour aire cette expression. Ils appartiennent à la composante « *Traduire algébriquement dans différentes représentations (géométrique, graphique, langage naturel)* » de la compétence algébrique et mettent en jeu la capacité « *Traduire une expression algébrique comme une aire d'une surface* ». La tâche consiste à cliquer sur les zones de la figure dont la somme des aires correspond à l'expression donnée. Ainsi, cette classe d'exercices propose une réponse préformatée (choix de une ou plusieurs zones) et correspond comme les précédentes à une question fermée (Tableau 5.a).

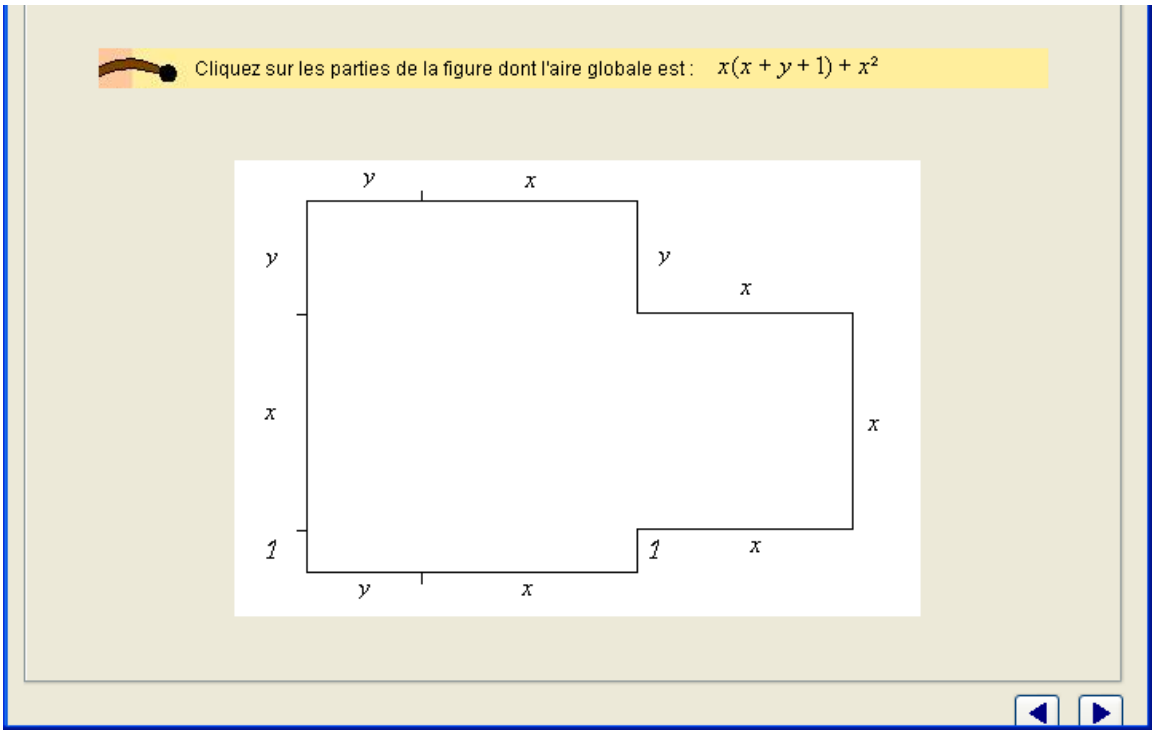


Figure 4 : Interface de l'exercice originel de la classe « Correspondance entre aire et expression »

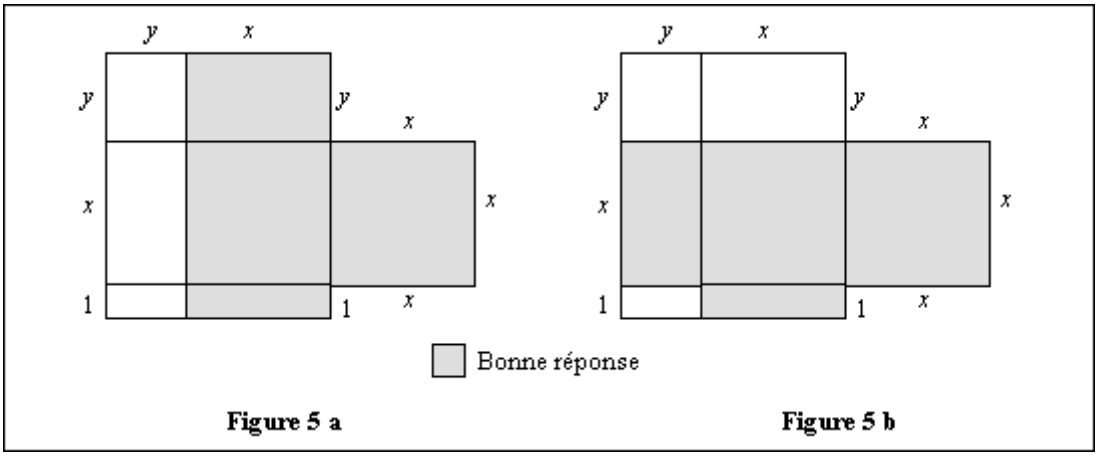


Figure 5 : Réponses correctes de l'exercice originel

Indexation didactique	
Objectifs	Rechercher si un élève sait associer une expression algébrique à un domaine plan qui a pour aire cette expression
Composante	Traduire algébriquement dans différents cadres (géométrique, arithmétique et langage naturel)
Capacité	Traduire une expression algébrique comme une aire d'une surface
Type de tâche	- Calculer l'aire d'un domaine plan - Associer une expression algébrique à l'aire d'un domaine plan
Critères d'évaluation	V1, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V3 : incorrect
	T1, T3 : critères d'évaluation de la dimension « Traduction » T1 : Traduction correcte T2 : Traduction incorrecte

Tableau 5.a : Caractérisation de la classe « Correspondance entre aire et expression »

6.2. Génération de l'exercice

Étudions les informations concernant l'interface élève et la génération de l'énoncé.

6.2.1. Interface élève

Du point de vue abstrait, l'interface est constituée d'un *énoncé* comportant un *texte* invariant, une *expression algébrique* variable et une *figure* géométrique variable. Les possibilités d'actions de l'élève sur le système sont le coloriage de l'aire correspondant à l'expression algébrique donnée dans l'énoncé en sélectionnant des zones cliquables.

Interface
- Un énoncé composé d'un texte invariant et d'une expression algébrique - Une figure géométrique F2 composée de zones cliquables (question fermée)

Tableau 5.b : Caractérisation de la classe « Correspondance entre aire et expression »

6.2.2. Génération de l'énoncé

L'expression algébrique donnée dans l'énoncé de l'exercice originel correspond à l'aire d'une figure ou de plusieurs figures géométriques F1, construites à l'intérieur d'une figure F2. La figure F2 est constituée d'un ensemble de zones cliquables par les élèves (les rectangles et les carrés représentés dans la Figure 6). Pour donner à l'auteur la possibilité de créer différents exercices, la figure F2 est à l'intérieur d'une troisième figure F3 qui sert de cadre. La figure F3 est un *invariant* de la classe d'exercices. Les figures F1, F2 et F3 sont composées de rectangles et de carrés. A l'intérieur de la figure F3 différents choix sont possibles pour obtenir d'une part la figure F2 et d'autre part la figure F1 et l'aire associée. La figure F2, la figure F1 incluse dans la figure F2 ainsi que l'expression algébrique de l'aire de F1 sont des *paramètres* de l'exercice.

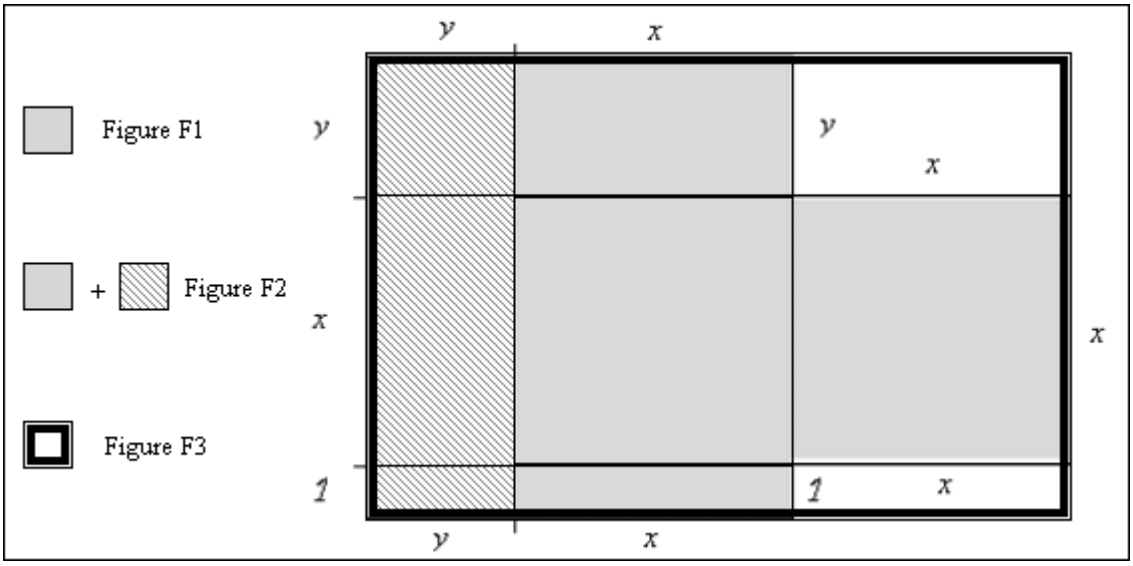


Figure 6 : Figures F1, F2 et F3

La complexité de la question dépend de la nature des figures géométriques qui constituent F1 et de l'expression algébrique qui exprime son aire. Les *contraintes* définies par les didacticiens portent sur la nature des figures qui composent F1, F2 et F3 (carrés ou rectangles contigus), les noms des variables utilisées pour repérer les côtés d'une figure géométrique (x et y ou a et b ou u et v ou m et n) et le choix d'utiliser seulement deux lettres et une valeur numérique pour repérer les côtés des rectangles ou des carrés (Tableau 5.c).

Le Tableau 6 présente les valeurs des paramètres pour un clone de cette classe qui illustre notre discours (Figure 4).

Génération de l'énoncé	
Invariants	Figure F3 : le cadre
Paramètres	- Deux figures F1, F2 - Deux variables l_1, l_2 et un nombre c - Expression algébrique égale à l'aire de F1
Contraintes	- Les figures F1 et F2 sont composées de rectangles et de carrés contigus - F2 ⊂ F3 et F1 ⊂ F2 - Les noms des côtés sont repérés par deux lettres consécutives l : l ∈ [*a*, *b*, *x*, *y*, *m*, *n*, *u*, *v*] - Un coté d'une des figures qui compose F1 a une valeur numérique c : $1 \leq c \leq 9$
Type de génération	Génération assistée - Choix des figures F2 et F1 par l'auteur - Génération automatique des différentes expressions de l'aire de F1 (utilisation du module de calcul formel Pépinière) - Choix d'une expression de l'aire de F1 par l'auteur

Tableau 5.c : Caractérisation de la classe « Correspondance entre aire et expression »

6.3. Génération du diagnostic

Les *réponses anticipées* correctes sont toutes les façons de sélectionner les rectangles et les carrés pour obtenir l'aire de la figure F1 (dans l'exercice originel deux figures ont pour aire l'expression donnée dans l'énoncé (Figure 5), les autres sélections étant les réponses erronées. La *grille de codage* est limitée à deux *codes* : V1T1 (validation et traduction correcte) si l'aire des zones qui ont été sélectionnées est correcte et V3T3 (validation et traduction incorrecte) dans le cas contraire. La difficulté de la génération du diagnostic résulte du fait que plusieurs figures peuvent avoir la même aire : l'auteur ayant choisi une figure F1, le système doit calculer l'aire de F1 puis déterminer quelles sont les figures situées à l'intérieur de F2 qui ont une aire égale à celle de F1. L'ensemble des informations concernant la génération d'un clone de cette classe d'exercices est présenté dans le Tableau 5.c.

Génération du diagnostic local		
Solution correcte	Sélection de zones dont la somme des aires est égale à l'expression	V1, T1
Solution incorrecte	Sélection d'autres zones	V3, T3
Type de génération	Génération automatique de la grille de codage après détermination par PépiGen des figures incluses dans F2 dont l'aire est égale à celle de F1	

Tableau 5.d : Caractérisation de la classe « Correspondance entre aire et expression »

6.4. Type de génération

La génération de clones de cette classe d'exercices est associée d'une part à la génération de la figure F2 dans laquelle est incluse la figure F1, et, d'autre part, au choix de l'expression

algébrique souhaitée pour exprimer l'aire de la figure F1. Une figure ayant été choisie par un auteur, plusieurs expressions sont possibles pour exprimer son aire. Pour l'exercice originel, les expressions algébriques $x(y+x+1) + x^2$ (Figure 5a) ou $x(y+2x) + x^2$ (Figure 5b) ou encore l'expression obtenue en faisant la somme des aires des rectangles et des carrés qui constituent la figure F1, soit $x\,y + 2\,x^2 + x$, correspondent à l'aire de la figure F1. Les enseignants de l'équipe ont choisi l'expression algébrique $x\,(y+x+1) + x^2$ car, parmi les expressions algébriques que l'on peut construire à partir de l'expression de l'aire de la figure F1, certaines ne correspondent pas aux objectifs fixés pour cet exercice ou apparaissent comme peu probables à ce niveau scolaire.

Pour construire ces expressions, nous proposons une génération assistée : (i) l'auteur choisit dans un cadre (F3) formé de rectangles et de carrés contiguës la figure F2, puis la figure F1 incluse dans F2, (ii) PépiGen calcule différentes expressions algébriques équivalentes à l'expression algébrique exprimant l'aire de F1 et les propose à l'auteur (iii) qui choisit l'une de ces expressions. Pour le diagnostic, PépiGen détermine toutes les figures incluses dans F2 dont l'aire est aussi égale à celle de F1. Pour cette classe d'exercices la grille de codage générée est encore indépendante du clone

Exercice originel (Figure 4)	**F1**	Figure
	F2	Figure
	l_1	x
	l_2	y
	c	1
	Aire associée à la figure de F1	$x(y+x+1) + x^2$

Tableau 6 Valeurs des paramètres

7. Classe d'exercices « Expression littérale de l'aire d'un rectangle »

Un autre exercice de PépiTest porte sur la correspondance entre aire et expression. Il comporte deux parties. Les figure 7.a et 7.b présentent l'interface élève de l'exercice originel de la classe d'exercices « Expression littérale de l'aire d'un rectangle ». Pour cette classe d'exercices les informations concernant les objectifs, la composante, les capacités et le type de tâche sont identiques à celles de la classe d'exercices « Correspondance entre aire et expression », mais les tâches proposées à l'élève s'en distinguent. Pour la première partie, la tâche consiste à calculer l'aire d'une figure, pour la deuxième partie la tâche de type QCM a pour objet de reconnaître différentes formes de l'expression algébrique correspondant à l'aire demandée dans la première partie.

Indexation didactique	
Objectifs	Rechercher si un élève sait associer une expression algébrique à un domaine plan qui a pour aire cette expression
Composante	Traduire algébriquement dans différentes représentations (algébrique, arithmétique et langage naturel)
Capacité	Traduire une expression algébrique comme une aire d'une surface
Type de tâche	- Calculer l'aire d'un domaine plan - Associer une expression algébrique à l'aire d'un domaine plan
Critères de validation	V1, V2, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V2 : correct partiel ou non attendu V3 : incorrect
	EA1, EA31, EA33, EA41, EA42 : critères d'évaluation de la dimension « Utilisation des règles d'Ecriture et de réécriture Algébrique » EA1 : Utilisation correcte des règles de transformation EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct EA33 : Utilisation de règles de transformation fausses identifiées EA41 : Les règles de transformation utilisées linéarisent les expressions : EA42 : Les règles de transformation utilisées « assemblent » les termes
	T1, T3, T4 : critères d'évaluation de la dimension « Traduction » T1 : Traduction correcte T3 : Traduction incorrecte T4 : Traduction abréviative

Tableau 7.a: Caractérisation de la classe « Expression littérale de l'aire d'un rectangle »

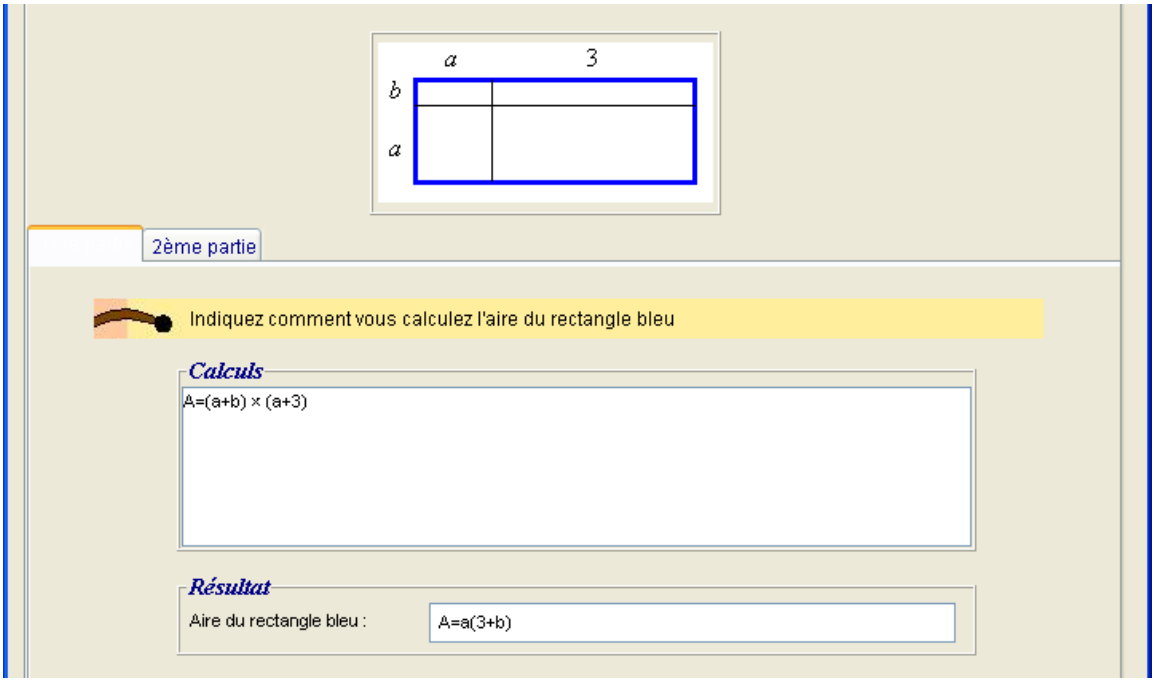


Figure 7.a : Interface de l'exercice originel de la classe
« Expression littérale de l'aire d'un rectangle » (partie 1)

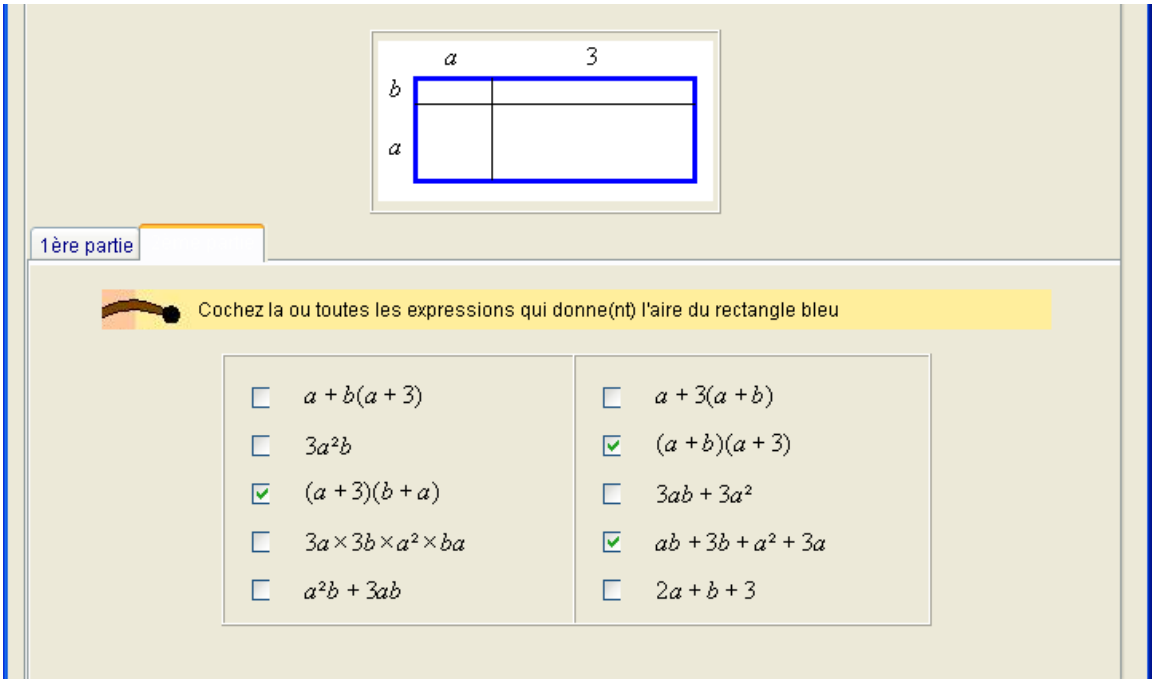


Figure 7.b : Interface de l'exercice originel de la classe
« Expression littérale de l'aire d'un rectangle » (partie 2)

7.1. Génération de l'exercice

La génération des deux parties de l'exercice s'appuie sur une figure, le rectangle R de côté (a+3)(b+a).

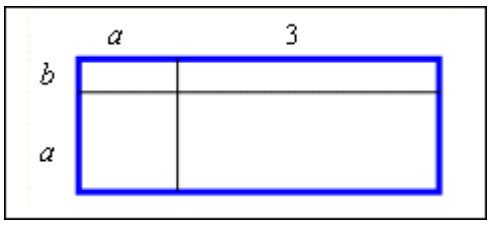


Figure 8 : Rectangle R (traits épais)

7.1.1. Interface élève

L'interface élève est composée d'une *figure*, un rectangle découpé en quatre rectangles. Pour la première *partie*, l'interface comporte une *question ouverte*, et pour la deuxième *partie* une question à *choix multiples*.

7.1.2. Génération des énoncés

7.1.2.1. Génération de la figure

L'aire du rectangle R est déterminée par la donnée des côtés des rectangles qui le composent (Figure 8). Le rectangle R, est un *invarian*t de l'exercice. Pour les deux parties de l'énoncé, les contraintes imposées par les didacticiens et les enseignants portent sur le choix d'utiliser seulement deux lettres (x et y ou a et b ou u et v ou m et n) et une valeur numérique entière strictement comprise entre 1 et 10 pour repérer les côtés des rectangles qui composent le rectangle R. Les différentes façons de repérer les côtés des quatre rectangles qui composent le rectangle R, par deux variables et un nombre, génèrent des clones d'exercices de la classe (Figure 9). Ainsi, les variables utilisées et la valeur numérique sont les *paramètres* de l'exercice.

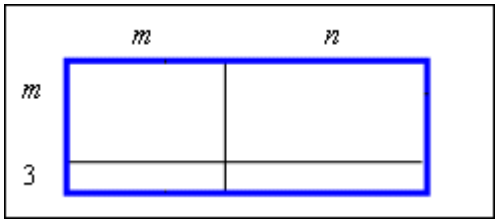


Figure 9 : Une autre configuration du rectangle R

La figure proposée pourrait être une figure quelconque composée de rectangles ce qui nous ramènerait à la classe d'exercices présentée dans la section 6, cependant la complexité des expressions algébriques obtenues avec cette figure pour la deuxième partie de l'exercice ne correspondrait pas aux capacités exigibles en algèbre au niveau scolaire pour lequel cet exercice est prévu.

7.1.2.2. Génération de la deuxième partie de l'exercice

La génération des questions de la deuxième partie de cette classe d'exercices prend en compte les réponses correctes ou erronées observées plus particulièrement dans les tâches associant différentes formes d'une expression algébrique à l'aire d'un domaine plan. Pour la deuxième partie, les réponses à proposer sont des expressions algébriques obtenues à partir de la formule générale de l'aire d'un rectangle (Longueur × Largeur) ou en appliquant des propriétés des opérations arithmétiques comme la commutativité (e.g. $(a+3)(b+a)$ et $(a+b)(a+3)$), des règles correctes comme la somme des aires des différentes sous-figures (e.g. $ab+a^2+3b+3a$) ou erronées comme la reconnaissance des figures qui composent le rectangle R avec des erreurs dans la traduction en une expression algébrique (e.g. $a+3(b+a)$). Une des réponses est l'expression du périmètre à la place de celle de l'aire (e.g. $2a+b+3$). Pour obtenir ces expressions algébriques, six

modes opératoires ont été identifiés (Tableau 7.b). Ces modes opératoires sont les *invariants* de la deuxième partie de l'exercice.

Pour des élèves de niveau cinquième ou quatrième, les expressions algébriques peuvent être obtenues à partir des périmètres.

Interface	
- Énoncé constitué d'une figure géométrique - Partie 1 : une question ouverte avec un énoncé dont le texte est invariant et deux zones de saisie : une pour les calculs et une pour le résultat (expressions algébriques) . - Partie 2 : une question à choix multiples (dix choix dont les contenus sont des expressions algébriques générées à partir des six modes d'obtention décrits dans les invariants)	
Génération de l'énoncé	
Invariant partie 1 et 2	- Une figure : le rectangle R
Invariants (partie 2 seulement)	- Modes d'obtention des types de choix Type 1 : expressions correctes de l'aire sous la forme d'un produit Longueur × Largeur Type 2 : expression de l'aire obtenue en additionnant les aires des différents rectangles Type 3 : reconnaissance de sous-figures mais erreur de traduction au niveau des parenthèses (e.g. $a+3 \times a+b$) Type 4 : confusion entre aire et périmètre Type 5 : traduction par assemblage des termes (e.g. $a^2 b + 3 ab$) Type 6 : traduction de l'aire avec confusion entre les opérateurs + et × (e.g. $3a \times 3b \times a^2 \times ba$)
Paramètres	- Deux lettres (l_1, l_2) et un nombre c - La position des types de règles à l'interface (partie 2) - Dix expressions générées à partie des 6 types de règles
Contraintes	- Les noms des côtés sont repérés par deux lettres consécutives l : l ∈ [*a*, *b*, *x*, *y*, *m*, *n*, *u*, *v*] - Un coté d'une des figures qui compose le rectangle bleu a une valeur numérique c : $1 \leq c \leq 9$
Type de génération	- Génération automatique de deux lettres et d'une valeur numérique - Génération automatique des dix expressions à partir des invariants (partie 2) - Génération automatique de la position des dix expressions

Tableau7.b : Caractérisation de la classe « Expression littérale de l'aire d'un rectangle »

7.1.3. Génération du diagnostic

7.1.3.1. Première partie de l'exercice

Pour la première partie, les réponses anticipées sont les expressions correctes ou erronées obtenues pour le calcul de l'aire : les choix proposés dans la question à *choix multiples* (QCM) de la deuxième partie sont les *réponses anticipées* de la première partie. Une partie de la *grille de codage* est donc identique à celle de la deuxième partie. Pour cette première partie, les codages proposés ne permettent pas de coder toutes les réponses possibles, mais ils permettent d'analyser

les réponses observées dans le corpus de réponses d'élèves obtenu dans le cadre du projet Pépite (e.g. confusion entre aire et périmètre, reconnaissance des figures qui composent le rectangle R avec une erreur d'écriture dans l'expression algébrique correspondante).

7.1.3.2. Deuxième partie de l'exercice

Nous commençons par la génération du diagnostic pour la deuxième partie de l'exercice car les *réponses anticipées* sont les choix proposés. La *grille de codage* élaborée par les didacticiens associe à chaque choix un *code* qui caractérise le mode d'obtention de la réponse : règle correcte ou erronée mise en œuvre, confusion entre aire et périmètre par exemple.

7.1.4. Type de génération

Remarquons d'abord qu'une des difficultés à surmonter pour générer cet exercice est le calcul des expressions correctes ou erronées qui sont obtenues à partir de l'expression de l'aire du rectangle, et, qui constituent à la fois les choix proposés dans le QCM et les réponses anticipées pour la génération du diagnostic de la première partie. Le choix des lettres utilisées ainsi que les différentes possibilités de nommer les côtés des rectangles étant limitées, nous utilisons une génération automatique : (i) le système PépiGen génère de façon automatique deux lettres l_1 et l_2 et un valeur numérique, (ii) les expressions algébriques proposées dans la question à choix multiples de la deuxième partie selon les modes d'obtention décrits dans le Tableau 7.b, (iii) l'auteur choisit, parmi ces expressions, celles qui compose le QCM, (iv) la grille de codage est obtenue à partir des expressions générées pour la deuxième partie car les codes sont caractérisés par le mode d'obtention de l'expression (Cf. 7.1.3).

Génération du diagnostic local		
Solution correcte	Partie 1 : deux démarches sont attendues	
	Réponse de type 1	V1, T1
	Réponses de type 2	V2, T1
	Calcul correct pour obtenir une autre forme de l'expression	V1, T1, EA1
	Partie 2 : Réponses de type 1 ou type 2	V1T1
Solution incorrecte Partie 1 et partie 2	Réponse de type 3	V3, EA31, T3
	Réponse de type 5	V3, EA42, T4
	Réponse de type 6	V3, EA41, T3
	Réponse de type 4	V3, EA41, T3
	Formule incorrecte ou partielle	V3, EA33, T3
Type de génération	Génération automatique du diagnostic : la grille de codage est générée au cours de la génération des expressions algébriques proposées dans la question à choix multiples de la partie 2	

Tableau 7.c : Caractérisation de la classe « Expression littérale de l'aire d'un rectangle »

Le Tableau 8 présente les valeurs des paramètres pour le représentant de cette classe qui illustre notre discours.

<table>
<tr><td rowspan="3">Partie 1 et 2</td><td>l_1</td><td>a</td></tr>
<tr><td>l_2</td><td>b</td></tr>
<tr><td>c</td><td>3</td></tr>
<tr><td rowspan="10">Partie 2 : type de l'expression
☐ $a+b(a+3)$ ☐ $a+3(a+b)$
☐ $3a^2b$ ☐ $(a+b)(a+3)$
☐ $(a+3)(b+a)$ ☐ $3ab+3a^2$
☐ $3a\times3b\times a^2\times ba$ ☐ $ab+3b+a^2+3a$
☐ a^2b+3ab ☐ $2a+b+3$</td><td>type 3</td><td>$a + b(a+3)$</td></tr>
<tr><td>type 5</td><td>$3\ a^2b$</td></tr>
<tr><td>type 1</td><td>(a+3) (b+a)</td></tr>
<tr><td>type 6</td><td>$3a \times 3b \times a^2 \times ba$</td></tr>
<tr><td>type 5</td><td>$a^2b + 3\ ab$</td></tr>
<tr><td>type 3</td><td>a + 3(a+b)</td></tr>
<tr><td>type 1</td><td>(a+b)(a+3)</td></tr>
<tr><td>type 5</td><td>$3\ ab + 3\ a^2$</td></tr>
<tr><td>type 2</td><td>$ab + 3b + a^2 + 3a$</td></tr>
<tr><td>type 4</td><td>2a + b + 3</td></tr>
</table>

Tableau 8 : Valeurs des paramètres

7.2. Discussion

Nous avons choisi de générer de façon automatique les paramètres l_1, l_2 et c car ces paramètres sont fortement contraints (Tableau 7.b). Le choix des expressions algébriques qui constituent la deuxième partie a pour objectifs d'identifier les règles de traduction, correctes ou erronées, utilisées par les élève pour passer d'une représentation géométrique à une représentation algébrique. Les expressions algébriques sont générées selon les règles que nous avons établies (Tableau 7.b) pour obtenir des clones équivalents du point de vue du diagnostic cognitif. Cependant, à une même règle correspondent plusieurs expressions (e.g. pour la règle de type 5 : 3 $a^2\,b$ ou $a^2\,b + 3ab$ ou $3ab + 3a^2$). Pour générer la deuxième partie du clone, prenant en compte le contexte dans lequel cet exercice est proposé aux élèves, l'auteur choisi dix expressions parmi celles proposées par PépiGen.

8. Classe d'exercices « Preuve et programme de calcul »

Contrairement aux classes précédentes, cette classe d'exercices comporte l'analyse d'une réponse ouverte sous la forme d'un raisonnement algébrique assez complexe.

La figure 10 présente l'interface élève de l'exercice originel de cette classe d'exercices dont les objectifs sont d'étudier :

- le type de preuve utilisée (preuve par l'exemple ou preuve algébrique) pour prouver qu'un énoncé est vrai.
- le type de traitement algébrique mobilisé (utilisation d'une expression globale parenthésée prenant en compte l'enchaînement des opérations ou calculs pas à pas faisant apparaître les résultats intermédiaires),
- la gestion de l'articulation entre deux représentations (algébrique et langage naturel),
- la signification accordée au signe d'égalité,
- la maîtrise du calcul algébrique.

Cette classe est donc particulièrement riche du point de vue didactique et les objectifs s'inscrivent dans les nouveaux programmes, où, dès la cinquième, les élèves sont amenés à « *Produire une expression littérale, utiliser une expression littérale* » [BO 2007].

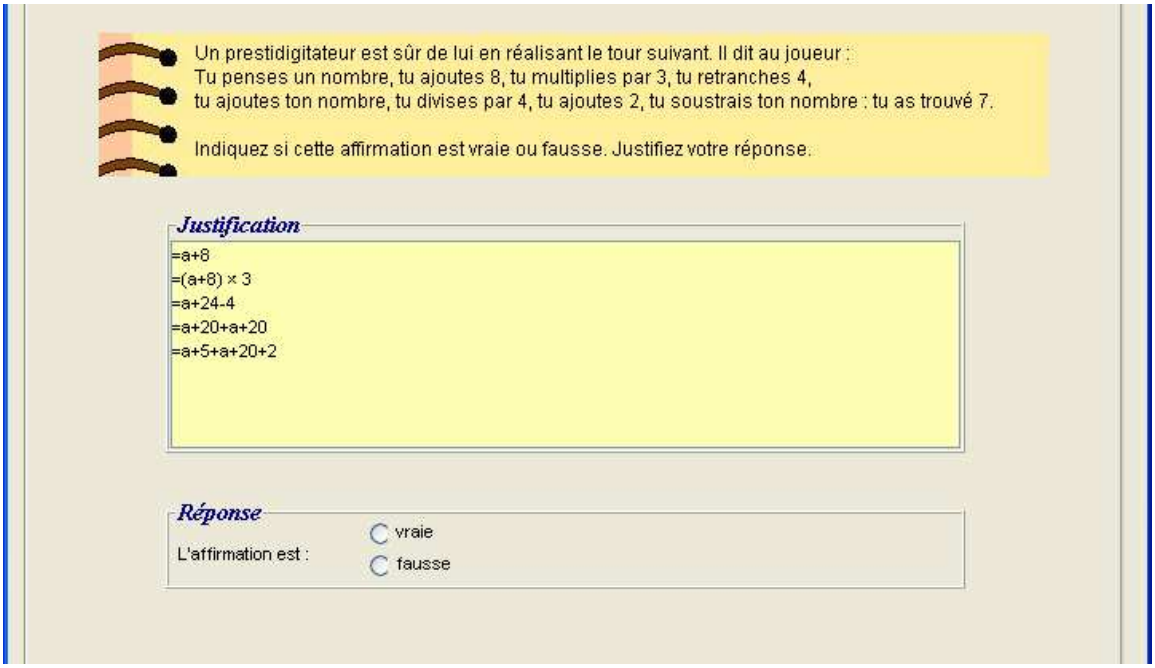


Figure 10 : Interface de l'exercice originel de la classe « Preuve et programme de calcul »

Cette classe appartient aux trois composantes « *Effectuer du calcul algébrique* », « *Utiliser l'algèbre pour résoudre des problèmes* », « *Traduire algébriquement dans différents représentations* (*géométrique, graphique, langage naturel*) » et mobilise plusieurs capacités : « *Réduire une expression littérale* » pour la première composante, « *Produire une expression littérale* » et « *Prouver des propriétés numériques* » pour la seconde, « *Traduire un programme de calcul en langage naturel en une expression algébrique* » pour la troisième. Le type de tâches consiste à produire une expression littérale à partir d'un programme de calcul énoncé en langage naturel et prouver qu'une affirmation est vraie.

Indexation didactique	
Objectifs	Etudier - le type de preuve utilisée (preuve par l'exemple ou preuve algébrique) pour prouver qu'un énoncé est vrai - le type de traitement algébrique mobilisé (utilisation d'une expression globale parenthésée ou calculs pas à pas faisant apparaître les résultats intermédiaires) - la gestion de l'articulation entre différentes représentationss (algébrique, arithmétique et langage naturel) - la signification accordée au signe d'égalité
Composantes	1 - Effectuer du calcul algébrique 2 - Utiliser l'algèbre pour résoudre des problèmes 3 - Traduire algébriquement dans différents cadres (géométrique, graphique, langage naturel)
Capacités	- Composante 1 : Réduire une expression littérale - Composante 2 : Produire une expression littérale, démontrer des règles de calcul, des propriétés, des identités - Composante 3 : Traduire un programme de calcul en langage naturel en une expression algébrique » pour la troisième
Type de tâche	- Produire une expression littérale à partir d'un programme de calcul - Prouver qu'une affirmation est vraie
Critères d'évaluation	V1, V2, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V2 : correct partiel ou non attendu V3 : incorrect
	L1, L2,L3, L5 : critères d'évaluation de la dimension « Utilisation des lettres » L1 : Utilisation correcte des lettres L2 : Utilisation des lettres pour leur substituer des valeurs numériques L3 : Utilisation des lettres pour faire du calcul algébrique avec des règles fausses L5 : Aucune utilisation des lettres
	EA1, EA2, EA31, EA32, EA33, EA41, EA42 : critères d'évaluation de la dimension « Utilisation des règles d'Ecriture et de réécriture Algébrique » EA1 : Utilisation correcte des règles de transformation EA2 : Maîtrise technique fragile EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct EA32 : Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect EA33 : Utilisation de règles de transformation fausses identifiées EA41 : Les règles de transformation utilisées linéarisent les expressions : EA42 : Les règles de transformation utilisées « assemblent » les termes
	T1, T2, T3, T4 : critères d'évaluation de la dimension « Traduction » T1 : Traduction correcte T3 : Traduction incorrecte T2 : Traduction correcte non attendue T4 : Traduction abréviative
	J1, J2, J3 : critères d'évaluation de la dimension « Type de justification » J1 : Justification par l'algèbre J3 : Justification de type scolaire J2 : Justification par l'exemple numérique

Tableau 9.a : Caractérisation de la classe d'exercices « Preuve et programme de calcul »

8.1. Génération de l'exercice

8.1.1. Interface élève

Du point de vue abstrait, l'interface comporte un *énoncé* constitué d'un programme de calcul en langue naturelle, d'une *question à choix exclusif* (« Indique si cette affirmation est vraie ou fausse. ») et d'une *question ouverte* (« Justifie ta réponse »). Pour définir des clones de cet exercice, il suffit de faire varier le programme de calcul.

8.1.2. Génération de l'énoncé

Prenons deux exemples de représentants de cette classe. Le premier est l'exercice originel de PépiTest et l'autre est posé au niveau quatrième.

> Enoncé 1 : Un prestidigitateur est sûr de lui en réalisant le tour suivant. Il dit au joueur : « Tu prends un nombre, tu ajoutes 8, tu multiplies par 3, tu retranches 4, tu ajoutes ton nombre, tu divises par 4, tu ajoutes 2, tu soustrais ton nombre : tu trouves 7. Prouve-le ».
>
> Enoncé 2 : « Tu prends un nombre, tu ajoutes 6 à ce nombre, tu multiples le résultat par 3, tu soustrais 3 fois le nombre de départ : tu trouves 18. Prouve-le ».

Après une présentation du contexte « *Un prestidigitateur est sûr de lui en réalisant le tour suivant. Il dit au joueur* » pour l'énoncé 1, les deux énoncés proposés sont des programmes de calcul exprimés en langage naturel. Les termes utilisés permettent de construire une expression algébrique à une variable : par exemple « *tu prends un nombre* », « *tu multiplies* », « *tu ajoutes* ». Ces termes ont été choisis par les didacticiens des mathématiques pour un niveau de collège ou de seconde. Nous avons choisi de regrouper ces termes dans une palette (Figure 11) qui est utilisée par l'auteur pour écrire le programme de calcul. Les termes de cette palette sont les *invariants* de l'exercice. Ils définissent le vocabulaire de la grammaire qui permet d'engendrer le programme de calcul. Nous reviendrons sur le choix d'une palette dans la section 8.3. Les deux clones d'exercice sont caractérisés par un programme de calcul qui peut être exprimé dans deux registres, celui du langage naturel, (e.g. « *Tu prends un nombre, tu ajoutes* 8, *tu multiplies par* 3 ») et celui de l'algèbre (e.g. $(x+8)\times 3$), et, par l'expression réduite qui est une constante (7 pour l'énoncé 1, 18 pour l'énoncé 2). Le programme de calcul exprimé dans deux registres (langage naturel et algébrique) ainsi que l'expression réduite sont les *paramètres* de l'exercice (Tableau 9.b). A ces paramètres sont associées des *contraintes* qui caractérisent la complexité de l'exercice (niveau de parenthèses, nombre et type d'opérations), le type d'expression réduite (expression constante ou expression affine) qui dépendent du niveau scolaire ($5^{ième}$, $4^{ième}$, $3^{ième}$ ou 2^{de}) .

Interface	
- Une énoncé composé d'un programme de calcul exprimé en langage naturel, variable, d'une question fermée et d'une question ouverte dont les énoncés sont invariants. - Pour la question ouverte : une zone de saisie pour les justifications - Pour la question fermée : deux choix exclusifs Vrai - Faux	
Génération des questions	
Invariants	- L'ensemble des termes de la palette - Le texte « Indique si cette affirmation est vraie ou fausse. Justifie ta réponse » - Les deux choix de la question fermée
Paramètres	- Un programme de calcul exprimé dans deux registres : - langage naturel : texte - algèbre : expression algébrique
Contraintes	- Le degré de complexité de l'expression algébrique - nombres de parenthèses - nombre et type des opérations (addition, soustraction, multiplication, division) - L'expression réduite du programme de calcul est une constante ou une expression affine
Type de génération	Génération assistée : - L'auteur choisit les termes de la palette pour écrire le programme de calcul - PépiGen construit l'expression algébrique traduisant le programme de calcul et calcule la forme réduite de cette expression

Tableau 9.b : Caractérisation de la classe d'exercices « Preuve et programme de calcul »

8.2. Génération du diagnostic

Pour cette classe d'exercices, des analyses théoriques et empiriques sur des cohortes d'élèves nombreuses et diverses [Grugeon 1995] ont permis de spécifier les réponses anticipées qu'elles soient correctes, partielles ou erronées. Ces études, si elles ne permettent pas de garantir le codage de toutes les réponses possibles, permettent d'analyser les réponses observées dans notre corpus et de repérer des cohérences de fonctionnement des élèves en algèbre. Selon ces études, pour cette classe d'exercices, les élèves utilisent deux types de preuve : une preuve algébrique ou une preuve pragmatique (démarche arithmétique et justification par l'exemple). A chaque type de preuve envisagée, l'analyse didactique associe un type de démarche et pour chacune de ces démarches, les différentes expressions (algébriques ou numériques) obtenues en appliquant des *règles* de réécriture correctes ou erronées à l'expression qui traduit le programme de calcul donné dans l'énoncé. L'ensemble de ces *réponses anticipées* et le code associé est présenté dans les tableaux 9.c et 9.d. Dans la section 6, nous avons déjà évoqué la difficulté à produire des calculs algébriques. Pour cette classe d'exercices, les expressions algébriques produites dépendent du programme de calcul, donc de l'expression de départ : la difficulté est liée à la complexité de cette expression. Pour l'exercice originel, la génération du diagnostic présentée dans les tableaux 9.c et 9.d donne l'ampleur du travail à réaliser pour générer les réponses anticipées et le code

associé. Dans le chapitre 7 nous reviendrons sur la génération des réponses anticipées. Pour cette classe d'exercices, la *grille de codage* dépend de l'expression algébrique du programme de calcul qui est un des paramètres du clone.

8.3. Type de génération

Pour générer un programme de calcul, deux possibilités s'offrent à nous : utiliser le registre du langage naturel ou utiliser celui de l'algèbre en donnant l'expression algébrique. Nous avons choisi la première solution car ce registre est mis en oeuvre dans d'autres exercices de PépiTest et les études menées pour générer cet exercice sont réutilisables. Nous avons décidé d'employer une palette, car elle garantissait la production de programmes de calcul adaptés à différents niveaux de classe (quatrième, troisième ou début de seconde). Le choix des termes de la palette a été réalisé par les didacticiens de mathématiques et les enseignants de l'équipe. En ce qui concerne la génération des différentes réponses anticipées, les analyses menées dans cette section, ainsi que dans la section 6, montrent la nécessité d'un module de calcul formel pour produire les différentes étapes de calculs algébriques. Nous présentons ce module, Pépinière, dans le chapitre 6.

La génération de clones de cette classe d'exercices est donc associée à la génération du programme de calcul, de l'expression algébrique correspondante et du résultat du calcul. La génération est assistée : (i) l'auteur crée un programme de calcul en choisissant les termes à l'aide de la palette, (ii) PépiGen construit l'expression algébrique correspondant aux choix des termes et calcule le résultat final, (iii) pour le diagnostic, PépiGen génère l'ensemble des réponses anticipées, c'est-à-dire l'ensemble des solutions correctes, correctes mais non attendues, incorrectes présentées dans la grille de codage.

La grille de codage dépend de l'expression algébrique correspondant au programme de calcul qui est un des paramètres de l'exercice.

Génération du diagnostic local		
	Réponses et commentaires	**Code**
Solution correcte	Preuve algébrique avec expression globale (parenthésée) Pour les expressions comportant des fractions, deux tactiques de calcul sont envisageables : (1) on réduit les deux expressions au même dénominateur (2) la première expression est simplifiée	 V1, L1, EA1, T1, J1 V1, L1, EA2, T1, J1

Tableau 9.c : Caractérisation de la classe d'exercices « Preuve et programme de calcul »

Génération du diagnostic local (suite)		
Solution correcte non attendue	Preuve algébrique avec des calculs pas à pas	V2, L1, EA1 T2, J1
	Preuve algébrique avec l'interprétation de l'énoncé comme une équation admettant une infinité de solutions	V2, L1, T1, J1 EA1 : tactique (1) EA2 : tactique (2)
Solution partielle	Une preuve algébrique est engagée avec l'une des trois interprétations précédentes. Mais une erreur de calcul conduit à un résultat faux ou a un abandon ou à une égalité non justifiée	
	- Cas d'une expression globale	V2, L1, EA33 T1, J1
	- Cas d'expressions partielles	V2, L1, EA33 T2, J1
Solution incorrecte	Démarche algébrique	
	Écriture globale non parenthésée	V3, L3, T3, J
	Calculs pas à pas	V3, L3, T4, J3
	On affine le codage de la dimensions Calcul Algébrique (EA) pour les 3 cas avec deux possibilités :	
	Possibilité 1 : il y a manipulation formelle	
	- avec mémoire de l'énoncé (par ex, $4x + 20 \div 5 = x + 5$)	EA31
	- sans mémoire de l'énoncé (par ex, $4x + 20 \div 5 = 4x + 5$)	EA32
	- avec erreurs d'application d'une règle de calcul	EA33
	- pseudo-opératoire en assemblage	EA42
	par exemple : $23\,x + x = 23\,x^2$ ou $3x+24 = 27x$ ou $8x-x=7$	
	Possibilité 2 : il y n'a pas manipulation formelle mais substitution à x de valeurs numériques (preuve par l'exemple) :	V3, L2, T3, J2
	- avec mémoire de l'énoncé	EA31
	- sans mémoire de l'énoncé	EA32
	- avec erreurs d'application d'une règle de calcul	EA33
	Démarche non algébrique (preuve par l'exemple)	
	1) Les solutions mettent en jeu des écritures correctes	
	- expression globale parenthésée	V3, L5, EA1, T1, J2
	- expressions partielles	V3, L5, EA1, T2, J2
	2) les solutions mettent en jeu des écritures incorrectes.	
	- écriture pas à pas enchaînée en succession d'opérations	V3, L5, T4, J2
	- écriture linéaire globale non parenthésée	V3, L5, T3, J2
	- avec mémoire	EA31
	- sans mémoire	EA32
Type de génération	Génération automatique du diagnostic -PépiGen génère l'ensemble des réponses anticipées, c'est-à-dire l'ensemble des solutions correctes, correctes mais non attendues, incorrectes présentées dans la grille de codage - La grille de codage dépend de l'expression algébrique correspondant au programme de calcul	

Tableau 9.d : Caractérisation de la classe d'exercices « Preuve et programme de calcul »

Le Tableau 10 présente les valeurs des paramètres pour le représentant de cette classe qui illustre notre discours.

Exercice originel (Figure 10)	**Programme de calcul**	Langage naturel	Tu prends un nombre, tu ajoutes 8, tu multiplies par 3, tu retranches 4, tu ajoutes ton nombre, tu divises par 4, tu ajoutes 2, tu soustrais ton nombre : tu trouves 7. Prouve-le
		Expression	$((x+8)3-4+x)/4+2 - x$
	Expression réduite	7	
Clone	**Programme de calcul**	Langage naturel	Tu prends un nombre, tu ajoutes 6 à ce nombre, tu multiples le résultat par 3, tu soustrais 3 fois le nombre de départ : tu trouves 18. Prouve-le
		Expression	(x+6)3-3x
	Expression réduite	18	

Tableau 10 : Valeurs des paramètres pour deux représentants de la classe « Preuve et programme de calcul »

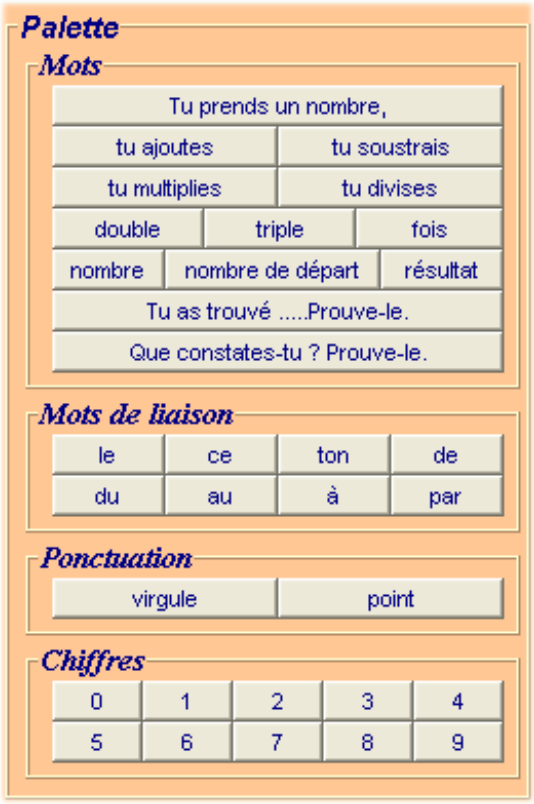


Figure 11 : Palette de termes de l'interface auteur pour la classe d'exercices « Preuve et programme de calcul »

9. Classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales »

Parmi les exercices à questions ouvertes de PépiTest, plusieurs classes d'exercices mettent en œuvre des transformations d'expressions algébriques dont certaines sont associées à l'utilisation d'identités remarquables appliquées aux polynômes du second degré. La Figure 12 présente l'interface élève d'une de ces classes d'exercices « Déterminer si des expressions algébriques du second degré sont égales ».

Cette classe d'exercices a pour objectif d'étudier si un élève attribue plusieurs formes à une expression algébrique et sait mobiliser la forme adaptée au problème rencontré. Elle appartient à la composante « *Effectuer du calcul algébrique* » et met en jeu les capacités « *développer une expression littérale de la forme* (*a*+*b*)(*c*+*d*) » et « *Utiliser les identités remarquables dans les deux sens, pour factoriser ou développer* ». Le type de tâches consiste à associer plusieurs formes à une expression algébrique, effectuer du calcul algébrique en mobilisant la forme de l'expression adaptée au calcul (Tableau 11.a). Un exercice de cette classe comporte trois tâches de type questions à choix exclusif avec des justifications de type réponse ouverte.

Indexation	
Objectifs	Étudier si un élève attribue plusieurs formes à une expression algébrique et sait mobiliser la forme adaptée au problème rencontré
Composante	Effectuer du calcul algébrique
Capacité	- Développer une expression littérale de la forme (a+b)(c+d) - Utiliser les identités remarquables dans les deux sens, pour factoriser ou développer
Type de tâche	- Associer plusieurs formes à une expression algébrique - Effectuer du calcul algébrique en mobilisant la forme de l'expression adaptée au calcul.
Critères d'évaluation	V1, V2, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V3 : incorrect
	EA1, EA2, EA31, EA32, EA33, EA41, EA42 : critères d'évaluation de la dimension « Utilisation des règles d'Ecriture et de réécriture Algébrique » EA1 : Utilisation correcte des règles de transformation EA2 : Maîtrise technique fragile EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct EA32 : Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect EA33 : Utilisation de règles de transformation fausses identifiées EA41 : Les règles de transformation utilisées linéarisent les expressions : EA42 : Les règles de transformation utilisées « assemblent » les termes

Tableau 11.a : Caractérisation de la classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales »

9.1. Génération de l'exercice

9.1.1. Génération de l'interface

Un exercice de cette classe comporte trois *questions* composées d'une question avec deux *choix exclusifs* (Vrai, Faux) et d'une justification. Ces trois questions sont des *questions composées* d'une *question fermée* et d'une *question ouverte*. (Tableau 11.b). Les *justifications* se présentent sous la forme d'un *calcul algébrique*

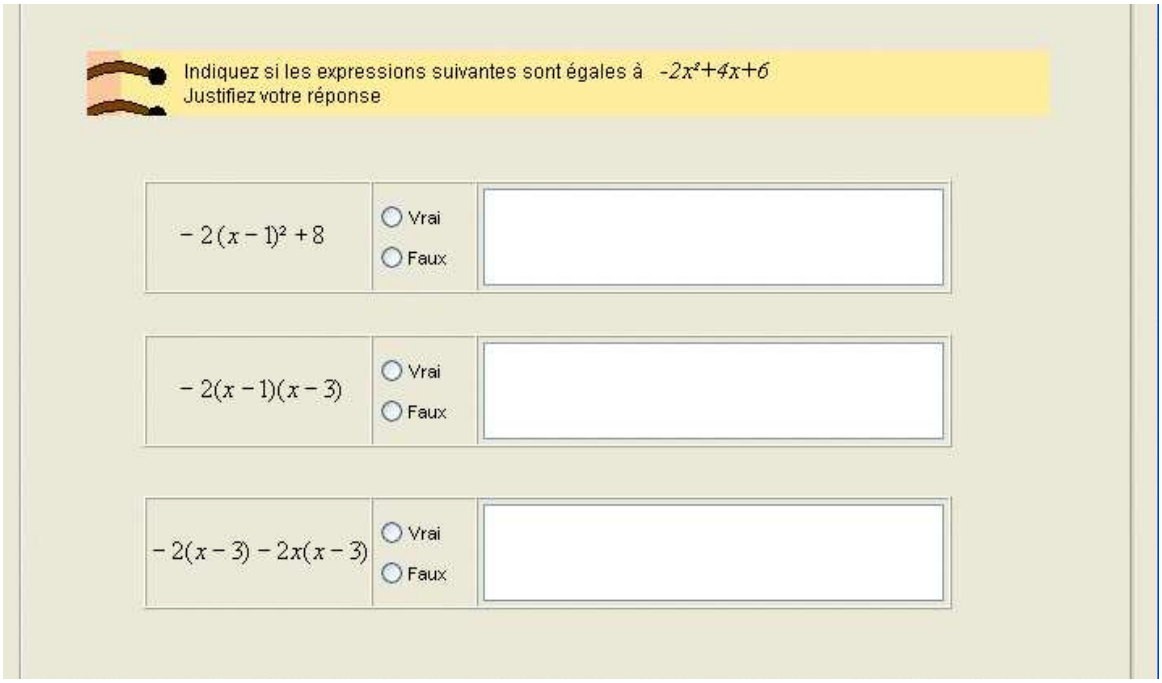


Figure 12 : Interface de l'exercice originel de la classe « Déterminer si des expressions algébriques du second degré sont égales »

9.1.2. Génération des énoncés

L'expression de l'énoncé est un polynôme du second degré : (E1) $-2x^2+4x+6$

Exprimons (E1) et les trois expressions proposées (E2), (E3) et (E4) en fonction des racines x_1 et x_2 de ce polynôme, de leur somme $s = x_1 + x_2$ et de leur produit $p = x_1 x_2$

(E1)	$-2x^2+4x+6$	peut s'écrire	(E'1)	$ax^2- a(x_1 + x_2)x + a x_1 x_2$
(E2)	$-2(x-1)^2+8$	peut s'écrire	(E'2)	$a(x-s/2)^2-a(s^2-4p)/4$
(E3)	$-2(x-3)(x-1)$	peut s'écrire	(E'3)	$a(x-x_1)(x+x_2)$
(E4)	$-2(x-3)-2x(x-3)$	peut s'écrire	(E'4)	$-ax_2(x-x_1)+ax(x-x_1)$

L'expression (E'1) est l'expression d'un trinôme en fonction de la somme et du produit des racines : $ax^2- a(x_1 + x_2)x + a x_1 x_2$, (E'2) s'obtient à partir de l'expression canonique d'un trinôme du second degré car $a((x+b/(2a))^2-\Delta/(4a^2))$ peut s'écrire $a((x-s/2)^2-(s^2-4p)/4)$ soit en développant

une première fois $a(x\text{-}s/2)^2\text{-}a(s^2\text{-}4p)/4$. Les expressions (E'34) et (E'4) sont obtenues à partir de l'expression d'un trinôme en fonction de ses racines : $a(x - x_1)(x - x_2)$. (E'3) en est une forme erronée et (E'4) est obtenue par la suite de calculs :

$$a(x - x_1)(x - x_2) = (x - x_1)(ax - ax_2) = -ax_2(x - x_1) + ax(x - x_1)$$

Ces quatre modes d'obtention des expressions sont les *invariants* de l'exercice. La donnée des racines et de la valeur du coefficient *a* du polynôme de l'énoncé permet de générer des clones de l'exercice : *a*, x_1 et x_2 sont les *paramètres* de l'exercice. Ces paramètres sont soumis à des *contraintes* car ils déterminent la complexité des expressions générées. Les valeurs numériques choisies par les didacticiens sont des nombres entiers relatifs ce qui implique que *a*, x_1, x_2 vérifient les relations :

$$\forall\ a, x_1, x_2, \in \mathrm{Z} \quad \text{on a : } -10 < a < 10\ ,\ -10 < x_1 < 10\ ,\ -10 < x_2 < 10$$

Le Tableau 12 présente les valeurs des paramètres pour le représentant de cette classe qui illustre notre discours et qui est présenté dans la Figure 12.

Interface	
- Un énoncé composé d'un texte invariant et d'une expression algébrique variable : $ax^2 - a(x_1 + x_2)x + a x_1 x_2$ x_1 et x_2 étant les racines du polynôme - Trois questions ouvertes dont les énoncés sont les expressions algébriques générées à partir des invariants associées à trois questions à choix exclusif Vrai - Faux	
Génération de l'énoncé	
Invariants	Type 1 : expression d'un trinôme en fonction des racines $ax^2 - a(x_1 + x_2)x + a x_1 x_2$ Type 2 *:* premier développement de l'expression canonique $a(x\text{-}s/2)^2\text{-}a(s^2\text{-}4p)/4$ Type 3 : expression erronée de a $(x - x_1)(x - x_2)$ Type 4 : expression correcte obtenue à partir de l'expression a $(x - x_1)(x - x_2)$ après avoir effectué dans l'un des produits puis développé une fois $-ax_2(x - x_1) + ax(x - x_1)$
Paramètres	- *a*, x_1, x_2 - quatre expressions générées à partir des quatre types de règles - trois choix
Contraintes	$\forall\ a, x_1, x_2, \in \mathrm{Z}$ $-10 < a < 10$, $-10 < x_1 < 10$, $-10 < x_2 < 10$, $x_1 + x_2$ est paire
Type de génération	Génération assistée - Choix des valeurs des racines x_1 et x_2 et du coefficient *a* du polynôme du second degré - Génération automatique des quatre expressions algébriques correspondantes - Génération automatique de la position des questions ouvertes

Tableau 11.b : Caractérisation de la classe d'exercices
« Déterminer si des expressions algébriques du second degré sont égales »

9.2. Génération du diagnostic

Deux éléments interviennent dans l'élaboration de la *grille de codage* par les didacticiens : la réponse au choix exclusif qui est indépendante du clone et l'analyse des justifications de l'élève. Le Tableau 11.c présente la génération du diagnostic. Les différentes *réponses anticipées* sont les expressions obtenues en appliquant des règles de calcul correctes ou erronées aux trois expressions algébriques (E2), (E3), (E4). La grille de codage dépend des expressions algébriques obtenues en faisant varier les valeurs des paramètres a, x_1, x_2 de la classe d'exercices.

Génération du diagnostic local		
	Réponses et commentaires	**Code**
Solution correcte	Les calculs algébriques sont correctes ainsi que les choix vrai et faux -Maîtrise technique attendue en fin de troisième (développement par application d'une identité remarquable $(a + b)^2$ ou $(a - b)^2$)	V1, EA1
	-Maîtrise technique faible (pas de reconnaissance de l'identité remarquable)	V1, EA2
Solution incorrecte	- Absence de parenthèses dans l'écriture avec résultat correct	V3, EA31
	- Absence de parenthèses dans l'écriture avec résultat faux	V3, EA32
	- Erreur liée à l'utilisation de règles incorrectes : $2ab = 2a2b$, $(a - b)^2 = a^2 – b^2$	V3, EA33
	- Calculs incorrects se ramenant au premier degré $(a + b)(c + d) = (a + b) + (c + d)$, $a^2 = 2\ ab$	V3, EA41
	- Calculs incorrects avec regroupement par exemple $a^2 + a = a^3$	V3, EA42
Type de génération	Génération automatique du diagnostic - PépiGen génère l'ensemble des réponses anticipées obtenus à partir des expressions algébriques proposées dans les trois questions, et, le code associé - La grille de codage dépend des expressions algébriques proposées dans les questions	

Tableau 11.c : Caractérisation de la classe d'exercices

« Déterminer si des expressions algébriques du second degré sont égales »

9.3. Type de génération

Les paramètres a, x_1 et x_2 sont fortement contraints. Nous avons choisi une génération automatique de ces paramètres et de la position des trois questions : (i) PépiGen génère de façon aléatoire les valeurs des racines x_1 et x_2 et du coefficient a du polynôme du second degré, (ii) PépiGen construit les quatre expressions algébriques selon les modes d'obtention décrits dans le Tableau 11.b (iii) pour le diagnostic, PépiGen génère l'ensemble des réponses anticipées, c'est-à-dire les calculs algébriques (développements et réductions) obtenus à partir des expressions algébriques proposées dans les trois questions, et le code associé.

9.4. Discussion

Dans cette classe d'exercices ainsi que dans la classe d'exercices « Expression littérale de l'aire d'un rectangle » (Cf. 7.1.2), nous générons de façon automatique les expressions algébriques qui constituent les contenus des questions. Pour obtenir des clones équivalents du point de vue du diagnostic, nous avons élaboré des règles d'obtention de ces expressions algébriques en étudiant les exercices originels. Cette étude s'impose si nous voulons obtenir des clones équivalents.

Le Tableau 12 présente les valeurs des paramètres pour le représentant de cette classe qui illustre notre discours.

Exercice originel (Figure 12)	a	-2
	x_1	-1
	x_2	3
	Expression de type 1	$ax^2 - a\,(x_1 + x_2)\,x + a\,x_1\,x_2$
	Expression de type 2	$a(x\text{-}s/2)^2\text{-}a(s^2\text{-}4p)/4$
	Expression de type 3	$a\,(x - x_1)\,(x + x_2)$
	Expression de type 4	$-ax_2\,(x - x_1) + ax\,(x - x_1)$

Tableau 12 : Valeurs des paramètres

10. Classe d'exercices « Expressions algébriques d'un programme de calcul»

La dernière classe d'exercices que nous étudions dans ce document porte sur la production de texte en langage naturel contraint (Figure 13.a et Figure 13.b). Cette classe d'exercices a pour objectifs de savoir si un élève sait associer des expressions algébriques aux différentes étapes d'un programme de calcul décrit en langage naturel, identifier les règles de traduction et utiliser les deux représentations d'un programme de calcul (passer du langage naturel à l'expression algébrique et vice-versa). Elle appartient à la composante « *Traduire algébriquement dans différents représentations* (*géométrique, graphique, langage naturel*) » et met en œuvre la capacité « *Traduire une expression algébrique comme un programme de calcul et vice-versa* ». Le type de tâches consiste donc à écrire l'expression algébrique traduisant chaque étape d'un calcul et réciproquement à écrire les phrases traduisant les étapes d'un calcul.

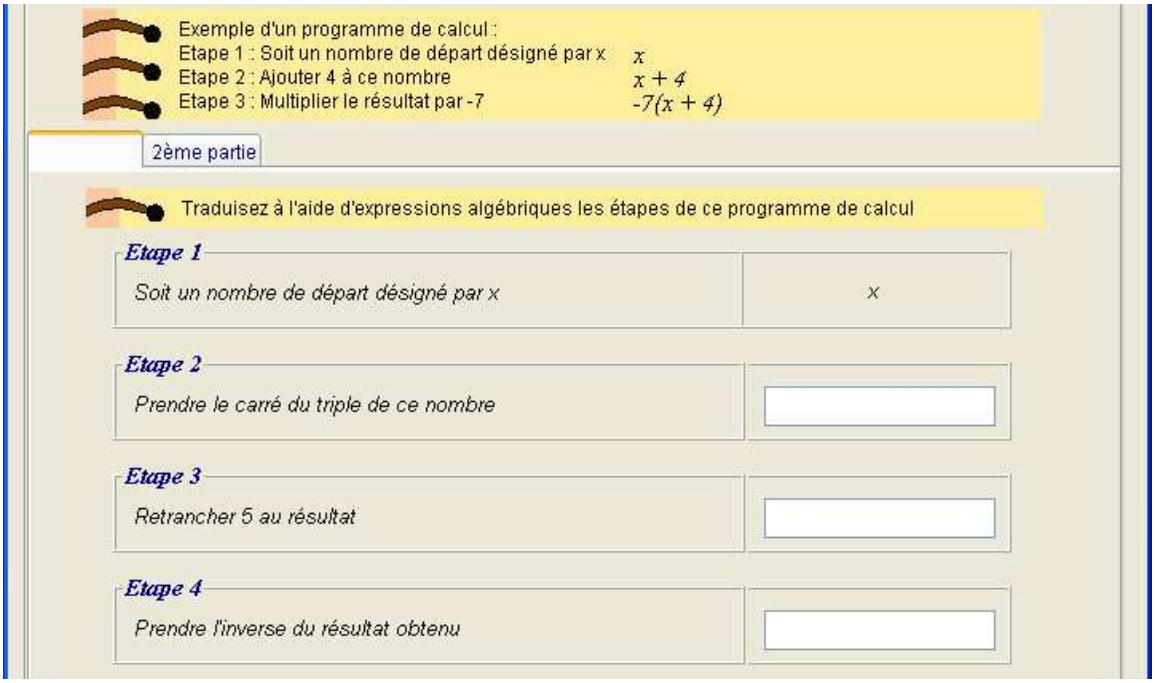


Figure 13.a : Interface de la première partie de l'exercice originel de la classe « Déterminer des expressions d'un programme de calcul » - Première partie

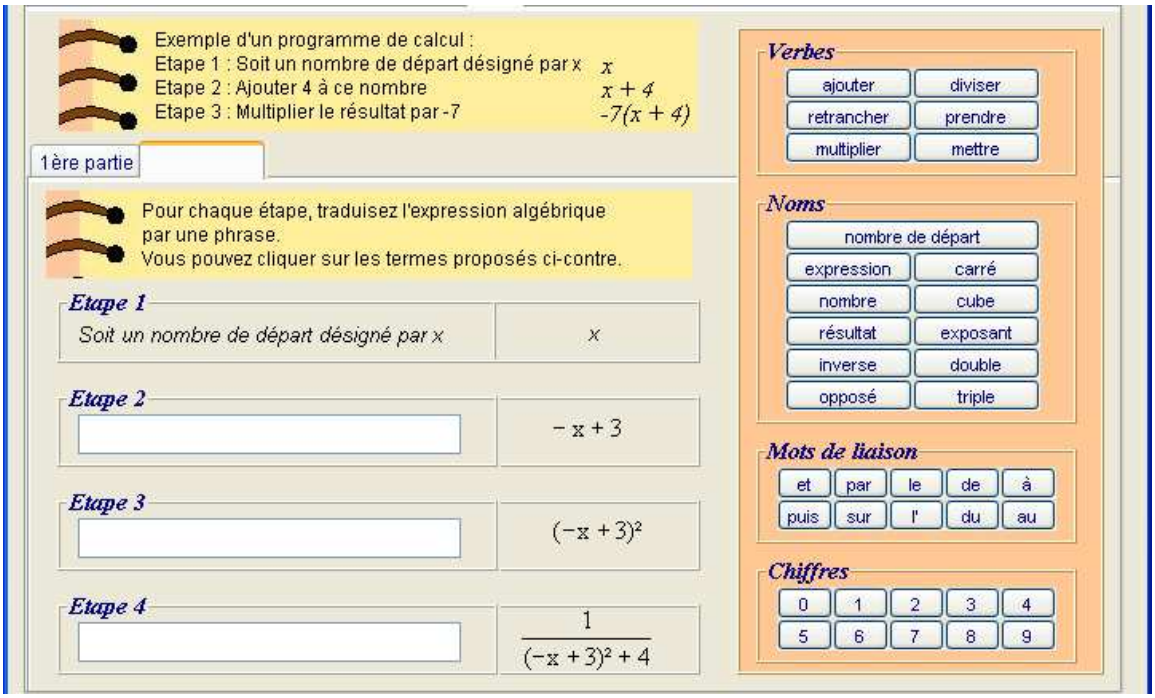


Figure 13.b : Interface de la deuxième partie de l'exercice originel de la classe «Déterminer des expressions d'un programme de calcul » - Deuxième partie

Indexation didactique	
Objectifs	- Savoir si un élève sait associer des expressions algébriques aux différentes étapes d'un programme de calcul et vice-versa - Identifier les règles de traduction pour passer du langage naturel au écritures algébriques et vice-versa
Composante	Traduire algébriquement dans différents représentations (géométrique, graphique, langage naturel)
Capacité	Traduire une expression algébrique comme un programme de calcul et vice-versa
Type de tâche	- Écrire l'expression algébrique traduisant chaque étape d'un calcul - Réciproquement écrire les phrases traduisant les étapes d'un calcul
Critères d'évaluation	V1, V3 : critères d'évaluation de la dimension « Validité » V1 : correct V3 : incorrect
	T1, T3 : critères d'évaluation de la dimension « Traduction » T1 : Traduction correcte T3 : Traduction incorrecte
	EA1, EA2, EA31, EA32, EA33, EA41, EA42 : critères d'évaluation de la dimension « Utilisation des règles d'Ecriture et de réécriture Algébrique » EA1 : Utilisation correcte des règles de transformation EA2 : Maîtrise technique fragile EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct EA32 : Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect EA33 : Utilisation de règles de transformation fausses identifiées EA41 : Les règles de transformation utilisées linéarisent les expressions : EA42 : Les règles de transformation utilisées « assemblent » les termes

Tableau 13.a: Caractérisation de la classe d'exercices
« Déterminer des expressions d'un programme de calcul »

10.1. Génération de l'exercice

10.1.1. Interface Élève

Cet exercice comporte deux parties. D'un point de vue abstrait l'interface est composée d'un *énoncé*, commun aux deux *parties*, dont le *texte* est invariant et, pour chacune des parties, d'un *énoncé* spécifique et de trois *questions ouvertes* avec une zone de saisie.

10.1.2. Génération des énoncés

Pour la première partie, les trois questions décrivent trois étapes d'un programme de calcul et la réponse attendue est l'expression algébrique traduisant l'étape correspondante de ce programme. La modification du programme de calcul génère un clone. Pour la deuxième partie, les trois questions sont des expressions algébriques, la réponse attendue est la traduction de cette expression en un programme de calcul exprimé en langage naturel et la modification de

l'expression algébrique génère un clone. La réponse est exprimée avec les termes d'une palette de mots, choisis par les didacticiens. Ainsi, les *invariants* de la classe d'exercices sont les mots de cette palette. Nous avons choisi d'utiliser cette palette pour générer des clones de cette classe d'exercices. L'auteur écrit un programme de calcul en langage naturel avec la palette qui génère aussi l'expression algébrique correspondante. Le clone est caractérisé par le programme de calcul en langage naturel (e.g. « *prendre le triple du nombre de départ* ») et l'expression algébrique correspondante (e.g. x^3). La phrase en langage naturel et l'expression algébrique sont les *paramètres* de la classe d'exercices. Ainsi la génération d'un exercice de cette classe est assistée.

Interface	
- Un énoncé commun aux deux parties de l'exercice dont le texte est invariant - Un énoncé spécifique à chaque partie dont le texte est invariant Partie 1 : Trois questions ouvertes avec une zone de saisie Partie 2 : Une palette pour écrire des phrases et trois questions ouvertes avec une zone de saisie	
Génération des énoncés	
Invariants	Les termes de la palette
Paramètres	Un programme de calcul exprimé dans deux représentations : - En langage naturel - Sous la forme d'une expression algébrique
Contraintes	- Aucune car le programme de calcul est limité par les termes de la palette
Type de génération	Génération assistée - L'auteur choisit les termes de la palette pour écrire le programme de calcul - PépiGen construit les quatre expressions algébriques correspondantes

Tableau 13.b : Caractérisation de la classe d'exercices
« Déterminer des expressions d'un programme de calcul »

10.2. Génération du diagnostic

L'ensemble des *réponses anticipées* sont les expressions algébriques générées à partir des programmes de calcul. La *grille de codage* dépend des programmes de calcul. Le Tableau 13.c donne la grille de codage de l'exercice. Pour cette classe d'exercices, nous utilisons de nouveau les analyses théoriques et empiriques [Grugeon 1995] qui ont permis de spécifier les différentes expressions (algébriques ou numériques) obtenues en appliquant des *règles* de réécriture correctes ou erronées à l'expression qui traduit le programme de calcul donné dans l'énoncé. L'ensemble de ces *réponses anticipées* et le code associé est présenté dans le Tableau 13.c. Par exemple, le programme de calcul « *prendre le carré du nombre de départ* » traduit par $2x$ sera codé V3,EA41,T3 car la réponse n'est pas valide, la traduction est incorrecte et la règle de transformation erronée utilisée linéarise l'expression (x^2 est transformée en $2\,x$).

Pour cette classe d'exercices, la grille de codage dépend de l'expression algébrique du programme de calcul qui est un des paramètres du clone.

Génération du diagnostic local		
Première partie		
	Commentaire	**Code**
Solution correcte	Les expressions algébriques correspondent aux étapes du programme de calcul proposé	V1, T1
Solution incorrecte	Écriture non parenthésée	
	- qui garde la mémoire du calcul	V3, EA31,T3
	- qui ne garde pas la mémoire de calcul	V3, EA32,T3
	Confusion entre carré et double	V3, EA41,T3
	Confusion entre inverse et opposé	V3, EA33,T3
Deuxième partie		
	Commentaire	**Code**
Solution correcte	les étapes du programme de calcul correspondent aux expressions algébriques proposées	V1, T1
Solution incorrecte	Écriture non parenthésée	
	- qui garde la mémoire du calcul	V3, EA31,T3
	- qui ne garde pas la mémoire de calcul	V3, EA32,T3
	Confusion entre carré et double	V3, EA41,T3
	Confusion entre inverse et opposé	V3, EA33,T3
Type de génération	Génération automatique - PépiGen génère l'ensemble des réponses anticipées obtenus à partir des expressions algébriques correspondant aux programmes de calcul proposées dans les trois questions, et, le code associé - La grille de codage dépend des expressions algébriques proposées dans les questions	

Tableau 13.c : Caractérisation de la classe d'exercices
« Déterminer des expressions d'un programme de calcul »

10.3. Type de génération

La génération de clones de cette classe d'exercices est associée soit à la génération du programme de calcul, soit à la génération de l'expression algébrique correspondant à un programme de calcul. Une génération assistée est possible : (i) l'auteur choisit les mots appropriés en utilisant une palette de termes pour concevoir un programme de calcul. Ce programme de calcul est l'énoncé de la question pour la première partie, la traduction de l'expression algébrique proposée pour la deuxième partie, (ii) pour les deux parties PépiGen construit l'expression algébrique correspondant aux choix des mots de la palette, (iii) pour le diagnostic, PépiGen génère l'ensemble des réponses anticipées présentées dans la grille de codage.

La grille de codage dépend des paramètres de la classe d'exercices : le programme de calcul et l'expression algébrique.

11. Élaboration d'un modèle conceptuel d'une classe d'exercices

Dans les sections précédentes nous avons étudié plusieurs classes d'exercices pour identifier les informations à prendre en compte dans la fabrication de clones des exercices de PépiTest : elles concernent l'indexation didactique des exercices, la génération d'un exercice et la génération du diagnostic associé et les informations concernant le type de génération. Nous avons analysé dans quelle mesure ces informations interviennent dans la génération des clones d'un exercice et du diagnostic local associé et fait apparaître les concepts mis en jeu pour construire le modèle conceptuel d'une classe d'exercices. Dans cette section, nous synthétisons ce travail en présentant un modèle conceptuel qui décrive les différentes classes d'exercices de PépiTest et qui puisse s'adapter à d'autres classes d'exercices. Pour représenter ce modèle, nous utilisons le langage de modélisation UML (Unified Modeling Language).

La description d'une classe d'exercices[1] (***ClasseExercice***) est une composition des éléments relatifs aux catégories suivantes :

- les données d'indexation didactique (**IndexationDidactique**) qui caractérisent les exercices d'un point de vue didactique,
- la génération des énoncés (**GenerationEnonce**) qui regroupe les informations pour générer les contenus des énoncés des différentes parties et des questions d'un exercice,
- l'interface abstraite (**InterfaceAbstraite**) qui caractérise les éléments nécessaires à la génération de l'interface élève du clone d'exercice,
- la classe **GenerationGrilleCodage** qui correspond au diagnostic local associé au clone d'exercice.

La classe UML, ***ClasseExercice***, qui décrit les spécifications d'une classe d'exercices est une classe abstraite.

[1] Une classe d'exercices représente des exercices équivalents du point de vue du diagnostic cognitif (section 2). Pour différencier cette notion de classe (concept mathématique) avec celle de classe UML ou dans le chapitre 7 de classe Java nous utilisation un style différent pour noter les classes UML ou Java (Police Arial, style gras).

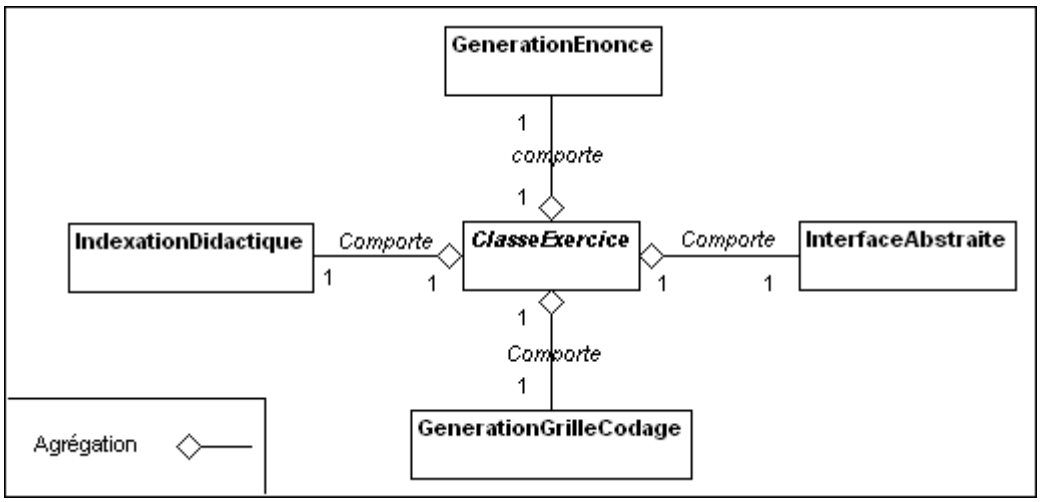


Figure 14 : Modèle conceptuel partiel d'une classe d'exercices

11.1. Indexation didactique des exercices

Pour chaque classe d'exercices, les informations concernant l'indexation didactique comportent dix éléments principaux : les objectifs, les composantes, les capacités, et les critères d'évaluation (Figure 15). Pour une classe d'exercices, les objectifs (**Objectif**), le type de tâches diagnostiques (**TypeTache**), les dimensions d'évaluation (**DimensionEvaluation**), les critères d'évaluation (**CritereEvaluation**) et le code (**Code**) associé s'appuient sur le modèle multidimensionnel de la compétence algébrique défini par Grugeon [Grugeon 1995] et les travaux des membres du projet Lingot. Selon ce modèle, une dimension d'évaluation (**DimensionEvaluation**) comporte plusieurs critères d'évaluation. Un critère d'évaluation est associé à un code : par exemple le critère d'évaluation « Maîtrise technique fragile » appartenant à la dimension d'évaluation « Utilisation des règles d'Écriture et de réécriture Algébrique » est codé EA2 (tableaux Indexation didactique de ce chapitre et annexe). Les capacités (**Capacite**) ainsi que les connaissances (**Connaissances**) sont décrites, par niveau scolaire (**NiveauScolaire**), dans les programmes d'enseignement des mathématiques [B.O. 2007]. Une composante (**Composante**) comporte plusieurs capacités (Annexe A.2). Le niveau scolaire n'est pas précisé pour chacune des classes d'exercices que nous avons étudiées car les exercices originels de PépiTest ont tous été conçus pour un niveau de troisième. Il en est de même des connaissances qui ne sont pas spécifiées dans les exercices originels. Elles sont décrites dans les programmes d'enseignement et sont un élément à prendre en compte pour une indexation didactique des classes d'exercices.

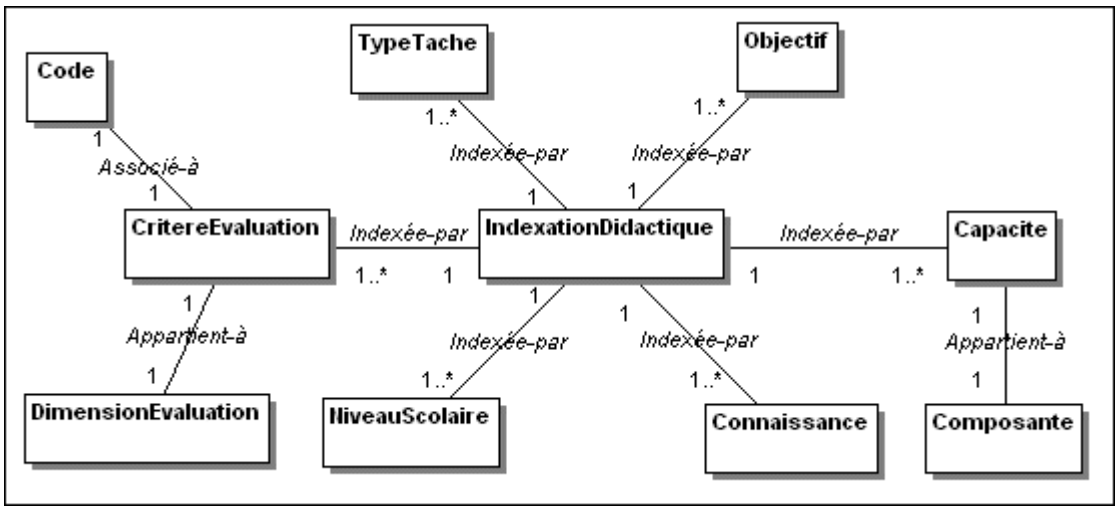


Figure 15 : Modèle conceptuel des données d'indexation didactique d'une classe d'exercices

11.2. Génération des énoncés

Pour générer des clones d'exercices équivalents du point de vue du diagnostic cognitif, notre étude a fait apparaître des *invariants* (**Invariant**) et des *paramètres* (**Parametre**) à partir desquels sont générés les énoncés de ces clones. Les *invariants* et les *paramètres* ont des contenus (**Contenu**) générés selon deux modes (**ModeObtention**) : une génération automatique pour les exercices dont les paramètres sont fortement contraints et une génération assistée pour laquelle l'auteur d'un exercice peut saisir les paramètres par le biais d'une interface (Figure 16). Les contenus des paramètres sont soumis à des *contraintes* (**Contraintes**) qui peuvent dépendre du niveau scolaire. La classe **Contenu** représente le contenu des invariants, des paramètres, des contraintes mais aussi celui des énoncés (Figure 18). Cette classe UML abstraite est une agrégation dont les composants sont une expression numérique ou algébrique, un texte, une figure. C'est aussi une généralisation de deux sous-classes, **ContenuSimple** et **ContenuComplexe**, la classe **ContenuComplexe** étant une composition. Cette dernière classe permet de décrire les contenus de toutes les classes d'exercices que nous avons rencontrées dans PépiTest et plus particulièrement des contenus complexes comportant à la fois des textes, des figures et des expressions algébriques.

Enfin, un contenu peut être obtenu par la donnée explicite de l'expression, du texte ou de la figure (e.g. les expressions qui constituent les invariants de la classe d'exercices « Reconnaître des égalités vraies » (section 3)), par application de propriétés, de règles d'obtention spécifiques à la classe d'exercices (e.g. les paramètres de la partie 2 de la classe d'exercices « Expression littérale de l'aire d'un rectangle » sont des expressions générées à partir de 6 types de règles (section 7.1.2)) ou encore être généré de façon aléatoire (e.g. les paramètres des classes

d'exercices « Reconnaître des égalités vraies ») ou de façon assistée (e.g. avec une palette de termes).

Pour décrire le modèle conceptuel de la génération des énoncés, nous avons donc utilisé des classes abstraites d'une part, des relations de généralisation et d'agrégation d'autre part. L'utilisation de classe abstraites factorise ainsi le code et les données communes à chacune des classes.

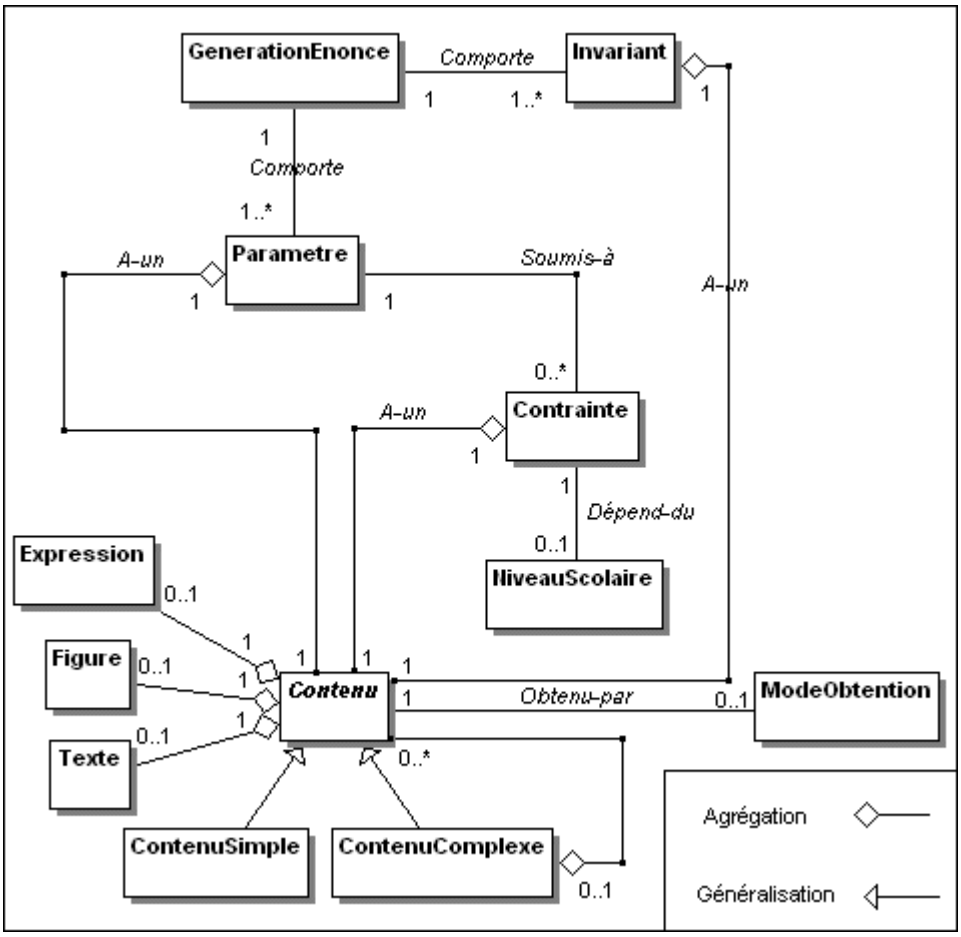


Figure 16 : Modèle conceptuel de la génération des énoncés

11.3. Description de l'interface

L'étude de plusieurs classes d'exercices de ce chapitre a montré deux types d'organisation : certains exercices comportent seulement une seule partie avec des questions, d'autres comportent plusieurs parties avec une ou plusieurs questions. Notre objectif étant de construire un modèle qui s'applique à tous les cas, tous les exercices sont structurés en une ou plusieurs parties contenant des questions. Pour une classe d'exercices, l'interface est constituée d'un *énoncé*, d'une ou

plusieurs *parties*, chaque partie comportant une ou plusieurs *questions*. Une partie ou une question peuvent avoir un énoncé particulier (Figure 18). Ainsi la classe d'exercices « Reconnaître des égalités vraies » a un énoncé dont le contenu est « Cocher dans chaque cas la ou les égalités correctes », une partie et quatre questions a choix multiples, chaque choix ayant un contenu qui est une expression numérique (e.g. $5^2 \times 5^3 = 5^6$). La classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales » a un énoncé dont le contenu est un contenu complexe : un texte et une expression algébrique suivie d'un texte et de trois questions avec un énoncé dont le contenu est un contenu simple comportant une expression algébrique (e.g. $2(x\text{-}1)^2 + 8$) et un choix exclusif avec un contenu de type texte (e.g. Vrai,Faux).

Pour décrire les différentes questions que nous avons rencontrées dans notre étude, nous avons choisi une classe abstraite, ***Question***. La description de certains concepts par une classe abstraite nous permet de définir un protocole général, d'obtenir un modèle plus générique adaptable à d'autres classes d'exercices. Nous distinguons d'abord deux catégories de questions : des questions simples (**QuestionSimple**) et des questions composées (**QuestionComposee**). Une question simple est la généralisation d'une *question fermée* (**QuestionFermée**) et d'une *question ouverte* (**QuestionOuverte**), une question composée est une composition de ***Question***.

Ainsi, pour la classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales », la question présentée dans la Figure 17 est une question composée d'une question simple à choix exclusif (Vrai, Faux) et d'une question ouverte (« Justifiez votre réponse »). Une question fermée est une généralisation de questions à *choix multiples* et de questions à *choix exclusif*. Les classes **ChoixMultiple** et **ChoixExclusif** sont des agrégations dont la valeur de multiplicité est de deux à plusieurs.

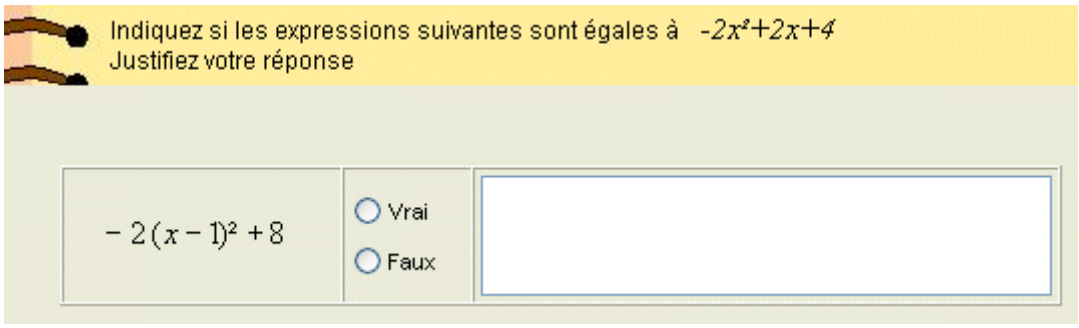


Figure 17 : Exemple de question composée : « Déterminer si des expressions algébriques du second degré sont égales » - question 1

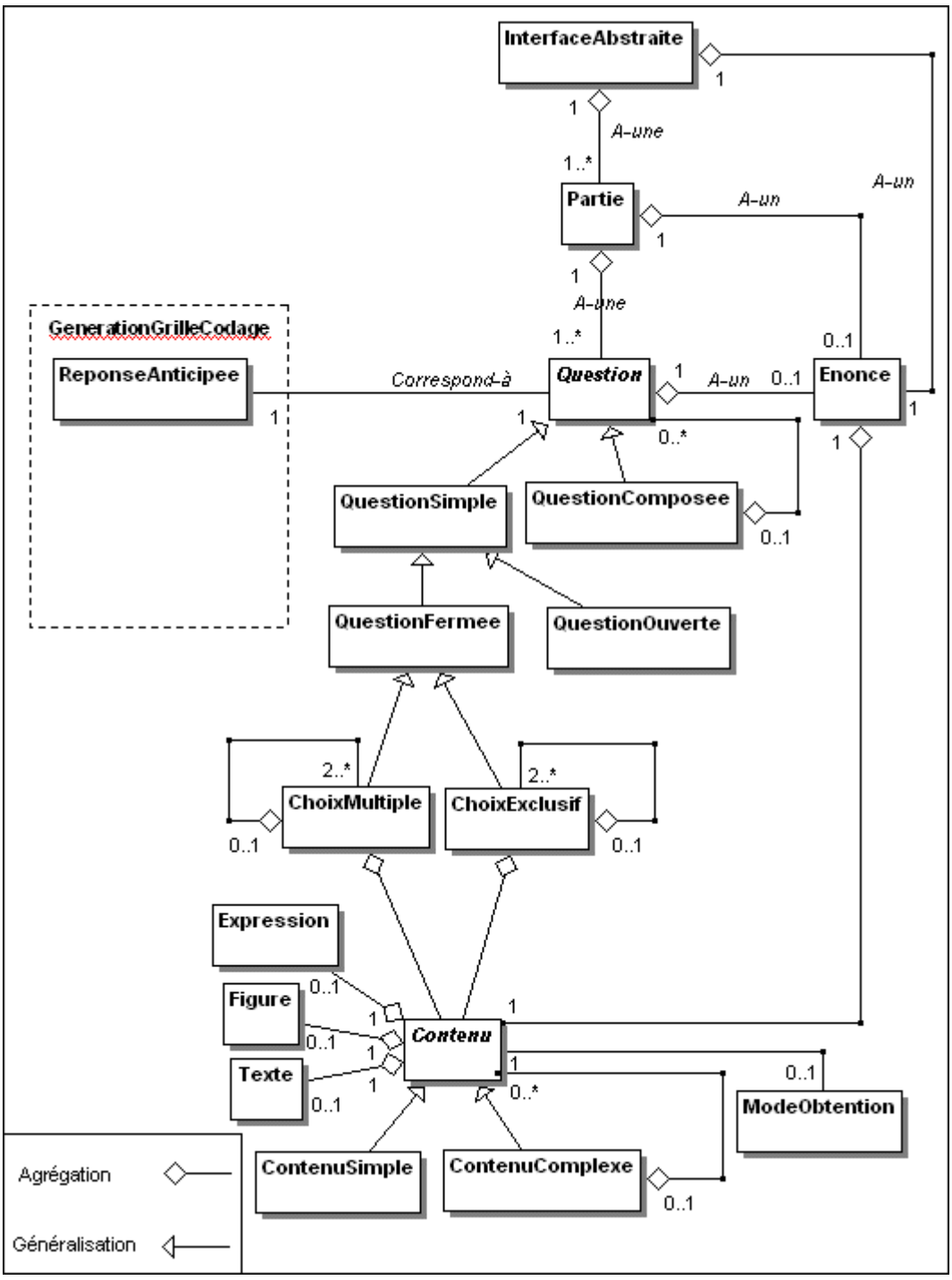


Figure 18 : Modèle conceptuel de l'interface abstraite

11.4. Génération du diagnostic

A chaque classe d'exercices est associée une *grille de codage* des réponses qui comporte l'ensemble des *réponses anticipées*, qu'elles soient correctes ou erronées, et le diagnostic local associé à ces réponses. Une réponse anticipée est attachée à une question (Figure 18 et Figure 19). Le diagnostic est composé d'un *code* qui caractérise une *dimension d'évaluation* et un *critère d'évaluation* ainsi que d'une *interprétation* et, éventuellement, de la *règle* de calcul *correcte* ou *erronée* utilisée pour obtenir la réponse anticipée. Les dimensions d'évaluation, les critères d'évaluation et le code associé sont définis dans la *grille d'analyse multidimensionnelle* des compétences définie en annexe 2. Pour les raisons déjà évoquées dans la section 11.3, nous utilisons une classe abstraite pour les classes **ReponseAnticipee** et **Regles** (Figure 19). Une réponse anticipée est la généralisation des trois sous-classes **ReponseQuestionFermee**, **ReponseQuestionOuverte** et **ReponseQuestionComposee**. Une réponse à une question fermée est une agrégation de un à plusieurs choix. Pour une réponse à une question ouverte, la réponse anticipée peut être sous la forme de lignes de calcul ou d'un résultat ou encore sous les deux formes (lignes de calcul et résultat), d'où les valeurs de multiplicité de zéro à plusieurs pour l'une et zéro ou un pour l'autre. La classe **ReponseQuestionComposee** est une composition de **ReponseAnticipee**. Ainsi, pour la classe d'exercices « Déterminer si des expressions algébriques du second degré sont égales », la réponse anticipée à la question présentée dans la Figure 17 comporte une réponse à une question fermée (Vrai, Faux) et une réponse à une question ouverte sous la forme de plusieurs lignes de calculs.

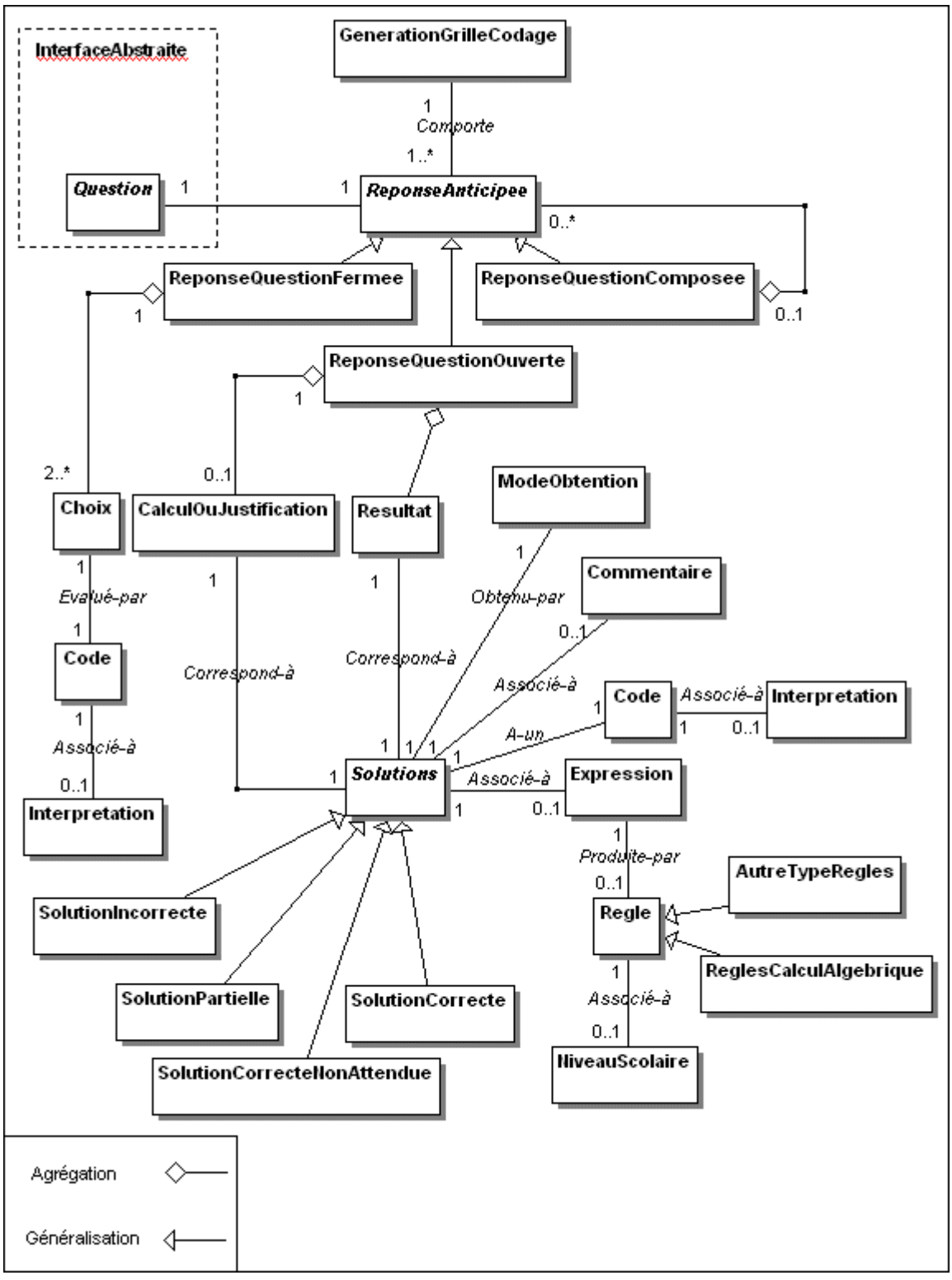


Figure 19 : Modèle conceptuel de la Grille de Codage des réponses

12. Conclusion

Dans ce chapitre nous avons décrit le modèle conceptuel des classes d'exercices que nous avons mis au point pour générer des banques d'exercices de diagnostic de compétences en algèbre élémentaire. Une classe d'exercices définit d'une part, les caractéristiques communes à tous les exercices équivalents à un exercice originel du point de vue du diagnostic cognitif et, d'autre part, les paramètres sur lesquels jouer pour générer de tels exercices. Lorsque les paramètres sont fortement contraints leurs valeurs sont générées automatiquement de façon aléatoire. Par contre les enseignants de l'équipe ont souhaité fixer eux-mêmes les valeurs des paramètres pour générer des exercices exactement adaptés à leur attente, lorsque les paramètres donnent plus de degré de liberté. Dans les deux cas l'analyse des réponses est générée automatiquement. Rappelons que, dans ces deux cas, un enseignant utilisateur final qui veut faire passer un test à ces élèves peut se contenter de le composer avec des exercices prêts à l'emploi, soit en demandant la génération automatique d'un test, soit en choisissant les exercices dans une banque d'exercices.

Nous avons montré comment ce modèle permet de décrire des exercices « équivalents » à ceux du test diagnostic originel mis au point par [Grugeon 1995]. Nous nous sommes restreints dans ce mémoire à des exercices de type questions fermées dont les énoncés portent sur des expressions algébriques ou sur des figures géométriques et, d'autre part, à des exercices de type questions ouvertes où les réponses attendues sont une expression algébrique ou une suite d'expressions algébriques traduisant une preuve.

Dans les chapitres suivants nous montrons comment nous avons construit un prototype qui nous permet de tester l'opérationnalité de ce modèle conceptuel. En effet, ce modèle conceptuel a été élaboré avec les enseignants et les didacticiens de l'équipe pour proposer un format de description des données qui prenne en compte les contraintes à respecter pour générer des clones qui soient équivalents du point de vue du diagnostic cognitif. C'est donc un modèle descriptif que nous avons d'abord représenté sous la forme de tableaux. Puis nous avons formalisé les entrées de ces tableaux en décrivant les classes d'exercices sous la forme de diagrammes UML. Du point de vue Génie Logiciel, comme nous le montrons dans le chapitre 7, cette formalisation constitue un métamodèle qui sert de fondement au développement de la couche domaine de notre système. En effet, ce métamodèle est le modèle des données des différentes classes métiers de PépiGen . Il permet ainsi de générer automatiquement les schémas XML qui décrivent la structure et les caractéristiques des fichiers XML contenant, d'une part les données caractérisant une classe d'exercices et, d'autre part, les données caractérisant chaque exercice représentant de ces classes.

Pour le projet Pépite, c'est un saut qualitatif très important. En effet, la thèse de Stéphanie Jean-Daubias [Jean 2000] avait démontré qu'il était possible d'automatiser l'outil diagnostic papier-crayon mis au point par Brigitte Grugeon, ce qui, à l'époque laissait les didacticiens très dubitatifs. Pour cela, elle a construit un prototype composé de trois logiciels qui proposait une solution à chacun des trois problèmes soulevés par ce travail. Tout d'abord, PépiTest a prouvé que malgré la difficulté de saisir des expressions algébriques au clavier, des interfaces soigneusement mises au point permettaient de recueillir des productions d'élèves signifiantes pour le diagnostic. Ensuite, qu'il était possible de coder automatiquement au moins une bonne partie des réponses ouvertes pour alléger le travail de codage fastidieux pour un humain. Enfin, Stéphanie Jean a mis au point une interface pour faciliter l'exploitation du diagnostic. Cette interface comporte de nombreuses fonctionnalités encore absentes des logiciels actuels d'évaluation. Ce premier prototype a été tout a fait primordial pour le projet. Il a permis de mener de nombreux tests dans les classes. Il a aussi donné une visibilité à l'expertise de Brigitte Grugeon en permettant à d'autres personnes (chercheurs ou praticiens) de la mettre œuvre. Cependant, comme de nombreux prototypes de type « preuve de concept », il était difficile à faire évoluer et à maintenir. Dans ce genre de prototype, il s'agit d'abord de montrer ce que l'on peut faire avant de le faire dans les règles de l'art. Ainsi, l'expertise didactique était souvent embarquée dans le code informatique alors que dans ce travail de thèse nous l'avons réifiée dans des fichiers au format XML. De plus, en nous fondant sur ce premier prototype nous avons pu mettre au point notre modèle conceptuel pour créer des banques d'exercices de diagnostic au lieu des exercices uniques originels.

Du point de vue EIAH, un problème de recherche très actuel concerne la constitution de banques de ressources pédagogiques et particulièrement l'indexation des ressources pédagogiques [De La Passardiere et Jarraud 2005, Rebaï et al 2008]. Le modèle conceptuel que nous proposons peut constituer une contribution pour élaborer un profil de métadonnées spécifique à l'évaluation diagnostique. En effet, il résulte d'une étude très détaillée des caractéristiques à prendre en compte pour la génération d'exercices de diagnostic. Si les valeurs des éléments du modèle sont spécifiques à l'analyse didactique qui sous-tend le projet Pépite, les éléments eux-mêmes sont génériques (critères d'évaluation, dimensions d'évaluation, capacités, connaissances). C'est une des perspectives de notre travail. Nous revenons sur ces aspects dans la conclusion de notre thèse dans le chapitre 8.

Avant d'aborder la mise en œuvre informatique du modèle conceptuel que nous venons de présenter, nous étudions dans le chapitre suivant, la façon de résoudre un des problèmes de l'évaluation diagnostique : l'analyse de réponses à des questions ouvertes.

Chapitre 6

Génération et analyse des expressions algébriques : Pépinière

1. Introduction 161

2. Scénarios d'utilisation de Pépinière 162
2.1. Scénario 1 : diagnostic d'une réponse algébrique 162
2.1.1. Construction d'un arbre d'expression 163
2.1.2. Comparaison d'expressions 164
2.2. Scénario 2 : génération automatique de la grille de codage d'un exercice 164
2.3. Architecture fonctionnelle de Pépinière 167

3. Analyse syntaxique des expressions algébriques dans Pépinière 167
3.1. Représentation des expressions algébriques dans Pépinière 168
3.1.1. Représentation usuelle 169
3.1.2. Représentation linéaire usuelle 169
3.1.3. Représentation linéaire 169
3.1.4. Représentation sous forme d'arbre binaire 169
3.1.5. Traduction d'une représentation linéaire en arbre d'expression 170
3.2. Analyse lexicale 171
3.3. Analyse syntaxique 171
3.3.1. Définition de la grammaire algébrique 172
3.3.2. Élimination de la récursivité à gauche 173
3.3.3. Conditions de déterminisme 174
3.3.4. Algorithme d'analyse 177
3.3.5. Étiquette des arbres d'expression obtenus par l'analyseur 178

4. Transformation des expressions algébriques 180
4.1. Transformation et règles de réécriture 182
4.2. Classes de transformations et terminaison 182
4.3. Transformations dans Pépinière 183
4.3.1. Types de règles 183
4.3.2. Heuristiques et terminaison des transformations 186
4.3.3. Discussion 187
4.4. Mécanisme d'inférence : unification 189
4.4.1. Principe de l'algorithme d'unification 190
4.4.2. Exemple d'unification 191
4.4.3. Mise en œuvre de l'unification 194
4.4.4. Modélisation des règles de réécriture 195
4.5. Traitement particulier des règles de réécriture de type 6 et 7 196
4.5.1. Poids d'un arbre 197
4.5.2. La procédure « AjoutTermes » 197
4.5.3. Exemples 199

5. Générateur des solutions anticipées 200
5.1. Construction de l'arbre des solutions anticipées 201
5.2. Modèle général de l'arbre des solutions 203

6. Comparaison des expressions algébriques 204
6.1. Arbres d'expression équivalents 204
6.1.1. Définitions 207
6.2. Procédures de transformation des arbres d'expression 208
6.2.1. Remplacement des soustractions 208
6.2.2. Nœuds dont les deux fils sont des feuilles 209
6.2.3. Nœuds dont les deux fils sont de poids différents 209
6.2.4. Sommes de plusieurs termes et produits de plusieurs facteurs 209
6.3. Utilisation de la comparaison d'expressions 210

7. Conclusion 210

1. Introduction

La construction d'EIAH dans les disciplines scientifiques se heurte depuis de nombreuses années au problème de représentation et de transformation des expressions algébriques. Plusieurs approches ont été suivies. La première consiste à se limiter à des expressions simples, numériques ou affines : c'est par exemple l'approche retenue par l'équipe de Pittsburgh [Koedinger et al. 1997, 2004]. Une seconde approche s'appuie sur l'utilisation de logiciels existants de calcul formel (e.g. Reduce, Derive, Mathematica, Maple) : les projets Camelia [Vivet 1984], ActiveMaths [Melis 1999, 2004], Casyopée [Lagrange 2002] ont développé ou développent des recherches dans ce sens. Une troisième approche consiste à développer des systèmes de calcul formel spécifiques à une matière (e.g. [Shapiro 2005] en physique pour le projet Andes, [Beeson 2002] en calcul intégral pour le logiciel MathXpert) ou pour un niveau scolaire donné (e.g. [Cerrulli 2002] pour les preuves algébriques dans le projet Algebrista ou [Nicaud 1987, 2004, 2007], pour les raisonnements algébriques dans le projet Aplusix).

Concernant le projet Pépite, les problèmes à résoudre par les élèves sont trop complexes et trop divers pour être traités en suivant la première approche. Nous avons écarté la seconde approche car, si l'utilisation de logiciels de calcul formel est envisageable au niveau universitaire, pour les adapter au contexte scolaire pré-universitaire, les chercheurs tentent de développer des sur-couches. Au niveau de l'enseignement obligatoire où nous nous situons, la manipulation par les élèves de tels systèmes s'avère beaucoup trop complexe [Artigue et al. 1998]. De plus, dans une perspective de diagnostic, pour pouvoir analyser des compétences en cours de construction, ou des expressions mal formées ou des calculs erronés saisis par les élèves, Pépite doit disposer d'une représentation des expressions et des règles de transformation (correctes ou erronées) ce qui est impossible avec des logiciels de calcul formel qui sont des boîtes noires dont le code est propriétaire. Nous nous situons donc dans la troisième approche. Nos travaux sont en grande partie voisins de ceux de l'équipe Aplusix qui, à notre sens, a produit les recherches les plus abouties et les plus documentées sur les problèmes de factorisation et de résolution d'équations. Nos travaux s'appuient principalement sur les résultats de cette équipe pour ce qui concerne le cadre théorique sur le calcul formel [Gelis 1994], [Nicaud et al 2001]. Dans le projet Pépite, l'analyse des réponses des élèves sur ce type de problèmes est beaucoup moins fine. En revanche, nous nous intéressons à d'autres types de problèmes et, outre la dimension calcul algébrique sur laquelle s'est centré Aplusix, nous nous intéressons à trois dimensions supplémentaires : le type d'utilisation des lettres, le type de traduction d'un registre sémiotique à un autre et le type de justification (Chapitre 5). Ainsi, nous avons complété sur certains points les travaux de cette

équipe sur les manipulations formelles d'expressions, pour les adapter aux problèmes de diagnostic et au modèle multidimensionnel de la compétence algébrique sur lequel se fonde le projet Pépite.

Ce chapitre présente Pépinière, le logiciel de calcul formel que nous avons conçu et développé pour le projet Pépite. C'est un logiciel générique qui s'occupe du traitement des expressions algébriques indépendamment des classes d'exercices. A partir d'une chaîne de caractères contenant une expression algébrique, il assure principalement l'analyse syntaxique des expressions algébriques, la génération de l'ensemble des solutions anticipées pour les problèmes de développement et de réduction de polynômes de degré au plus égal à deux et, enfin, il teste si deux expressions algébriques sont équivalentes du point de vue de l'analyse didactique. C'est le socle de calcul formel sur lequel s'appuie PépiGen pour générer d'une part, des banques d'exercices et, d'autre part, le diagnostic des réponses habituellement observées à ces exercices.

Nous commençons ce chapitre par une présentation à l'aide de scénarios d'utilisation, des quatre principaux problèmes auxquels nous avons fait face : la représentation des expressions algébriques, la transformation des expressions algébriques en expressions algébriques équivalentes, la génération des solutions anticipées aux problèmes de développement et de réduction et, enfin, la comparaison des expressions algébriques. Nous terminons cette présentation par une description de l'architecture fonctionnelle de Pépinière. Puis nous abordons chacun de ces quatre problèmes en détaillant pour chacun d'eux le cadre théorique sur lequel nous nous appuyons et les solutions logicielles que nous avons élaborées. Ce chapitre se termine par un résumé de nos propositions, les limites et les perspectives de notre travail.

2. Scénarios d'utilisation de Pépinière

Appuyons-nous sur deux scénarios d'utilisation de Pépinière pour mettre en évidence les principales fonctionnalités du logiciel Pépinière : d'une part le diagnostic de la réponse d'un élève saisie sous la forme d'une expression algébrique, et, d'autre part la génération automatique de la grille de codage des réponses anticipées à un exercice.

2.1. Scénario 1 : diagnostic d'une réponse algébrique

Pour répondre à un exercice, un élève saisit au clavier une expression algébrique sous une forme linéaire (E1) :

(E1) $2x + (x+5)^2$

Cette expression est mémorisée par PépiTest dans une chaîne de caractères et affichée à l'écran sous la forme plane usuelle (E2) :

(E2) $2x + (x+5)^2$

Remarquons que les expressions (E1) et (E2) contiennent des implicites : (i) l'écriture $2x$ signifie $2*x$ et (ii) la multiplication et l'exponentiation sont prioritaires sur l'addition (iii) à ordre de priorité égale, l'évaluation se fait de la gauche vers la droite. Pour analyser cette réponse, le système de diagnostic envoie la chaîne de caractères représentant (E1) à Pépinière qui construit un arbre d'expression, puis, le compare avec les arbres d'expression représentant les réponses correctes ou erronées à cet exercice. Ces réponses anticipées ainsi que le code du diagnostic local que nous appelons *grille de codage* ont été générés automatiquement par Pépinière lors de la génération de l'exercice et sont mémorisés dans un fichier XML qui contient la banque d'exercices (Chapitre 7, section 7.2).

2.1.1. Construction d'un arbre d'expression

Pépinière commence par effectuer un pré-traitement sur la chaîne de caractères (E1) pour ajouter les opérateurs implicites et supprimer de l'expression les caractères inutiles (e.g. les espaces) ou qui ne sont pas des symboles habituellement utilisés dans une expression algébrique. La chaîne de caractère obtenue est :

(E3) $2*x+(x+5)\hat{}2$

Pépinière analyse la chaîne de caractères correspondant à l'expression (E3) et construit un arbre binaire étiqueté que nous appelons arbre d'expression [Aho et al. 1993] représenté par la Figure 1.

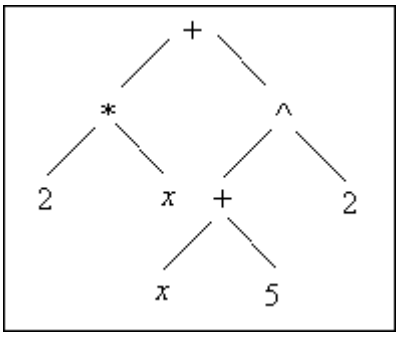


Figure 1 : arbre d'expression de l'expression algébrique $2*x+ (x+5)\hat{}2$

Si l'expression algébrique proposée par l'élève n'est pas une expression algébrique bien formée, Pépinière envoie un message d'erreur approprié et s'arrête. Sinon Pépinière retourne l'arbre d'expression.

2.1.2. Comparaison d'expressions

Pour établir le diagnostic, le système compare la réponse de l'élève aux réponses anticipées à cet exercice mémorisées dans le fichier qui contient la grille de codage. Dans ce fichier, supposons que la réponse attendue soit l'expression :

(E4) $(5+x)^2+2*x$

Le système de diagnostic parcourt les expressions anticipées de la grille de codage et envoie les deux expressions algébriques (la réponse de l'élève et la réponse anticipée) à Pépinière qui construit les arbres d'expression représentés par la Figure 1et la Figure 2 et les compare. Ces deux arbres représentent des expressions équivalentes compte tenu de la commutativité de l'addition. Le système de diagnostic associe à la réponse de l'élève le code correspondant associé à (E4) dans le fichier qui contient la grille de codage : V1, T1, EA1, J1.

Remarque : considérons l'expression :

(E5) $2x + (x+5)(x+5)$

D'un point de vue calcul formel (E5) est une expression équivalente à (E4). Pour les élèves au niveau scolaire où nous travaillons, reconnaître cette équivalence est une compétence à acquérir. Pépinière distingue donc ces deux expressions, qui ne conduisent pas au même codage.

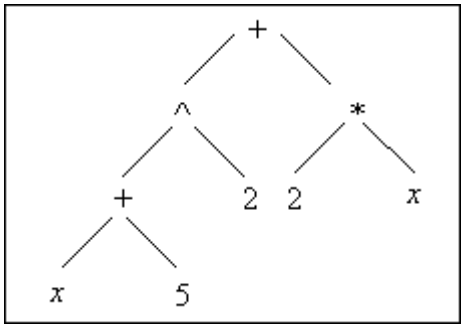


Figure 2: arbre d'expression de l'expression algébrique (5+x)^2 + 2*x

2.2. Scénario 2 : génération automatique de la grille de codage d'un exercice

Dans ce scénario, le concepteur d'exercices vient de saisir l'énoncé d'un exercice de type particulier, par exemple celui du prestidigitateur. Il a saisi l'expression simple suivante :

(E6) $3\,(x+6) - 3\,x$

Une fois les paramètres de l'énoncé saisis par le concepteur d'un exercice, PépiGen génère automatiquement la grille de codage de l'exercice, c'est-à-dire l'ensemble des réponses anticipées à cet exercice et le codage associé à ces réponses. Dans l'exercice que nous prenons en exemple, le raisonnement algébrique consiste à produire des expressions équivalentes en utilisant des règles de développement et de réduction des expressions jusqu'à obtenir la forme réduite (la

valeur 18 pour l'expression algébrique donnée en exemple). Pépinière produit les différentes suites d'expressions anticipées : les réponses correctes attendues et les réponses partielles ou erronées. La Figure 3 présente une copie d'écran décrivant une partie de la grille de codage. Deux grands types de justifications sont anticipées : les justifications utilisant le calcul algébrique (codées J1, L1 si elles sont correctes et J3, L3 sinon) et celles utilisant un exemple numérique (codées J2, L5). Certains élèves tentent une justification par l'algèbre qu'ils abandonnent pour prendre un exemple numérique (démarche codée J2, L2). Les expressions numériques ont la même structure que les expressions algébriques. D'un point de vue informatique, il suffit donc de générer ces dernières. Les élèves traduisent le programme de calcul en langage algébrique soit par une expression globale (codée T1 si elle est correctement parenthésée, codée T3 sinon), soit par des expressions partielles (Figure 3) traduisant chaque pas de calcul (codées T2 si la suite d'expressions est correcte et T4 sinon), soit par une équation (codée T2 si le premier membre est correctement parenthésé, codée T3 sinon). Ensuite, chaque passage d'une expression à une autre met en œuvre des règles de calcul correctes (codées EA1) ou incorrectes (dans ce cas le code indique le type d'erreur).

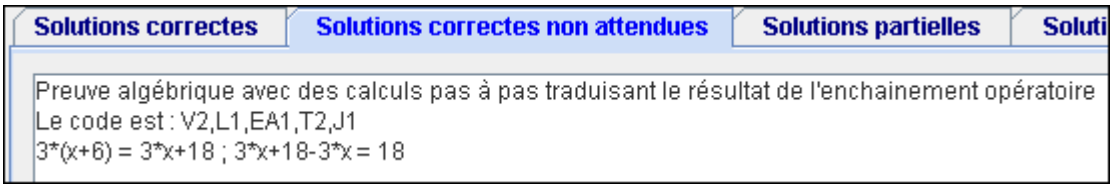


Figure 3 : Génération des expressions partielles pour l'expression 3 (x+6)3x

Pépinière construit d'abord l'arbre d'expression correspondant à l'expression (E6) selon la procédure détaillée dans la section 2.1.1. Il retourne l'arbre représenté à la Figure 4.

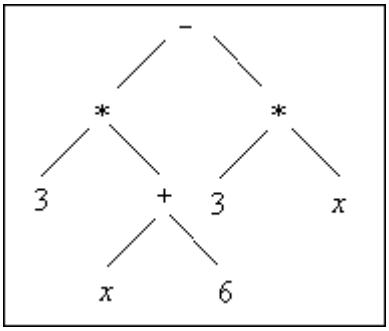
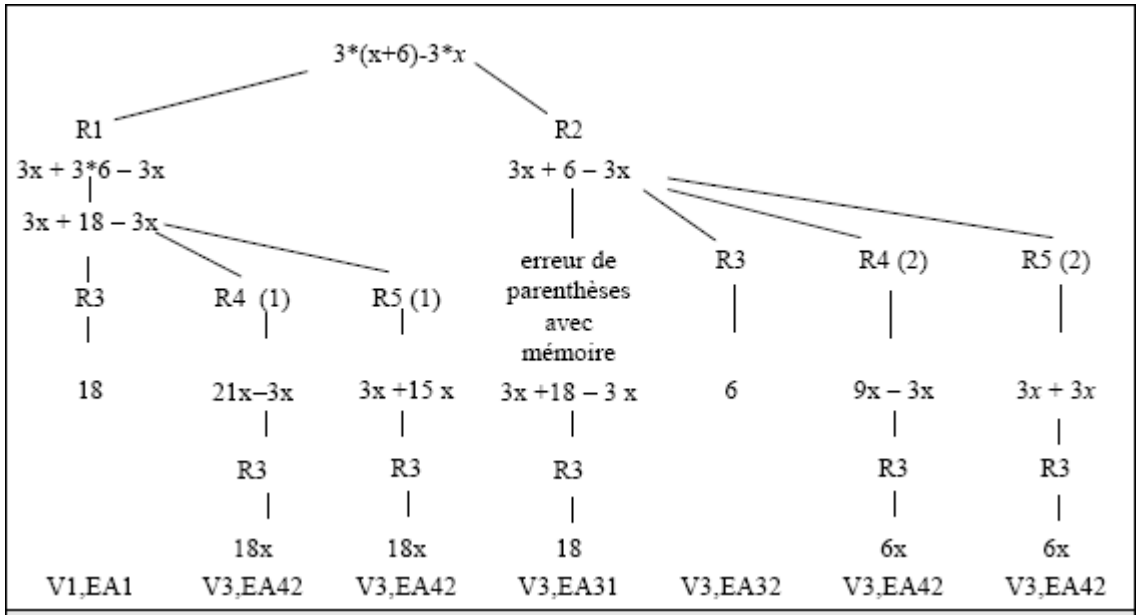


Figure 4 : arbre d'expression de l'expression algébrique 3 (x+6)-3x

PépiGen demande à Pépinière de produire les réponses anticipées pour chacune des démarches prévues dans l'analyse didactique. Partant d'une expression algébrique initiale, Pépinière applique les règles correctes ou erronées qui permettent de la transformer. Puis, Pépinière construit un arbre général dont les nœuds sont étiquetés par les expressions engendrées,

la règle qui a permis de l'engendrer et les codes qui ont pu être attribués à cette étape. Les feuilles de l'arbre que nous appelons *arbre des solutions anticipées* contiennent le code obtenu. La Figure 5 représente l'arbre des solutions anticipées construit pour la démarche algébrique utilisant une expression globale parenthésée. La section 6 décrit la construction de l'arbre des solutions anticipées. Dans la section 4 nous présentons les heuristiques et les stratégies que nous avons définies dans le choix des règles que nous appliquons aux expressions algébriques pour éviter les bouclages et l'explosion combinatoire ainsi que la méthode de construction de l'arbre des solutions anticipées.



Description des règles appliquées		
Règles correctes	R1 : A(B+C) → AB+AC	R3 : AC+BC→ (A+B)C
Règles erronées	R2 : A(B+C) → AB+C	appliquée à : 3*(x+6)
	R4 : AB+C→ B*(A+C)	appliquée à : (1) 3x + 18 (2) 3x + 6
	R5 : A+BC→ (A+B)*C	appliquée à : (1) 18 – 3x (2) 6 – 3x
Description des codes		
Validité	V1 : traitement correct	V3 : traitement incorrect
Utilisation des règles d'Écriture et de réécriture Algébrique	EA1 : Utilisation correcte des règles de transformation	
	EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct Remarque : certains élèves n'indiquent pas de parenthèses mais en conservent le sens. Cette erreur est codée EA31 (erreur de parenthèse avec mémoire) alors qu'une erreur de parenthèse sans mémoire de l'énoncé est codée EA32.	
	EA32 : Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect	
	EA42 : Les règles de transformation utilisées assemblent les termes	

Figure 5 : Démarche algébrique : expression globale parenthèsée

2.3. Architecture fonctionnelle de Pépinière

Pépinière est composé de quatre modules :

- Un module de traitement des expressions algébriques (**CalculAlgébrique**) qui est le point d'entrée de Pépinière. Il gère les différentes étapes des calculs algébriques dont les transformations d'expression algébrique en utilisant des règles de transformation d'écriture,
- Un analyseur syntaxique (**Analyseur**) qui, étant donnée une expression algébrique, vérifie si c'est une expression bien formée et construit un arbre binaire la représentant que nous appelons *l'arbre de l'expression*,
- Un module qui manipule les arbres d'expression (**ArbreExpression**): création de l'arbre, transformations de l'arbre par application des règles de réécriture, et fait la comparaison des expressions algébriques en vérifiant si deux expressions algébriques sont équivalentes (au sens didactique de la remarque du scénario 1),
- Un générateur des solutions anticipées (**GenerateurSolution**) qui génère l'arbre général appelé *arbre des solutions anticipées*. Chaque branche représente un raisonnement pour résoudre un problème de type « développer et réduire ». Cet arbre contient toutes les solutions anticipées obtenues par application de règles de transformation correctes ou erronées, mises en évidence dans le cadre du projet Pépite.

La Figure 6 illustre cette architecture et les principales classes Java utilisées dans ces modules et les sections suivantes décrivent précisément le cadre théorique, la conception et la réalisation de chacun de ces modules.

3. Analyse syntaxique des expressions algébriques dans Pépinière

Au niveau collège les programmes officiels mentionnent les « expressions littérales » sans en donner de définition. Le terme « expressions algébriques » n'apparaît qu'au niveau seconde [B.O. 2001]. Depuis les nouveaux programmes de mathématiques parus en 2005 [BO 2005], certains manuels les définissent, à destination des élèves, comme des expressions dans lesquelles des nombres et des grandeurs sont représentés par des lettres. Pour le manuel scolaire de quatrième [Transmath 2005] une expression littérale ou expression algébrique exprime un programme de calcul sur les nombres. En ce qui concerne Pépinière notre objectif est de formaliser les objets mathématiques en jeu, aussi nous définissons une expression algébrique comme une suite de symboles qui peut être obtenue en appliquant une grammaire décrite dans cette section. Examinons d'abord les différentes représentations des expressions algébriques utilisées dans le projet Pépite : celles des utilisateurs (élèves, professeurs, concepteurs d'exercices) et celles de la

machine. Puis nous détaillons la façon dont Pépinière traduit les expressions algébriques saisies par l'utilisateur en représentation machine.

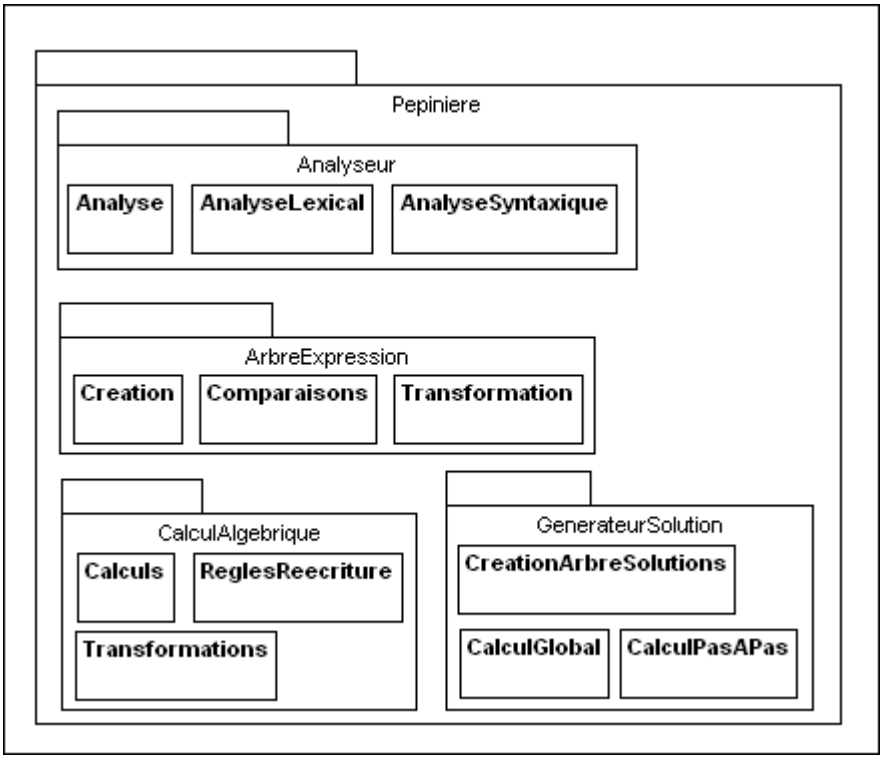


Figure 6 :Architecture de Pépinière

3.1. Représentation des expressions algébriques dans Pépinière

Du côté des utilisateurs, trois types de représentation sont utilisés dans Pépite : les représentations usuelles pour l'écriture papier-crayon, les représentations pour la saisie au clavier et les descriptions en langage naturel. Une représentation plane d'arbres d'expression est proposée par certains enseignants ou par certains logiciels (e.g. par Gélis dans le logiciel AILE [Delozanne et al. 2005] ou dans Aplusix [Nicaud 2007]). Cette représentation n'est cependant pas très répandue dans les classes. Nous ne l'avons donc pas retenue dans le travail présenté ici.

Du côté machine, de façon classique, deux types de représentation sont utilisés : les représentations linéaires (e.g. polonaises postfixée, infixée, préfixée) et les représentations sous forme d'arbres binaires ou d'arbres généraux. Pour des raisons de simplicité, nous avons retenu, pour la mémorisation des expressions, la représentation infixe qui est proche de la représentation usuelle et, pour les comparaisons et les transformations d'expression, la représentation en arbre binaire.

Précisons chacune de ces représentations et leur contexte d'utilisation qui sont résumés dans le Tableau 1.

3.1.1. Représentation usuelle

Exemple : $x+3x^2-\frac{x}{2}$

C'est une représentation plane (pour l'écriture des puissances et des quotients) utilisée par les humains, sur le papier ou au tableau. Cette représentation s'appuie sur des implicites : l'opérateur de multiplication n'est pas écrit (e.g. 2x signifie 2*x) et les règles de priorité des opérateurs sont sous-entendues. PépiGen, utilise cette représentation pour afficher des expressions à l'écran à l'aide d'un logiciel nommé Jaxe [Jaxe].

3.1.2. Représentation linéaire usuelle

Exemple : x + 3x ^ 2 – x / 2

Lorsqu'un utilisateur saisit une expression sur un clavier, il ne peut pas utiliser les quotients ni les exposants ou les indices. Il doit faire précéder les exposants par ^, et les numérateurs et dénominateurs doivent être parenthésés. Cette représentation linéaire infixe sous-entend encore certains symboles multiplicatifs et les règles de priorité pour les opérateurs. Dans Pépite, les humains utilisent donc cette représentation pour saisir des expressions au clavier.

3.1.3. Représentation linéaire

Exemple : x + 3*x ^ 2 – x / 2

C'est la représentation linéaire usuelle qui explicite tous les symboles multiplicatifs mais sous-entend encore les règles de priorité pour les opérateurs. Dans Pépite, comme c'est un cas particulier de la représentation précédente, les humains peuvent aussi l'utiliser pour saisir des expressions. Principalement, cette représentation est celle que le système utilise pour la sauvegarde des expressions.

3.1.4. Représentation sous forme d'arbre binaire

Exemple : la Figure 7 donne une représentation plane sous forme d'arbre binaire de l'arbre de l'expression : $x + 3\ x^2 -\ x / 2$.

Nous adoptons la définition récursive suivante [Aho et al. 1993]. Un arbre d'expression est un arbre binaire étiqueté qui est :

- soit une feuille dont l'étiquette est un nombre ou une variable,

- soit un nœud dont l'étiquette est un opérateur et les deux fils sont des arbres d'expressions.

Cette représentation fait apparaître la structure de l'expression. Elle n'a pas besoin d'être complétée par des règles de priorité. De façon classique, au niveau informatique, Pépinière utilise cette représentation pour transformer les expressions algébriques et les comparer [Aho et al., 1993]. Du point de vue programmation, ces arbres sont réalisés en utilisant des pointeurs « fils gauche », « fils droit ».

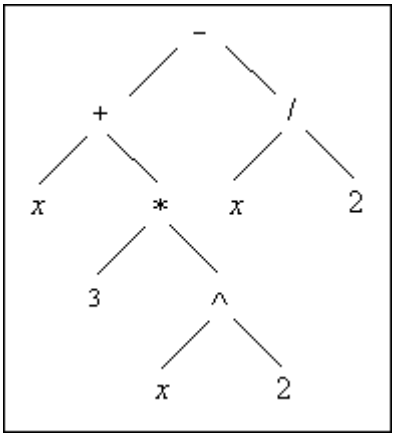


Figure 7 : Représentation plane de l'arbre de l'expression $x + 3\,x^2 - x / 2$

Nom de la représentation	Exemple	Humain	Machine
Usuelle plane	$x+3x^2-\frac{x}{2}$	- Ecriture Papier-crayon - Au tableau	Affichage à l'écran
Usuelle linéaire	x + 3x ^ 2 – x / 2	- Saisie au clavier - Dans Pépite	Affichage à l'écran
Linéaire	x + 3 *x ^ 2 – x / 2	- Saisie au clavier - Dans Pépite	Sauvegarde
Arbre binaire	figure 7	Pour faire apparaître la structure de l'expression	Arbres binaires réalisés par pointeurs fils gauche, fils droit

Tableau 1 : Représentation des expressions algébriques et contextes d'utilisation

3.1.5. Traduction d'une représentation linéaire en arbre d'expression

Lorsqu'un utilisateur saisit au clavier une expression algébrique (sous forme linéaire usuelle), Pépite la mémorise dans une chaîne de caractères. Pépinière effectue sur cette chaîne de caractères d'abord une analyse lexicale puis une analyse syntaxique.

3.2. Analyse lexicale

Tout d'abord, Pépinière applique un prétraitement qui consiste en une analyse lexicale de la chaîne de caractères: (i) suppression des caractères qui ne font pas partie des symboles attendus dans une expression algébrique (e.g. espace, ?), (ii) conversion des majuscules en minuscules, des crochets en parenthèses, des caractères spéciaux « × » et « ² » en « * » et « ^ 2 », (iii) ajout des symboles multiplicatifs implicites, (iv) mise en évidence des éléments lexicaux qui sont les nombres entiers relatifs, les variables (les 26 lettres de l'alphabet), les 5 symboles opératoires (le plus, le moins, l'étoile, la barre de fraction et l'accent circonflexe) et les parenthèses ouvrantes et fermantes. Le Tableau 2 résume les éléments lexicaux (les lexèmes) pris en compte dans cette analyse. L'ajout des symboles multiplicatifs est réalisé par un automate d'états finis.

Types	Entiers relatifs	26 lettres de l'alphabet	Opérateurs	Parenthèses
Lexèmes	*nombre*	*lettre*	+ – * / ^	( )

Tableau 2 : Les lexèmes de Pépinière

3.3. Analyse syntaxique

Après cette analyse lexicale, la représentation linéaire de l'expression saisie, qu'elle soit bien formée ou non, est une liste de lexèmes. Ensuite, Pépinière traduit cette liste en un arbre d'expression à la condition qu'elle représente une expression algébrique bien formée. Pour mener à bien cette analyse syntaxique nous avons adapté à notre contexte, les techniques éprouvées d'informatique fondamentale.

[Aho et al. 1993] décrivent une méthode d'analyse syntaxique descendante à partir de la définition d'une grammaire hors-contexte. Ils illustrent leur démarche avec une grammaire simple sur les expressions arithmétiques. Ces auteurs et [Jouannaud 2007] précisent les conditions à respecter pour que cette méthode soit déterministe : cette grammaire doit être de type LL(1). Nous avons donc choisi d'adopter ce cadre théorique pour fonder l'analyseur que nous avons mis au point. Les sections qui suivent décrivent les étapes de la construction de la grammaire correspondant à notre analyseur.

D'abord nous présentons la grammaire hors-contexte que nous avons retenue pour prendre en compte les opérations adaptées au niveau scolaire où nous nous situons : les nombres sont des nombres rationnels, les opérations sont l'addition, la soustraction, la multiplication, la division, l'opposé et l'exponentiation. Puis nous transformons cette grammaire en grammaire LL(1) pour assurer le déterminisme de l'analyse. Enfin nous présentons l'algorithme mis en œuvre dans notre analyseur.

3.3.1. Définition de la grammaire algébrique

Les symboles terminaux V_t de la grammaire G sont les lexèmes représentés dans le Tableau 2.

$$V_t = \{+ , \ - , * , \ / , \wedge , (,) , \textit{lettre}, \ \textit{nombre}\}$$

Nous traduisons la définition d'une expression algébrique et les règles de priorité associées aux opérateurs par les règles de dérivation présentées dans le Tableau 3.

Soit
- un ensemble de symboles non terminaux V_n représentés par les lettres E, T, S, F,
- E l'axiome de la grammaire,

1. $E \rightarrow ST \mid E + T \mid E - T$
2. $S \rightarrow - \mid \varepsilon$ ($S \rightarrow \varepsilon$ est une production vide).
3. $T \rightarrow F \mid T * F \mid T / F$
4. $F \rightarrow P \mid F \wedge P$
5. $P \rightarrow \textit{nombre} \mid \textit{lettre} \mid (E)$

Tableau 3: Grammaire G des expressions algébriques de Pépinière

A titre d'exemple, l'expression a + b*c est obtenue à partir de l'axiome E par les suites de dérivations suivantes :

$E \rightarrow E + T \rightarrow ST + T \rightarrow T + T \rightarrow F + T \rightarrow P + T \rightarrow a + T \rightarrow a + T * F \rightarrow a + F * F \rightarrow a + P * F \rightarrow a + b * F \rightarrow a + b * P \rightarrow a + b * c$

Sur cet exemple, pour obtenir cette dérivation à partir de l'axiome, plusieurs choix ont été nécessaires dans l'application des règles de dérivation. Par exemple : la suite de dérivations (1) ne permet pas d'aboutir à l'expression a + b * c et implique un retour en arrière pour choisir une autre règle de dérivation. Pour éviter ces retours en arrière, le choix entre plusieurs règles de dérivation peut se faire grâce à la connaissance du prochain symbole terminal à produire, appelé aussi caractère d'avance. Ainsi dans l'exemple proposé la présence du symbole terminal « + » oriente vers le choix de la règle de dérivation $E \rightarrow E + T$.

(1) $E \rightarrow ST \rightarrow T \rightarrow T * F \rightarrow \ldots$

Une grammaire pour laquelle le choix de l'application d'une règle de dérivation est déterminé par la connaissance du caractère suivant est dite grammaire de type LL(1). Selon Jouannaud, pour obtenir une grammaire LL(1) équivalente à la grammaire initiale, il faut transformer celle-ci, en particulier éliminer la récursivité gauche, afin que la condition de déterminisme énoncée dans le

tableau 5 soit vérifiée. De plus une grammaire LL(1) est non ambiguë puisque le choix de la règle à appliquer est entièrement déterminé par la connaissance du caractère suivant. Ainsi, pour chaque chaîne analysée, il existe un seul arbre d'expression construit avec la grammaire.

L'analyse d'une expression algébrique s'arrête lorsque l'analyseur arrive à la fin de la chaîne des symboles terminaux (ici les lexèmes de Pépinière) qui représente l'expression algébrique. Par facilité, nous ajoutons un symbole terminal (noté #) à la grammaire ainsi qu'une règle ($A \rightarrow E\ \#$) qui devient l'axiome de la grammaire : ce symbole marque la fin de l'analyse.

3.3.2. Élimination de la récursivité à gauche

Dans notre grammaire trois règles sont récursives à gauche :

$$1.\ E \rightarrow ST \mid E + T \mid E - T$$

$$3.\ T \rightarrow F \mid T * F \mid T / F$$

$$4.\ F \rightarrow P \mid F \wedge P$$

Nous remplaçons la règle 1. par les deux règles :

$$E \rightarrow S\ T\ E' \quad \text{et} \quad E' \rightarrow + T'\ E' \mid -TE' \mid \varepsilon$$

Nous remplaçons la règle 3. par les deux règles :

$$T \rightarrow F\ T' \quad \text{et} \quad T' \rightarrow * F\ T' \mid / F\ T' \mid \varepsilon.$$

Nous faisons de même pour la règle 4. qui se transforme en :

$$F \rightarrow F'\ P \quad \text{et} \quad F' \rightarrow \wedge P\ F' \mid \varepsilon.$$

Nous obtenons ainsi la grammaire G' dont les règles de dérivation sont données dans le Tableau 4. Dans la section suivante nous montrons que la grammaire G' est déterministe.

Soit
- un ensemble de symboles non terminaux Vn représentés par lettres E, E', T, T', S, F, F'.
- A l'axiome de la grammaire

1. A $\rightarrow$ E #
2. E $\rightarrow$ S T E'
3. S $\rightarrow$ − | ε
4. E' $\rightarrow$ + T E'| –T E' | ε
5. T $\rightarrow$ F T'
6. T' $\rightarrow$ * F T' | / F T' | ε
7. F $\rightarrow$ P F'
8. F' $\rightarrow$ ^ P F' | ε
9. P $\rightarrow$ *nombre* | *lettre* | (E)

Tableau 4 **:** Grammaire G' des expressions algébriques de Pépinière

3.3.3. Conditions de déterminisme

Sans reprendre les démonstrations de Jouannaud, nous présentons ici comment nous avons appliqué sa méthode en l'illustrant avec un exemple simple. Le choix entre plusieurs règles applicables se faisant par la connaissance du caractère à analyser, pour toute chaîne α formée de symboles terminaux et non terminaux, nous déterminons l'ensemble appelé *Premier* des symboles terminaux qui peuvent débuter une chaîne dérivée à partir de α par G.

Ainsi, *Premier* (P) = {*nombre*, *lettre*, (} et nous pouvons choisir l'une des trois règles de dérivation pour le symbole P ou signaler une erreur si l'un de ces symboles terminaux n'est pas le caractère à analyser mais ceci n'est pas suffisant : une règle donnant une chaîne vide ne peut pas être choisie. Ainsi, pour la règle 4, *nombre* n'appartient ni à *Premier*(+ T E'), ni à *Premier*(– T E') et, une erreur est produite si le premier caractère à analyser est *nombre*.

Aussi pour les règles de production donnant une chaîne dérivée vide, nous déterminons l'ensemble des terminaux qui peuvent se trouver juste après un non terminal dans la chaîne dérivée. Cet ensemble est noté *Suivant.*

Prenons pour exemple une étape de l'analyse de l'expression algébrique a+b*c, où nous avons pour chaîne dérivée, aT'E'#, et pour caractère à analyser le symbole + :

ce symbole terminal étant un élément de l'ensemble *Suivant*(T'), nous pouvons choisir la règle T' $\rightarrow$ ε pour obtenir la chaîne dérivée aE'# puis a+TE' par application de la règle 4 E' $\rightarrow$ + T E'.

Pour chaque règle telle que N $\rightarrow \alpha$, déterminons l'ensemble de tous les symboles terminaux qui peuvent débuter une dérivation par cette règle, ensemble appelé *Prédiction* par Jouannaud. Si les ensembles Prédiction sont disjoints pour tous les couples de règles N $\rightarrow \alpha$ et N $\rightarrow \beta$, l'analyse descendante est déterministe [Jouannaud 2007]. Le Tableau 5 donne une définition formelle des concepts *Premier*, *Suivant* et *Prédiction* et énonce la condition de déterminisme.

Définitions

Soit $V = V_t \cup V_n$, V_t étant l'ensemble des symboles terminaux et V_n l'ensemble des symboles non-terminaux de la grammaire.

V* est l'ensemble des mots, dont le mot vide ε, que l'on peut former avec le vocabulaire V.

$\overset{*}{\rightarrow}$ représente la suite de dérivations qui transforme $\alpha \rightarrow \beta$ avec $\alpha \in V^*$ et $\beta \in V^*$.

Premier ($N \in V_n$) = $\{a \in V_t \mid \exists \beta \in V^* \quad N \overset{*}{\rightarrow} a\beta\}$

Premier ($\alpha \in V^*$) = $\{a \in V_t \mid \exists \beta \in V^* \quad \alpha \overset{*}{\rightarrow} a\beta\}$

Suivant ($N \in V_n$) = $\{a \in V_t \mid \exists P \in V_n \quad P \overset{*}{\rightarrow} \alpha N\beta$ avec $a \in$ *Premier*(β) ou $\beta \overset{*}{\rightarrow} \varepsilon$ et $a \in$ *Suivant*(P)$\}$

Prédiction ($N \rightarrow \alpha$) = *Premier* ($\alpha \in V^*$) $\cup$ *Suivant* ($N \in V_n$) si $\alpha \overset{*}{\rightarrow} \varepsilon$, *Premier* ($\alpha \in V^*$) sinon

avec *Premier* (ε) = $\varnothing$

Condition de déterminisme

Une grammaire est dite LL(1) si elle satisfait la condition :

Pour chaque paire de règles N $\rightarrow \alpha \mid \beta$, *Prédiction* ($N \rightarrow \alpha$) $\cap$ *Prédiction* ($N \rightarrow \beta$) = $\varnothing$

Tableau 5 : Définition des ensembles *Premier*, *Suivant* et *Prédiction* et condition de déterminisme [Jouannaud 2007, p. 54]

Le Tableau 6 présente les ensembles *Premier*, et *Suivant* pour les symboles non-terminaux et pour les chaînes de symboles terminaux et non-terminaux situées en membre droit des règles de dérivation de la grammaire G' utilisée dans Pépinière. En utilisant les résultats du Tableau 6 et la définition donnée dans le Tableau 5, nous en déduisons l'ensemble *Prédiction* présenté dans le Tableau 7. Ce tableau montre que pour les règles 3a et 3b les ensembles *Prédiction* sont disjoints. Il en est de même pour les règles 4, 6, 8 et 9. La condition de déterminisme est vérifiée.

V_n	*Premier*	*Suivant*	Chaîne	*Premier*
A	{– , *nombre* , *lettre*, (}		E #	{– , *nombre* , *lettre*, (}
E	{– , *nombre* , *lettre*, (}	{) , #}	ST E'	{– , *nombre* , *lettre*, (}
S	{–}	{*nombre* , *lettre* , (}	+ T E'	{ + }
E'	{ +, – }	{) , #}	–T E'	{ – }
T	{*nombre* , *lettre* , (}	{+, – ,) , # }	F T'	{*nombre* , *lettre* , (}
T'	{ *, / }	{+, – ,) , # }	* F T'	{ * }
F	{*nombre* , *lettre* , (}	{* , / , – , + , # ,)}	/ F T'	{ / }
F'	{ ^ }	{* , / , – , + , # ,)}	P F'	{*nombre* , *lettre* , (}
P	{*nombre* , *lettre* , (}	{ ^ , * , / , – , + , # ,)}	^ P F'	{ ^ }
			P → *nombre*	{ *nombre* }
			P → *lettre*	{ *lettre* }
			P → (E)	{ (}

Tableau 6 **:** Ensembles *Premier* et *Suivant* pour la grammaire G'.

Le Tableau 8 présente la table d'analyse de la grammaire que nous avons construite à partir de l'ensemble *Prédiction*. Les colonnes correspondent aux symboles terminaux et les lignes aux membres gauches des règles de dérivation de la grammaire G'. A l'intersection d'une ligne et d'une colonne figure le membre droit de la règle de dérivation à appliquer selon le caractère lu d'avance. Les cases comportant le symbole ε correspondent à l'application de la règle de dérivation qui produit le mot vide et les cases vides correspondent à une erreur. En effet, si le caractère lu d'avance n'appartient pas aux ensembles *Prédiction* des règles de dérivation concernées, l'analyse syntaxique s'arrête et un message d'erreur précise le caractère attendu.

> Par exemple la règle 1, A → E # qui est l'axiome de la grammaire ne pourra être appliquée que si le caractère lu appartient à l'ensemble *Prédiction*(A), soit à l'ensemble {– , *nombre* , *lettre*, (}.

La table d'analyse reprend l'ensemble des résultats du Tableau 7 et a été utilisée pour écrire l'algorithme d'analyse descendante et les différentes procédures qui en découlent.

N°	Règle	*Prédiction*
1	A → E #	{– , *nombre* , *lettre*, (}
2	E → S T E'	{– , *nombre* , *lettre*, (}
3a	S → –	{–}
3b	S → ε	{*nombre* , *lettre* , (}
4a	E' → + T E'	{ + }
4b	E' → –T E'	{ – }
4c	E' → ε	{) , #}
5	T → F T'	{*nombre* , *lettre* , (}
6a	T' → * F T'	{ * }
6b	T' → / F T'	{ / }
6c	T' → ε	{+, – ,) , # }
7	F → P F'	{*nombre* , *lettre* , (}
8a	F' → ^ P F'	{ ^ }
8b	F' → ε	{* , / , – , + , # ,)}
9 a	P → *nombre*	{ *nombre* }
9 b	P → *lettre*	{ *lettre* }
9 c	P → (E)	{ (}

Tableau 7 : Ensemble *Prédiction* des règles de dérivation de la grammaire G'**.**

3.3.4. Algorithme d'analyse

L'algorithme d'analyse descendante consiste à implémenter la grammaire par plusieurs procédures imbriquées, chacune correspondant à un ou plusieurs symboles non terminaux [Aho et al 1993]. Nous avons fait le choix de construire l'arbre d'expression au cours de l'analyse de l'expression algébrique. Nous distinguons cinq procédures pour cette analyse. Une première procédure, ANALYSE, qui correspond à l'axiome de la grammaire G', initialise le processus d'analyse, le symbole # correspondant à la fin de la liste des lexèmes produite par l'analyse lexicale. Une deuxième procédure, EXPRESSION, correspond au traitement des symboles non-terminaux E, S et E', une troisième procédure, TERME traite les symboles non-terminaux T et T', une procédure appelée FACTEUR traite les symboles non-terminaux F, F' et, enfin, une dernière procédure, TERMINAL, traite le symbole non-terminal P. La procédure d'analyse se termine lorsqu'une erreur est détectée (l'expression algébrique est mal formée) ou après la lecture du dernier lexème de la liste fournie par l'analyse lexicale. Le Tableau 9 présente l'algorithme mis en œuvre dans la procédure TERME et la figure illustre les étapes de la construction d'un arbre d'expression avec cette procédure. La Figure 8 présente deux arbres d'expression construits au cours de l'analyse des expressions $3(x+6)-3x$ et $(4x+6)/2-2x$, dessinés par une procédure graphique pour illustrer le résultat de la procédure ANALYSE.

3.3.5. Étiquette des arbres d'expression obtenus par l'analyseur

Les étiquettes des nœuds de l'arbre que ce soit un nœud intérieur ou un nœud de type feuille, sont des éléments structurés qui comportent plusieurs champs dont le genre du nœud (opérande ou opérateur), le signe, la valeur absolue, le type (variable, nombre, autre), le poids et le numéro du nœud qui sont utilisés dans certaines transformations des expressions algébriques ainsi que pour leur comparaison.

V_t / V_n	–	+	–	*	/	^	*nombre*	*lettre*	(	)	#
A	E #						E #	E #	E #		
E	S T E'						S T E'	S T E'	S T E'		
S	–						ε	ε	ε		
E'		+ T E'	– T E'							ε	ε
T							F T'	F T'	F T'		
T'		ε	ε	* F T'	/ F T'					ε	ε
F							P F'	P F'	P F'		
F'		ε	ε	ε	ε	^ P F'				ε	ε
P							*nombre*	*lettre*	(E)		

Tableau 8 : Table d'analyse de la grammaire des expressions algébriques de Pépinière

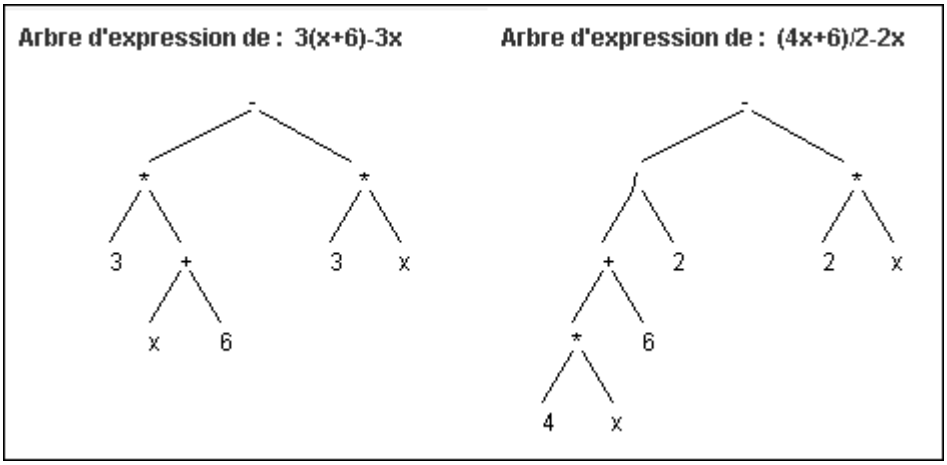


Figure 8 : Représentation graphique d'arbres d'expression générés par l'analyseur.

Procédure TERME

Remarque : les numéros seront utilisés dans la figure 9

Traitement de T (T → F T')

si *lexème courant* appartient à l'ensemble {nombre, lettre, (}

alors *lexème courant* = FACTEUR (*lexème courant*) ①

sinon

erreur et retour

fin si

Traitement de T' (T' → * F T' ou T' → / F T')

tant que *lexème courant* appartient à l'ensemble { *, / }

- Création d'un *nœud* étiqueté par le symbole * ou / ②

- Le fils gauche de *nœud* pointe vers le sous-arbre obtenu en retour des appels précédents de la procédure FACTEUR ③

- Lecture de *lexème suivant* ④

si *lexème suivant* appartient à l'ensemble {nombre, lettre, (}

alors (traitement du non-terminal F)

lexème courant = FACTEUR (*lexème suivant*) ⑤

sinon

erreur et retour

fin si

Le fils droit de *nœud* pointe vers le sous-arbre obtenu en ⑤ au retour de l'appel de la procédure FACTEUR ⑥

fin tant que

si *lexème courant* ≠) ou si ce n'est pas la fin de la chaîne à analyser **(T' → ε)**

alors

erreur

fin si

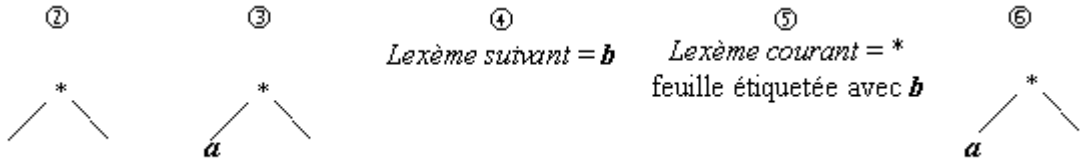
Tableau 9 : Algorithme de la procédure TERME

Construction de l'arbre d'expression de a*b*c+d

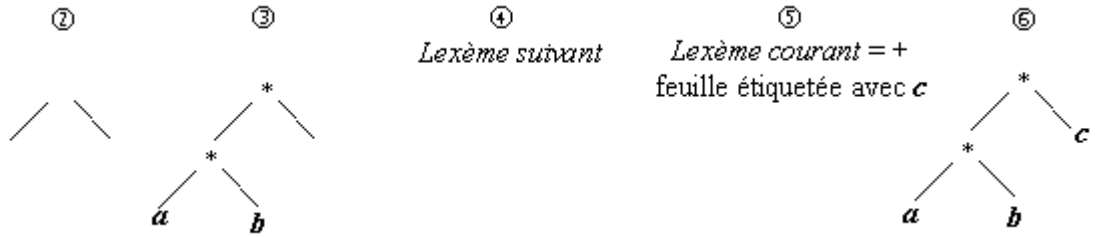
La numérotation reprend celle utilisée dans le tableau 9
Après les appels de la procédure ANALYSE puis EXPRESSION le *lexème courant* est la lettre ***a***.

En ① après le premier appel de la procédure FACTEUR nous avons :
- *Lexème courant* = *
- Une feuille étiquetée avec ***a***

Premier passage dans la boucle **tant que** de la procédure TERME



Deuxième passage dans la boucle **tant que** de la procédure TERME



Sortie de la boucle **tant que** de la procédure TERME

Retour dans la procédure EXPRESSION pour construire le nœud étiqueté avec + ayant pour fils gauche le sous-arbre **a*b*c**

Appel des Procédures TERME, FACTEUR et TERMINAL : nous obtenons une feuille étiquetée avec ***d*** qui devient le fils droit du nœud étiqueté avec +

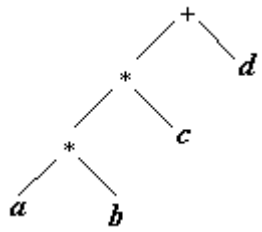


Figure 9 : Construction de l'arbre d'expression a*b*c+d

4. Transformation des expressions algébriques

Dans les programmes scolaires officiels, l'utilisation d'expressions littérales donnant lieu à des calculs numériques est préconisée dès la classe de cinquième. L'apprentissage du calcul

littéral est conduit pour développer et réduire des expressions littérales en cinquième puis pour les factoriser en quatrième [B.O. 2001, 2005, 2007]. Ces apprentissages sont repris et développés au lycée. Dans l'idéal, pour réaliser ces exercices, les élèves utilisent des propriétés (e .g. la distributivité de la multiplication sur l'addition puis les identités remarquables) et mobilisent des connaissances stratégiques pour reconnaître les différentes écritures d'une expression et choisir la forme la plus adaptée au travail demandé. Dans la pratique, les enseignants et les didacticiens des mathématiques ont mis en évidence un certain nombre de démarches inadaptées et de règles erronées qui apparaissent de façon récurrente dans les productions des élèves. Le scénario trois (section 2.3) donne des exemples de tels dysfonctionnements.

Pour établir un diagnostic, la première étape consiste à expliciter l'ensemble de ces règles correctes ou erronées et à leur associer un code. Le Tableau 11 donne la liste des règles définies par les didacticiens de l'équipe pour les exercices de PépiTest. A partir de cet ensemble de règles, notre travail d'informaticien a consisté à concevoir un générateur des réponses anticipées des élèves qui applique ces règles de transformation correctes ou erronées en évitant les bouclages et l'explosion combinatoire. Pour cela, nous nous sommes appuyés sur différents travaux d'intelligence artificielle, en particulier ceux de Laurière [Laurière 1988] et de Gélis [Gelis 1994]. Nous avons organisé cet ensemble de règles puis mis au point un ensemble d'heuristiques qui permettent d'en contrôler l'application (pour éviter les bouclages et l'explosion combinatoire). C'est le travail que nous présentons dans cette section. Nous nous restreignons à l'étude du diagnostic sur la dimension calcul algébrique qui concerne l'identification des règles de calcul utilisées par les élèves. Le diagnostic sur les dimensions « type de justification », et « traduction du registre de l'algèbre à un autre registre sémiotique » est traité au chapitre 7.

Dans cette section, nous présentons d'abord les concepts de transformation et de règle de réécriture ainsi que les questions de terminaison analysées dans la littérature. Puis, nous détaillons la modélisation des règles et l'organisation de ces règles que nous avons mises au point pour aborder le problème de la terminaison des transformations. Ensuite, nous explicitons les stratégies que nous avons mises en œuvre pour limiter le choix des règles et assurer la terminaison du programme tout en générant les réponses correctes ou erronées anticipées sur les exercices de Pépite. Enfin, nous décrivons comment nous utilisons l'algorithme d'unification, mécanisme d'inférence qui permet d'engendrer une nouvelle expression à partir d'une expression donnée et d'un ensemble de règles de transformation.

5. Transformation et règles de réécriture

Pour résoudre de nombreux exercices portant sur des expressions algébriques, les humains utilisent des *transformations* d'écriture. Ces transformations permettent d'engendrer une nouvelle expression à partir d'une expression donnée. Dans leur théorie des réécritures, Dershowitz et Jouannaud [Dershowitz & Jouannaud 1990] définissent ce mécanisme d'inférence, *remplacement d'égaux par des égaux* appelé *substitution*. Une sous-expression étant une suite de symboles de l'expression qui forme une expression algébrique au sens donné dans la section 3, une substitution permet de remplacer dans une expression, une sous-expression membre d'une égalité, par l'autre membre de l'égalité.

Prenons un exemple simple d'exercice du niveau scolaire qui nous préoccupe : « développer puis réduire l'expression $(x+5)^2+2x-5$ ». Dans l'expression $(x+5)^2+2x-5$, un élève idéal reconnaît l'identité remarquable $(a+b)^2 = a^2 +2ab + b^2$ telle qu'elle est enseignée actuellement en classe de troisième. Il remplace $(x+5)^2$ par $x^2+10x+25$ et obtient l'expression $x^2+10x+25+2x-5$ qu'il réduit pour aboutir à l'expression $x^2+12x+20$. Il a appliqué deux transformations que nous représentons avec le symbole $\rightarrow$ selon :

$$(x+5)^2+2x-5 \rightarrow x^2+10x+25+2x-5 \rightarrow x^2+12x+20$$

La première transformation utilise la *règle de réécriture* $(A+B)^2 \rightarrow A^2 +2AB + B^2$. Une règle de réécriture impose un sens dans l'utilisation d'une identité, ainsi, à l'identité utilisée dans l'exemple précédent, nous associons deux règles de réécriture : $(A+B)^2 \rightarrow A^2 +2AB + B^2$ et $A^2 +2AB + B^2 \rightarrow (A+B)^2$. Les lettres A et B sont appelées variables de réécriture et nous les notons avec des lettres majuscules afin de les différencier des symboles de variables utilisés dans les expressions algébriques. Ces variables ont pour rôle d'être remplacées par des expressions quelconques (e.g. A par x et B par 5).

5.1. Classes de transformations et terminaison

Gelis [Gelis 1994], dans son travail de thèse, s'est attaché à prouver un théorème concernant la terminaison d'un ensemble de règles et de transformations sur un ensemble d'expressions qui est celui sur lequel travaille Pépite. L'objectif de ce travail était de construire un cadre théorique mathématique pour fonder la conception d'un résolveur cognitif de référence pour des problèmes algébriques. Ce résolveur concerne les problèmes suivants : développer et réduire, simplifier des fractions, factoriser, et résoudre des équations. Dans le cadre de Pépite, nous nous limitons actuellement aux deux premiers types de problèmes qui, d'après [Nicaud et al 2005] sont les plus simples. Concernant la propriété de terminaison, ces travaux montrent entre autres que :

- Les transformations utilisant des règles de réduction se terminent,
- Les problèmes de développement et de réduction et les problèmes de simplification de fractions se terminent si l'on s'attache à limiter l'espace de choix des règles par l'utilisation de méta-règles [Laurière 88] ou de principes de stratégies [Gélis 1994] qui évitent d'appliquer une règle puis son inverse.

5.2. Transformations dans Pépinière

Dans la version actuelle, au niveau stratégique nous distinguons cinq types de problèmes qui utilisent des règles de transformation : (i) réduire une expression algébrique, (ii) développer une expression algébrique, (iii) factoriser une expression algébrique en utilisant une identité remarquable, (iv) effectuer des opérations sur les fractions, (v) effectuer des opérations sur les puissances. La résolution d'un type de problème peut nécessiter la résolution d'un sous-problème d'un autre type. Ainsi, par exemple, développer une expression demande ensuite de la réduire. De plus, pour éviter les bouclages, il faut contrôler la résolution en évitant d'appliquer en séquence des règles qui s'inhibent (par exemple AB + AC $\rightarrow$ A (B+C) et A (B+C) $\rightarrow$ AB + AC). Pour cela nous avons organisé les règles de transformation en distinguant plusieurs types de règles, puis, nous avons défini des heuristiques pour contrôler l'ordre d'application de ces différents types de règles en fonction du problème à résoudre. La mise au point de cette organisation est très délicate car une même règle peut dans certains contextes être utilisée pour résoudre un problème et dans un autre pour résoudre un autre problème. Par exemple la règle AB + AC $\rightarrow$ A (B+C) est une règle de factorisation (e.g. $2x^2 + 3x \rightarrow x(2x+3)$); mais sous certaines contraintes, si B et C sont des nombres et A est un monôme, c'est une règle de réduction (e.g. $2x^2 + 3x^2 \rightarrow (2+3)\ x^2$ ce qui donne $5x^2$ en évaluant la somme des deux nombres). On ne peut donc pas classer les règles uniquement en fonction du type de problème à résoudre.

5.2.1. Types de règles

Nous avons défini sept types de règles de réécriture numérotés de 1 à 7 et, pour chacun de ces types, deux catégories de règles : les règles correctes et les règles erronées. Le Tableau 11 donne la liste de ces règles. Pour déterminer ces types, nous sommes partis des propriétés de l'algèbre qui fondent le calcul algébrique. A une égalité entre deux expressions algébriques correspondent deux règles de réécriture qui s'inhibent. Par exemple, à l'égalité a(b+c) = ab + ac, qui traduit la distributivité de la multiplication par rapport à l'addition, correspondent les règles de réécriture A (B+C) $\rightarrow$ AB + AC et AB + AC $\rightarrow$ A (B+C) : une de ces règles est un développement et l'autre une factorisation ou une réduction de monômes. Afin d'éviter que des boucles

n'apparaissent et n'engendrent une suite infinie de transformations, ces règles n'appartiennent pas au même type de règles. Toujours, pour éviter ces bouclages, d'autres règles de réécriture qui ne sont cependant pas obtenues à partir de la même égalité doivent aussi appartenir à des types différents. Par exemple, en ce qui concerne les fractions, les règles qui permettent de réduire deux fractions au même dénominateur (e.g . règles d'addition de deux fractions) n'appartiennent pas au même type de règles que celles qui permettent de simplifier une fraction. Les expressions algébriques concernées par Pépinière étant des polynômes de degré deux maximum, le nombre de règles de réécriture utilisées est limité. Le mécanisme d'inférence utilisé pour appliquer les règles de réécriture de type 1, 2, 3, 5 ainsi que certaines règles de type 4 est l'algorithme d'unification. Pour les règles de type 6 et 7, nous appliquons un traitement particulier que nous présentons ainsi que les raisons de ce choix dans la section 5.4.

5.2.1.1. Règles de réécriture de type 1

Appartiennent à ce type les règles de réécriture qui correspondent à des développements et qui ne comportent pas le symbole de la division. Bien que les deux règles de réécriture $A(B+C) \rightarrow AB + AC$ et $(B+C)A \rightarrow BA + CA$ se déduisent l'une de l'autre en appliquant la propriété de commutativité de la multiplication, nous avons fait le choix de garder la possibilité d'appliquer l'une ou l'autre de ces deux règles car les réponses anticipées différent selon la règle utilisée. A la première règle de réécriture est associée une réponse erronée usuelle [Grugeon 1995] qui correspond à une application erronée de cette règle sous la forme $A (B+C) \rightarrow AB + C$ (règle erronée numéro 1) ce qui n'est pas le cas pour la règle de réécriture et $(B+C)A \rightarrow BA + CA$. Cette règle erronée permet ainsi d'identifier, au niveau du diagnostic local à l'exercice, une utilisation inadaptée des parenthèses qui aboutit à un résultat incorrect (codée EA32). Nous avons aussi choisi de mettre dans le même type les règles $(A+B)^2 \rightarrow A^2 + 2\,AB + B^2$ et $(A+B)(C+D) \rightarrow AC + BC + AD + BD$. Ces deux règles sont correctes. Mais selon le niveau scolaire de l'élève, leur utilisation est codée différemment. En quatrième, les élèves n'ont pas encore étudié les identités remarquables, les deux réponses constituent la réponse correcte attendue (codée V1) mais en troisième, la première est la réponse correcte attendue (codée V1) alors que la deuxième est correcte mais non attendue (codée V2) car elle dénote une maîtrise encore fragile ou en construction puisque l'identité remarquable n'est pas mobilisée.

5.2.1.2. Règles de réécriture de type 2

Ces règles de réécriture engendrent des transformations qui correspondent à la simplification de fractions, le dénominateur étant un nombre entier (e.g. la transformation $(5x+10)/2 \rightarrow 5x/2+5$)).

Ici, la règle de réécriture utilisée est (A+B)/C → A/C + B/C et dans ce cas il lui correspond la règle erronée (A+B)/C → A + (B/C) ou (A/C) + B.

5.2.1.3. Règles de réécriture de type 3

Les règles de type 3 permettent de réduire des fractions au même dénominateur, le dénominateur étant un nombre entier (e.g. la transformation (5x/2 + 3x → (5x+6x)/2). Ici la règle de réécriture utilisée est B/A+C → (B+AC)/A. Ces règles correspondent aux opérations avec des fractions (addition, soustraction de fraction). Elles produisent les transformations inverses de celles produites par les règles de type 2.

5.2.1.4. Règles de réécriture de type 4

Les transformations obtenues sont des factorisations. Nous nous limitons actuellement à quatre règles de réécriture pour les factorisations. Les règles de récriture qui correspondent aux identités remarquables au programme de la classe de troisième et la règle de réécriture AB + AC → A(B+C) avec A de type entier ou de type variable (e.g. 4x+6 = 2(x+3)). Cette dernière règle intervient notamment pour simplifier des fractions. L'application de l'algorithme d'unification ne peut pas être généralisé à ce type de règles qui nécessitent un pré-traitement : recherche préalable du carré d'une expression ou d'un nombre pour les identités remarquables, recherche d'un facteur commun pour la règle de réécriture AB + AC → A(B+C).

5.2.1.5. Règles de réécriture de type 5

Ces règles de réécriture concernent les propriétés des puissances.

5.2.1.6. Règles de réécriture de type 6

Cette règle de réécriture, AC+BC → (A+B) *C, où A et B sont des nombres et C est une puissance d'une variable, consiste à effectuer la somme de deux monômes semblables (de même degré) et correspond à la réduction d'une expression algébrique. L'application de cette règle nécessite un pré-traitement de l'expression (détaillé dans la section 5.2.1.7.) Nous utilisons une procédure que nous détaillons dans la section 5.4 qui recherche les monômes de même degré pour en faire ensuite la somme. Cette même procédure est appliquée pour les règles erronées de ce type : citons AC+B → (A+B)*C où A et B sont des nombres, C une puissance d'une variable (e.g. 3x+18 → 21x). A ce type de règles sont associées plusieurs règles (Tableau 11) qui assemblent les termes (codées EA41).

5.2.1.7. Règles de réécriture de type 7

Ces règles de réécriture font l'objet d'un traitement particulier et nous n'appliquons pas ici non plus l'algorithme d'unification pour engendrer une nouvelle expression avec ces règles. La règle $A \rightarrow 1*A$ qui pourrait s'appliquer à toutes les expressions ou sous-expression algébriques rencontrées est appliquée par Pépinière aux seuls monômes x et x^2 avant la réduction d'une expression algébrique (e.g. 2*x + x est transformé en 2*x + 1*x pour faire la somme des deux termes et obtenir 3 x). Après la réduction, la règle $1*A \rightarrow A$, inverse de la règle citée précédemment, ainsi que les deux règles $A * 0 \rightarrow 0$ et $A + 0 \rightarrow A$ sont utilisées si cela est nécessaire (e.g. le résultat de 3*x-2*x = 1*x est transformé en x).

5.2.2. Heuristiques et terminaison des transformations

Dans cette section nous présentons les heuristiques que nous avons définies et les stratégies de choix des règles de réécriture pour engendrer des transformations qui se terminent.

5.2.2.1. Réduction d'une expression algébrique

Nous avons déjà évoqué dans la section 5.2.1.7 l'utilisation des deux règles de réécriture $A \rightarrow 1*A$ et $1*A \rightarrow A$. Pour réduire une expression algébrique nous appliquons dans l'ordre :

1. La règle $A \rightarrow 1*A$ à tous les monômes x de l'expression algébrique à réduire,
2. La règle correcte de type 6, $AC+BC \rightarrow (A+B)*C$ où C est une puissance d'une variable, ainsi que les règles erronées,
3. Les règles $1*A \rightarrow A$, $A * 0 \rightarrow 0$ et $A + 0 \rightarrow A$.

La règle $A \rightarrow 1*A$, qui est une règle expansive, est utilisée une seule fois dans la réduction d'une expression algébrique et est inhibée par la règle $1*A \rightarrow A$ qui est utilisée ensuite. La transformation obtenue en utilisant les règles de réécriture dans cet ordre est une réduction qui se termine.

5.2.2.2. Développement d'une expression algébrique

Pour développer une expression algébrique nous appliquons :

1. Les règles de réécriture de type 1 pour développer l'expression,
2. Les règles de la section 5.2.2.1 dans l'ordre pour réduire l'expression algébrique obtenue.

La première transformation consiste à développer l'expression algébrique et la deuxième transformation consiste à réduire l'expression algébrique obtenue. La suite de transformations qui utilise ces règles de réécriture se termine.

5.2.2.3. Simplification de fractions

Pour simplifier une fraction nous appliquons :

1. Les règles de réécriture de type 2 pour simplifier l'expression,
2. Les règles de la section 5.2.2.1 dans l'ordre pour réduire l'expression algébrique obtenue.

Une première transformation consiste à simplifier une fraction et une deuxième transformation consiste à réduire l'expression algébrique obtenue. La suite de transformations qui utilise ces règles de réécriture se termine.

5.2.2.4. Réduction de fractions au même dénominateur

Pour réduire une fraction au même dénominateur nous appliquons :

1. Les règles de réécriture de type 3 pour réduire les deux fractions au même dénominateur,
2. Les règles de la section 5.2.2.1 dans l'ordre pour réduire l'expression algébrique obtenue,
3. Les règles de la section 5.2.2.3 dans l'ordre pour simplifier la fraction obtenue,
4. Puis de nouveau, les règles de la section 5.2.2.1 dans l'ordre pour réduire l'expression algébrique obtenue.

Nous utilisons les règles de réécriture de type 3 plusieurs fois si nécessaire pour réduire au même dénominateur et le processus s'arrête lorsque toutes les fractions de l'expression algébrique ont le même dénominateur. Cette transformation est suivie d'une réduction de l'expression formant le numérateur puis d'une simplification. Les transformations « Réduction au même dénominateur » et « Simplification de fraction » utilisent des règles de réécriture de type 2 et 3 qui s'inhibent et l'ordre dans lequel nous les utilisons, aboutit à une transformation qui se termine.

5.2.2.5. Factorisation

Comme nous l'avons déjà évoqué dans la section 1 nous n'avons pas particulièrement étudié les factorisations. Le Tableau 10 et le Tableau 11 présentent l'ensemble des règles de réécritures utilisées par Pépinière.

5.2.3. Discussion

Cette étude montre qu'une application systématique de l'algorithme d'unification ne permet pas d'assurer la terminaison des transformations. En particulier pour la réduction de monômes semblables nous n'utilisons pas cette algorithme mais une procédure spécifique qui gère aussi le problème des règles $A \rightarrow 1*A$ et $1*A \rightarrow A$ et que nous décrivons à la section 4.5.

Type	Numéro	Règles correctes « V »	Numéro	Règles erronées « F »
		1 : Développement		
1	1	$(A+B)\,(C+D) \rightarrow AC+BC+AD+BD$		
	2	$(A-B)\,(C+D) \rightarrow AC - BC+AD - BD$		
	3	$(A+B)\,(C-D) \rightarrow AC+BC - AD - BD$		
	4	$(A-B)\,(C-D) \rightarrow AC - BC - AD+BD$		
	5	$A\,(B+C) \rightarrow AB + AC$	5	$A\,(B + C) \rightarrow AB + C$
	6	$A\,(B-C) \rightarrow AB - AC$	6	$A\,(B-C) \rightarrow AB - C$
	7	$(B+C)\,A \rightarrow BA + CA$		
	8	$(B-C)\,A \rightarrow BA - CA$		
	9	$(A+B)^2 \rightarrow A^2 + 2\,AB + B^2$	8	$(A+B)^2 \rightarrow A^2 + AB + B^2$
			9	$(A+B)^2 \rightarrow A^2 + B^2$
	10	$(A-B)^2 \rightarrow A^2 - 2\,AB + B^2$	10	$(A-B)^2 \rightarrow A^2 + B^2$
	11	$(A-B)(A+B) \rightarrow A^2 - B^2$	11	$(A-B)(A+B) \rightarrow A^2 + B^2$
	12	$(A+B)(A-B) \rightarrow A^2 - B^2$		
		2 : Simplification des fractions		
2	13	$(A+B)/C \rightarrow (A/C) + (B/C)$	13	$(A+B)/C \rightarrow A + (B/C)$
	14	$(A-B)/C \rightarrow (A/C) - (B/C)$	14	$(A-B)/C \rightarrow A - (B/C)$
	15	$(AB)/C \rightarrow (A/C)\ B$		
		3 : Réduction au même dénominateur		
3	16	$(A/B)+(C/D) \rightarrow (AD+BC)/(BD)$	15	$(A/B)+(C/D) \rightarrow (A+C)/(B+D)$
			16	$(A/B)+(C/D) \rightarrow (A+C)/(BD)$
	17	$(A/B)-(C/D) \rightarrow (AD-BC)/(BD)$	17	$(A/B)-(C/D) \rightarrow (A-C)/(B-D)$
	18	$(A/B)+C \rightarrow (A+BC)/B$	18	$(A/B)+C \rightarrow (A+C)/B$
	19	$(A/B)-C \rightarrow (A-BC)/B$	19	$(A/B)-C \rightarrow (A-C)/B$
	20	$(A/B)(C/D) \rightarrow (AC)/(BD)$		
	21	$A(B/C) \rightarrow (AB)/C$	21	$A(B/C) \rightarrow (AB)/(AC)$
		4 : Factorisation		
4	22	$A^2 + 2\,AB + B^2 \rightarrow (A+B)^2$		
	23	$A^2 - 2\,AB + B^2 \rightarrow (A-B)^2$		
	24	$A^2 - B^2 \rightarrow (A+B)\,(A-B)$		
	25	$AB + AC \rightarrow A(B+C)$		
		5 : Puissance		
5			25	$A^m + A^n \rightarrow A^{m+n}$
	26	$A^m \times A^n \rightarrow A^{m+n}$	26	$A^m \times A^n \rightarrow A^{m*n}$
	27	$(A^m)^n \rightarrow A^{m*n}$	27	$(A^m)^n \rightarrow A^{m+n}$
	28	$(AB)^n \rightarrow A^n * B^n$	28	$(AB)^n \rightarrow A\,B^n$
	29	$(A/B)^n \rightarrow A^n / B^n$	29	$(A/B)^n \rightarrow A^n / B$
			30	$A^n = n\,A$

Tableau 10 : Règles de réécriture de type 1 à 5 utilisées par Pépinière

Type	Numéro	Règles correctes « V »	Numéro	Règles erronées « F »
		6 : Réduction d'un polynôme		
6		A et B sont des nombres, C une puissance d'une variable		A et B sont des nombres, C une puissance d'une variable
		addition de monômes sembables	31	AC+B $\rightarrow$ (A+B) C
				exemple : 3*x*+24 = 27*x*
	31	AC+BC $\rightarrow$ (A+B) C	32	A+BC $\rightarrow$ (A+B) C
				exemple : 3+24x = 27*x*
			33	AC-C $\rightarrow$ A-1
				exemple : 8x – x = 7 ou 23x + x = 24
			34	AC-C $\rightarrow$ A
				exemple : 8x – x = 8 ou 23x + x = 23
			35	AB + B $\rightarrow$ AB^2
				exemple :3x+x = $3x^2$
		7 : Propriétés particulières		
7	36	A $\rightarrow$ 1 * A élément neutre		
	37	A + 0 $\rightarrow$ A		
	38	A * 0 $\rightarrow$ 0 élément absorbant		
	39	1 * A $\rightarrow$ A		

Tableau 11 : Règles de réécriture de type 6 et 7 utilisées par Pépinière

5.3. Mécanisme d'inférence : unification

Reprenons l'exemple donné dans la section 4.1. L'expression de départ $(x+5)^2+2x-5$ est transformée en $x^2+10x+25+2x-5$ par application de la règle de réécriture $(A+B)^2 \rightarrow A^2 +2AB + B^2$. Nous disons que la substitution de A par x et de B par 5 permet d'unifier la sous-expression algébrique $(x+5)^2$ et le membre gauche de la règle de réécriture car nous obtenons $(x+5)^2$ en remplaçant A par x et B par 5. Cette substitution effectuée dans le membre droit de la règle de réécriture aboutit à l'expression algébrique $x^2+10x+25$ puis à la transformation de l'expression de départ. Une règle de réécriture étant choisie est-il toujours possible d'unifier une expression donnée avec le membre gauche de la règle ? Pour répondre à cette question nous utilisons l'algorithme d'unification. « *L'algorithme d'unification répond à la question : Comment appliquer un théorème à une expression ? Cet algorithme est totalement indépendant du système formel dans lequel on se place. Pour un théorème et une expression donnés, il procède de façon rigoureuse par un double parcours dans un ordre préfixé sans choix ni retour arrière.*» [Laurière 1988]. Cet algorithme a été établi en 1966 par Jacques Pitrat dans sa thèse et indépendamment par John Robinson. Il peut être utilisé de façon systématique dans différents contextes mathématiques (e.g. résolution d'équations trigonométriques avec le programme PRET [Grandbastien 1974], résolution d'exercices d'arithmétique avec le programme PARI [Bourgoin

1978]). D'abord nous présentons le principe de l'algorithme d'unification illustré par deux exemples, puis nous précisons les contextes dans lesquels nous avons mis en œuvre cet algorithme.

5.3.1. Principe de l'algorithme d'unification

E étant une expression algébrique et R une règle de réécriture telle que G → D, l'expression E et la règle R vont être parcourues en parallèle et unifiées au fur et à mesure. Si l'on atteint la fin de la partie gauche G de la règle alors l'unification est réussie et les substitutions effectuées dans la partie droite D de la règle donnent le résultat : la partie droite de la règle est alors la transformée de l'expression E par la règle de réécriture. Dans le cas contraire il n'est pas possible d'unifier l'expression avec cette règle. Cet algorithme s'appuie essentiellement sur une procédure de parcours d'expressions algébriques qui correspond à l'ordre de parcours préfixé de l'arbre binaire représentant l'expression (Figure 10). Seuls les symboles qui représentent des variables libres sont substituables et les seules substitutions autorisées sont celles qui consistent à remplacer une variable libre de E ou G par une sous-expression de E ou de G, cette sous-expression ne comportant pas elle-même cette variable libre. Si les expressions E et G ont des noms de variable communs, il est nécessaire de les renommer pour éviter toute ambiguïté. Si les variables des règles de réécriture sont des variables libres, les variables d'une expression algébrique sont liées et les seules substitutions autorisées dans Pépinière seront les substitutions des variables de la règle de réécriture, appelées aussi variables « muettes ». Ainsi dans l'expression $(x+5)^2+2x-5$ nous appliquons la règle de réécriture $(A+B)^2 \rightarrow A^2 +2AB + B^2$ à la sous-expression $(x+5)^2$ en substituant la variable x à A et la constante 5 à B, A et B étant des variables libres.

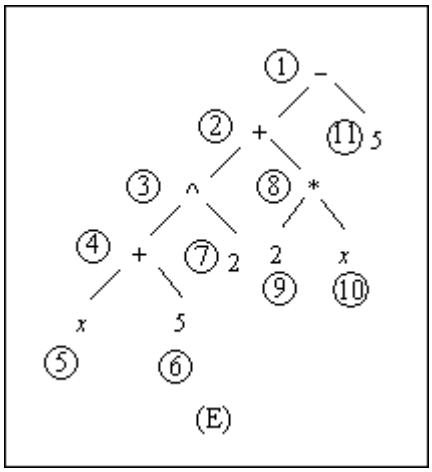


Figure 10 : Parcours préfixé de l'arbre d'expression de $(x+5)^2+2x-5$

Le Tableau 12 présente les procédures utilisées dans cet algorithme qui est décrit de façon détaillée dans [Laurière 1988]. L'algorithme converge avec un nombre de substitutions au plus égal au nombre initial de variables substituables dans l'expression algébrique et dans le membre gauche de la règle de réécriture [Laurière 1988].

L'algorithme que nous avons mis au point est constitué de deux procédures récursives :

- la procédure UNIFICATION qui parcourt en parallèle l'arbre d'expression de l'expression algébrique à transformer et l'arbre d'expression du membre gauche de la règle de réécriture et appelle la procédure SUBSTITUTION,
- La procédure SUBSTITUTION qui remplace les variables libres dans les deux membres de la règle de réécriture.

Le Tableau 12 et le Tableau 13 présentent ces algorithmes.

5.3.2. Exemple d'unification

5.3.2.1. Exemple1

Reprenons l'expression $(x+5)^2+2x-5$ dont l'arbre d'expression est représenté par la Figure 10. Nous appliquons la règle de réécriture $(A+B)^2 \rightarrow A^2 +2AB + B^2$ à la sous-expression $(x+5)^2$. La procédure UNIFICATION est appelée de façon récursive pour les nœuds numérotés ① à ③ dont les étiquettes sont égales. L'étiquette A du nœud du nœud ③ est une variable. La procédure « Instanciation » substitue la variable x à chaque occurrence de la variable A dans les expressions G et D parties gauche et droite de la règle de réécriture. La Figure 11 présente l'expression (E), les règles gauches (G) et droites (D) à cette étape de la substitution. A l'appel de la procédure UNIFICATION au niveau du nœud numéroté ④ l'étiquette du nœud est la variable B. La procédure SUBSTITUTION substitue la valeur 5 à chaque occurrence de la variable B dans les expressions de (G) et (D).

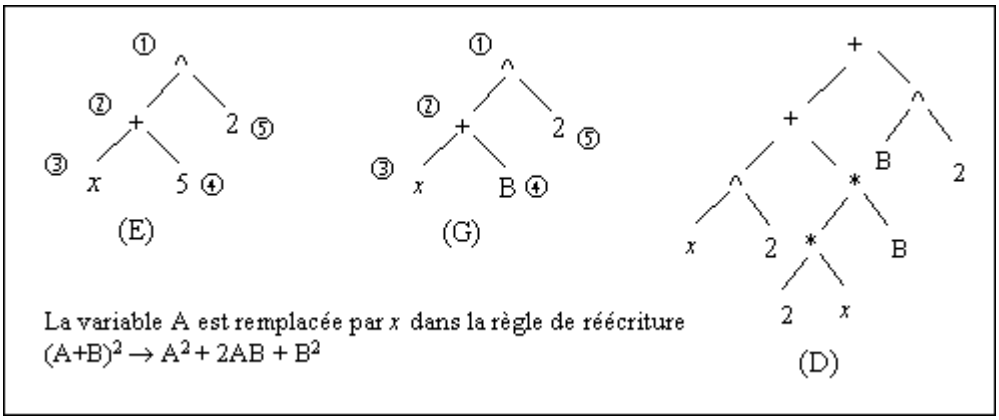


Figure 11 : Unification de l'expression $(x+5)^2$ avec la règle $(A+B)^2 \rightarrow A^2 +2AB + B^2$

Procédure UNIFICATION

Données

E : expression

G et *D* : respectivement partie gauche et partie droite de la règle de réécriture *R*

e : nœud courant dans *E*

g : nœud courant dans *G*

Les nœuds *e* et *g* sont étiquetés par des opérateurs, par des variables ou des nombres.

Algorithme

```
si le parcours en parallèle des arbres d'expression de E et de G est terminé
    alors  si l'un des deux arbres a été totalement parcouru  mais pas l'autre
               alors
               unification = Faux
               retour
               sinon
               unification = Vrai
               retour
           fin si
    sinon  si les nœuds e et g ont les mêmes étiquettes
               alors
               unification = Vrai
               sinon  si l'étiquette de g est une variable libre
                          alors
                          SUBSTITUTION de g par e dans la partie gauche de la règle
                          SUBSTITUTION de g par e dans la partie droite de la règle
                          unification = Vrai
           fin si         fin si
           UNIFICATION du fils gauche du nœud e avec le fils gauche du nœud g
           UNIFICATION du fils droit du nœud e avec le fils droit du nœud g
fin si
```

Résultats

Si l'unification est réussie (*unification = Vrai*), la partie droite de la règle est la transformée de l'expression par cette règle

Tableau 12 : procédure UNIFICATION

Procédure SUBSTITUTION

Données

Un arbre d'expression : celui de *G* ou de *D*, respectivement partie gauche et partie droite de la règle de réécriture *R*

v : variable muette à instancier

e : sous-expression de *E* à substituer à la variable *v*

Algorithme

si le parcours de l'arbre d'expression est terminé
  alors **retour**
  sinon **si** l'étiquette du nœud courant est égale à *v*
    alors
    - Remplacement de *v* par *e*
    - SUBSTITUTION de *v* par *e* dans le fils gauche du nœud courant
    - SUBSTITUTION de *v* par *e* dans le fils droit du nœud courant
  fin si
fin si

Résultat

Toutes les occurrences de *v* sont remplacées par *e* dans l'arbre d'expression (*G* ou *D*)

Tableau 13 : Procédure SUBSTITUTION

5.3.2.2. Exemple2

Prenons une exemple un peu plus complexe, le développement de l'expression 2 (x^2+5x+6) qui est une étape de calcul envisageable dans des exercices permettant d'évaluer la compétence « *utiliser des identités remarquables pour développer des expressions littérales* ». Ici la règle de réécriture utilisée est A (B+C) → AB + AC. La procédure UNIFICATION est appelée de façon récursive. L'étiquette A du nœud ② est une variable. La procédure SUBSTITUTION substitue la valeur 2 à chaque occurrence de la variable A dans les expressions G et D parties gauche et droite de la règle de réécriture. A l'appel de la procédure UNIFICATION au niveau du nœud numéroté ④ l'étiquette du nœud est la variable B. La procédure SUBSTITUTION substitue la sous-expression (x^2+2x) à chaque occurrence de la variable B dans les expressions G et D. La Figure 12 présente l'expression (E), la partie gauche de la règle (G) avant les substitutions, la partie gauche de la règle (G') à cette étape des substitutions et la partie droite (D) de la règle à cette étape des substitutions.. Ensuite la valeur 6 est substituée à la variable C. La partie droite de la règle après instanciation est l'expression 2*(x^2+2x)+2*6. Une deuxième application de la même règle de réécriture permet d'obtenir au final l'expression 2*x^2+2*2x+2*6.

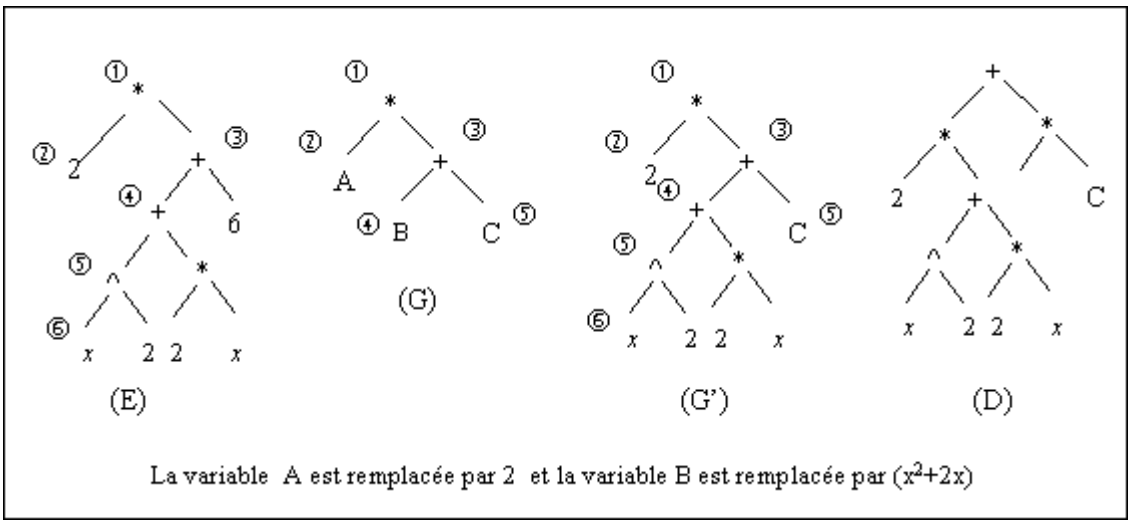


Figure 12 : Unification de l'expression 2 (x^2+5x+6) avec la règle A (B+C) → AB + AC

5.3.3. Mise en œuvre de l'unification

L'algorithme d'unification est mis en œuvre pour chaque sous-expression de l'expression de départ et pour toutes les règles de réécriture d'un même type dans un ordre donné. Ainsi nous avons fait le choix d'appliquer la règle (A+B)*(C+D) → AC+BC+AD+BD avant la régle A*(B+C) → AB+AC. En effet ces deux règles de développement peuvent être unifiées avec un produit de deux facteurs mais au niveau de quatrième « *développer une expression de la forme (a+b) (c+d)* » est une compétence à acquérir [BO 2007]. Illustrons l'utilisation de ces deux règles sur un exemple.

> Pour développer (x+5) (x+2) nous pouvons unifier cette expression avec la partie gauche de la règle de réécriture A*(B+C) → AB+AC. Nous substituons (x+5) à A, x à B et 2 à C pour obtenir (x+5)*2+(x+5)*x, puis nous unifions la partie gauche de la régle de réécriture (B+C) A → BA + CA avec les deux sous-expressions (x+5)*2 et (x+5)*x. Après avoir réduit et ordonné les polynômes obtenus nous avons le même résultat que si nous avions unifié la partie gauche de la règle (A+B) (C+D) → AC+BC+AD+BD avec l'expression (x+5) (x+2) soit x^2+7*x+10.

L'algorithme d'unification applique donc les règles dans un ordre choisi pour analyser des compétences en cours de construction à un niveau scolaire donné. La procédure s'arrête lorsqu'il n'y a plus de règles applicables pour un même type de règle.

5.3.4. Modélisation des règles de réécriture

Les règles sont stockées sous la forme d'un document XML. La Figure 13 présente la structure du fichier XML des règles de réécritures. Pour les décrire nous avons repris la notion de type de règle (section 5.2.1), choisi d'appeler genre la qualité de règle correcte ou erronée (genre « V » pour une règle correcte et « F » pour une règle erronée). Une règle est associée à un niveau scolaire et comporte une partie gauche (par exemple A*(B+C) pour la règle A*(B+C) → AB+AC) et une partie droite (AB+AC). Le Tableau 14 présente le schéma des règles de réécriture et le Tableau 15, un extrait du document XML. Le fichier complet est présenté en annexe C.2.

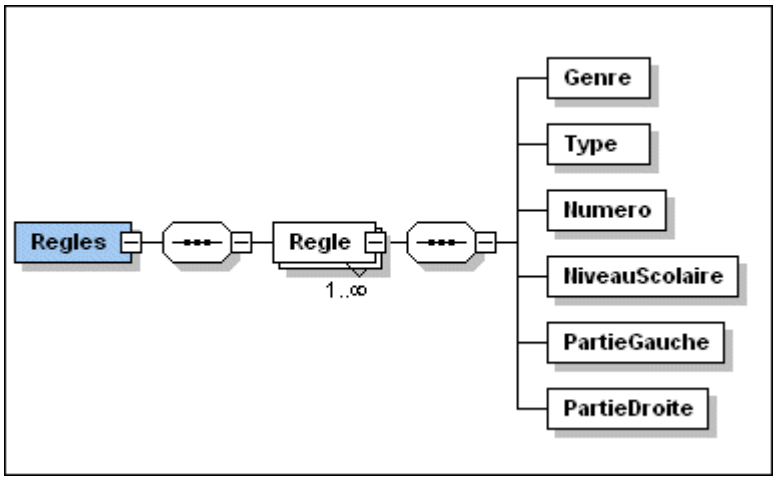


Figure 13 : Structure du fichier XML des règles de réécritures

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Regles">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Regle" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Genre"/>
                            <xs:element name="Type"/>
                            <xs:element name="Numero"/>
                            <xs:element name="NiveauScolaire"/>
                            <xs:element name="PartieGauche"/>
                            <xs:element name="PartieDroite"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Tableau 14 : schéma XML des règles de réécriture

```
<?xml version="1.0" encoding="UTF-8"?>
<Regles >
   <Regle>
            <Genre>V</Genre>
      <Type>1</Type>
      <Numero>1</Numero>
      <NiveauScolaire>4,3,2</NiveauScolaire>
      <PartieGauche>(a+b)*(c+d)</PartieGauche>
      <PartieDroite>a*c+a*d+b*c+b*d</PartieDroite>
   </Regle>
   <Regle>
      <Genre>V</Genre>
      <Type>1</Type>
      <Numero>2</Numero>
      <NiveauScolaire>4,3,2</NiveauScolaire>
      <PartieGauche>(a-b)*(c+d)</PartieGauche>
      <PartieDroite>a*c+a*d-b*c-b*d</PartieDroite>
   </Regle>
   <Regle>
      <Genre>V</Genre>
      <Type>1</Type>
      <Numero>3</Numero>
      <NiveauScolaire>4,3,2</NiveauScolaire>
      <PartieGauche>(a+b)*(c-d)</PartieGauche>
      <PartieDroite>a*c-a*d+b*c-b*d</PartieDroite>
   </Regle>
   …./….
</Regles>
```

Tableau 15 : Extrait du document XML des règles de réécriture

5.4. Traitement particulier des règles de réécriture de type 6 et 7

La règle de réécriture de type 6, AC+BC → (A+B)C, où A et B sont des nombres et C une puissance d'une variable permet plus particulièrement d'évaluer la compétence « *Réduire une expression littérale à une variable* » en effectuant la somme des monômes de même degré. Dans une perspective de diagnostic, pour pouvoir analyser cette compétence en construction dès la classe de cinquième, nous avons souhaité produire l'ensemble des réponses anticipées obtenues en additionnant les monômes deux par deux ou en appliquant des règle erronées sans modifier l'ordre dans lequel ces sommes sont effectuées, ce qui n'est pas le cas si nous utilisons l'algorithme d'unification. Pour unifier une sous-expression avec la partie gauche AC+BC de la règle de réécriture nous modifions l'ordre dans lequel les réductions sont faites dans un environnement *papier-crayon* et nous faisons apparaître des réponses anticipées qui ne correspondent pas toujours à la réalité. Le choix d'un traitement particulier pour appliquer cette règle de réécriture est déterminé par la préoccupation d'être au plus près des démarches de calculs mises en œuvre dans un environnement *papier-crayon*.

Prenons un exemple : la réduction de l'expression (E7).

(E7) $x^2+5x+15-2x$

Dans un environnement *papier-crayon* et une lecture de l'expression de la gauche vers la droite, nous avons les étapes suivantes :

(E8) $x^2+5x+15-2x = x^2+3x+15$

L'application des règles erronées de ce type de règles produit avec cette expression plusieurs réponses erronées anticipées dont deux règles qui assemblent les termes.

(E9) $x^2+5x+15-2x = x^2+20x-2x$ en appliquant la règle $A+BC \rightarrow (A+B)C$ à la sous-expression $5x+15$

(E10) $x^2+5x+15-2x = x^2+3x+13x$ en appliquant la règle $AC+B \rightarrow (A+B)\ C$ à la sous-expression, ou encore :

(E11) $x^2+5x+15-2x = x^2+3x+15 = x^2+18x$ en appliquant la règle erronée précédente au résultat final pour une « dernière réduction ».

Reprenons cet exemple en utilisant l'algorithme d'unification avec la règle $AC+BC \rightarrow (A+B)C$ pour faire la somme des termes de degré un après avoir additionné les constantes. Nous devons d'abord changer l'ordre des termes puis, unifier la sous-expression 5x-2x en substituant 5 à A et -2 à B :

(E12) $x^2+5x+15-2x = 5x-2x+x^2+15 = 3x + x^2 +15$

Nous ne retrouvons pas dans cette expression les mêmes termes consécutifs qui permettraient d'obtenir l'expression erronée (E11).

Dans les sections qui suivent nous définissons d'abord le poids d'un arbre qui est utilisé pour repérer les monômes et pour comparer des arbres (Cf. 7.2.3), nous présentons la procédure AJOUTERMES qui effectue des sommes algébriques de monômes de même degré, puis, nous illustrons par deux exemples.

5.4.1. Poids d'un arbre

Gauche(A) étant le sous-arbre gauche d'un arbre A et Droit(A) le sous-arbre droit, nous définissons le poids de l'arbre A récursivement par :

Poids(A) = 0 si A est vide sinon Poids(A) = 1 + Poids(Gauche(A)) + Poids(Droit(A)).

Le poids d'un arbre est donc la taille de l'arbre ou encore le nombre de nœuds de l'arbre.

5.4.2. La procédure « AjoutTermes »

Les expressions algébriques que nous analysons avec Pépinière sont généralement des expressions du second degré maximum.

La somme algébrique de deux ou plusieurs monômes se fait selon le principe suivant :

1. Parcourir selon l'ordre postfixé l'arbre d'expression, afin d'établir une liste ordonnée des nœuds qui sont la racine d'un sous-arbre d'expression représentant un monôme pour un degré donné,
2. Pour chacune des listes établies, réaliser la somme algébrique des monômes de même degré en utilisant les listes établies précédemment.

La forme générale d'un monôme est $a_n x^n$, a_n étant une constante et x la variable, soit $a_n * x^n$ dans une représentation linéaire. La racine des sous-arbres d'expression qui les représentent qui est étiquetée par le symbole opératoire de la multiplication doit donc comporter la constante (un nombre dans le cas de Pépinière) en fils gauche ce qui nécessite une transformation si ce n'est pas le cas. Pour repérer les monômes, le poids du sous-arbre est une première indication, le poids est égal à 1, 3 ou 5 selon que le monôme est de degré 0, 1 ou 2, à laquelle il faut ajouter la présence de l'opérateur * ou de l'exposant.

L'ordre de parcours postfixé ordonne les monômes dans le sens croissant selon une lecture de l'expression de la gauche vers la droite (e.g. dans l'expression x^2+5x+15-2x, le monôme 5x est placé avant le monôme -2x) ce qui permet d'effectuer les sommes des monômes de même degré de la gauche vers la droite.

Le Tableau 16 donne l'algorithme de la procédure réalisant la somme algébrique des monômes.

Procédure SOMMETERMES

Données

n : nœud courant d'un arbre d'expression *A*

Algorithme

si le parcours de *A* est terminé
  alors
  retour
  sinon **si** l'ordre de *n* est égal à *ordre 2*
    alors **tant que** l'étiquette du noeud gauche de *n* est égale à + ou – et que l'étiquette de *n* n'est pas égal à *ordre 1*
      parcours gauche en profondeur de l'arbre
      fin tant que
      - Calcul de la somme des monômes d'ordre 1 et 2
      - Application des règles erronées
      - Remplacement du monôme d'ordre 1 par la somme
      - Suppression du nœud d'ordre 2
    fin si
  SOMMETERMES pour le fils gauche de *n*
  SOMMETERMES pour le fils droit de *n*
fin si

Résultats

La somme des deux premiers monômes de même degré dans un ordre de parcours de l'expression algébrique sous une forme linéaire de la gauche vers la droite

Tableau 16 : Algorithme de la procédure SOMMETERMES

5.4.3. Exemples

5.4.3.1. Exemple 1

Reprenons l'exemple donné dans la section 5.4.

(E7) $x^2+5x+15-2x$

Après un parcours postfixé de l'arbre d'expression représenté par la Figure 14, la liste des monômes de premier degré est [7, 13]. La procédure SOMMETERMES ajoute les monômes 5*x* et -2*x* . Le nœud numéroté 7 a pour fils droit le sous-arbre d'expression de 3*x, le nœud numéroté 13 est supprimé. On obtient l'expression :

(E13) $x^2+3x+15$

5.4.3.2. Exemple 2

Prenons pour exemple la réduction de l'expression :

(E14) (2x+5x+15)(2+3x+2x)

Après un parcours postfixé de l'arbre d'expression représenté par la Figure 14, la liste des monômes de premier degré est [7, 7, 14, 18]. La procédure SOMMETERMES ajoute les

monômes 2*x* et 5*x*, puis les monômes 3x et 2x. Le nœud numéroté 9 a pour fils droit le sous-arbre d'expression de 7*x et nœud numéroté 14 a pour fils droit le sous-arbre d'expression de 5*x . Les nœuds numérotés 7 et 18 sont supprimés. On obtient l'expression :

(E15) (7x+15)(2+5x)

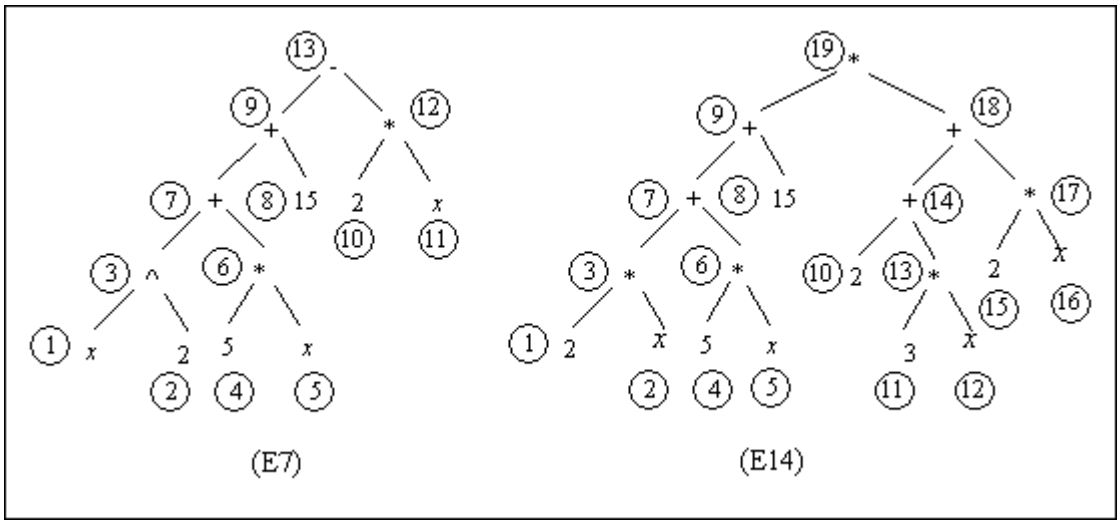


Figure 14 : arbres d'expression de x^2+5x+15+2x et (2x+3x+15)(2+3x+2x)

6. Générateur des solutions anticipées

Dans la section 4, nous avons exprimé les heuristiques qui nous permettent d'éviter les suites infinies de transformation dans les calculs algébriques comportant des développements et des réductions. Dans la section 5, nous montrons comment Pépinière utilise ces transformations. Nous avons vu dans le scénario présenté à la section 2.2 que PépiGen génère automatiquement la grille de codage de l'exercice proposé par le concepteur d'exercices, c'est-à-dire l'ensemble des réponses anticipées à l'exercice et le codage associé à ces réponses. Pour les exercices qui mobilisent les compétences du domaine « *Effectuer du calcul algébrique* » PépiGen fait appel à Pépinière pour créer un arbre des solutions anticipées. C'est un arbre général dont chaque nœud a pour étiquette une expression et pour fils chaque expression obtenue en appliquant les règles de transformation étudiées à la section 4. Chaque branche de l'arbre correspond à un raisonnement anticipé pour résoudre l'exercice et ce raisonnement est codé sur les dimensions « Calcul algébrique » et « Validité ». De plus, l'arbre des solutions mémorise le numéro des règles de réécriture appliquées pour obtenir un nœud à partir de la racine : ceci permet de connaître les règles correctes ou erronées utilisées par un élève.

Remarquons que Pépinière ne détermine pas entièrement le code associé à chaque solution. En effet les autres dimensions du codage (type d' « Utilisation des lettres », type de « Traduction » d'un registre sémiotique à un autre et type de « Justification ») sont spécifiques à une classe d'exercices et sont donc générées par PépiGen. L'attribution de ces codes peut également amener PépiGen à modifier la dimension « Validité ».

Dans cette section, en nous appuyant sur les heuristiques que nous venons de définir dans la section 4 pour générer des expressions algébriques équivalentes, nous décrivons la construction de l'arbre des solutions anticipées, puis nous présentons le modèle général de cette arbre.

6.1. Construction de l'arbre des solutions anticipées

Pour chacune des transformations décrites dans la section 4, nous construisons un arbre général de la solution anticipée. Nous utilisons pour parcourir cet arbre une représentation « fils-aîné-frère-droit ». La racine est l'expression de départ. Chaque nœud est étiqueté par l'expression produite, le numéro de la règle appliquée pour la produire à partir du père, le code associé qui peut être éventuellement vide. Le fils-aîné est le nœud étiqueté par l'expression obtenue en appliquant la règle correcte, les frères droits étant les nœuds obtenus en appliquant des règles erronées. La règle de réécriture appliquée correspond à l'une des quatre transformations : réduction de l'expression algébrique, développement, réduction au même dénominateur ou simplification d'une fraction. Chaque expression obtenue à cette étape (étape 1) est réduite en appliquant la règle correcte ainsi que les règles erronées correspondantes. Certains codes ne peuvent être attribués qu'en étudiant deux étapes successives. C'est le cas des réponses de certains élèves qui n'indiquent pas de parenthèses mais en conservent le sens. Cette erreur est codée EA31 (erreur de parenthèse avec mémoire de l'énoncé) alors qu'une erreur de parenthèse sans mémoire de l'énoncé est codée EA32.

> Par exemple pour développer l'expression 3(*x*+6)-3*x*, Pépinière développe l'expression 3(x+6)3-3x en appliquant la règle correcte A(B+C) → A*B+A*C pour obtenir l'expression 3x+3*6-3*x soit 3*x+18-3*x. La règle erronée A(B+C) → A*B+C est aussi appliquée à l'expression 3(*x*+6)-3*x*, ce qui donne pour résultat 3x+6-3x. Cependant à l'étape suivante, certains élèves donnent pour résultat 18 (et non 6), ce qui correspond à l'application de la règle correcte A(B+C) → A*B+A*C dont ils ont gardé le sens puis à la réduction de l'expression obtenue 3x +18-3x. Ensuite, nous appliquons à l'expression obtenue les règles correctes ou erronées concernant la réduction d'un polynôme.

La Figure 5 présente l'arbre des solutions anticipées correspondant à cette exemple. Le Tableau 17 présente le parcours « fils-aîné-frère-droit » de l'arbre des solutions anticipées. Nous retrouvons ainsi les sept feuilles terminales de l'arbre représenté par la Figure 5.

Parcours « fils-aîné-frère-droit » de l'arbre des solutions anticipées

Expression	Règle	Type de règle	Codage
3*(x+6)-3*x			
3*x+3*6-3*x	A(B+C) --> AB + AC	Règle correcte	V1,EA1
3*x+18-3*x			
18	AC+BC --> A(B+C)	Règle correcte	V1,EA1
3*(x+6)-3*x			
3*x+3*6-3*x	A(B+C) --> AB + AC	Règle correcte	V1,EA1
3*x+18-3*x			
21*x-3*x	AC+B --> A(B+C)	Règle erronée	V3,EA42
18*x	AC+BC --> A(B+C)	Règle correcte	V3,EA1
3*(x+6)-3*x			
3*x+3*6-3*x	A(B+C) --> AB + AC	Règle correcte	V1,EA1
3*x+18-3*x			
3*x+15*x	A+BC --> (A+B)C	Règle erronée	V3,EA42
18*x	AC+BC --> A(B+C)	Règle correcte	V3,EA1
3*(x+6)-3*x			
3*x+6-3*x	A(B+C) --> AB+AC	Règle erronée	V3,EA32
3*x+18-3*x	A(B+C) --> AB+AC	Mémoire énoncé	V3,EA31
18		Règle erronée	V3,EA31
3*(x+6)-3*x			
3*x+6-3*x	A(B+C) --> AB+AC	Règle erronée	V3,EA32
6	AC+BC --> A(B+C)	Règle correcte	V1,EA1
3*(x+6)-3*x			
3*x+6-3*x	A(B+C) --> AB+AC	Règle erronée	V3,EA32
9*x-3*x	AC+B --> A(B+C)	Règle erronée	V3,EA42
6*x	AC+BC --> A(B+C)	Règle correcte	V3,EA1
3*(x+6)-3*x			
3*x+6-3*x	A(B+C) --> AB+AC	Règle erronée	V3,EA32
3*x+3*x	A+BC --> (A+B)C	Règle erronée	V3,EA42
6*x	AC+BC --> A(B+C)	Règle correcte	V3,EA1

Tableau 17 : Parcours « fils-aîné-frère-droit » de l'arbre des solutions anticipées

L'arbre des solutions anticipées donne le code associé à la dimension « Utilisation des règles d'Ecriture et de réécriture Algébrique » (e.g. EA1, EA31, EA32, EA42) ainsi qu'à la « Validité » (e.g. V1 et V3). Le code évolue au fur et à mesure du parcours de l'arbre, les codes associés aux règles erronées étant retenu dès qu'une règle erronée est appliquée. Ainsi pour l'exemple du

Tableau 17, le premier parcours sera codé V1,EA1, le deuxième et le troisième V3,EA42, le quatrième V3, EA31 (erreur de parenthèse avec mémoire de l'énoncé) et le cinquième V3, EA32 (erreur de parenthèse sans mémoire de l'énoncé) et les deux derniers comportent V3, EA32, EA42.

6.2. Modèle général de l'arbre des solutions

Le modèle général de cet arbre est donnée par la Figure 15.

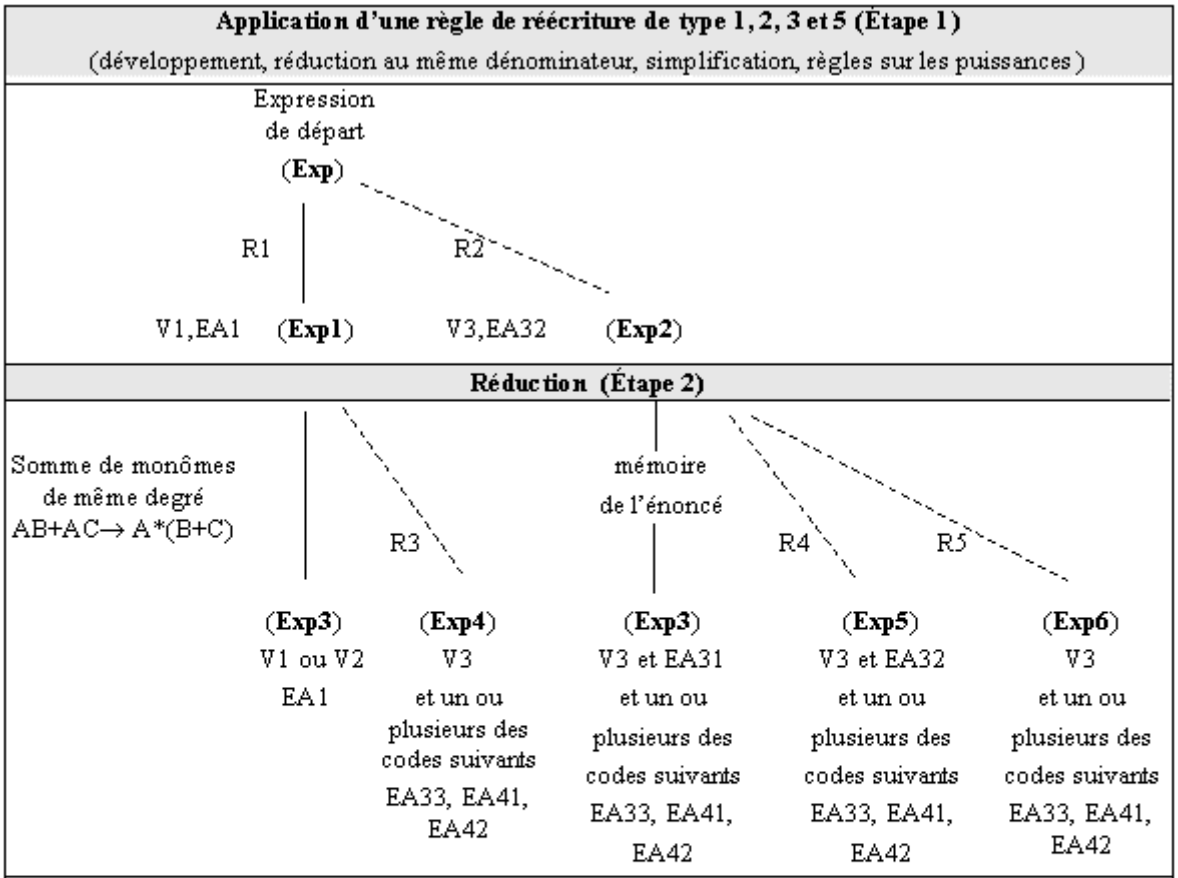


Figure 15 : Arbre général des solutions anticipées

Dans le chapitre 7, nous expliquons comment PépiGen utilise l'arbre des solutions anticipées pour générer le fichier contenant la grille de codage des exercices et nous présentons des exemples.

7. Comparaison des expressions algébriques

Reprenons l'exemple proposé dans le scénario 1 de la section 2. Dans cet exercice, le raisonnement algébrique consiste à produire des expressions équivalentes en utilisant des règles de développement et de réduction des expressions jusqu'à obtenir la forme réduite. Pour établir le diagnostic, le système compare la réponse de l'élève aux réponses anticipées à cet exercice mémorisées dans le fichier qui contient la grille de codage. Pour pouvoir comparer les deux réponses, nous travaillons sur la représentation des expressions algébriques sous la forme d'arbre d'expression. Dans cette section, nous explicitons ce choix en nous appuyant sur l'exemple proposé, nous définissons ce que nous appelons des arbres d'expression équivalents, puis, nous décrivons les procédures effectuées sur les arbres d'expression pour prouver qu'ils sont équivalents.

7.1. Arbres d'expression équivalents

Dans le scénario 1 (Cf. 2.1), la réponse attendue à un niveau de troisième est l'expression algébrique (E4) dont l'arbre d'expression est représenté par la Figure 2.

(E4) (5+*x*)^2+2**x*

Nous supposons que la réponse d'un élève est l'expression algébrique (E1) dont l'arbre d'expression est représenté par la Figure 1 et que la réponses d'un autre élève est l'expression algébrique (E5) dont l'arbre d'expression est représenté par la Figure 16 :

(E1) 2*x* + (*x*+5)^2 (E5) 2*x* + (*x*+5)(*x*+5)

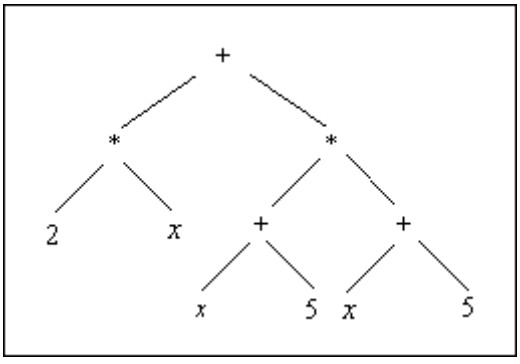


Figure 16 :Arbre d'expression de 2x + (x+5) (x+5)

Les expressions algébriques que nous comparons dans cet exemple et plus généralement que nous analysons avec Pépinière sont des expressions qui se ramènent, après réduction, à des polynômes à une indéterminée du second degré maximum et tout polynôme s'exprime d'une seule façon comme une combinaison linéaire de la forme $\Sigma a_n x^n$. Ainsi, pour les expressions (E4), (E1) et (E5) nous obtenons après réduction et ordonnancement le même polynôme :

Réponse attendue : (E4) $(5+x)^2+2x = x^2 + 12x + 25$

Réponse 1 : (E1) $2x + (x+5)^2 = x^2+ 12x + 25$

Réponse 2 : (E5) $2x + (x+5)(x+5) = x^2 + 12x + 25$

Sur le plan formel, les trois expressions algébriques sont équivalentes, cependant le système de diagnostic associe le code V1, T1, EA1, J1 à la réponse (E1) qui correspond à la réponse correcte attendue et le code V2, T2, EA1, J1 à la réponse (E5) qui correspond à une réponse correcte mais qui n'est pas attendue à ce niveau de classe comme nous l'avons remarqué en 2.1.2. Pour comparer la réponse de l'élève à la réponse attendue et établir un diagnostic nous ne pouvons pas réduire et ordonner le polynôme pour vérifier si les expressions sont équivalentes mais nous devons comparer l'expression algébrique donnée par l'élève à l'expression attendue. Or l'expression (E1) qui correspond à une réponse correcte, n'est pas identique à l'expression (E4) : pour les rendre identiques nous devons appliquer la propriété de commutativité de l'addition pour transformer $2x+(x+5)^2$ en $(x+5)^2+2x$ puis transformer $(x+5)^2+2x$ en $(5+x)^2+2x$ et nous pouvons alors attribuer le code V1, T1, EA1, J1 à la réponse (E1).

Pour comparer deux expressions nous avons fait le choix d'utiliser les arbres d'expression car la représentation sous forme d'arbre est bien adaptée pour appliquer les propriétés des opérations. Ainsi, l'arbre d'expression représentant (E1) peut être transformé en appliquant la propriété de commutativité de l'addition qui se traduit par un échange entre les fils gauches et droits des nœuds numérotés ① et ② et nous obtenons un arbre d'expression superposable à celui de (E4) (Figure 17). Il n'en est pas de même pour l'arbre d'expression représentant (E5). Le terme superposable est pris ici avec le sens utilisé en géométrie pour vérifier que deux figures sont superposables : l'une des deux figures est reproduite avec un calque puis on vérifie que le calque peut se superposer à l'autre figure.

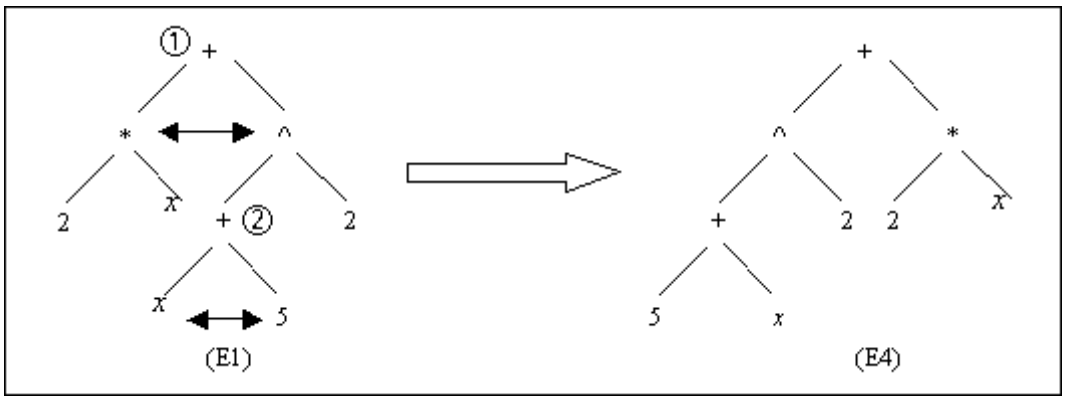


Figure 17 : Application de la commutativité de l'addition à l'arbre d'expression de 2*x* + (*x*+5)^2

Supposons maintenant que la compétence évaluée dans l'exemple proposé au scénario 1 soit la maîtrise de l'utilisation des identités remarquables pour développer l'expression algébrique (E4). La première étape de calcul attendue à un niveau de troisième est l'application de l'identité remarquable pour développer le carré de la somme (E16). Soit (E17) la réponse d'un élève, les arbres d'expression de ces réponses étant représentés par la Figure 18 :

Expression à développer : (E4) (5+*x*)^2+2**x*

Réponse attendue à la première étape de calcul : (E16) 25 + 10**x* + *x*^2 + 2**x*

Réponse d'un élève : (E17) 2x + x^2 + 25 + 10x

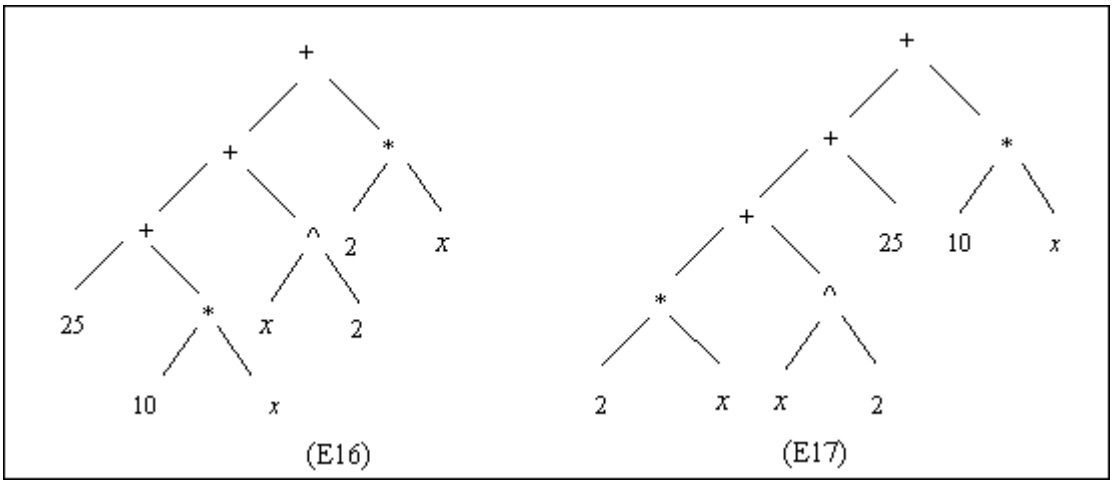


Figure 18 : Arbre d'expression de 25 + 10**x* + *x*^2 + 2**x* et 2x + x^2 + 25 + 10x

Dans cet exemple les expressions algébriques (E16) et (E17) ne sont pas identiques, cependant en ce qui concerne la validité de la réponse et l'utilisation du calcul algébrique, la réponse de l'élève est la réponse correcte attendue à l'ordre près des termes (codée V1, EA1). Pour vérifier que les arbres d'expression sont superposables, nous utilisons de nouveau les propriétés de commutativité et d'associativité de l'addition pour échanger les fils droits des nœuds étiquetés

par l'opérateur + et ordonner dans les deux arbres les termes de la somme selon les puissances décroissantes de la variable *x* ce qui revient à transformer ces deux arbres pour obtenir l'arbre représenté par la Figure 19 qui correspond à la représentation de l'expression (E18) :

(E18) x^2 + 10x + 2x + 25

Ces deux exemples montrent que pour comparer la réponse de l'élève aux réponses anticipées afin d'établir le diagnostic, nous avons transformé les arbres d'expression des deux réponses pour les rendre superposables en appliquant les propriétés des opérations. Nous disons que les arbres obtenus sont équivalents.

7.1.1. Définitions

Dans la section 7.1 nous avons donné la définition du terme « superposable » dans le cadre géométrique. Nous transposons cette définition aux arbres d'expression pour ensuite définir l'équivalence de deux arbres d'expression.

7.1.1.1. Arbres superposables

Deux arbres d'expression sont *superposables* si étant parcourus parallèlement selon le même ordre de parcours (e.g. préfixé) les étiquettes des nœuds qui sont des opérateurs ou des opérandes sont égales.

7.1.1.2. Arbres équivalents

Nous disons que deux arbres sont équivalents si nous pouvons les rendre superposables en appliquant des traitements qui s'appuient sur les propriétés de commutativité et d'associativité des opérations.

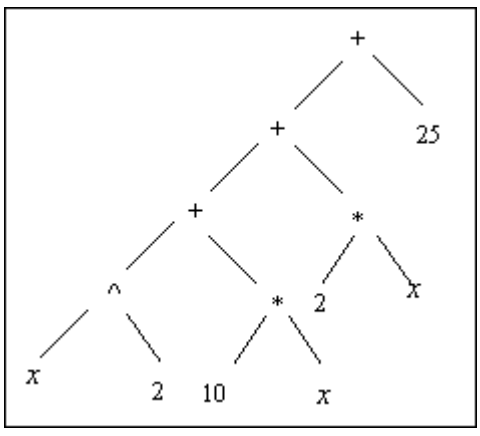


Figure 19 : Arbre d'expression de x^2 + 10x + 2x + 25

7.2. Procédures de transformation des arbres d'expression

Pour obtenir des arbres équivalents nous avons transformé les arbres d'expressions en appliquant les propriétés des opérations (commutativité et associativité). Pour appliquer ces propriétés nous avons défini des stratégies afin d'aboutir à des arbres superposables. Ces choix ont été déterminés à la fois par le souci de s'appuyer au plus près des démarches de calcul observées dans un environnement papier-crayon et des conventions habituellement utilisées en calcul algébrique.

Nous définissons ainsi une normalisation des arbres pour pouvoir les comparer. Elle s'appuie sur deux principes :

- Transformation des soustractions en addition qui est une opération commutative ce qui n'est pas le cas pour la soustraction,
- Définition d'un ordre croissant sur les arbres pour les trier en profondeur.

Ainsi, pour vérifier que deux arbres d'expression sont équivalents nous leur appliquons plusieurs procédures de transformations :

1. Remplacement des soustractions en appliquant la règle a – b = a + opposé(b)
2. Traitement des nœuds étiquetés par un opérateur commutatif (+, *) dont les deux fils sont des feuilles,
3. Traitement des nœuds étiquetés par un opérateur commutatif (+, *) dont les deux fils sont de poids différents,
4. Traitements des sommes de plusieurs termes ou des produits de plusieurs facteurs.

Ces trois derniers traitements qui utilisent les propriétés de commutativité et d'associativité de la somme et du produit permettent d'ordonner les nœuds.

Notons qu'avant de transformer les arbres d'expression, Pépinière vérifie les noms des variables utilisées dans les expressions à comparer : par exemple, pour une expression utilisant une seule variable Pépinière produit les réponses anticipées en la nommant *x*. Si la réponse de l'élève comporte un autre nom de variable, Pépinière renomme la variable.

7.2.1. Remplacement des soustractions

Les soustractions sont d'abord remplacées par une addition selon la règle :

a – b = a + opposé(b)

Puis, les règles qui permettent de calculer l'opposé d'une somme, d'un produit, d'un quotient ou d'une puissance sont appliquées. Ensuite, nous remplaçons l'opposé d'un nombre qui est un opérateur unaire par le nombre lui-même affecté du signe convenable. Ces règles sont présentées dans le Tableau 18.

7.2.2. Nœuds dont les deux fils sont des feuilles

Si le père est un opérateur commutatif (+, *) et les feuilles sont des opérandes de type lettre ou nombre, les feuilles gauche et droite sont triées dans l'ordre gauche-droite selon l'ordre lexicographique si les deux feuilles sont des lettres, l'ordre des entiers relatifs si les deux feuilles sont des nombres. Si l'une des feuilles est un nombre et l'autre feuille une lettre, le nombre est placé en fils gauche. Ce dernière procédure est aussi appliquée avant de faire la somme de monômes (Cf. 5.4.2)

opposé (a+b) = opposé(a) + opposé(b)
opposé (a-b) = opposé(a) + b
opposé (a*b) = opposé(a) * b
opposé (a/b) = opposé(a) / b
opposé (a^n) = opposé(a) * a^(n-1)
opposé (a) > 0 si a < 0 opposé (a) < 0 si a > 0

Tableau 18 : Règles de remplacement des soustractions

7.2.3. Nœuds dont les deux fils sont de poids différents

Si le père est un opérateur commutatif (+, *) et les sous-arbres qui sont les fils gauches et droits sont de poids différents, les deux fils sont échangés si le poids du fils gauche est inférieur à celui du fils droit. Remarquons que ce cas est différent de celui des nœuds traités dans la section 7.2.2 pour lesquels les deux fils sont de poids égaux.

7.2.4. Sommes de plusieurs termes et produits de plusieurs facteurs

Dans une somme de plusieurs termes, les fils droits des nœuds étiquetés par le signe + sont triés en profondeur par ordre de poids croissant, le fils droit du nœud étiqueté + le plus proche de la racine ayant le poids minimal. Ainsi dans l'exemple de la section 7.1, l'arbre d'expression de $2x + x\text{^}2 + 25 + 10x$ est transformé en un arbre représentant l'expression $x\text{^}2 + 10x + 2x + 25$ pour lequel la racine a pour fils droit le nombre 25 dont le poids est égal à un. Ce choix s'appuie sur la convention utilisée en mathématiques qui consiste à ordonner les polynômes selon les puissances décroissantes. Lorsque les fils droits sont de même poids nous comparons les opérateurs, selon une hiérarchie qui prend en compte l'ordre de priorité des opérateurs (+ < * < ^), ainsi $x\text{^}2$ est placé plus en profondeur que $10x$. Enfin si les opérateurs sont les mêmes nous comparons leurs fils gauches en prenant les mêmes conventions que celles données dans la section 7.2.2 (e.g. 10 x est placé plus en profondeur que 2x). Nous appliquons les mêmes règles pour un produit de plusieurs facteurs.

7.3. Utilisation de la comparaison d'expressions

Dans le chapitre 7, nous illustrons notre démarche pour prouver que deux arbres sont équivalents en appliquant les procédures décrites dans la section 7.2 au corpus de réponses collecté dans le cadre du projet Pépite et nous montrons comment PépiDiag utilise le comparateur d'expression de Pépinière pour coder le raisonnement d'un élève en le comparant aux raisonnements contenus dans la grille de codage de l'exercice.

8. Conclusion

Dans ce chapitre nous avons présenté le cœur de notre travail pour l'analyse des questions ouvertes : le logiciel Pépinière qui gère le calcul formel nécessaire à la génération des exercices et du diagnostic des réponses. Sans ce module nous serions limités à des exercices simples du type des exercices générés par QTI.

Nous avons présenté la méthode d'analyse syntaxique que nous avons adoptée et qui s'appuie sur les résultats en informatique fondamentale en particulier [Aho et al 1993] et [Jouannaud 2007]. Cette analyse s'appuie sur une grammaire algébrique de type LL(1). Dans la version actuelle, elle ne prend pas en compte le traitement des racines carrées mais elle permet de couvrir tous les exercices de Pépite. De plus, notre grammaire prévoit l'ajout de fonctions et donc pourrait être complétée par exemple pour la fonction racine carrée (elle traite déjà la fonction puissance).

À partir de cette analyse syntaxique, nous avons décrit la structure de données que Pépinière utilise pour représenter les expressions algébriques. Puis nous avons explicité les règles de réécriture correctes ou erronées que Pépinière utilise pour transformer les expressions. Dans la version actuelle nous nous sommes limités aux erreurs recensées dans l'analyse a priori de Pépitest. En nous appuyant sur les travaux de Gélis [Gélis 1995], nous avons, de plus, défini les heuristiques qui permettent d'éviter des suites infinies de calculs. Ensuite, nous avons présenté l'algorithme d'unification et nous avons détaillé la construction de l'arbre des solutions anticipées pour les problèmes de développement et de réduction des polynômes de degré inférieur ou égal à deux. À terme, nous envisageons d'ajouter des règles et des heuristiques par exemple en réinvestissant les travaux de recherche de l'équipe d'Aplusix [Nicaud 2005] sur la résolution d'équations et les factorisations. Nous n'avons pas à ce jour approfondi ce type de problèmes mais nous pensons qu'il est possible de limiter l'espace de recherche des solutions anticipées en se fixant des objectifs de diagnostic limités aux erreurs classiques qui permettent

aux didacticiens de repérer les cohérences de fonctionnement et de construire un profil cognitif de l'élève en algèbre élémentaire.

Enfin, nous avons expliqué comment Pépinière compare les expressions algébriques pour tenir compte des opérations commutatives et associatives mais aussi de contraintes particulières liées au diagnostic cognitif à ce niveau scolaire. Ceci est d'ailleurs le principal apport de notre travail par rapport à la littérature sur la manipulation des expressions algébriques.

Pépinière représente, là encore, un saut qualitatif pour le projet Pépite, puisqu'il fonde le calcul formel nécessaire au fonctionnement du système sur les cadres conceptuels solides de l'informatique théorique. Si ces théories (grammaires algébriques, unification) sont bien connues des informaticiens, leur mise en œuvre sur des problèmes qui dépassent les exercices d'application est loin d'être simple et nécessite de définir des heuristiques rigoureuses. C'est de loin, le problème qui m'a le plus préoccupée pendant mon travail de thèse. Le chapitre suivant montre comment Pépinière permet de fiabiliser l'analyse des réponses ouvertes et de l'étendre à des raisonnements complexes (par exemple de type « Prouver » par développement et réduction d'expressions algébriques du second degré). Il montre aussi comment Pépinière est utilisé pour générer les exercices et la grille d'analyse des réponses.

Comme nous l'avons remarqué dans notre état de l'art, dans la grande majorité des travaux d'EIAH, les réponses ouvertes sont évaluées par des évaluateurs humains et l'analyse automatique des réponses se limite à l'analyse de réponses simples et préformatées ou bien à la vérification de la validité de la réponse à l'aide d'une boîte noire (compilateur ou système de calcul formel qui vérifie la conformité de la réponse, pas de la démarche). L'analyse multidimensionnelle des productions d'apprenants qui ne soient pas préformatées est un problème qui est loin d'être résolu même dans des domaines réputés assez formels comme l'arithmétique ou l'algèbre comme en témoigne les travaux de Beeson et Nicaud par exemple. Sans prétendre avoir construit un analyseur d'expressions algébriques qui soit générique et réutilisable, ce qui me semble hors de portée actuellement, notre contribution est de proposer, en nous appuyant sur une analyse didactique a priori et la définition d'heuristiques, un logiciel qui permet d'anticiper et de caractériser des démarches de résolution et pas seulement une étape et ceci pour des problèmes d'algèbre de l'enseignement secondaire.

Chapitre 7

Le système PépiGen

1. Introduction 215

2. Diagrammes de cas d'utilisation 216
2.1. Créer un exercice 216
2.2. Composer un test diagnostic 217

3. Architecture fonctionnelle de PépiGen 218
3.1. Réalisation informatique 218
3.2. Architecture fonctionnelle 219
3.2.1. Le système auteur 219
3.2.2. Interpréteur et diagnostiqueur 220

4. Architecture logicielle 221
4.1. Couche domaine 221
4.2. Couche présentation 222
4.3. Couche application 223
4.4. Couche persistance 224

5. Génération des deux représentations d'un programme de calcul 225

6. Génération de la grille de codage 227
6.1. Codage associé à une branche de l'arbre des solutions 230
6.2. Génération de la grille de codage 230
6.2.1. Solutions correctes non optimales 233
6.2.2. Solutions partielles 233
6.2.3. Solutions incorrectes 233

7. Persistance des données 236
7.1. Structure du fichier XML « ModelesClasseExercices » 236
7.1.1. Racine du schéma XML du fichier « ModelesClasseExercices » 236
7.1.2. Schéma XML des données d'indexation didactique 237
7.1.3. Schéma XML de la classe GenerationEnonce 238
7.1.4. Schéma XML de l'interface abstraite 239
7.1.5. Schéma XML de la grille de codage 241
7.2. Structure du fichier XML « BanqueExercices » 244

8. Interpréteur d'exercice : PépiTest 246
8.1. Principales fonctionnalités 246
8.2. Exemple 247

9. Diagnostiqueur : PepiDiag 249
9.1. Principales fonctionnalités 249

9.2. Algorithme de diagnostic de la classe d'exercices « Preuve et programme de calcul » .. 250
9.3. Tests 251

10. Conclusion 255

1. Introduction

L'objectif de notre projet est de concevoir des outils facilitant le développement de banques d'exercices de diagnostic afin de permettre à des enseignants de produire des tests pour établir des bilans de compétences à différents moments de la construction de la compétence algébrique des élèves.

Dans le chapitre 5 nous avons proposé un modèle conceptuel pour la génération d'exercices de diagnostic. Dans le chapitre 6, nous avons présenté le logiciel Pépinière que nous avons conçu et développé pour le traitement des expressions algébriques. Nous avons montré comment Pépinière génère de façon automatique la grille de codage des réponses anticipées aux exercices, pour la dimension « Utilisation des règles d'Écriture et de réécriture Algébrique ».

Dans le présent chapitre nous décrivons le prototype PépiGen qui met en oeuvre notre proposition. Pour éviter une présentation fastidieuse de chacune des étapes de la méthodologie de conception, nous nous centrons sur les modules qui implémentent les trois principales fonctionnalités du système, à savoir : le module auteur qui permet de créer des exercices, l'interpréteur qui permet aux élèves de résoudre les exercices et le module de diagnostic qui code automatiquement les réponses des élèves sur plusieurs dimensions.

Notre chaîne logicielle est développée en Java et, afin de faciliter l'interopérabilité, toutes les données sont mémorisées dans des fichiers XML validés par des schémas XML. La modélisation UML des classes d'exercices a grandement facilité la modélisation objet du logiciel et la génération automatique par Eclipse (la plateforme de développement que nous avons utilisée) des classes Java du domaine, ainsi que la création des fichiers XML pour mémoriser les données et la création des schémas XML qui les valident.

Nous commençons la description du système auteur par la présentation et la mise en oeuvre de deux cas d'utilisation élaborés à partir de deux des scénarios décrits dans le chapitre 4 : « Un auteur enrichit la banque d'exercices » (que nous avons implémenté) et « Un enseignant, habitué, compose un test diagnostic » (que nous avons étudié). Nous décrivons ensuite l'architecture fonctionnelle du système PépiGen, puis, son architecture logicielle. Nous montrons plus particulièrement comment PépiGen s'articule avec le logiciel de traitement des expressions algébriques, Pépinière, pour générer la grille de codage. Ensuite nous présentons l'interpréteur d'exercices, PépiTest, et le module de diagnostic local à un exercice, PépiDiag.

2. Diagrammes de cas d'utilisation

Deux cas d'utilisation (Figure 1 et Figure 2) se dégagent des deux scénarios retenus au chapitre 4 pour décrire les principales fonctionnalités du système auteur : « Créer un exercice » et « Composer un test diagnostique ».

2.1. Créer un exercice

Le scénario « Un auteur enrichit la banque d'exercices » (scénario 8 du chapitre 4) met en jeu un auteur qui paramètre une instance d'une classe d'exercices. La Figure 1 résume les fonctionnalités du système. L'auteur choisit dans une base d'exercices (**ModelesClasseExercices**) un modèle[1] d'exercice qui représente une instance particulière (c'est l'exercice originel de PépiTest). L'auteur conçoit un clone en donnant des valeurs aux paramètres de la classe d'exercices. Ces valeurs sont des valeurs numériques, des expressions algébriques, des textes ou des figures selon la classe d'exercices choisie. Une interface permet à l'auteur de saisir ces paramètres dans des formulaires de façon conviviale. L'auteur contrôle ses choix en passant en mode élève et il les modifie si cela est nécessaire.

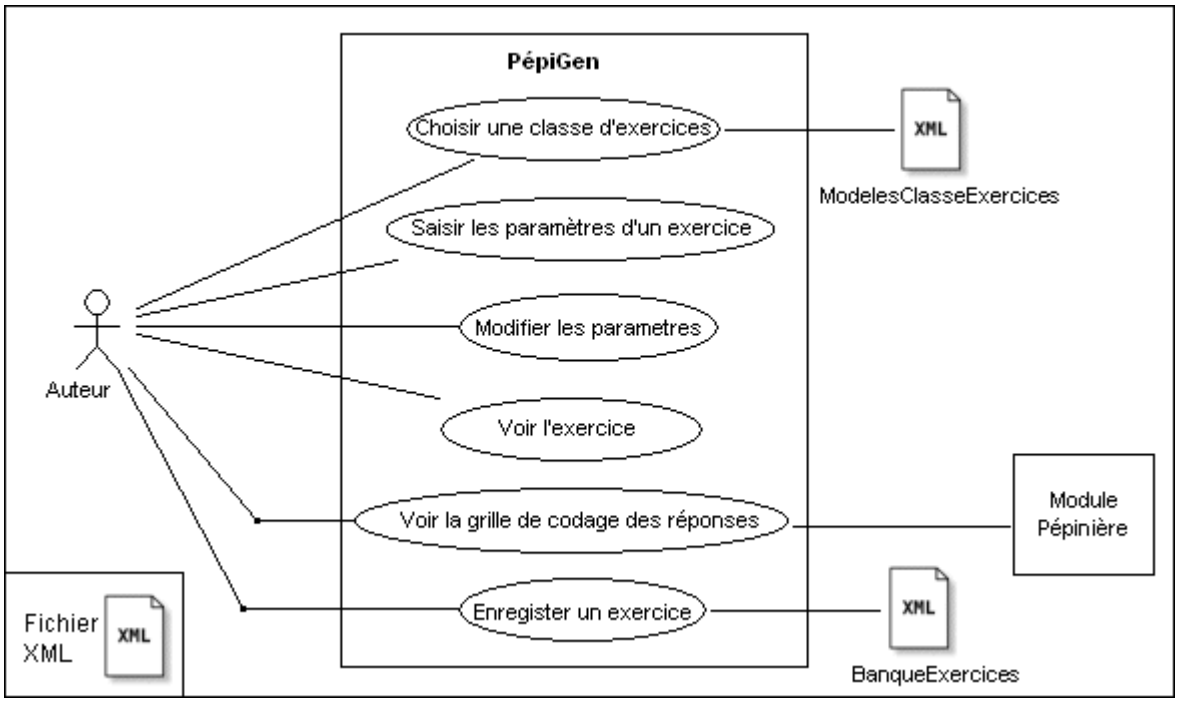


Figure 1 : Cas d'utilisation « Créer un exercice » - acteur : un auteur d'exercices

[1] Pour l'auteur nous utilisons le terme « modèle » (modèle d'exercices), pour les didacticiens le terme « classe » (classe d'exercices) et pour les informaticiens le terme patron (patron d'exercices).

Généralement, pour les didacticiens, les classes d'exercices dont la conception est assistée sont des exercices clés dans l'évaluation de la compétence algébrique (par exemple les classes d'exercice « Preuve et programme de calcul », « Déterminer des expressions d'un programme de calcul »). Ils comportent des questions ouvertes. PépiGen génère les réponses anticipées et le diagnostic local associé à ces réponses, c'est-à-dire la grille d'analyse associée à ces réponses. L'auteur peut contrôler cette génération en visualisant la grille d'analyse. Il enregistre ce clone (les paramètres et la grille de codage) dans la banque d'exercices (le fichier XML **BanqueExercices**).

2.2. Composer un test diagnostic

Les scénarios : « Un enseignant découvre SuperPépite et s'approprie un test type », « Un enseignant, habitué, compose un test diagnostic », « Un enseignant veut étudier l'évolution des compétences de ses élèves depuis le dernier test » mettent en jeu un enseignant. Les principales actions de ce cas d'utilisation « Composer un test diagnostic » sont présentées dans la Figure 2.

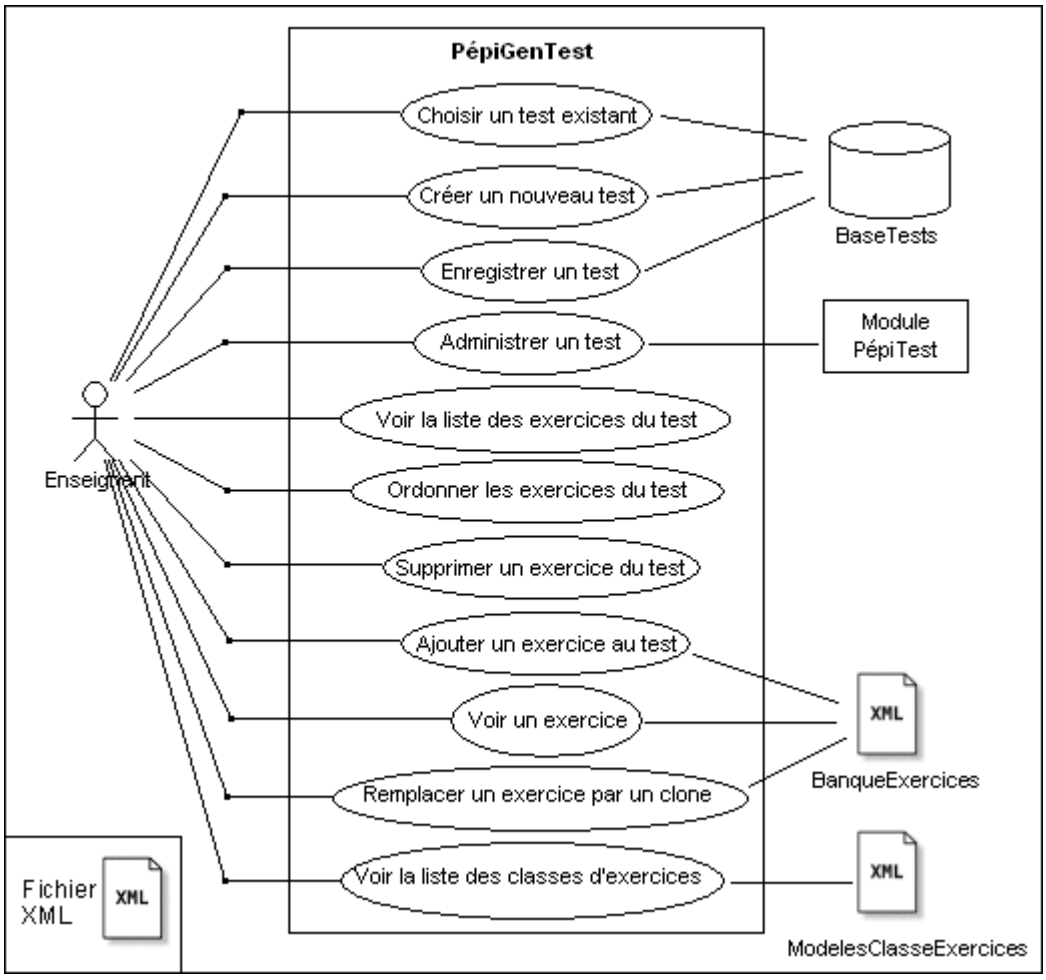


Figure 2 : Cas d'utilisation « Composer un test diagnostic » - acteur : un enseignant

Pour composer un test, soit l'enseignant crée le test en choisissant des exercices, soit il utilise un test « prêt à l'emploi » qu'il modifie éventuellement, en ajoutant ou supprimant un exercice en fonction du temps dont il dispose pour faire passer le test, ou, du niveau de difficulté souhaité. Nous avons réfléchi à ce cas d'utilisation mais nous ne l'avons pas (encore) mis en oeuvre.

3. Architecture fonctionnelle de PépiGen

Le modèle conceptuel que nous avons présenté au chapitre 5 a été élaboré par abstractions successives. A partir de l'étude des exercices originels du premier logiciel PépiTest, nous avons d'abord dégagé les principales caractéristiques de chaque classe d'exercices sous la forme de tableaux qui constituent un formalisme avec lequel les experts sont familiers. Puis, à partir de ces tableaux, nous avons élaboré un modèle conceptuel d'une classe d'exercices que nous avons décrit en utilisant le formalisme d'UML (Chapitre 5) qui sert de fondement à la conception informatique dans un langage objet.

3.1. Réalisation informatique

Le système auteur (PépiGen), l'interpréteur (PépiTest) utilisé par les élèves pour résoudre les exercices et le diagnostiqueur (PépiDiag) qui constituent la chaîne logicielle que nous avons mise au point sont développés avec le langage Java à l'aide de la plate-forme de développement Eclipse. Dans un premier temps Eclipse a été utilisée pour implémenter les principales classes Java du domaine à partir du diagramme de classes UML.

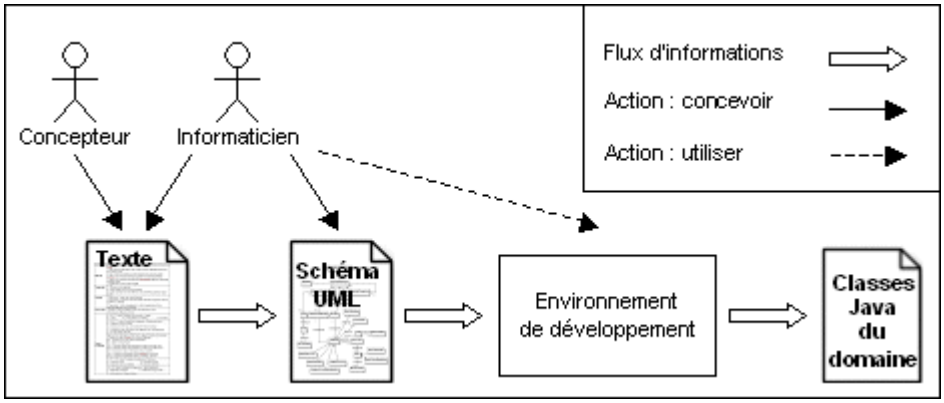


Figure 3 : De la conception du modèle à l'implémentation des classes Java du domaine.

Actuellement le module auteur PépiGen comporte environ 22000 lignes de code et 60 classes Java, le diagnostiqueur PépiDiag 17000 et 40 classes et l'interpréteur PépiTest 18000 et 45 classes. Les trois modules comportent environ 20 % de lignes de commentaires. Les classes Java qui effectuent les calculs algébriques, qui réalisent les opérations sur les arbres (transformations,

recherches, parcours) et qui effectuent le diagnostic sont les plus importantes (1500 lignes de codes en moyenne) suivies par celles qui effectuent l'analyse syntaxique et celles qui gèrent les échanges entre les objets Java et les documents XML.

Les données du système PépiGen étant stockées dans des fichiers XML, les transformations d'un fichier XML en objet Java et réciproquement sont réalisées en utilisant un ensemble de classes implémentant l'API DOM (Document Object Model). DOM est une bibliothèque qui définit la structure logique, les modes de gestion et d'accès aux contenus des documents XML. L'intérêt de ce modèle objet DOM est qu'il est indépendant d'un langage de programmation. Il spécifie des méthodes et des propriétés qui permettent de créer, modifier, supprimer ou extraire des données d'un document XML et définir les relations entre les nœuds et les déplacements dans l'arborescence XML.

Pour définir les classes logicielles impliquées dans les cas d'utilisation de la section 2, nous avons appliqué de façon méthodique des patterns de conception, notamment des patterns GRASP (General Responsability Assignment Software Patterns [Larman 2005]) et des patterns GoF (Gang of Four) [GHJV95]. La description détaillées des principales classes utilisées et de la mise en oeuvre de ces patterns est reportée en annexe D.

3.2. Architecture fonctionnelle

Les cas d'utilisation décrivent les interactions entre les acteurs (acteurs humains ou systèmes informatiques) et le système logiciel à concevoir.

Le système PépiGen est composé de trois modules :

- un système auteur, PépiGen, qui génère les clones d'exercices et le diagnostic,
- un interpréteur, PépiTest, destiné aux élèves, qui charge un ensemble d'exercices constituant un test, les affiche et enregistre les réponses des élèves dans un fichier XML,
- un diagnostiqueur, PépiDiag, qui génère le diagnostic local aux exercices du test, c'est à dire qu'il mémorise le codage de chaque réponse d'un élève sur plusieurs dimensions dans un fichier XML.

3.2.1. Le système auteur

Le système auteur PépiGen charge le fichier XML (**ModelesClasseExercices**) contenant les modèles des classes d'exercices. Quand un auteur demande la génération d'un clone, d'un exercice donné, deux cas se présentent. Pour les clones à génération assistée il saisit les paramètres. Pour les autres, il valide les valeurs calculées par PépiGen ou demande de nouveau la génération d'un clone si ces valeurs ne lui conviennent pas.

Pour les exercices mettant en œuvre une expression algébrique, PépiGen fait appel au module Pépinière en lui passant cette expression et le problème à résoudre (par exemple prouver par développement et réduction). Pépinière construit l'arbre des solutions anticipées et le retourne à PépiGen sous la forme d'une liste de pointeurs (Chapitre 6, section 5). PépiGen exploite alors cet arbre pour générer les différentes solutions anticipées ainsi que la grille de codage associée au clone d'exercice (Cf. section 6). Ce clone est enregistré dans le fichier XML (**BanqueExercices**) qui constitue la banque d'exercices.

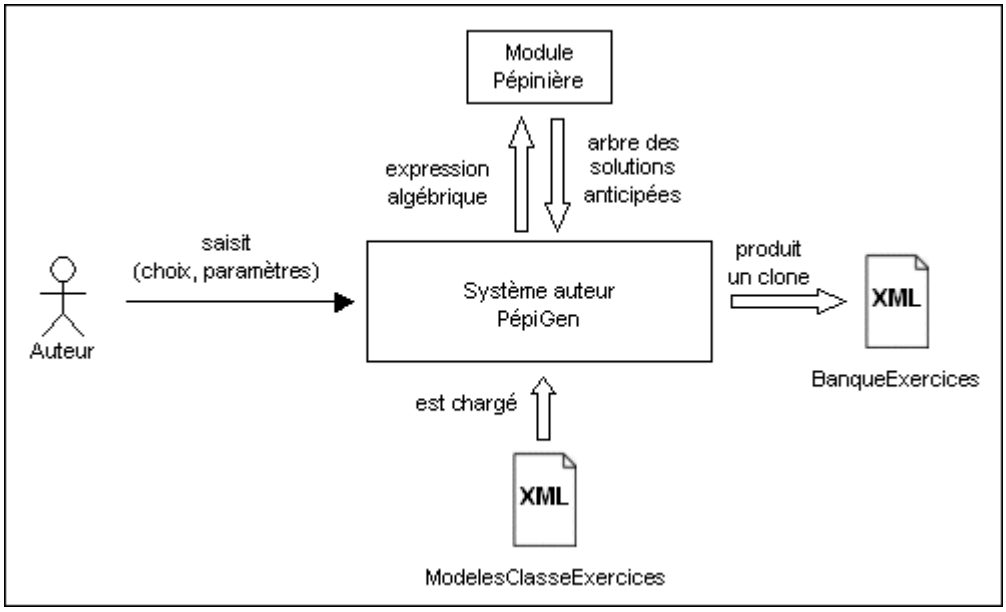


Figure 4 : Architecture fonctionnelle de PépiGen

3.2.2. Interpréteur et diagnostiqueur

L'interpréteur permet à un élève de répondre à l'ensemble des exercices d'un test et mémorise les réponses de l'élève. Ce module charge le fichier XML contenant les clones d'exercices qui constituent un test et les affiche. L'interpréteur enregistre les réponses de l'élève dans un fichier XML qui contient une référence au test et les réponses de l'élève. Puis, ce fichier de réponses est chargé par le diagnostiqueur qui effectue le codage des réponses aux exercices du test. Lorsque les réponses anticipées sont des expressions algébriques, le diagnostiqueur fait appel au module Pépinière pour tester l'équivalence de deux expressions algébriques : la réponse de l'élève et la réponse anticipée.

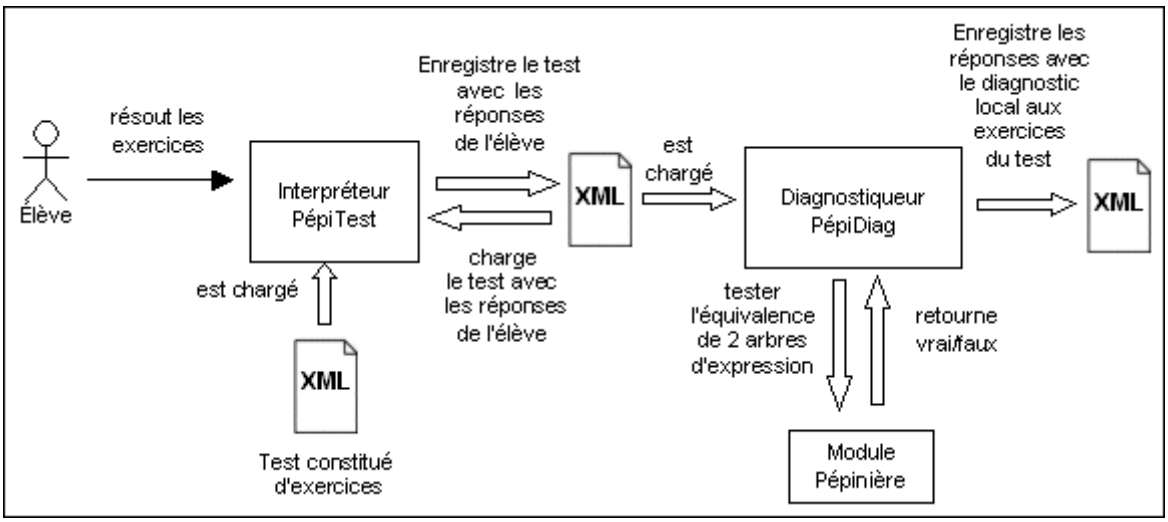


Figure 5 : Architecture fonctionnelle de l'interpréteur et du diagnostiqueur.

4. Architecture logicielle

Pour organiser la structuration logique de notre système nous utilisons le pattern « Couches » [Larman 2003]. Selon ce pattern, la structure logique du système est organisée en couches ayant chacune une responsabilité distincte. De façon classique nous distinguons quatre couches : une couche présentation, une couche application, une couche domaine (appelée aussi couche métier), une couche persistance (Figure 6). Chacune de ces couches correspondent à un package. Certaines sont découpées en plusieurs packages qui regroupent les classes autour de fonctionnalités ou de concepts.

4.1. Couche domaine

La couche domaine est formée de deux packages : un package, pepiniere, correspond au module Pépinière (Chapitre 6) et l'autre package, exercices, contient l'ensemble des classes issues du modèle conceptuel d'une classe d'exercices. Seules les principales classes ont été représentées sur la Figure 6. Notons que le package pepiniere est un module de calcul formel indépendant qui peut être utilisé dans d'autres contextes.

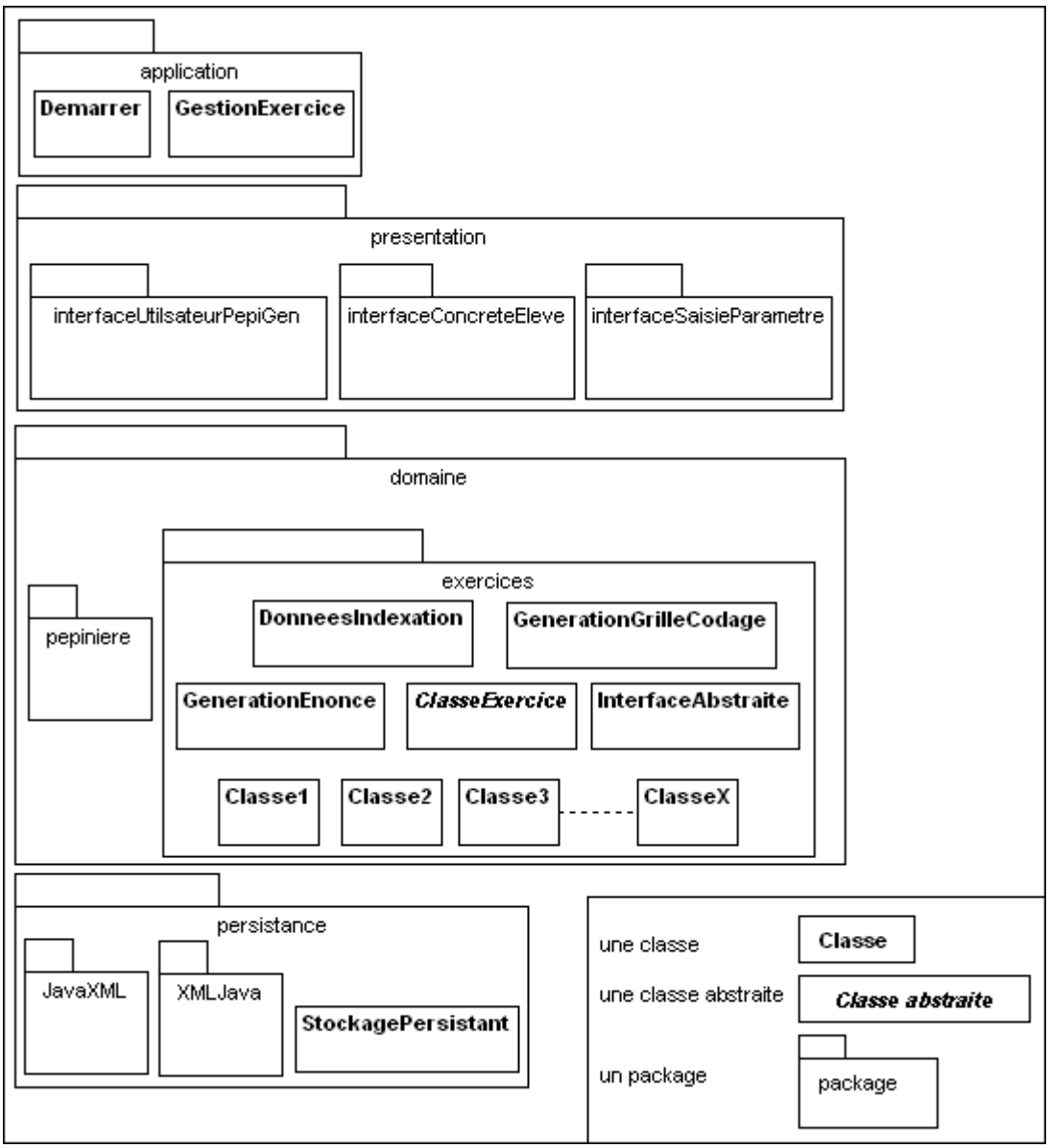


Figure 6 : Architecture logicielle de PépiGen (vue partielle)

4.2. Couche présentation

Les deux cas d'utilisation présentés dans la section 2 mettent en jeu des interfaces graphiques. Pour les actions « Choisir un modèle d'exercice » et « Saisir les paramètres d'un exercice », les interfaces graphiques utilisées gèrent les interactions entre l'auteur et le système **PépiGen** alors que l'action « Voir l'exercice » affiche l'interface élève du clone d'exercice. Nous avons regroupé dans le package interfaceUtilisateurPepiGen toutes les classes qui mettent en œuvre les interfaces graphiques qui gèrent les interactions entre un utilisateur et **PépiGen**.

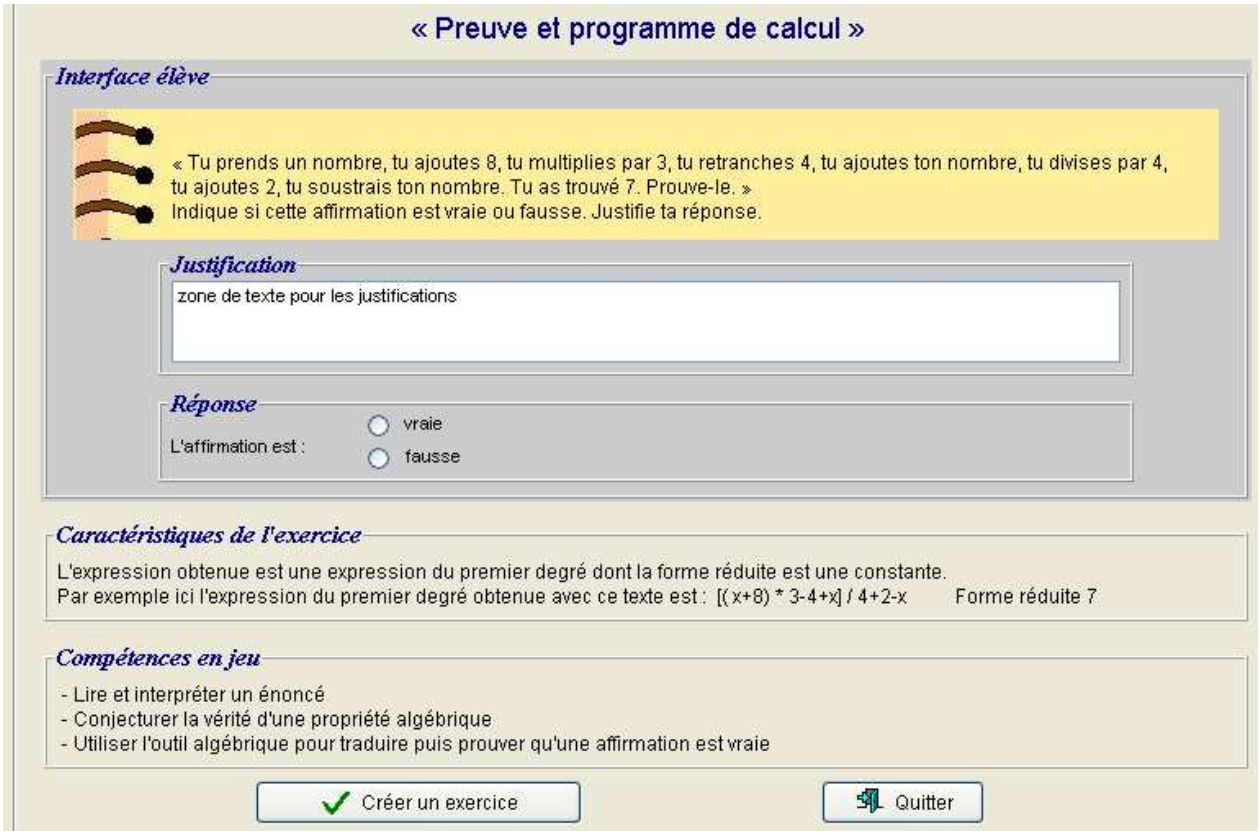


Figure 7 : « Créer un exercice » pour la classe « Preuve et programme de calcul »

Le package interfaceSaisieParametre correspond aux interfaces graphiques de saisie des paramètres utilisés pour les exercices dont la génération est assistée et qui sont spécifiques à chaque exercice (Figure 7, Figure 8).

La section 5 présente l'interface de saisie des paramètres de l'exercice de la classe « Preuve et programme de calcul » pour la génération des énoncés comportant un programme de calcul. Le package interfaceConcreteEleve regroupe toutes les classes qui participent à l'affichage des interfaces élèves des clones d'exercices, et, qui sont affichées pour les actions « Voir un exercice ».

Le package interfaceUtilisateurPepiGen est en relation avec la couche application et la couche persistance tandis que les deux autres packages sont en relation avec la couche domaine car ils affichent des interfaces étroitement liées au domaine de PépiGen.

4.3. Couche application

Cette couche sert d'intermédiaire entre la couche domaine et les autres couches. La classe **Demarrer** correspond à l'opération système qui consiste à lancer l'exécution de PépiGen. Cette classe démarre l'exécution de PépiGen en présentant la première interface utilisateur de PépiGen

et en procédant aux tâches d'initialisation. Ainsi la classe **Demarrer** appelle la classe **StockagePersistant** qui gère les opérations sur les fichiers XML. Ce package est en relation avec les trois autres packages.

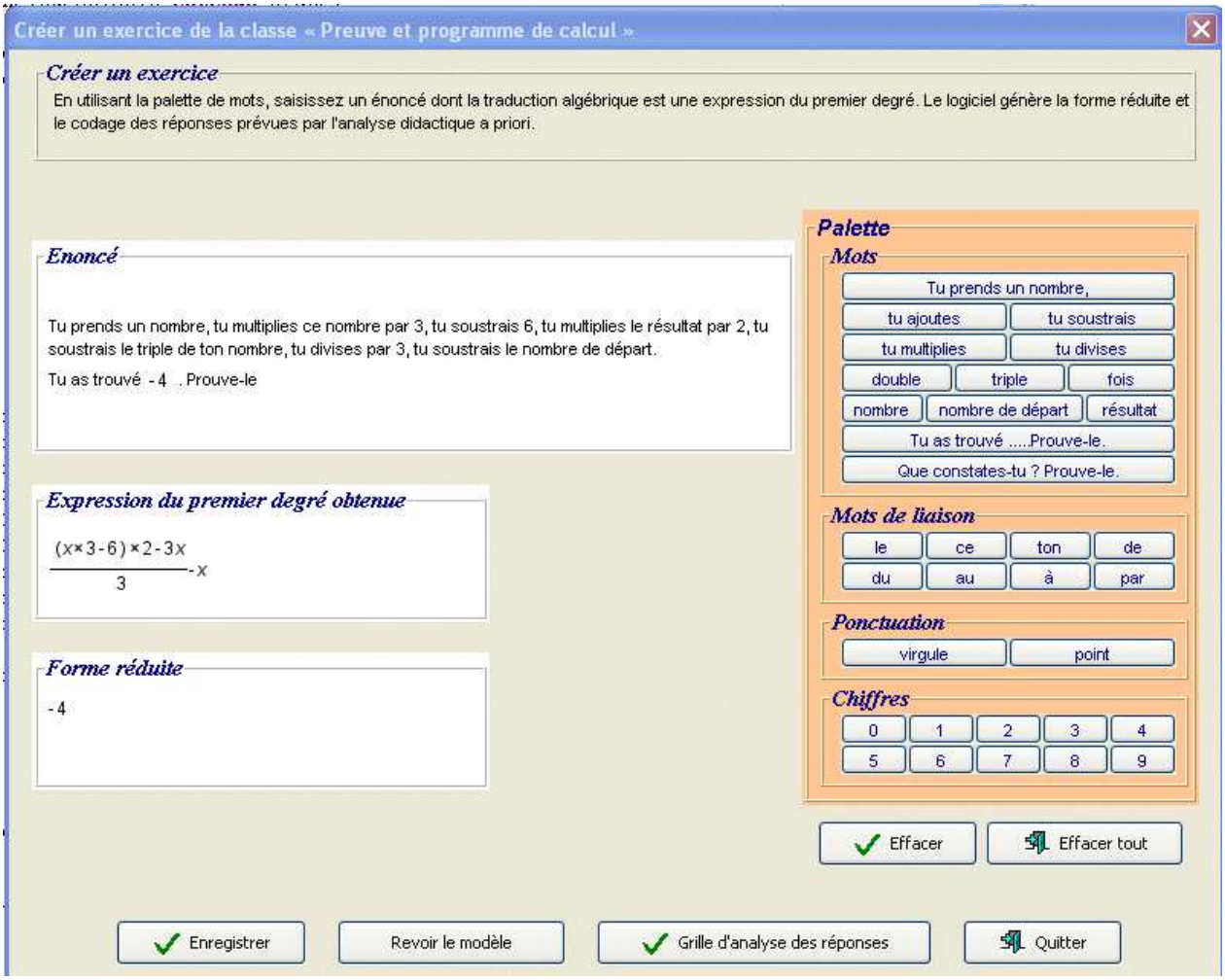


Figure 8 : « Saisir les paramètres » pour la classe « Preuve et programme de calcul »

4.4. Couche persistance

La couche persistance a pour objet les échanges entre les fichiers XML et PépiGen : les classes de ce package réalisent les actions spécifiques aux systèmes de gestion de fichiers (e.g. création, ouverture, lecture et écriture) et utilisent l'API DOM. La structure des fichiers XML est décrite aux sections 7.1 et 7.2. La classe **StockagePersistant** gère les opérations de gestion des fichiers **BaseTests**, **BanqueExercices** et **BaseModelExercices**.

5. Génération des deux représentations d'un programme de calcul

Dans le système PépiGen, tant au niveau de génération d'énoncés d'exercices que de la résolution d'exercices, un problème récurent consiste à transformer un programme de calcul écrit en langage naturel en une expression algébrique et réciproquement. Pour cela nous avons défini un composant logiciel réutilisable réalisé par un automate fini déterministe complet.

Pour écrire un programme de calcul, l'utilisateur dispose d'une palette de termes qui a été mise au point par un travail approfondi avec les enseignants et des tests auprès des élèves[2]. Chaque touche constitue le vocabulaire de l'automate. La frappe d'une touche provoque ainsi un changement d'état de l'automate (Figure 9 et Figure 10). Par exemple, la frappe de la touche *tu ajoutes* provoque le passage à l'état numéroté 2. Cet automate reconnaît des phrases telles « *tu prends un nombre, tu soustrais le nombre de départ, tu ajoutes* » ou « *tu prends un nombre, tu multiplies par, tu ajoutes* » (Figure 10). Les différentes transitions ont été définies par une analyse précise avec les experts et les praticiens du domaine des programmes de calcul proposés à un certain niveau de compétences en algèbre élémentaire. La suite des transitions permet de générer un programme de calcul en langage naturel et l'expression algébrique obtenue.

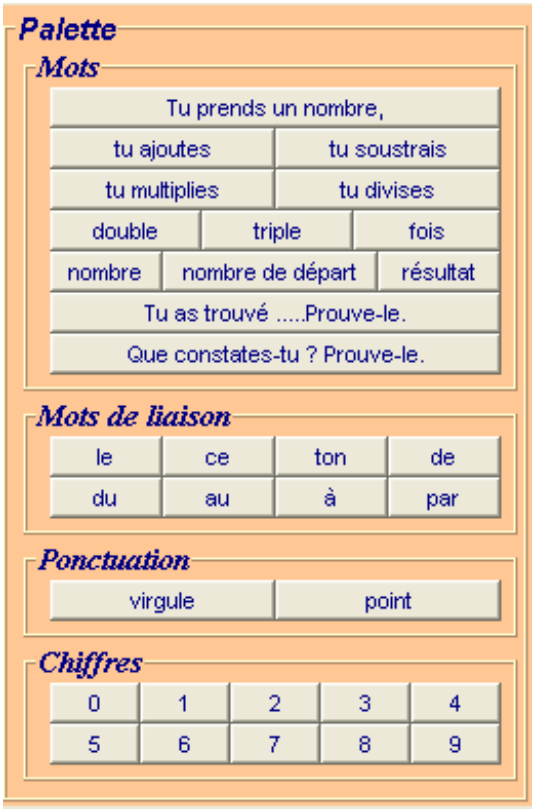


Figure 9 : Palette de termes de la classe d'exercices « Preuve et programme de calcul »

[2] Par exemple le choix entre « tu soustrais » et « tu retranches ».

Cet automate est défini par le quintuplet (V, Q, T, I, F) où V est le vocabulaire de l'automate, Q l'ensemble fini des états de l'automate, T la fonction de transition. L'état initial, I = {q_0}, correspond à la frappe de la touche « *Tu prends un nombre* ». Il y a deux état finals F = {q_{41} et q_{40}} qui correspondent respectivement à la frappe des touches « *Tu as trouvé* ou *Que constates-tu ?* » . T fonction de transition de l'automate est définie par : T : Q×V → Q avec T(q_i, a) = q_j. q_i et q_j sont des éléments de l'ensemble Q, a un élément de V (e.g. T(q_7, « *tu ajoutes* ») = q_{25}). Le vocabulaire V est l'ensemble des expressions, des mots et des chiffres de la palette (Figure 9) et l'ensemble des états est Q = { ensemble des états numérotés de q_0 à q_{41} }.

Nous enrichissons cet automate en générant pour chaque transition la formule obtenue. Reprenons l'exemple précédent : la phrase « tu prends un nombre, tu soustrais 6, *tu ajoutes* » correspond à la formule « x – 6 + », la phrase « tu prends un nombre, tu multiplies par 8, *tu ajoutes* » à la formule « x × 8+ ».

La Figure 10 présente un extrait de la représentation de cet automate qui est donné incomplet pour en alléger la description. Pour chaque état, repéré par son numéro, nous avons représenté seulement les transitions qui construisent l'énoncé et la formule. Par exemple pour l'état q_1, seuls 4 éléments du vocabulaire provoquent un changement d'état : *tu ajoutes, tu soustrais, tu multiplies, tu divises*. Les autres éléments laissent l'automate dans l'état q_1.

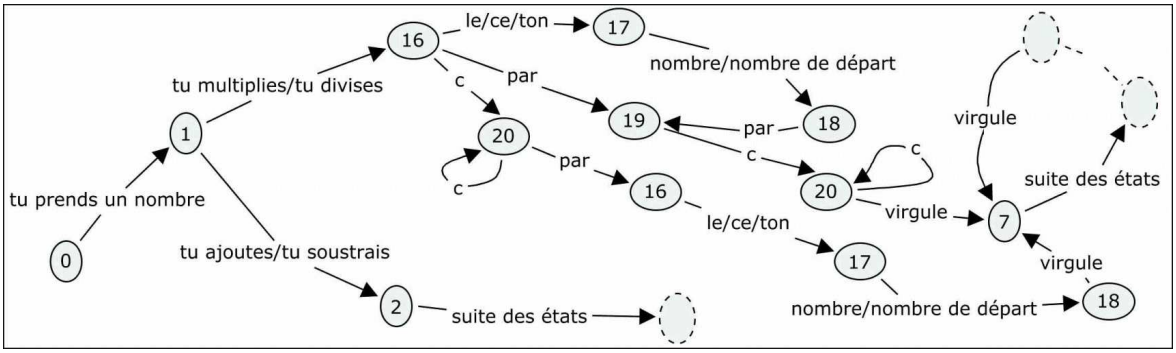


Figure 10 : Extrait de la représentation de l'automate (la lettre c représente un chiffre).

L'automate comportant des cycles, le langage engendré est infini. Généralement, pour la résolution d'exercices la longueur des énoncés est limitée par le nombre d'opérations de la formule de calcul et le nombre de parenthèses qui sont des variables didactiques à prendre en compte pour caractériser les exercices générés.

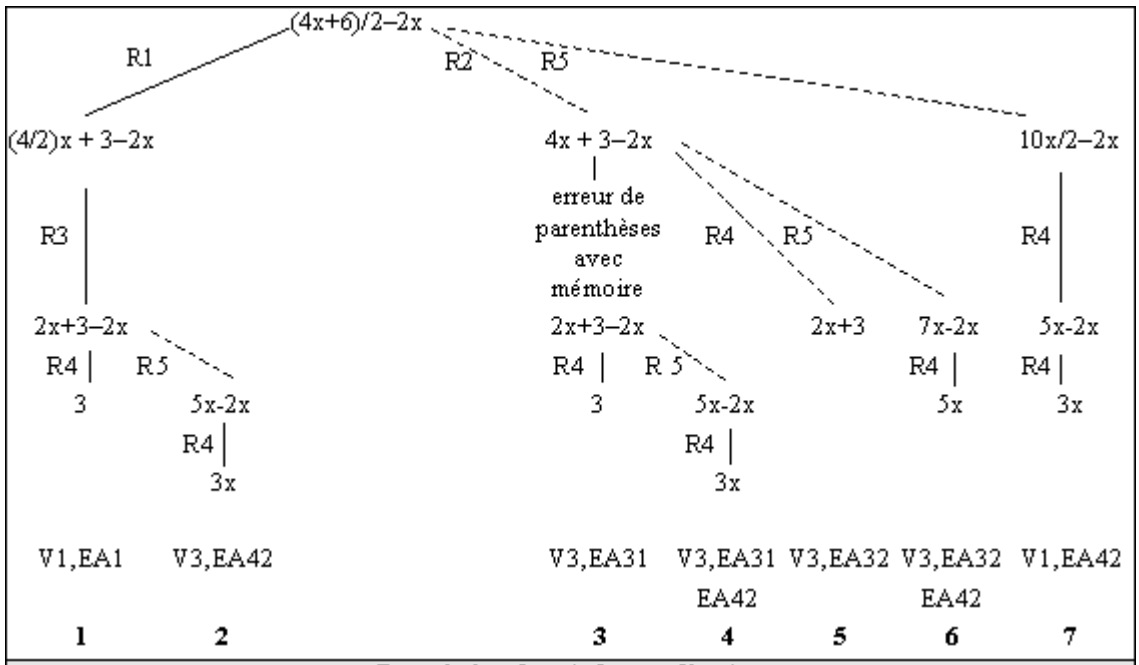
Cet automate est utilisé par un auteur pour la saisie des paramètres des exercices mettant en jeu des expressions d'un programme de calcul. Il est aussi utilisé par les élèves pour la saisie des réponses à un exercice comportant un programme de calcul (e.g. clones d'exercices des classes « Preuve et programme de calcul », « Expressions algébriques d'un programme de calcul »).

6. Génération de la grille de codage

C'est la fonctionnalité principale de notre système et ce qui fait son originalité. En effet en nous fondant sur l'analyse didactique a priori, nous générons l'ensemble des solutions plausibles anticipées et nous en proposons une analyse multidimensionnelle qui est associée à la caractérisation du clone d'exercice

Le schéma conceptuel (Chapitre 5, section 11.4) fait apparaître trois types de réponses : les réponses aux questions ouvertes, les réponses aux questions fermées et les réponses à des questions composées comprenant des questions ouvertes et des questions fermées. Les réponses aux questions fermées se présentent sous la forme de plusieurs choix (choix multiples ou choix exclusifs). A chaque choix correspond un code. Dans ce cas, la grille de codage est un invariant de la classe d'exercices.

Pour les réponses aux questions ouvertes sous la forme d'une suite de calculs algébriques, contrairement au cas précédent, la grille de codage est caractéristique du clone et non pas de la classe. PépiGen doit donc générer cette grille. Pour cela, il utilise l'arbre des solutions anticipées créé par Pépinière pour générer les différentes solutions qu'il mémorise dans le fichier XML décrivant le clone (Chapitre 6 section 5). Pour étayer la description du processus de génération de la grille de codage nous présentons d'abord sur un exemple l'arbre des solutions anticipées puis nous décrivons les différentes solutions rencontrées pour résoudre le problème donné.



Description des règles appliquées	
Règles correctes	R1 : (A+B)/C → (A/C) + (B/C) R3 : (AB)/C → (A/C) *B R4 : AC+BC→ (A+B)C : somme de monômes semblables
Règles erronées	R2 : (A+B)/C → A + (B/C) R5 : AC+B→ (A+B)C appliquée à : 2x+3, 4x+3
Description des codes	
Validité	V1 : traitement correct / V3 : traitement incorrect
Utilisation des règles d'Écriture et de réécriture Algébrique	EA1 : Utilisation correcte des règles de transformation
	EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct Remarque : certains élèves n'indiquent pas de parenthèses mais en conservent le sens. Cette erreur est codée EA31 (erreur de parenthèse avec mémoire) alors qu'une erreur de parenthèse sans mémoire de l'énoncé est codée EA32.
	EA32 : Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect
	EA42 : Les règles de transformation utilisées assemblent les termes

Figure 11: Arbre des solutions anticipées pour la tactique « simplifier les fractions » appliquée à l'expression (4*x*+6)/2-2*x*

La Figure 11 représente les branches de l'arbre des solutions anticipées. La Figure 12 est produite par PépiGen lors d'un parcours « fils-aîné-frère-droit » de l'arbre créé par Pépinière pour les solutions du problème « preuve par développement et réduction » de l'expression (4*x+6)/2-2*x (Figure 12). La colonne gauche du tableau a été ajoutée pour montrer la correspondance entre les branches de l'arbre présenté dans la Figure 11 et l'arbre créé par Pépinière. Les résultats complets se trouve en annexe B3.

	Expression	Règle	Type de règle	Codage
	../...			
1	(4*x+6)/2-2*x (4/2)*x+6/2-2*x 2*x+3-2*x 3	(A+B)/C ->A/C+B/C AC+BC ->(A+B)C	Règle correcte Règle correcte	V1,EA1 V1,EA1
2	(4*x+6)/2-2*x (4/2)*x+6/2-2*x 2*x+3-2*x 5*x-2*x 3*x	(A+B)/C ->A/C+B/C AC+B ->(A+B)C AC+BC ->(A+B)C	Règle correcte Règle erronée Règle correcte	V1,EA1 V3,EA42 V1,EA1
3	(4*x+6)/2-2*x 4*x+3-2*x 2*x+3-2*x 3	(A+B)/C ->A+(B/C) AC+BC ->(A+B)C	Règle erronée Mémoire énoncé Règle correcte	V3,EA32 V3,EA31 V1,EA1
4	(4*x+6)/2-2*x 4*x+3-2*x 2*x+3-2*x 5*x-2*x 3*x	(A+B)/C ->A+(B/C) AC+B ->(A+B)C AC+BC ->(A+B)C	Règle erronée Mémoire énoncé Règle erronée Règle correcte	V3,EA32 V3,EA31 V3,EA42 V1,EA1
5	(4*x+6)/2-2*x 4*x+3-2*x 2*x+3	(A+B)/C ->A+(B/C) AC+BC ->(A+B)C	Règle erronée Règle correcte	V3,EA32 V1,EA1
6	(4*x+6)/2-2*x 4*x+3-2*x 7*x-2*x 5*x	(A+B)/C ->A+(B/C) AC+B ->(A+B)C AC+BC ->(A+B)C	Règle erronée Règle erronée Règle correcte	V3,EA32 V3,EA42 V1,EA1
7	(4*x+6)/2-2*x 10*x/2-2*x (10/2)*x-2*x 5*x-2*x 3*x	 AC+B ->(A+B)C (AB)/C ->(A/C)B AC+BC ->(A+B)C	 Règle erronée Règle correcte Règle correcte	 V3,EA42 V1,EA1 V1,EA1

Figure 12 : Parcours « fils-aîné-frère-droit » de l'arbre des solutions anticipées

Pour générer la grille d'analyse des réponses anticipées, PépiGen travaille en deux étapes :

- Tout d'abord, PépiGen fait appel à Pépinière en lui fournissant en entrée un problème à résoudre. Pépinière retourne en résultat l'arbre des solutions anticipées qui est construit en appliquant à l'expression de départ puis aux différentes expressions obtenues les règles de calcul correctes ou erronées applicables. Pépinière mémorise dans chaque nœud de l'arbre l'expression obtenue ainsi que le numéro de la règle correcte ou erronée qui a été appliquée. Comme nous l'avons décrit au chapitre 6, Pépinière contrôle l'explosion combinatoire et les bouclages. Cet arbre ne prend en compte que les solutions algébriques et le codage n'est

effectué que sur les dimensions d'évaluation « Validité » et « Utilisation des règles d'Écriture et de réécriture Algébrique ».

- Puis, un algorithme spécifique à la classe d'exercices conçu à partir de l'analyse didactique prend en compte les différentes approches et les différentes tactiques de calcul. Il parcourt cet arbre et associe à chaque parcours un code sur les différentes dimensions d'évaluation en jeu dans la classe d'exercices. L'algorithme distingue les solutions correctes, les solutions incorrectes et selon la classe d'exercices les solutions correctes non optimales et les solutions partielles. Toutes ces solutions et le codage associé, sont mémorisés dans le fichier XML qui caractérise le clone.

6.1. Codage associé à une branche de l'arbre des solutions

En ce qui concerne le codage, l'arbre des solutions de la Figure 11 fait apparaître pour chaque branche le code associé aux deux dimensions « Validité » (V) et « Utilisation des règles d'Écriture et de réécriture Algébrique » (EA). Ce code est élaboré à partir des codes attribués aux différentes étapes des calculs. Ainsi, par exemple, pour la branche numéroté 4, les nœuds du parcours font apparaître les codes V3, EA31, puis V3, EA32, puis V3, EA42 et enfin V1,EA1. Pépinière associe à cette branche le code V3, EA31, EA42. En effet, la réponse n'est pas valide même si la dernière règle appliquée est correcte, il y a mémoire de l'énoncé (le code EA32 n'est pas retenu, le code EA31 est retenu) et une règle de transformation utilisée assemble les termes.

Pour les autres dimensions d'évaluation, le codage est complété à partir des informations spécifiques à la classe d'exercice et réifié dans le fichier XML **ModelesClasseExercices**. Nous présentons le schéma XML et la structure de ce fichier dans la section 7.1.

6.2. Génération de la grille de codage

PépiGen complète le codage effectué par Pépinière sur les autres dimensions d'évaluation et prend en compte des solutions non algébriques. PépiGen distingue les solutions correctes, les solutions correctes non attendues et les solutions incorrectes.

Prenons un exemple. La Figure 13 présente un extrait de la grille de codage pour un clone de la classe d'exercices « Preuve et programme de calcul » paramétré par l'expression algébrique ((x*3-6)*2-3*x)/3-x . La grille de codage est présentée dans son intégralité en annexe C11.2.

Pour cette classe d'exercices, le codage des solutions partielles ou incorrectes est complexe et c'est l'algorithme de diagnostic que nous détaillons dans la section 9 qui permet à partir de la grille de codage de repérer les erreurs. Dans la pratique, les enseignants et les didacticiens des

mathématiques ont mis en évidence un certain nombre de démarches inadaptées et de règles erronées qui apparaissent de façon récurrente dans les réponses des élèves. Nous ne prétendons pas pouvoir coder toutes les étapes des productions des élèves mais repérer des cohérences de fonctionnement relatives aux différents aspects de la compétence algébrique.

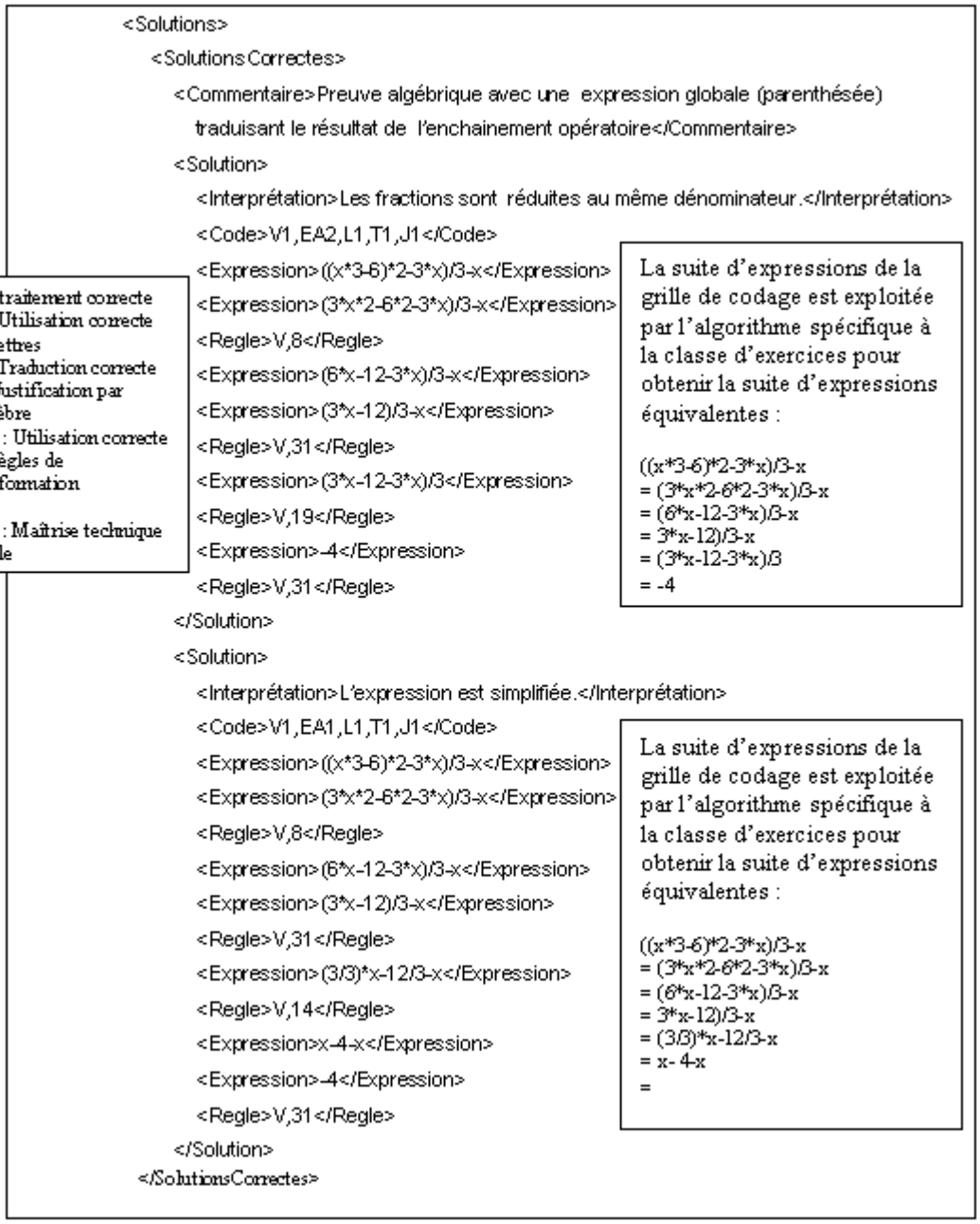


Figure 13 : Extrait de la grille de codage d'un clone de la classe d'exercices « Preuve et programme de calcul »

Une première difficulté concerne les solutions correctes non attendues. Dans l'exemple, l'expression comporte une fraction : deux tactiques de calcul sont envisageables. Elles sont

repérées dans l'arbre des solutions anticipées par le type des règles utilisées et sont codées différemment : les règles concernant la réduction des fractions au même dénominateur sont codées EA2 (maîtrise technique fragile), celles qui concernent la simplification des fractions sont codées EA1. Selon le modèle conceptuel (chapitre 5, Figure 21), la grille de codage des réponses comportant plusieurs lignes de calcul comporte la suite des expressions algébriques avec pour chaque expression les caractéristiques de la règle utilisée pour l'obtenir (e.g. V,15 pour la règle correcte numéro 15 (AB)/C → (A/C)B).

Ainsi, la solution correcte est obtenue à partir de la suite d'expressions équivalentes donnée dans la grille de codage. V,8 indique que la règle correcte numéro 8, (A+B)*C ->A*C+B*C a été appliquée pour transformer l'expression ((x*3-6)*2-3*x)/3-x en (3*x*2-6*2-3*x)/3-x . La Figure 14 présente une copie d'écran lorsque l'auteur demande l'affichage de la grille de codage obtenue pour les solution incorrectes.

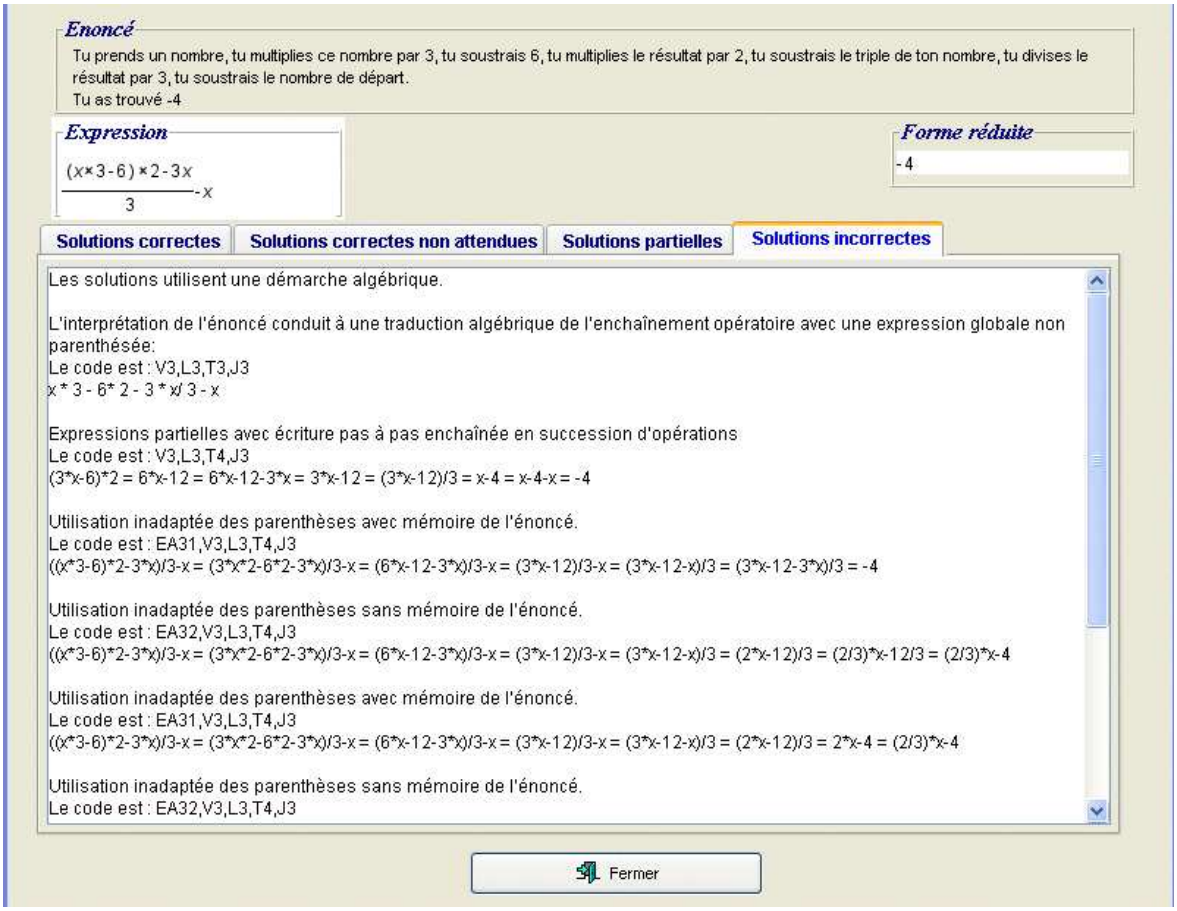


Figure 14 : « Voir la grille de codage des réponses »

6.2.1. Solutions correctes non optimales

Ce type de solution est rencontré lorsque les calculs sont corrects mais que la démarche utilisée n'est pas celle qui est attendue. A un niveau de classe de troisième dans le système scolaire français, le développement d'une expression du second degré utilise les identités remarquables. Une autre démarche est une solution non optimale.

Pour la classe d'exercices « Preuve et programme de calcul » deux démarches non optimales apparaissent : la première utilise un enchaînement de calculs pas à pas, la deuxième correspond à une interprétation de l'énoncé comme une équation admettant une infinité de solutions.

Pour l'exemple qui nous sert de guide, la première démarche se traduit par l'enchaînement des expressions :

(x*3-6)*2 = 6*x - 12 ; 6*x -12 - 3*x = 3*x - 12 ; (3*x - 12)/3 = x - 4 ; x - 4 - x = - 4

Pour générer cette solution, Pépinière construit à partir de l'expression de départ (e.g. (x*3-6)*2) un arbre des solutions anticipées (Figure 15) selon le même principe que l'arbre construit pour les solutions correctes (application des règles correctes ou erronées) avec à chaque pas de calcul l'ajout de l'opération proposée dans la suite du programme de calcul.

La deuxième démarche se présente sous la forme d'une résolution d'équations obtenues à partir de l'arbre des solutions anticipées construit pour les solutions correctes.

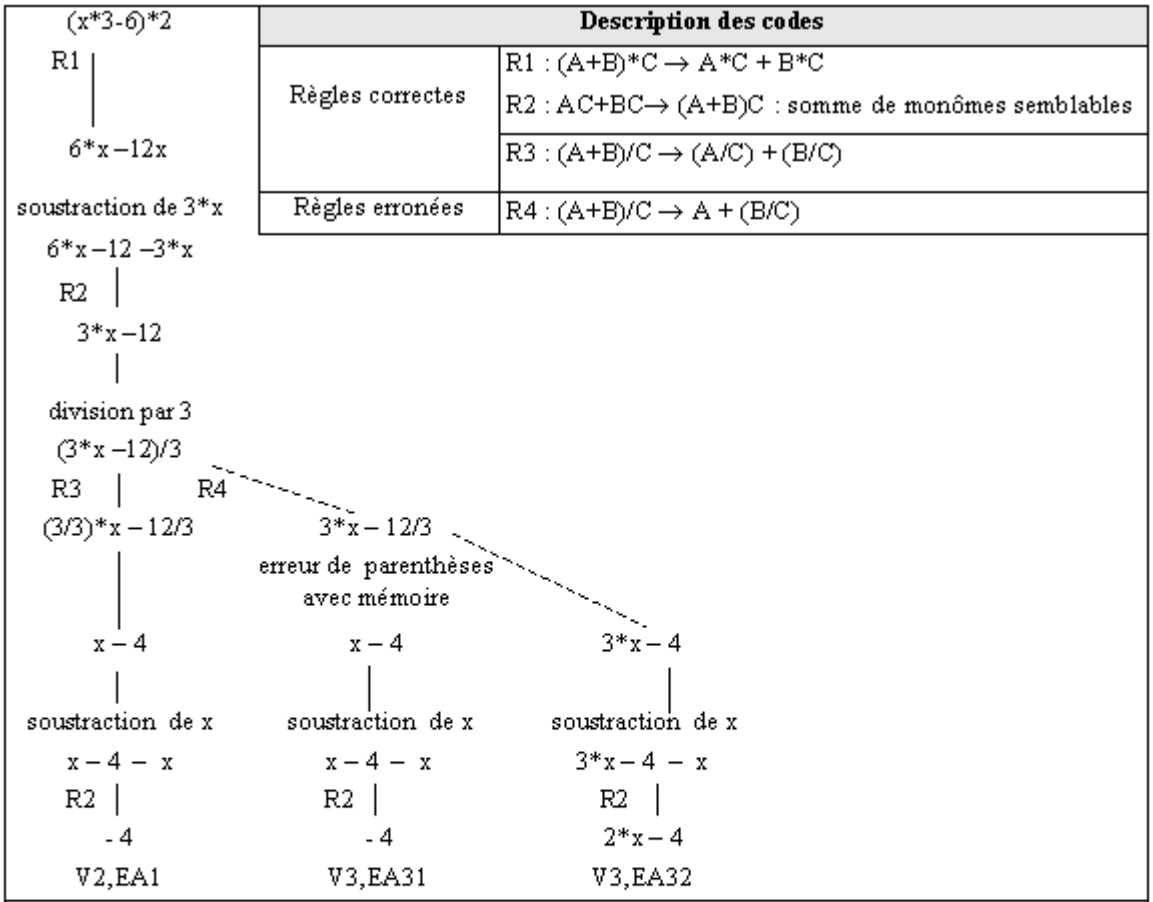
6.2.2. Solutions partielles

Une solution correcte ou une solution correcte non optimale est engagée mais une erreur de calcul conduit à un résultat faux. La grille de codage reprend celle des solutions précédentes (solutions correctes et correctes non attendues) ; c'est l'algorithme de diagnostic spécifique à la classe d'exercices (section 9) qui repère que la réponse n'est pas conforme à la réponse correcte anticipée et qui applique la partie de la grille de codage concernant les solutions incorrectes.

6.2.3. Solutions incorrectes

Dans l'exemple présenté pour la solution correcte, deux types de démarches sont envisagés : une démarche algébrique et une démarche numérique. Pour les démarches algébriques incorrectes, PépiGen exploite tous les parcours obtenus par application de règles erronées dans les arbres des solutions anticipées construits pour les solutions correctes et le raisonnement pas-à-pas. A ces parcours s'ajoutent des expressions algébriques particulières proposées dans l'analyse a priori (e.g. écriture de la formule de calcul privée des parenthèses). Pour les démarches numériques, c'est l'algorithme de diagnostic local à la classe d'exercices qui repère la

substitution de la variable par des valeurs numériques au début ou au cours des calculs (section 9). C'est aussi cet algorithme qui repère le codage de la dimension « Traduction » (écriture globale parenthésée ou pas-à-pas) mais les écritures formelles dans ce cas ne peuvent pas être codées. Nous présentons l'algorithme de diagnostic à la section 9 et nous présentons des tests de la génération de la grille de codage et de l'algorithme de diagnostic en annexe E.



Description des codes		
Règles correctes	R1 : (A+B)*C → A*C + B*C	
	R2 : AC+BC→ (A+B)C : somme de monômes semblables	
	R3 : (A+B)/C → (A/C) + (B/C)	
Règles erronées	R4 : (A+B)/C → A + (B/C)	

Description des codes	
Validité	V2 : traitement correct non attendu V3 : traitement incorrect
Utilisation des règles d'Écriture et de réécriture Algébrique	EA1 : Utilisation correcte des règles de transformation
	EA31 : Utilisation inadaptée des parenthèses qui conduit à un résultat correct Remarque : certains élèves n'indiquent pas de parenthèses mais en conservent le sens. Cette erreur est codée EA31 (erreur de parenthèse avec mémoire) alors qu'une erreur de parenthèse sans mémoire de l'énoncé est codée EA32.
	EA32 : Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect

Figure 15: Arbre des solutions anticipées pour une démarche de calcul pas à pas : (4*x*+6)/2-2*x*

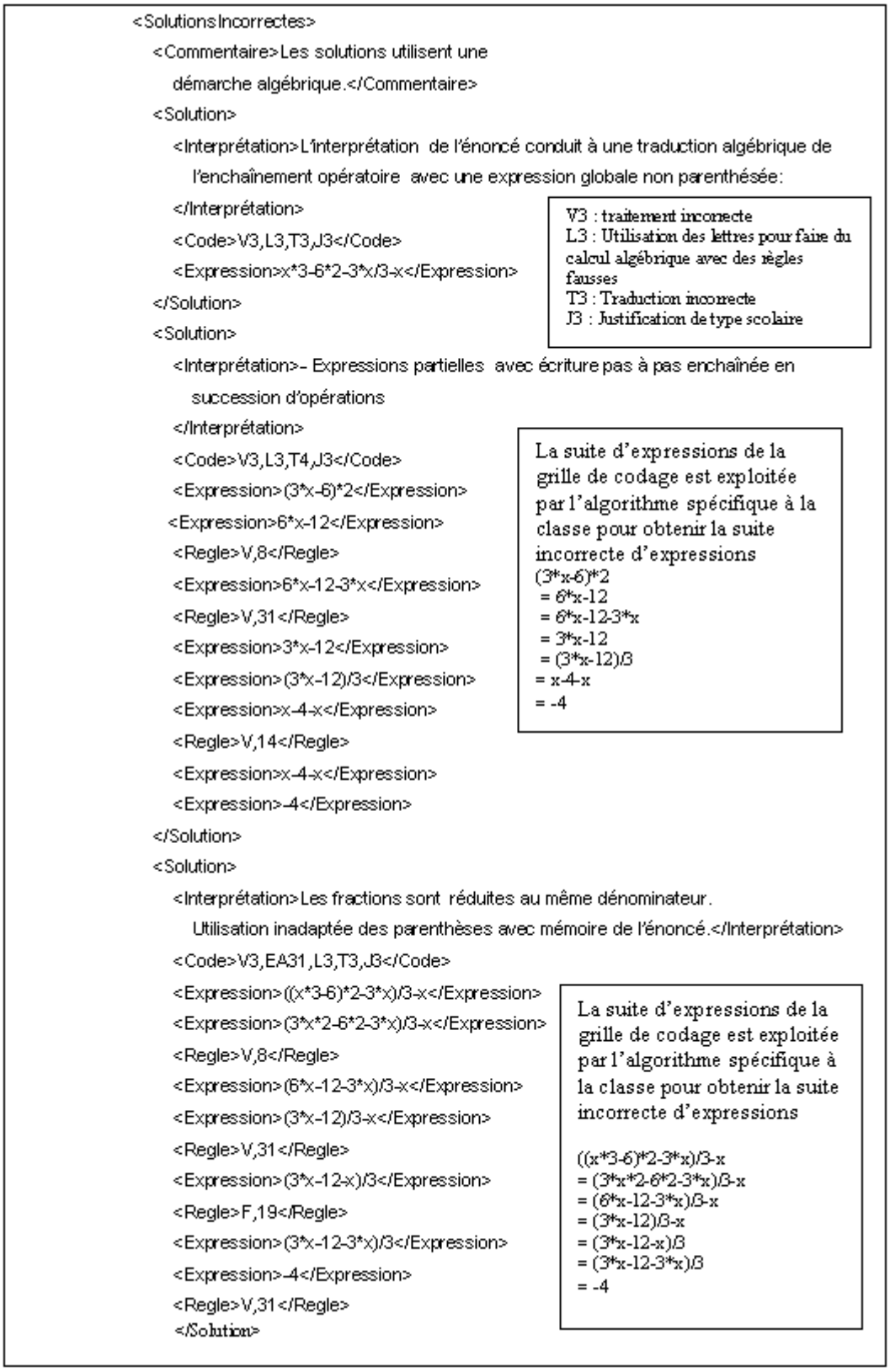


Figure 16 : Extrait de la grille de codage d'un clone de la classe d'exercices « Preuve et programme de calcul »

7. Persistance des données

Les clones d'exercices sont stockés dans le fichier XML **BanqueExercices** et les patrons d'exercices sont stockés dans le fichier XML **ModelesClasseExercices**. Pour vérifier que les documents XML contenus dans ces fichiers sont valides nous utilisons XML Schéma. Nous détaillons dans les sections qui suivent la structure de ces fichiers et les différents éléments du schéma XML correspondant.

7.1. Structure du fichier XML « ModelesClasseExercices »

Nous avons élaboré le modèle de données de ces fichiers XML à partir du modèle conceptuel d'une classe d'exercices défini dans le chapitre 5.

7.1.1. Racine du schéma XML du fichier « ModelesClasseExercices »

Le Figure 17 présente la racine de l'arbre du schéma XML du fichier **ModelesClasseExercice**. Ce fichier contient les modèles des classes d'exercices. Chacune de ces classes est décrite selon un type complexe (signalé par +). La classe **ClasseExercice** est une agrégation des classes **DonneeIndexation**, **GénérationEnonce**, **InterfaceAbstraite** et **GenerationGrilleCodage**.

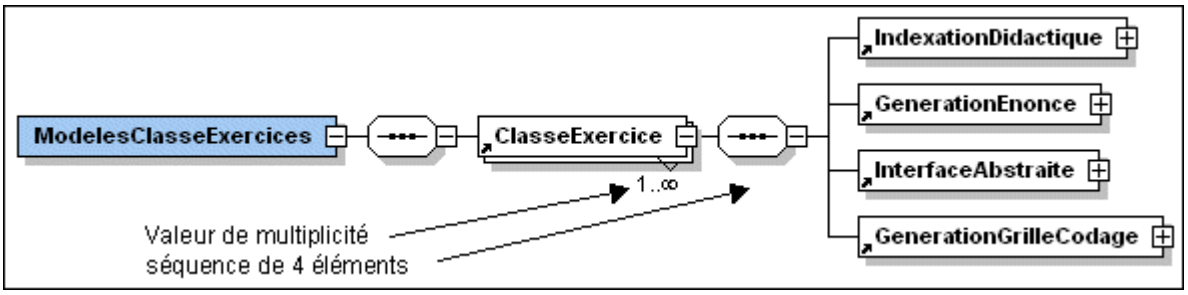


Figure 17 : Structure du fichier **ModelesClasseExercices**

```
<xs:element
name="ModelesClasseExercices">
        <xs:complexType>
            <xs:sequence>
                <xs:element
ref="ClasseExercice"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```

```
<xs:element name="ClasseExercice">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="IndexationDidactique"/>
                <xs:element ref="GenerationEnonce"/>
                <xs:element ref="InterfaceAbstraite"/>
                <xs:element ref="GenerationGrilleCodage"/>
            </xs:sequence>
            <xs:attribute name="Nom" type="xs:string" use="required"/>
            <xs:attribute name="Numero" type="xs:int" use="required"/>
        </xs:complexType>
    </xs:element>
```

Tableau 1.a : Schéma XML de **ModelesClasseExercices** - 1.b **:** Schéma XML de **ClasseExercice**

Une classe d'exercices a deux attributs qui sont « Nom » et « Nnuméro » (Tableau 1.b). « Preuve et programme de calcul » est le nom de la classe utilisée dans l'exemple de la section 6 et le numéro est 16 (numéro de l'exercice originel).

7.1.2. Schéma XML des données d'indexation didactique

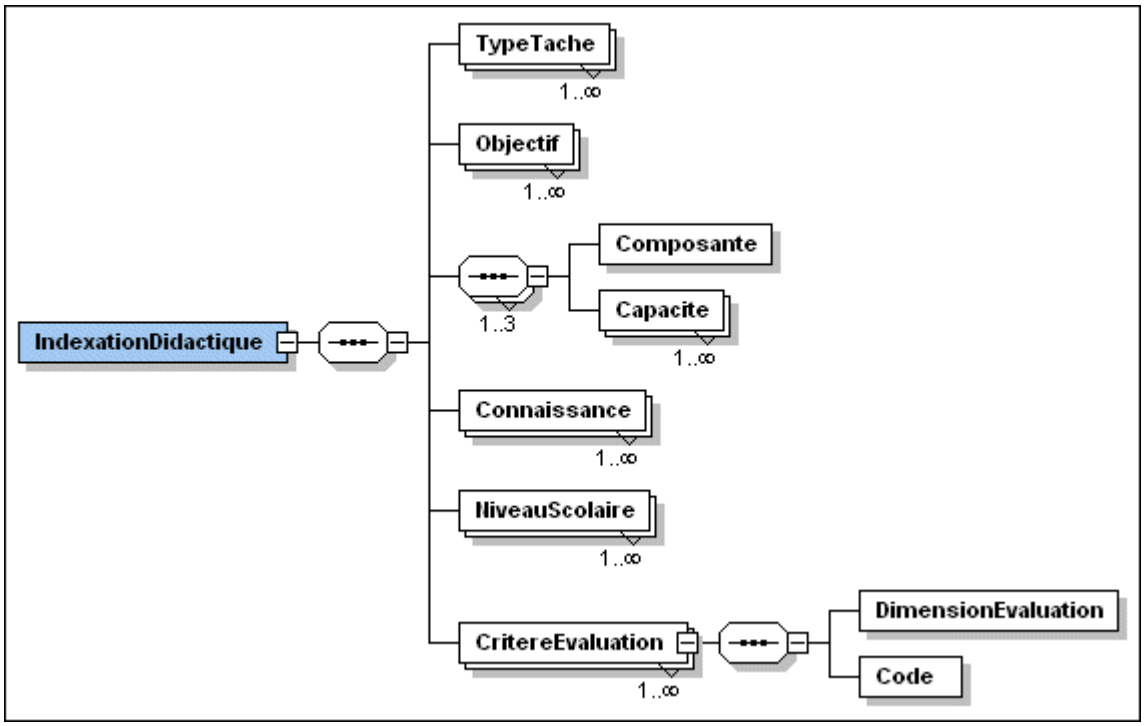


Figure 18 : Structure de la classe **IndexationDidactique**

```
<IndexationDidactique>
    <TypeTache>Calculer l'aire d'un domaine plan</TypeTache>
    <TypeTache>Associer une expression algébrique comme une aire de surface</TypeTache>
    <Objectif>Rechercher si un élève sait associer une expression algébrique à un domaine plan</Objectif>
    <Composante>Traduire algébriquement dans différents cadres et vice-versa</Composante>
    <Capacite>Traduire une expression algébrique comme une aire de surface</Capacite>
    <Connaissance></Connaissance>
    <NiveauScolaire>4,3</NiveauScolaire>
    <CritereEvaluation>
        <DimensionEvaluation>Validité</DimensionEvaluation>
        <Code>V1,V3</Code>
            </CritereEvaluation>
    <CritereEvaluation>
        <DimensionEvaluation>Traduction</DimensionEvaluation>
                <Code>T1,T3</Code>
    </CritereEvaluation>
</IndexationDidactique>
```

Tableau 2: Données d'indexation didactique - extrait du fichier XML de la classe d'exercices « Correspondance entre aire et expression »

La Figure 18 présente la structure des données d'indexation didactique et le Tableau 2 un extrait du fichier XML correspondant au modèle d'exercice « Correspondance entre aire et expression » (Chapitre 5, section 8)

7.1.3. Schéma XML de la classe GenerationEnonce

La Figure 19 présente la structure de la classe **GenerationEnonce**.

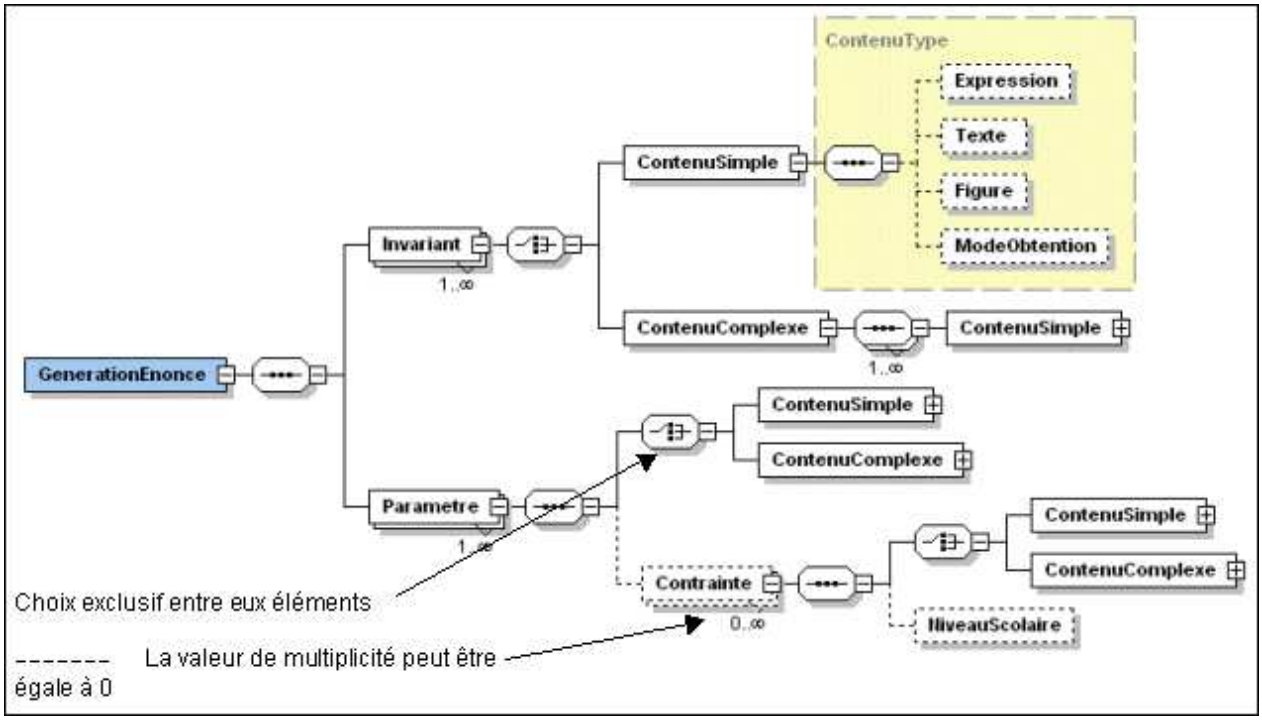


Figure 19 : Structure de la classe **GenerationEnonce**

Les classes **Parametre** et **Invariant** ont un attribut, « Identificateur », pour repérer leur contenu. Il est utilisé pour générer les énoncés des parties, des questions ou des choix dans l'interface abstraite. Dans le Tableau 3 le paramètre a pour identificateur « p1 ». C'est l'un des paramètres de la classe d'exercices « Preuve et programme de calcul » : le programme de calcul en langage naturel saisi avec la palette de termes (Figure 9). Ce programme de calcul est obtenu à l'exécution de PépiGen.

```
<Parametre Identificateur="p1">
    <ContenuSimple>
        <Texte></Texte>
        <ModeObtention>classe16.generer(programme)</ModeObtention>
    </ContenuSimple>
</Parametre>
```

Tableau 3 : Paramètres - extrait du fichier XML de la classe d'exercices « Preuve et programme de calcul »

Nous retrouvons l' « Identificateur » de ce paramètre dans la description du contenu de l'énoncé de cette classe d'exercices (Tableau 4).

```
<InterfaceAbstraite>
    <Enonce>
        <ContenuSimple>
            <Texte>p1</Texte>
        </ContenuSimple>
    </Enonce>
../…..
```

Tableau 4 : Énoncé - extrait du fichier XML de la classe d'exercices « Preuve et programme de calcul »

La classe **Contenu e**st une classe abstraite utilisée pour décrire les contenus des instances des classes **Invariant**, **Contrainte**, **Enonce**, **ChoixMultiple** et **ChoixSimple** (Chapitre 5, Figure 18 et Figure 19). C'st une généralisation des deux sous-classes **ContenuSimple** et **ContenuComplexe**. Nous avons créé un type **ContenuType** (Figure 20) pour la classe **Contenu**.

```
<xs:complexType name="ContenuType">
    <xs:sequence>
        <xs:element name="Expression"
minOccurs="0"/>
        <xs:element name="Texte" minOccurs="0"/>
        <xs:element name="Figure" minOccurs="0"/>
        <xs:element name="ModeObtention"
minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

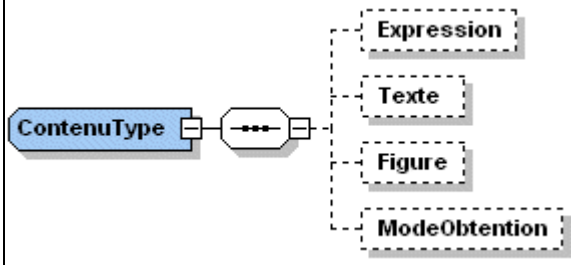


Figure 20 : Schéma XML et structure du type « ContenuType »

7.1.4. Schéma XML de l'interface abstraite

La Figure 21 présente les éléments qui composent la classe **InterfaceAbstraite**.

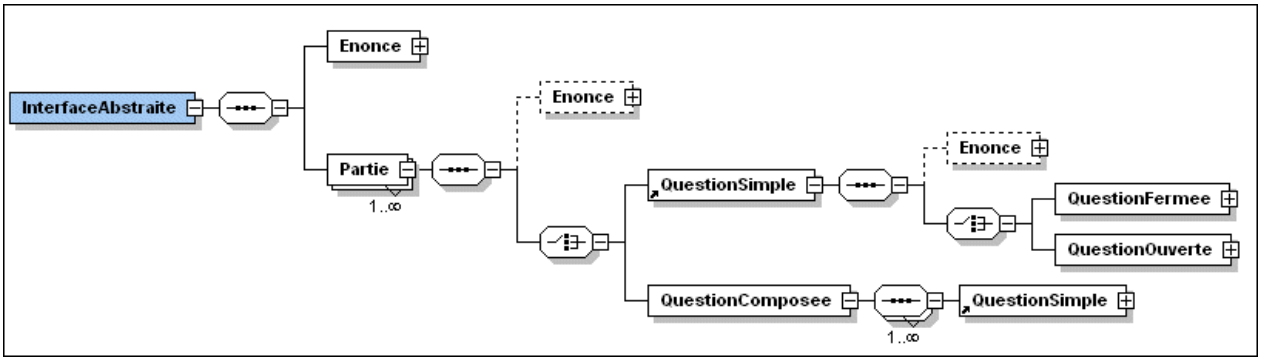


Figure 21 : Structure de la classe **InterfaceAbstraite**

La classe ***Question*** est une classe abstraite avec deux sous-classes **QuestionSimple** et **QuestionComposee**. Une question ayant une réponse anticipée (Chapitre 5, Figure 18 et 19), l'ajout d'un élément attribut « Identificateur » a pour objet d'établir la correspondance avec cette réponse. Une question fermée a deux sous-classes, **ChoixMultiple** et **ChoixExclusif**.

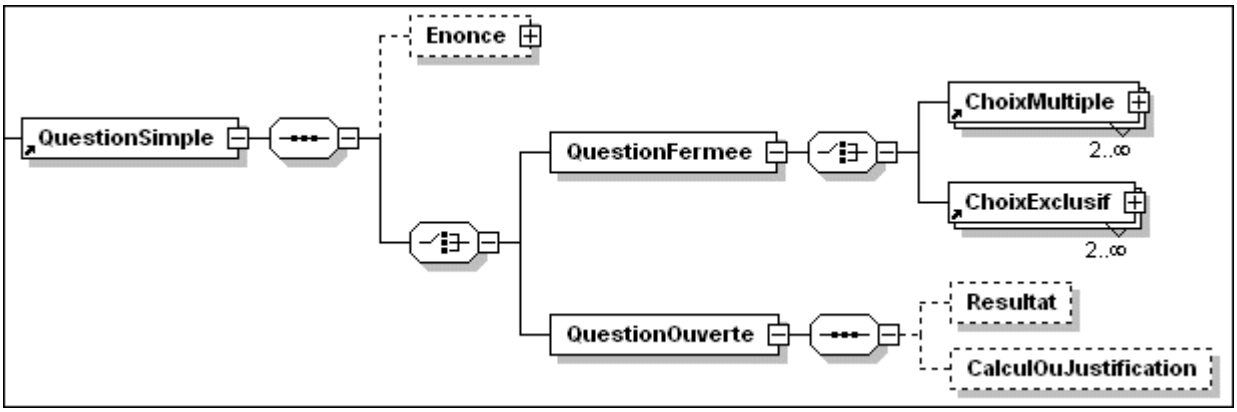


Figure 22 : Structure de la classe **QuestionSimple**

```
<InterfaceAbstraite>
    <Enonce>
        <ContenuSimple>
            <Texte>p1</Texte>
        </ContenuSimple>
    </Enonce>
    <Partie>
    <QuestionComposee>
        <QuestionSimple Identificateur="q1">
            <QuestionFermee>
            <ChoixExclusif Identificateur="c1">
                <ContenuSimple>
                    <Texte>Vrai</Texte>
                </ContenuSimple>
            </ChoixExclusif>
            <ChoixExclusif Identificateur="c2">
                <ContenuSimple>
                    <Texte>Faux</Texte>
                </ContenuSimple>
            </ChoixExclusif>
            </QuestionFermee>
        </QuestionSimple>
        <QuestionSimple Identificateur="q2">
            <QuestionOuverte>
                <CalculOuJustification  Identificateur="j1">
                </CalculOuJustification>
            </QuestionOuverte>
        </QuestionSimple>
    </QuestionComposee>
    </Partie>
</InterfaceAbstraite>
```

Tableau 5 : Interface Abstraite - extrait du fichier XML de la classe d'exercices « Preuve et programme de calcul »

Le Tableau 5 présente un extrait du fichier XML qui correspond à l'interface abstraite du modèle d'exercice « Preuve et programme de calcul » (Figure 23). Cette interface comporte une question composée avec deux questions simples. La première « q1 » est une question fermée avec deux choix exclusifs repérés par les identificateurs « c1 » et « c2 » et dont les contenus sont respectivement « Vrai » et « Faux ». La deuxième « q2 » est une question ouverte avec une justification « j1 ». Ces identificateurs sont utilisés pour associer une question à la réponse anticipée correspondante dans la grille de codage.

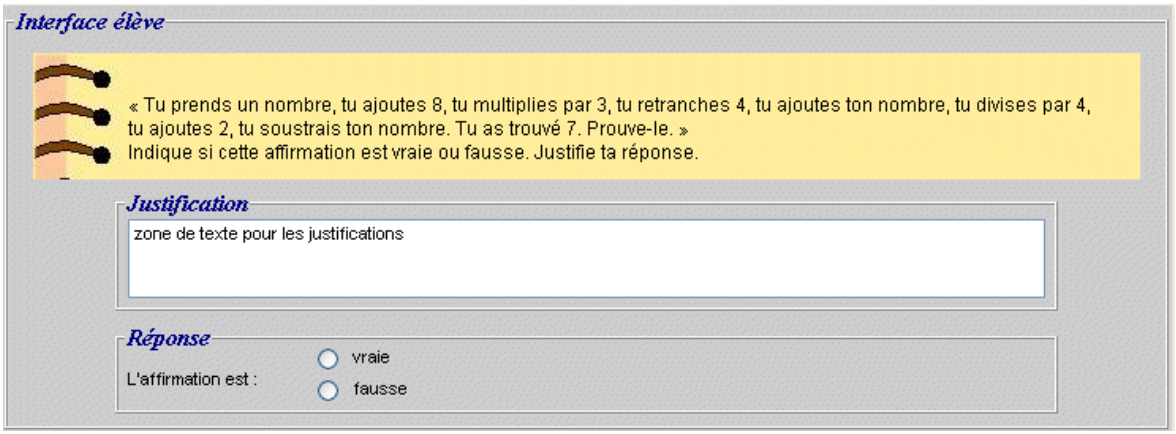


Figure 23 : Interface concrète de la classe d'exercices « Preuve et programme de calcul »

7.1.5. Schéma XML de la grille de codage

La Figure 25 présente les éléments qui composent la classe **GenerationGrilleCodage**. La classe **ReponseAnticipee** est une classe abstraite avec trois sous-classes **ReponseQuestionFermee**, **ReponseQuestionOuverte**, **ReponseQuestionComposee**.

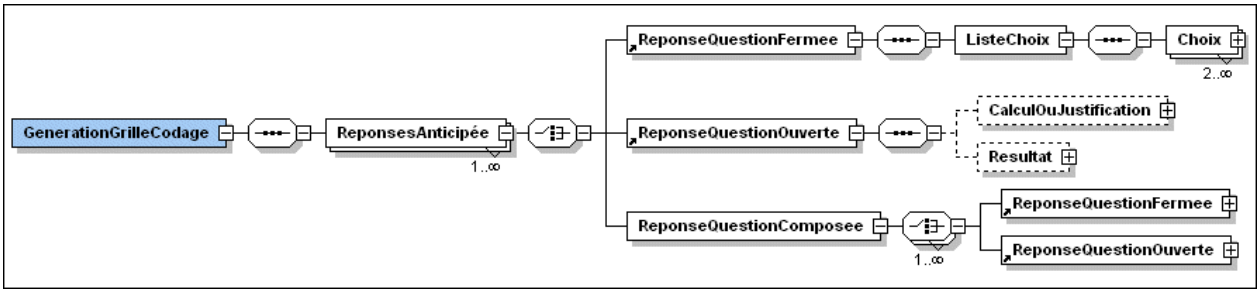


Figure 24 : Structure de la classe **GenerationGrilleCodage**

Les réponses aux questions fermées sont sous la forme de deux à plusieurs choix (**ListeChoix** et **Choix**). Les réponses aux questions ouvertes se présentent sous la forme de lignes de calculs (**CalculOuJustification**), d'une valeur numérique, une expression algébrique ou une formule de géométrie (**Resultat**).

7.1.5.1. Réponse à une question fermée

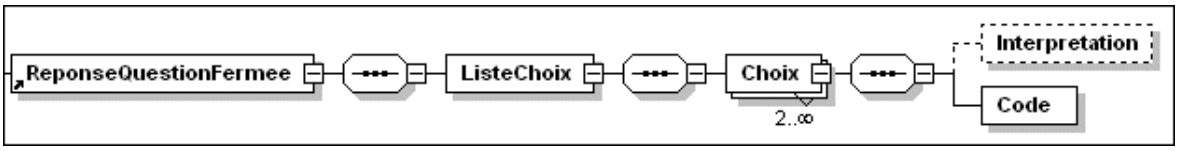


Figure 25 : Structure de la classe **ReponseQuestionFermee**

Reprenons l'exemple précédent pour montrer la correspondance entre les questions de l'interface abstraite et les réponses anticipées, puis la grille de codage obtenue. Le Tableau 6 présente un extrait du fichier XML qui présente la réponse correspondant à la question fermée de l'interface abstraite identifiée par « q1 ».

```
<GenerationGrilleCodage>
        <ReponsesAnticipée>
            <ReponseQuestionComposee>
                <ReponseQuestionFermee Identificateur="q1" >
                    <ListeChoix>
                        <Choix Identificateur="c1">
                            <Code>V1</Code>
                        </Choix>
                        <Choix Identificateur="c2">
                            <Code>V3</Code>
                        </Choix>
                    </ListeChoix>
                </ReponseQuestionFermee>
                            …/……
                </ReponsesAnticipée>
</GenerationGrilleCodage>
```

Tableau 6 : Génération de la grille de codage : extrait du fichier XML de la classe d'exercices « Preuve et programme de calcul »

Pour cette question le code est invariant : au choix « c1 » (Vrai) correspond le code V1 et au choix « c2 » (Faux) correspond le code V3.

7.1.5.2. Réponse à une question ouverte

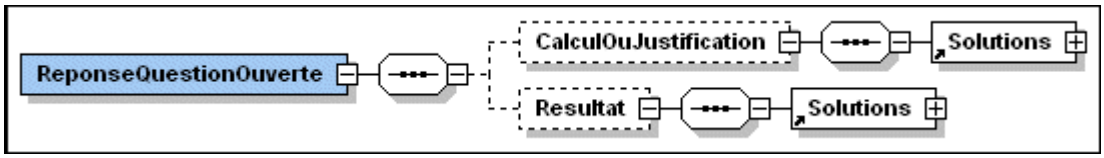


Figure 26 : Structure de la classe **RepponseQuestionOuverte**

Pour les réponses anticipées de type calcul, justification ou résultat, obtenues par l'application de règles de calcul correctes ou erronées, la génération de la grille de codage a déjà été présentée à la section 6. Nous avons donc retenu quatre types de solutions : les solutions correctes (**SolutionCorrecte**), les solutions incorrectes (**SolutionIncorrecte**) auxquelles s'ajoutent, pour

certaines classes d'exercices, des solutions correctes mais qui ne correspondent pas au degré de maîtrise attendu (**SolutionCorrecteNonAttendue**) à ce niveau scolaire (e.g. ne pas reconnaître la présence d'une identité remarquable en fin de troisième pour résoudre un exercice de la classe « Déterminer si des expressions algébriques du second degré sont égales », Cf. chapitre 5 - Tableau 15) et des solutions partielles (**SolutionPartielle**) où l'élève a engagé une démarche correcte mais une erreur de calcul ne lui permet pas d'aboutir au résultat.

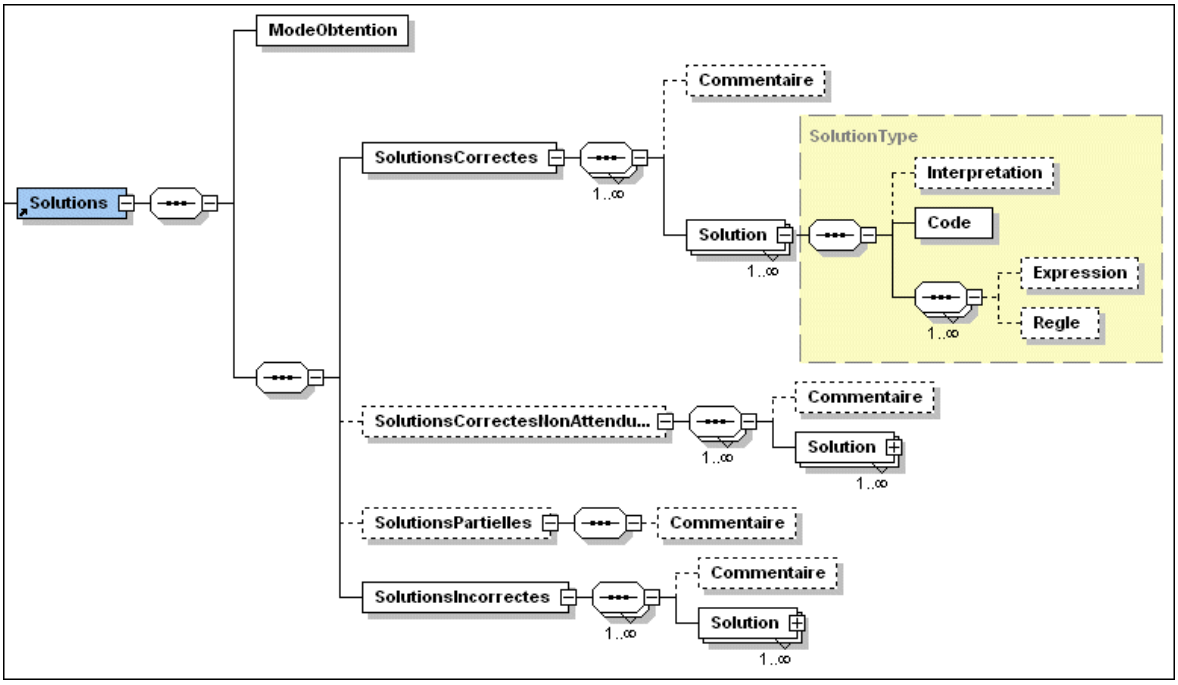


Figure 27 : Structure de la classe **Solutions**

La classe **Solutions** est une classe abstraite utilisée pour décrire les solutions de différents types. Nous avons créé un type **SolutionType** (Figure 28).

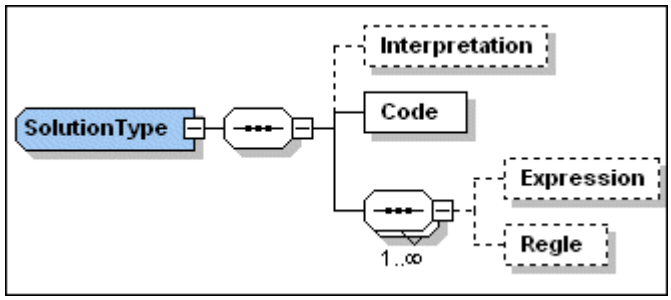


Figure 28 : Structure du type « SolutionType »

L'ensemble des structure sont présentées en annexe C.7

7.2. Structure du fichier XML « BanqueExercices »

Pour les clones d'exercices dont la génération est assistée, nous avons adopté la solution qui consiste à sauvegarder dans un même fichier les paramètres utilisés par l'interpréteur d'exercices, les réponses anticipées et la grille de codage nécessaires au programme de diagnostic des réponses. En ce qui concerne les clones dont la génération est automatique nous avons pris l'option de ne pas les sauvegarder dans la base d'exercices mais de les générer lorsque cela était nécessaire à partir du modèle de l'exercice (Annexes D).

La Figure 29 présente la structure du fichier **BanqueExercices**. Nous avons repris les éléments du schéma XML du fichier **ModelesClasseExercices**. Un clone d'exercice est caractérisé par un niveau scolaire, les valeurs des paramètres et le diagnostic local associé à ce clone. Le fichier **BaseExercices** est utilisé par l'interpréteur d'exercice (section 8) et par le module qui effectue le diagnostic local des réponses d'un élève à un test (section 9).

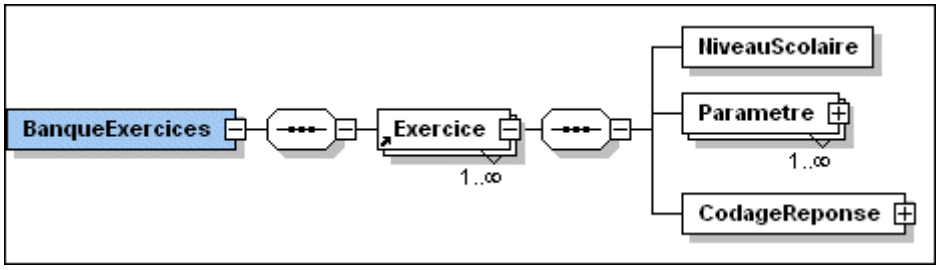


Figure 29 : Arborescence du schéma XML du fichier **BanqueExercices**

Les clones ont un ou plusieurs paramètres dont le contenu est du texte, une expression ou une figure. Un clone d'exercice (**Exercice**) à trois attributs pour pouvoir l'identifier : un nom, un numéro et un identificateur.

En ce qui concerne le codage des réponses, nous utilisons le schéma XML de **ReponseQuestionFermee** et **ReponsQuestionOuverte.** Les trois formes de réponses anticipées sont **CalculOuJustification**, **Resultat** et **ListeChoix**, et pour les deux premières formes les quatre types de solution de **ModelesClasseExercice.** .

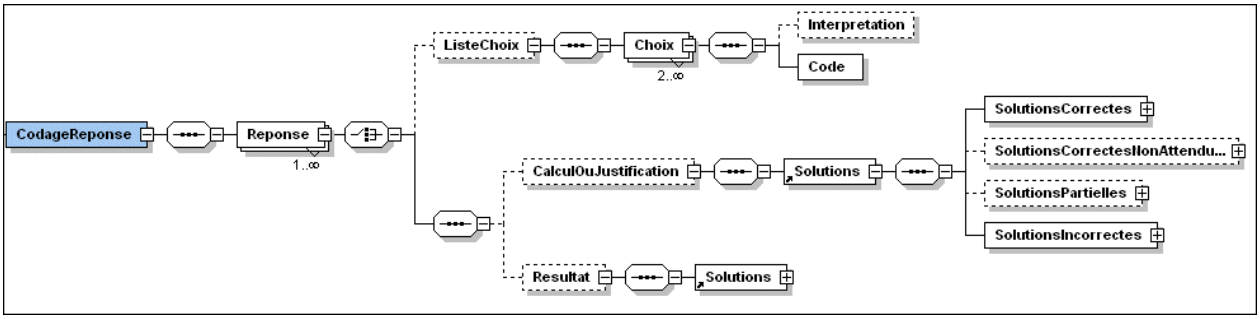


Figure 30 : Arborescence du schéma XML du fichier **CodageReponse**

La figure 34 présente un extrait de la grille de codage d'un clone d'exercice de la classe « Preuve et programme de calcul » dont le programme de calcul est : ((x*3 - 6)*2 - 3*x)/3 - x.

```
<Solutions>
  <SolutionsCorrectes>
    <Commentaire>Preuve algébrique avec une  expression globale (parenthésée)
      traduisant le résultat de l'  enchainement opératoire</Commentaire>
    <Solution>
      <Interprétation>Les fractions sont
        réduites au même dénominateur.</Interprétation>
      <Code>V1,EA2,L1,T1,J1</Code>
      <Expression>((x*3-6)*2-3*x)/3-x</Expression>
      <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
      <Regle>V,8</Regle>
      …/….
    </Solution>
    …/….
  </SolutionsCorrectes>
  <SolutionsCorrectesNonAttendues>
    ../….
     <Commentaire>Preuve algébrique avec  l'  interprétation de  l'  énoncé comme une équation
      admettant une infinité de solutions.</Commentaire>
    <Solution>
      <Interprétation>Les fractions sont
        réduites au même dénominateur.</Interprétation>
      <Code>V2,EA2,L1,T1,J1</Code>
      <Expression>((x*3-6)*2-3*x)/3-x = -4</Expression>
       ../…..
    </Solution>
    ../…
  </SolutionsCorrectesNonAttendues>
  <SolutionsPartielles>
    <Commentaire>Une preuve algébrique est  engagée avec l'  une des trois
      interprétations précédentes. Mais une  erreur de calcul opératoire conduit à un
      résultat faux ou à un abandon </Commentaire>
  </SolutionsPartielles>
  <SolutionsIncorrectes>
    <Commentaire>Les solutions utilisent une
      démarche algébrique.</Commentaire>
    ../….
  </SolutionsIncorrectes>
</Solutions>
```

Tableau 7 : Extrait du fichier XML BaseExercices (solutions)

Nous retrouvons les quatre types de solutions décrites à la section 7.2. L'ensemble du fichier se trouve en annexe C.11.

8. Interpréteur d'exercice : PépiTest

Le scénario 5 présenté dans le chapitre 4, « Des élèves passent un test diagnostic », fait apparaître les principales fonctionnalités de cet interpréteur. Ce scénario met en jeu un élève qui passe un test diagnostic préparé par un enseignant. C'est au cours de l'action « Administrer un test » (section 2.2) réalisée par un enseignant que les instances des exercices qui composent le test sont recopiées de la base de données d'exercices, **BanqueExercices**, dans un fichier pour une utilisation ultérieure par un élève ou par l'ensemble des élèves d'une classe.

8.1. Principales fonctionnalités

Un élève utilise l'interpréteur d'exercices pour réaliser un test. Ce programme affiche une interface instanciée avec les valeurs des paramètres des exercices enregistrés dans le fichier XML contenant les données du test. L'élève a la possibilité de réaliser les exercices dans l'ordre ou dans le désordre. L'interpréteur affiche les interfaces graphiques correspondant aux différents exercices du test. La Figure 32 présente l'interface élève de l'interpréteur. Les réponses de l'élève sont enregistrées dans un fichier de sorte que l'élève peut modifier ou rejouer le test ultérieurement. Ainsi, l'interpréteur sollicite des échanges de données fréquents avec le support de stockage contenant les données du test et les réponses de l'élève.

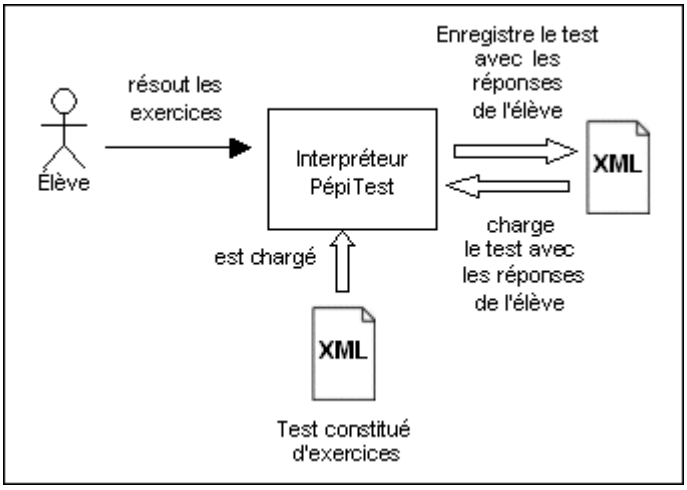


Figure 31 : Architecture fonctionnelle de PépiTest

8.2. Exemple

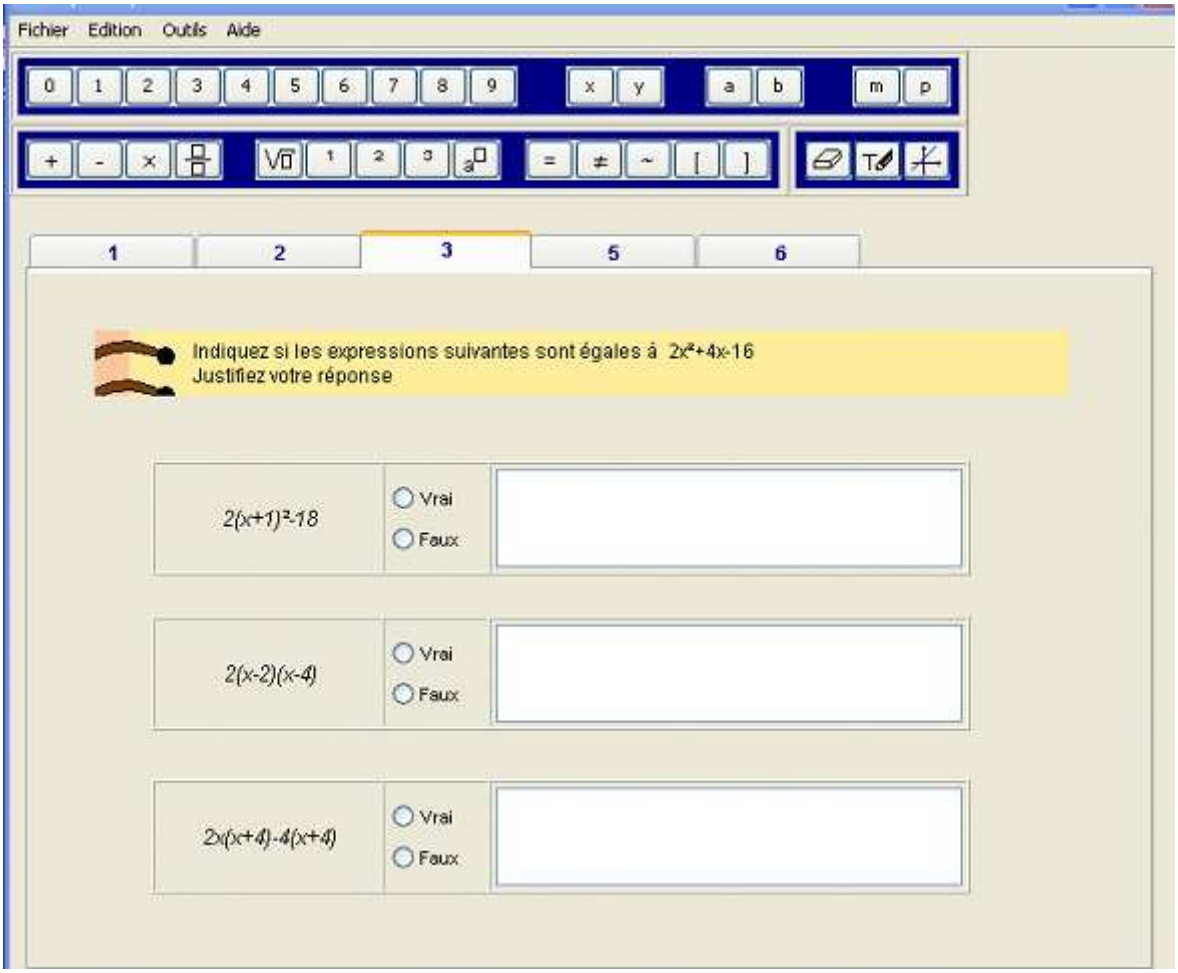


Figure 32 : Interface élève de SuperPépiTest

Le Tableau 8 donne un extrait des données utilisées par l'interpréteur d'exercices pour afficher un test de plusieurs exercices. Nous retrouvons les éléments qui caractérisent le clone d'exercices, le niveau scolaire, les paramètres et le codage des réponses.

L'interface élève du premier exercice décrit dans ce fichier est celle de la Figure 32. Pour cet exercice le codage des réponses correspond aux deux choix Vrai/Faux. Le codage complet des réponses de l'exercice dont l'identificateur est « 18-2-2008-4-53 » se trouve en annexe C.11.2.

```
<exercices version="date">
   <Exercice Identificateur="18-2-2008-4-50"
      Nom="Preuve et programme de calcul" Numero="16">
      <Niveau>4 ième - 3 ième</Niveau>
      <Parametre>
         <Expression>2x²+4x-16</Expression>
      </Parametre>
     <Parametre>
         <Expression>2(x+1)²-18</Expression>
      </Parametre>
      <Parametre>
         <Expression>2(x-2)(x-4)</Expression>
      </Parametre>
     <Parametre>
         <Expression>2x(x+4)-4(x+4)</Expression>
      </Parametre>
      <CodageReponse>
         <Reponse Identificateur="r1">
            <ListeChoix>
               <Choix Identificateur="c1">
                  <Code>V1</Code>
               </Choix>
               <Choix Identificateur="c2">
                  <Code>V2</Code>
               </Choix>
            </ListeChoix>
                  …. /…...
         </Reponse>
      </CodageReponse>
   </Exercice>
   <Exercice Identificateur="18-2-2008-4-53"
      Nom="Preuve et programme de calcul" Numero="16">
      <Niveau>4 ième - 3 ième</Niveau>
      <Parametre>
         <Texte>Tu prends un nombre, tu multiplies ce nombre par 3,  tu soustrais 6, tu multiplies le résultat par 2, tu
            soustrais le triple de ton nombre, tu divises le  résultat par 3, tu soustrais le nombre de départ.</Texte>
      </Parametre>
      <Parametre>
         <Expression>((x * 3 - 6)* 2 - 3 * x)/ 3 - x</Expression>
      </Parametre>
               …/…
    </Exercice>
</exercices>
```

Tableau 8 : Extrait d'un fichier XML contenant un test

9. Diagnostiqueur : PepiDiag

Le scénario 5 présenté dans le chapitre 4 « Des élèves passent un test diagnostic » fait apparaître les principales fonctionnalités du diagnostiqueur PépiDiag. Ce scénario met en jeu un enseignant qui réalise avec PépiDiag le diagnostic des réponses d'un élève qui a passé un test.

9.1. Principales fonctionnalités

Le diagnostiqueur PépiDiag réalise pour chaque exercice du test le diagnostic local des réponses d'un élève aux exercices de ce test. PépiDiag compare les réponses de l'élève aux réponses anticipées à ces exercices qui sont mémorisées dans la grille de codage. Nous avons vu dans la section 7.2 qu'il y avait trois formes de réponses anticipées : les réponses aux questions ouvertes sous la forme de lignes de calculs (**CalculOuJustification**), sous la forme d'un résultat (**Resultat**), les réponses aux questions fermées sous la forme d'un choix (**Choix**).

Un choix se traduit par une interaction de l'élève avec l'interface (e.g. clic, choix dans un menu déroulant) ; il est mémorisé par une valeur binaire (0 ou 1). La grille de codage est invariante (Chapitre 5) et à un choix est associé un code.

Pour les réponses aux questions ouvertes, la grille de codage comporte toutes les réponses correctes ou erronées anticipées et le code correspondant (section 6). Pour réaliser le diagnostic, PépiDiag compare les réponses de l'élève aux réponses anticipées de la grille de codage.

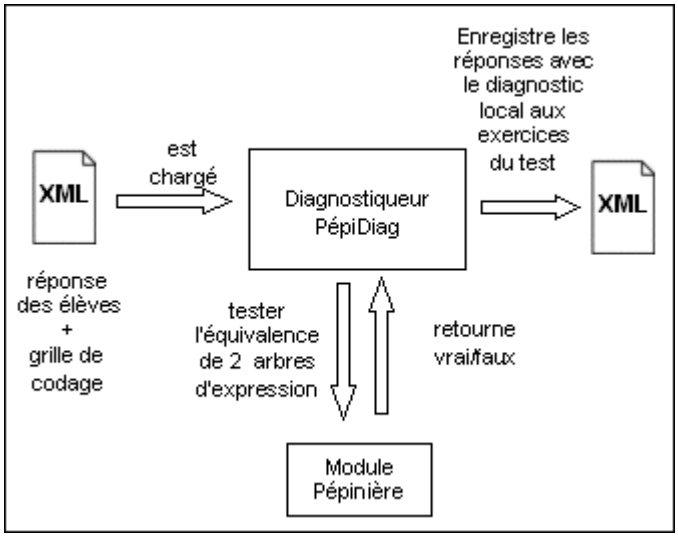


Figure 33 : Architecture fonctionnelle de PépiDiag

Plus particulièrement, pour les réponses comportant des expressions algébriques, PépiDiag utilise le module de calcul formel Pépinière (Chapitre 6). Pépinière construit les deux arbres d'expression correspondant, d'une part, à la réponse de l'élève et, d'autre part, à la réponse

anticipée et vérifie si ces deux arbres sont équivalents (Chapitre 6, section 6). Cette étape du diagnostic permet d'obtenir le codage des dimensions de l'algèbre concernant l'utilisation des règles d'écriture et de réécriture algébrique ou numérique pour toutes les réponses comportant des expressions algébriques.

Le codage de certaines dimensions (type d'utilisation des lettres, type de traduction d'une représentation dans une autre (langage naturel, algébrique, géométrique, graphique), type de justification) est spécifique à une classe d'exercice. Ce codage est généré par PépiGen sous la forme d'une référence à une méthode particulière de la classe d'exercices. Le parcours de l'ensemble des réponses anticipées générées par PépiGen en vue d'une comparaison avec les réponses de l'élève diffère aussi selon la classe d'exercices.

Dans la section suivante nous présentons l'algorithme de diagnostic spécifique aux exercices de la classe « Preuve et programme de calcul ».

9.2. Algorithme de diagnostic de la classe d'exercices « Preuve et programme de calcul »

La Tableau 9 donne les principales étapes de cet algorithme.

Si l'élève a répondu en partie ou complètement en utilisant le langage naturel, la première étape du diagnostic consiste à extraire si cela est possible les expressions algébriques. Une deuxième étape consiste à vérifier si les réponses de l'élève utilisent seulement des valeurs numériques (preuve par l'exemple).

Ensuite nous parcourons parallèlement la suite d'expressions algébriques constituant la réponse de l'élève et la grille de codage en commençant par les solutions correctes, puis pour chaque type de solution (correctes, correctes non optimales, incorrectes) en parcourant les solutions particulières jusqu'à ce qu'une solution corresponde à la réponse de l'élève ou non. Ainsi nous pouvons mémoriser au fur et à mesure du parcours effectué, les commentaires, les interprétations, les règles utilisées et la partie de la réponse de l'élève qui a pu être analysée.

A partir d'un corpus important de réponses obtenu dans les différentes expérimentations faites en classe (en France et à l'étranger) avec le premier prototype PépiTest, nous avons pu affiner nos heuristiques pour faire apparaître les principaux traits de fonctionnement qui permettent de dresser un profil cognitif de l'élève en algèbre élémentaire.

La section 9.3 présente un ensemble de résultats obtenus par application de cet algorithme et des raffinements successifs que nous lui avons apporté pour aboutir à des diagnostics significatifs et pertinents sur le plan didactique.

Si l'expression est numérique
Alors démarche = numérique
Sinon démarche = algébrique
Tant que la fin de la suite des expressions algébriques constituant la réponse de l'élève n'est pas atteinte
Faire
Pour chaque expression algébrique contenue dans la réponse de l'élève
Faire
Construire l'arbre d'expression A1
Pour chaque type de solution (correcte, correcte non attendu, incorrecte) et chaque solution anticipée de ce type de solution
Tant que la fin de la liste des réponses anticipées n'est pas atteinte
Pour chaque expression algébrique contenue dans la réponse anticipée
Construire l'arbre d'expression A2
- Si démarche = numérique remplacer la variable de l'arbre A2 par la valeur numérique A2
- Comparer A1 et A2
- Si A1 et A2 sont équivalents
Alors
- Mémoriser l'expression algébrique de l'élève reconnue
- Mémoriser éventuellement la règle correcte ou erronées utilisée pour passer d'une expression à une au
- Mémoriser éventuellement le commentaire correspondant
Sinon
Comparer A1 et l'arbre d'expression A3 obtenue avec la réponse anticipée suivante
Sinon avancer dans le type de solution anticipées
Fin Tant que
Fin Faire
Fin Faire
Fin Tant que
Si la fin de la suite des expressions algébriques constituant la réponse de l'élève est reconnue
Alors mémoriser le code correspondant
Sinon prendre en compte les éléments mémorisé pour attribuer un code
Sinon si aucun élément mémorisé diagnostic non réalisé

Tableau 9 : Algorithme du diagnostic pour la classe « Preuve et programme de calcul »

9.3. Tests

Nous avons testé PépiDiag sur un corpus regroupant les réponses de 83 élèves pour la résolution de l'exercice dit du « Prestidigitateur » (Tableau 10) au cours des expérimentations

menées avec le premier logiciel PépiTest [Jean 2000]. Cet exercice appartient à la classe d'exercices « Preuve et programme de calcul ».

Tu prends un nombre, tu ajoutes 8, tu multiplies par 3, tu retranches 4, tu ajoutes ton nombre, tu divises par 4, tu ajoutes 2, tu soustrais ton nombre : tu trouves 7. Prouve-le Expression algébrique : $((x+8)3-4+x)/4+2-x$

Tableau 10 : Programme de calcul de l'exercice dit du « Prestidigitateur »

Pour cela, nous avons d'abord généré avec PépiGen la grille de codage de cet exercice dans un fichier XML, puis, nous avons fait appel à PépiDiag pour faire le diagnostic local à cet exercice sur chacune des réponses de ce corpus. La Figure 34 montre l'interface que nous avons développée pour tester notre algorithme pour la classe « preuve et programme de calcul ».

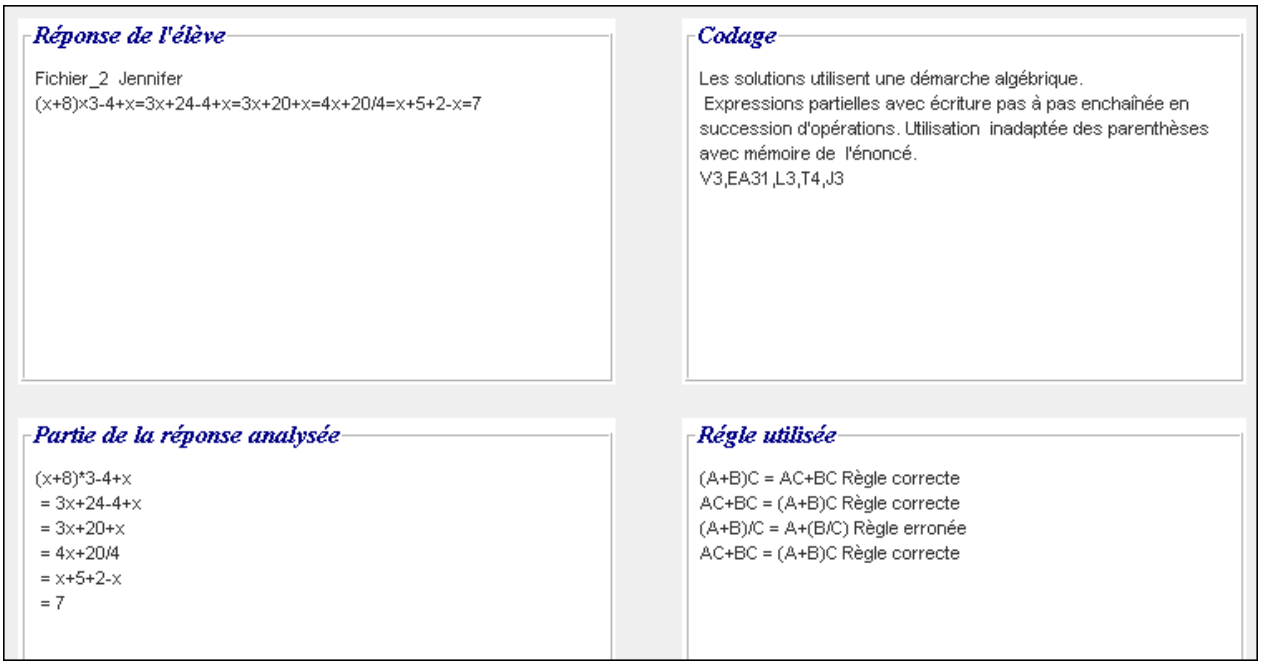


Figure 34 : Interface de test

Sur les 83 réponses, environ un quart des réponses n'ont pas été codées car les élèves n'ont pas donné de justification. Cinq réponses n'ont pas pu être codées car elle comportaient des symboles ou des notations imprévues (Tableau 11)

Fichier_7 christophe [E16] 01 3(x+8)-4-x []/[] 4+2-x <> 7 x+22 <> 4×7 x <> 28-22 x <> 6	Fichier_26 alexandre [E16] 00 {[(x + 8) × 3- 4+ x] {---------- + 2} - x = 7 { 4

Tableau 11 : Exemples de symboles ou notations imprévues

Nous codons toutes les réponses correctes. Pour les autres réponses nous repérons les deux types de démarche : la démarche algébrique et la démarche numérique. En ce qui concerne la démarche algébrique nous repérons trois types de solutions : les solutions correctes, les solutions correctes non optimales et les solutions incorrectes. Nous repérons les deux types d'utilisations inadaptées des parenthèses qui peuvent apparaître dans l'expressions algébrique de cet exercice (Tableau 12).

Laurent [(x+8)×3-4+x]/4+2-x ***(3x+24-4+x)/4+2-x*** ***4x+20/4+2-x*** x+5+2-x Expression diagnostiquée ((x+8)*3-4+x)/4+2-x (3x+24-4+x)/4+2-x 4x+20/4+2-x x+5+2-x Règles de réécriture utilisées (A+B)C = AC+BC Règle correcte AC+BC = (A+B)C Règle correcte ***(A+B)/C = A+(B/C) Règle erronée*** AC+BC = (A+B)C Règle correcte Les solutions utilisent une démarche algébrique avec une expression globale parenthésée. L'expression est simplifiée. ***Utilisation inadaptée des parenthèses avec mémoire de l'énoncé.*** Code V3,EA31,L3,T3,J3	Florence soit x le nombre choisi ***(x+8 × 3-4+x) []/[] 4+2-x=7*** ***(3x+24-4+x)*** []/[] 4+2-x=7 (4x+20) []/[] 4+2-x=7 x+5+2-x=7 7=7 Expression diagnostiquée (x+8 * 3-4+x) ()/() 4+2-x = 7 (3x+24-4+x) ()/() 4+2-x = 7 (4x+20) ()/() 4+2-x = 7 x+5+2-x = 7 7 = 7 Règles de réécriture utilisées ***(A+B)C = A+BC Règle erronée*** AC+BC = (A+B)C Règle correcte (A+B)/C = A/C+B/C Règle correcte AC+BC = (A+B)C Règle correcte Les solutions utilisent une démarche algébrique. Interprétation de l'énoncé comme une équation admettant une infinité de solutions. L'expression est simplifiée. ***Utilisation inadaptée des parenthèses avec mémoire de l'énoncé.*** Code V3,EA31,L3,T3,J3

Tableau 12 : Utilisation inadaptée des parenthèses pour les expressions (x+8)*3 et (4x+20)/4

Nous décelons les démarches algébriques avec utilisation d'une expression globale parenthésée ou non (Tableau 13). Pour deux démarches identiques nous mettons en évidence l'utilisation d'une règle incorrecte dans l'une des deux réponses (Tableau 14). Enfin pour les démarches numériques nous retrouvons les principaux types de réponses : utilisation d'une expression globale parenthésée (Tableau 15), d'expressions partielles traduisant les résultats intermédiaires (Tableau 15). Nous ne codons pas toutes les étapes des réponses des élèves mais

nous repérons des cohérences de fonctionnement relatives aux différents aspects de la compétence algébrique. Les résultats complets de ce test se trouvent en annexe C.12.

VIRGINIE (((x+8) × 3)-4+x)/4+2-x Expression diagnostiquée (((x+8) * 3)-4+x)/4+2-x Les solutions utilisent une démarche algébrique avec une ***expression globale parenthésée***. Code V3,L3,T3,J3	Fahriye (x+8)×4-4+x:4+2-x = Expression diagnostiquée (x+8)×4-4+x/4+2-x = L'interprétation de l'énoncé conduit à une traduction algébrique de l'enchaînement opératoire avec ***une expression globale non parenthésée:*** Code V3,L3,T3,J3

Tableau 13 : Deux types de traduction algébrique : globale parenthésée ou non

stephanie soit x le nombre [x+8] × 3 = 3x+24-4 = 3x+20 = 4x+20 = [4x+20]/4 = x+5 = x+5+2 = x+7 = x+7-x = 7 Expression diagnostiquée (x+8) * 3 = 3x+24-4 = 3x+20 = 4x+20 ***= (4x+20)/4*** ***= x+5*** = x+5+2 = x+7 = x+7-x = 7 Règles de réécriture utilisées (A+B)C = AC+BC Règle correcte ***(A+B)/C = A/C+B/C Règle correcte*** AC+BC = (A+B)C Règle correcte Les solutions utilisent une démarche algébrique. ***Expressions partielles avec écriture pas à pas enchaînée en succession d'opérations.*** Code V3,L3,T4,J3	Jennifer (x+8)×3-4+x=3x+24- 4+x=3x+20+x=4x+20/4=x+5+2-x=7 Expression diagnostiquée (x+8)*3-4+x = 3x+24-4+x ***= 3x+20+x*** ***= 4x+20/4*** ***= x+5+2-x*** = 7 Règles de réécriture utilisées (A+B)C = AC+BC Règle correcte AC+BC = (A+B)C Règle correcte ***(A+B)/C = A+(B/C) Règle erronée*** AC+BC = (A+B)C Règle correcte Les solutions utilisent une démarche algébrique. ***Expressions partielles avec écriture pas à pas enchaînée en succession d'opérations.*** ***Utilisation inadaptée des parenthèses avec mémoire de l'énoncé.*** Code V3,EA31,L3,T4,J3

Tableau 14 : Expression partielle avec utilisation de règles erronées ou non

Khemarak Soit 5 un nombre ((5+8)×3-4+5)/4+2-5=7 ? ((13)×3-4+5)/4+2-5=7 ? (39-4+5)/4+2-5=7 ? 10+2-5=7 ? 10-3=7 ? 7=7 ? Oui donc cela marche Expression diagnostiquée ((5+8)*3-4+5)/4+2-5 = 7 ((13)*3-4+5)/4+2-5 = 7 (39-4+5)/4+2-5 = 7 10+2-5 = 7 10-3 = 7 7 = 7 Les solutions correspondent à une démarche non algébrique preuve par des exemples. Les lettres ne sont pas disponibles. Expression globale parenthésée. Interprétation de l'énoncé comme une équation admettant une infinité de solutions. Code V3,L5,J2,T1	Laurent Je calcule par étapes,pour x=2: 2+8=10;10×3=30;30-4=26;26+2=28;28/4=7. Ce premier exemple parait justifier cette affirmation: Pour x=3 3+8=11;11×3=33;33-4=29;29+3=32;32/4=8;8-3=5 Si mes calcules sont justes,ce prestigitateur est un charlatan. Expression diagnostiquée 2+8 = 10 10 * 3 = 30 30 - 4 = 26 26+2 = 28 28/4 = 7 Les solutions correspondent à une démarche non algébrique preuve par des exemples. Les lettres ne sont pas disponibles. Expressions partielles traduisant pas à pas les résultats intermédiaires de l'enchainement opératoire.

Tableau 15 : Démarche numérique avec utilisation d’expressions globales ou partielles

10. Conclusion

Dans ce chapitre nous avons décrit le système PépiGen que nous avons mis au point pour générer des banques d’exercices de diagnostic. L’objectif de ce système est de permettre, à partir d’une base de données des modèles de classes d’exercices de diagnostic, d’obtenir des clones de ces exercices pour lesquels les réponses des élèves sont analysées automatiquement y compris les réponses ouvertes exprimant des raisonnements algébriques complexes. Ce système est l’objectif du travail que nous nous étions fixé dans le cadre du projet Lingot. En outre, sa réalisation permet de mettre à l’épreuve le modèle conceptuel que nous avons mis au point et de montrer la faisabilité de l’approche que nous proposons : fonder le diagnostic cognitif sur la génération automatique des réponses plausibles à un exercice en généralisant et en opérationalisant une analyse didactique a priori. Ce qui fait l’originalité de notre travail en EIAH, c’est que nous mettons en œuvre cette approche sur des raisonnements complexes et non sur des réponses préformatées ou sur une seule étape de calcul.

Concernant la génération des questions, nous avons présenté un logiciel auteur qui permet à un auteur de créer des énoncés à partir de classes paramétrées d'exercices. Les valeurs des paramètres qui permettent de passer d'un clone à un autre sont, soit générées automatiquement, soit saisies par un auteur humain. Nous avons donné un exemple d'interface de saisie des paramètres, interface qui a été testée par cinq enseignants de mathématiques qui ont créé sans problème une cinquantaine d'exercices.

Concernant le diagnostic automatique, nous avons montré tout d'abord comment, en nous fondant sur une analyse didactique a priori et sur le logiciel de calcul formel présenté au chapitre 6, il était possible de générer une grille d'analyse des réponses des élèves, c'est-à-dire, un fichier qui contient, pour chaque clone généré, les types de solutions plausibles anticipées par l'analyse a priori et leur codage. Nous avons ensuite montré qu'il était possible de définir une fonction qui, à partir de ce fichier et de la réponse de l'élève, établit le diagnostic sur les réponses des élèves à un clone. Nous avons testé notre proposition en la mettant en œuvre complètement sur une classe d'exercices complexe à analyser et importante pour établir le profil cognitif des élèves en algèbre élémentaire. Nous avons testé ce diagnostic sur une partie du corpus de réponses d'élèves recueilli avec le logiciel Pépite mis au point par Stéphanie Jean-Daubias. Pour les autres classes d'exercices plus simples, nous avons étudié des prototypes qui ne sont pas à l'heure actuelle intégré au système.

Ce chapitre montre la faisabilité de notre approche mais aussi la difficulté de mise en œuvre de notre proposition. Cependant, nous estimons qu'à partir des composants que nous avons d'ors et déjà développés, mettre en œuvre les autres classes d'exercices du test originel ne présente plus autant de difficultés.

Après avoir présenté les trois apports principaux de notre travail, il est temps de conclure en revenant sur nos questions de recherche et en réfléchissant sur les moyens d'évaluer nos propositions.

Chapitre 8

Conclusion

1. Introduction 259
2. Vers un métamodèle d'évaluation multicritère automatique 260
3. Évaluation multicritère des raisonnements algébriques 262
4. Conception d'une chaîne logicielle pour le diagnostic cognitif 264
5. Conclusion 267

1. Introduction

L'objectif du projet Lingot est de concevoir des logiciels supportant la prise en compte de la diversité cognitive des élèves dans la régulation des apprentissages. Dans ce cadre, l'objectif de cette thèse était d'étudier comment faciliter le développement d'exercices par des auteurs non informaticiens (enseignants et didacticiens), de façon à disposer de banques d'exercices diagnostics. Notre thèse consiste à proposer de partir de tests diagnostics validés pour définir des modèles paramétrés permettant à un logiciel de cloner les exercices de ces tests. Le clonage consiste à générer des questions équivalentes du point de vue du diagnostic, mais aussi à générer l'analyse automatique des réponses à ces questions, que ces réponses soit préformatées ou libres. Dans ce mémoire, nous avons présenté les modèles et les prototypes que nous avons conçus dans le domaine du diagnostic de compétence en algèbre élémentaire, pour étayer notre proposition. Dans ce chapitre de conclusion, nous discutons la portée de notre travail en reprenant nos questions de recherche.

La première question de recherche concerne la caractérisation d'exercices équivalents du point de vue du diagnostic permettant de générer des clones des exercices du premier prototype Pépite. Nous avons fait l'hypothèse qu'il était possible de modéliser des classes d'exercices en mettant en évidence des invariants et des paramètres dans les tâches diagnostiques de Pépite de façon à définir des clones en instanciant ces paramètres. Dans cette optique, nous avons défini des modèles pour décrire ces exercices et un métamodèle pour spécifier ces différents modèles.

La deuxième question de recherche concerne l'automatisation de l'évaluation multicritère de réponses ouvertes. Nous avons fait l'hypothèse qu'il était possible de trouver une solution de compromis entre une approche générique difficile à utiliser (voir jamais utilisée) par les enseignants, et une solution spécifique dont le champ d'application est limité mais utilisable [Murray 2003]. Dans le domaine de l'algèbre élémentaire, nous avons développé un logiciel de calcul formel spécifique, Pépinière, qui permet (i) de générer des énoncés d'exercices comportant des expressions algébriques, (ii) de générer les solutions anticipées à ce type d'exercices en fonction d'une analyse didactique a priori et (iii) de générer l'analyse automatique des solutions comportant des raisonnements algébriques.

La troisième question de recherche concerne le degré d'intervention des auteurs. Nous avons fait l'hypothèse de ne laisser à leur charge que ce qui relève directement de leur expertise, à savoir la génération des énoncés des exercices à partir de formulaires à compléter. Dans PépiGen, la génération des questions est, soit complètement automatique lorsqu'il s'agit de

donner des valeurs numériques ou prédéfinies à des paramètres, soit partiellement automatique quand l'énoncé requiert une certaine créativité. Toutes les autres tâches sont à la charge de la machine, à savoir,,la génération de l'interface et l'analyse des réponses qui sont automatiques. Pour mettre à l'épreuve cette hypothèse, nous avons conçu une chaîne logicielle qui permet à un auteur de créer des exercices, à un élève d'être testé sur ces exercices et à un enseignant d'obtenir un diagnostic sur les réponses de l'élève à cet exercice.

Nos hypothèses relèvent principalement de trois domaines différents de l'informatique : la première concerne la modélisation, la deuxième le génie logiciel et, la troisième, plus spécifiquement l'interaction humains-machines (IHM). Dans les sections qui suivent, nous résumons et discutons les résultats de notre travail de thèse qui a consisté à mettre à l'épreuve ces hypothèses. En nous appuyant sur les méthodes d'évaluation dans ces trois domaines informatiques, nous nous interrogeons sur le domaine de validité de nos résultats avant de conclure notre travail.

2. Vers un métamodèle d'évaluation multicritère automatique

Notre premier résultat est la mise au point de modèles pour caractériser chacun des exercices du premier prototype Pépite. Ces modèles sont génériques dans le sens où ils servent à générer des clones des exercices diagnostics originels. À partir de ces modèles, nous avons proposé un métamodèle pour spécifier et valider chacun de ces modèles.

Chacun de ces modèles a été mis au point en s'appuyant sur une étude didactique et cognitive rigoureuse validée sur les plans académique, institutionnel et expérimental, menée par les didacticiens de notre équipe. Nous avons montré sur plusieurs prototypes[1] que ces modèles descriptifs permettaient de créer des modèles exécutables. Ceci nous assure de la pertinence de ces modèles par rapport aux objectifs internes au projet Lingot.

Nous estimons que ce métamodèle est une première étape vers un métamodèle réutilisable pour spécifier une évaluation multicritère automatique des réponses à des exercices dans des formations en ligne. Un tel objectif est à moyen terme et à discuter dans une large communauté.

[Joosten-ten et al. 2007] proposent neuf critères adaptés de [Koper 2001] pour évaluer, sur le long terme, un modèle réutilisable d'évaluation pour l'enseignement en ligne. Nous nous sommes attachées à définir un métamodèle qui spécifie nos modèles et avons montré qu'ils étaient interprétables par des programmes ce qui satisfait le critère *formalisation*. Pour étayer

notre hypothèse sur la *flexibilité* de notre modèle, nous avons essayé de décrire des exercices en provenance d'autres systèmes reposant sur d'autres types d'analyse. En effet, les éléments de notre modèle sont génériques (e.g. critères, dimensions, objectifs d'évaluation) et les valeurs qui sont spécifiques à l'analyse cognitive qui sous-tend le diagnostic pourraient être définies à partir d'ontologies. Ainsi, il nous semble possible, en nous fondant sur la description des exercices de DIANE [Hakem et al. 2005], de les modéliser en utilisant le métamodèle car il s'agit aussi d'un logiciel de diagnostic cognitif qui caractérise des réponses ouvertes sur plusieurs dimensions. Les autres exemples que nous avons étudiés dans le chapitre 3 sont avant tout des systèmes d'apprentissage. La plupart d'entre eux associent une rétroaction au diagnostic local des réponses de l'élève. C'est un élément absent de notre modèle qu'il faudrait prendre en compte. Nous envisageons deux façons de spécifier les interactions : soit en attachant des rétroactions à chaque question ou à chaque exercice (comme le propose QTI par exemple), soit en spécifiant des gestionnaires de rétroactions en fonction du codage des réponses à l'instar des gestionnaires d'erreurs de Andes par exemple. Même s'il est parfois difficile dans les descriptions données par les auteurs de trouver les informations pour compléter les modèles de données, un premier travail exploratoire nous a montré que notre hypothèse était pertinente. Le critère de *complétude* n'est pas encore vérifié : ce dernier concerne la couverture de l'ensemble du processus d'évaluation, depuis la construction des tests jusqu'à son exploitation. Une des perspectives de notre travail dans le projet Pépite est de tester notre métamodèle sur un logiciel de conception de tests diagnostics à partir des banques d'exercices que nous proposons de développer. Cette perspective rejoint les travaux sur l'indexation de ressources pédagogiques. Pour aller vers un objectif de *réutilisabilité* nous avons réifié les éléments d'expertise didactique (e.g. connaissances, capacités, composantes de la connaissance algébrique, dimensions et critères d'évaluation) et nous avons *explicitement typé les objets* manipulés (e.g. énoncés, questions, critères, codes). Nous utilisons les technologies XML pour faciliter *la maintenance et l'interopérabilité* des données. Le critère de *reproductibilité* est assuré par la mémorisation des exercices dans une banque et non par la génération à la volée ce qui permet de poser plusieurs fois la même évaluation si cela s'avère souhaitable. D'un point de vue recherche, nous assurons la reproductibilité de notre démarche en explicitant et en publiant (Annexes C) tous les modèles pour mettre en œuvre notre proposition. La *compatibilité* avec les normes et standards est à notre avis prématurée, ceux-ci n'étant pas clairement établis en la matière. Sur ce point, une perspective est un travail en commun avec les chercheurs travaillant sur LOM-FR pour exploiter

[1] Pour les classes représentant les exercices 1, 2, 3, 4, 5, 6, 8, 9, 11, 13, 16, 19

les éléments de l'indexation didactique que nous avons mise au point dans notre modèle. Le dernier critère concerne *l'indépendance par rapport au médium* d'administration du test (en ligne ou papier crayon). Ce dernier critère est « assez » vérifié par notre modèle qui s'inspire d'un outil papier crayon. Par contre il nous semble très discutable ; l'analyse automatique de réponses numérisées nous apparaît ouvrir des possibilités d'évaluation différentes du papier-crayon comme le prouvent les travaux de fouilles de données pédagogiques, les logiciels d'analyse de texte permettant de détecter le plagiat ou le logiciel Pépite qui détecte des points forts ou des faiblesses que l'enseignant n'avait pas détectées.

Une autre façon d'évaluer la pertinence de notre modèle, consiste à le comparer à d'autres modèles proposés par les chercheurs du domaine. Dans le chapitre 3, nous avons présenté les limites, de notre point de vue, du formalisme de la spécification QTI. Le formalisme que nous proposons complète le formalisme de QTI principalement pour une évaluation multicritère des réponses. L'équipe de Koper a proposé un modèle très général d'évaluation [Joosten-ten et al. 2007]. Le notre est moins large mais plus approfondi sur la partie diagnostic. Une autre différence de taille avec ces travaux, est que ces deux modèles sont « contemplatifs » [Favre et al. 2006], c'est-à-dire purement descriptifs, alors que le notre est « productif » puisque nous avons produit un modèle interprétable et manipulable par une machine comme l'a démontré le chapitre 7.

Pour aller vers un objectif d'interopérabilité, nous envisageons outre le travail sur l'indexation des ressources pédagogiques que nous avons déjà mentionné, de traduire notre modèle en anglais en spécialisant le modèle de Joosten-ten et al. pour le diagnostic cognitif. Ce travail, de mise au point d'un métamodèle est une des perspectives ouvertes par ce travail de thèse et devra être mené bien sûr avec d'autres.

3. Évaluation multicritère des raisonnements algébriques

Notre deuxième résultat consiste en la conception et la mise en œuvre d'un logiciel de calcul formel spécifique, Pépinière, sur lequel repose l'analyse des expressions algébriques et toute la spécificité du domaine sur lequel nous travaillons. En effet, ce logiciel est utilisé par PépiGen, tout d'abord pour construire automatiquement des énoncés, par exemple de type QCM : il demande à Pépinière de générer des expressions équivalentes (du point de vue mathématique ou du point de vue des élèves). PépiGen utilise également Pépinière pour générer la grille d'analyse des réponses à un exercice. En entrée, Pépinière reçoit un type de problème (limité, dans l'état actuel, à « prouver par développement et réduction ») et une chaîne de caractères représentant

une expression algébrique, et il retourne l'arbre des solutions anticipées par l'analyse didactique a priori pour ce problème. Il construit cet arbre en appliquant par paquets des règles de réécriture (correctes et erronées) et des heuristiques pour limiter l'expansion et éviter les bouclages. Ces heuristiques et ces règles ont été réifiées à partir de l'analyse didactique a priori dans des fichiers XML présentés en annexe. Enfin, Pépinière est appelé par le module de diagnostic pour associer la solution d'un élève à une des solutions anticipées et ainsi obtenir la caractérisation de cette solution sur plusieurs dimensions.

Pour la conception de ce logiciel, nous nous sommes appuyés sur les théories en vigueur dans le domaine (les grammaires, la théorie des règles de réécriture, l'algorithme d'unification). Nous avons, de plus, mené des tests pour vérifier la solidité de nos heuristiques pour limiter l'expansion de l'arbre des solutions anticipées. Des exemples de tests sont fournis en annexe. Nous avons également vérifié la validité de la grille d'analyse produite sur une trentaine d'exercices saisis par différents enseignants dont deux extérieurs au projet Lingot. De plus, le diagnostic automatique fondé sur Pépinière a codé avec succès (en comparaison avec le codage humain) les réponses de notre corpus.

Pépinière, n'est pas un logiciel interactif. C'est un programme au sens d'une machine de Turing. Il reçoit des données en entrées, applique des algorithmes qui produisent des résultats. Son évaluation consiste donc à vérifier les critères de qualité des programmes. Sa validité interne est assurée par les théories sur lesquelles nous avons fondé sa conception. Il est fiable pour la classe de problèmes que nous avons développée (développer, réduire) et robuste tant que les coefficients sont entiers. Il est a priori extensible en ajoutant d'autres règles de réécriture, mais l'étendre aux factorisations et aux résolutions d'équations demanderait sans doute un gros effort de développement comme en témoigne les travaux de l'équipe d'Aplusix. De même, notre système analyse certains raisonnements algébriques mais n'analyse pas les raisonnements exprimés en langage naturel. Un exception concerne les « programmes de calcul » que nous analysons car ils sont produits par du langage contraint auquel nous associons une grammaire. Ce logiciel spécifique est « générique » dans la mesure où il permet de générer les solutions anticipées. Il est réutilisable dans la mesure où nous avons réifié l'analyse didactique dans des fichiers strictement formatés par des schémas XML dans lesquels Pépinière prend les informations nécessaires à son fonctionnement. En ce qui concerne la réutilisabilité, nous estimons qu'il pourrait être utilisable par exemple dans le cadre du projet C3 (Chaîne de Création de Contenus éducatifs) [Auzende et al. 2007].

La conception de Pépinière est fondamentale pour l'évolution du projet Lingot. Elle permet de remplacer les traitements ad hoc du premier logiciel Pépite, par des traitements communs aux différents exercices qui permettent d'envisager le développement d'exercices pour d'autres niveaux scolaires ou pour mettre en place des parcours d'apprentissage. Pépinière manipule la connaissance du domaine, correcte ou erronée, nécessaire pour mener à bien le diagnostic automatique de réponses qui ne sont pas préformatées. C'est donc le module « Domaine » de l'architecture classique des tuteurs intelligents [Wenger 1987, Nicaud et Vivet 1988, Devedzic et Harrer 2005]. L'originalité de la conception de Pépinière est de fonder le développement de l'arbre des solutions plausibles sur une analyse didactique a priori qui distingue plusieurs stratégies dans les réponses d'élèves, chacune de ces stratégies pouvant conduire à l'utilisation de règles de réécriture correctes ou erronées. Il permet ainsi la caractérisation fine de démarches et pas simplement d'une étape comme le font les systèmes tutoriels que nous avons décrits.

4. Conception d'une chaîne logicielle pour le diagnostic cognitif

Notre troisième résultat concerne la conception d'une chaîne logicielle accessible à des enseignants pour générer des exercices de diagnostic en leur épargnant le travail fastidieux et délicat de la prévision des réponses possibles et de l'analyse détaillée des réponses effectives. Pour évaluer ce résultat nous regardons vers les méthodes d'évaluation des logiciels interactifs développées en IHM ([Bastien et Scapin 2001], [Brangier et Barcenilla 2003]) et adaptées aux EIAH ([Tricot 2003], [Delozanne 2006].

Pour tester la faisabilité de notre approche, nous avons mis au point un prototype. La conception par prototypage est un des fondements de la conception centrée utilisateur ([ISO 13407:1999], [Beaudoin-Lafon et Mackay 2002]). Elle supporte la créativité, elle favorise la communication entre les différentes parties prenantes et elle permet des évaluations précoces des choix de conception. Le prototypage place l'évaluation au centre de la conception (Figure 1, Van Duyne et Landay 2003]). Beaudoin-Lafon et Mackay différencient quatre stratégies de prototypage (le prototypage horizontal, le prototypage vertical, le prototypage orienté tâche et le prototype fondé sur les scénarios) et trois techniques de prototypage (prototypage rapide, itératif et évolutif). Comme le recommande ces auteurs, nous avons combiné diverses approches.

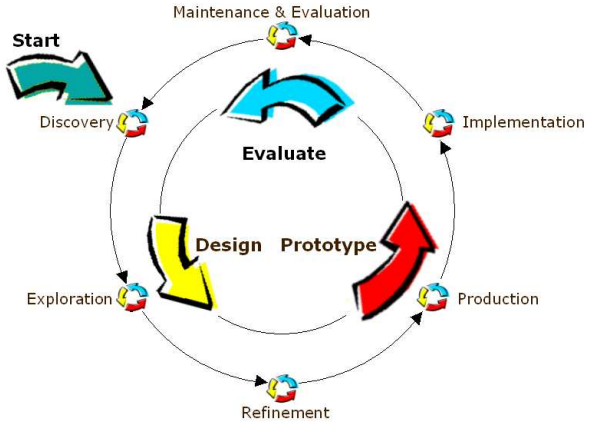


Figure 1 : Conception itérative [Van Duyne et Landay 2003]

Nous avons commencé par du prototypage rapide s'appuyant sur des scénarios. Ces premiers prototypes nous ont permis de tester nos choix de conception par les techniques usuelles en IHM : des techniques d'inspection (revues de conception et jugements d'experts) et des tests utilisateurs auprès d'enseignants et de didacticiens.

Puis, nous avons mis en œuvre un prototypage horizontal en testant le clonage de certains exercices sur des prototypes « à jeter » dont l'objectif est d'étudier des problèmes particuliers en proposant des solutions ad hoc pour mieux comprendre le problème. Des stagiaires de master ont ainsi proposé des solutions pour la génération de clones des exercices à réponses préformatées, des exercices dont les énoncés comportent des figures géométriques et des énoncés conduisant à la modélisation de problèmes de la vie courante (vitesse, débit, abonnement etc.). Ces prototypes, nous ont aidées à spécifier les traitements de Pépinière et aussi les paramètres de nos classes d'exercices et à abstraire le métamodèle qui les décrivent.

Les prototypes verticaux consistent à mettre en œuvre les différentes couches d'une partie du logiciel afin que celui-ci fonctionne complètement depuis l'interface utilisateur, jusqu'à la couche système. Ils servent à évaluer la faisabilité des choix de conception définis sur les prototypes rapides. Nous avons donc opté pour le prototypage vertical de la chaîne logicielle complète illustrant la création et l'utilisation d'une classe emblématique de tâches diagnostiques. Un utilisateur non informaticien peut saisir un énoncé d'exercice de la classe « Preuve et programme de calcul », poser l'exercice à des élèves et obtenir l'analyse multicritère automatique de leurs réponses.

La méthode de conception de notre chaîne logicielle se fonde sur ce que certains auteurs (e.g. [Mark et Greer 1993] appellent une « évaluation formative », c'est-à-dire en cours de conception. Par contre, la seule « évaluation sommative » que nous pouvons mettre en œuvre, dans le cadre

de cette thèse, concerne le prototype vertical, et non la chaîne complète. En effet, l'évaluation des usages n'a de sens qu'avec une intégration dans les pratiques usuelles [Bruillard et Baron 2006] qui exige un travail de développement pour passer du prototype au logiciel SuperPépite. Notre travail de recherche a consisté à préparer ce travail de développement en spécifiant le logiciel et en prouvant la faisabilité de la chaîne complète sur un exemple exécutable.

Notre prototype vertical a été conçu comme un prototype évolutif, contrairement aux prototypes horizontaux. Ainsi, afin d'assurer son évolutivité et de le compléter par la mise en œuvre d'autres classes de tâches diagnostiques, comme le recommande [Beaudoin-Lafon et Mackay 2002], nous avons veillé à fonder son développement sur des modèles UML dûment spécifiés, sur une architecture logicielle modulaire fondée sur l'utilisation de *design patterns* bien documentés. En particulier, nous avons réifié les éléments de l'analyse didactique en les modélisant et en les représentant dans des fichiers XML bien spécifiés : ceci concerne les dimensions et les critères d'évaluation, les composantes de la compétence algébrique et les différentes capacités, les connaissances et les types d'exercices. Nous avons de même spécifié le format des traces produites par le système pour mémoriser dans des fichiers XML les réponses des élèves et leur codage.

En ce qui concerne l'évaluation de notre prototype, du point de vue de l'interaction, nous avons organisé des tests d'utilisabilité avec d'une part, deux enseignants et deux didacticiens de l'équipe de recherche et, d'autre part, deux enseignants de l'IUFM de Rennes n'appartenant pas à l'équipe. Après une prise en main rapide, ils ont produit, chacun en moins d'une heure, une dizaine d'énoncés sans difficulté particulière. Une seule personne n'a pas réussi à installer le logiciel en raison de l'ancienneté des versions de la machine virtuelle java sur son ordinateur personnel. Avec les quelques dizaines d'exercices obtenus, il ne sert à rien de multiplier les expérimentations avant d'avoir développé les autres classes d'exercices et le générateur de tests.

Pour ces développements, afin de mettre en œuvre une autre classe d'exercices, en disposant de l'analyse didactique a priori, l'informaticien devra : (i) programmer la génération des questions pour tenir compte de contraintes ou faire saisir les paramètres par l'enseignant, (ii) modifier l'interface de PépiTest actuel pour remplacer les champs de texte existant par les données du fichier XML spécifiant l'exercice, (iii) et uniquement dans le cas des questions ouvertes, utiliser Pépinière pour la génération des réponses anticipées et de leur analyse.

Relativement à la première étape, nous avons testé la génération des exercices sur d'autres prototypes précoces. Notre approche s'applique à tous les exercices du test originel dont les énoncés sont générés par des programmes de calcul (2/20) ou par instanciation de formules

mathématiques (16/20). Seuls les exercices qui font intervenir des figures variables posent problèmes. Pour ces derniers (2/20), nous envisageons l'utilisation d'un logiciel de géométrie dynamique qui génère des fichiers XML. Pour la deuxième étape, la seule difficulté consiste dans l'affichage des expressions algébriques qui, dans l'ancienne version de Pépite, étaient affichées sous forme d'images, solution non compatible avec une approche générique. Actuellement, nous utilisons un logiciel [Jaxe] qui s'approche des habitudes scolaires. Nous envisageons d'étudier l'utilisation de l'éditeur d'Activemaths qui semble plus performant. Pour la troisième étape, c'est bien sûr la prévision des réponses anticipées aux questions réponses ouvertes, qui présente les principales difficultés. Cette prévision est menée par le module de génération pour les exercices où les réponses sont préformatées. Pour les autres exercices, comme nous l'avons déjà indiqué, cette génération est prévue pour les exercices qui consistent à prouver par développement et réduction ; par contre, l'analyse précise des techniques de résolution d'équations n'est pas envisageable sans un gros travail de développement. Remarquons cependant que cette analyse précise n'est pas nécessaire pour le diagnostic cognitif actuel qui nous sert de référence. Deux exercices mettent en jeu la résolution libre de systèmes d'équations, mais seule la façon dont l'élève effectue la mise en équation (ce qui a ce niveau scolaire est la difficulté majeure) et la validité de la résolution sont évalués. Pour conclure sur le développement de PépiGen, nous estimons à environ six mois d'un ingénieur à plein temps, le développement des classes d'exercices permettant de cloner PépiTest. À cela, nous estimons qu'il faut ajouter une quantité de travail équivalente pour mettre en œuvre les interfaces de composition de tests et de présentation du diagnostic aux élèves et aux enseignants, interfaces qui ont été spécifiés par d'autres membres de l'équipe [Bernard 2006] [Chenevotot et al. 2008].

5. Conclusion

La validation des résultats de recherche en EIAH est un problème complexe et ouvert [Mark et Greer 1993, Tricot 2003, Luengo et al. 2006, Tchounikine 2006]. Les informaticiens veulent des résultats originaux, génériques et réutilisables ; les chercheurs en sciences humaines veulent le droit de changer d'avis et d'expérimenter rapidement de nombreuses variables pour tester leurs hypothèses ; enfin les praticiens veulent des logiciels robustes, fiables et d'une ampleur suffisante pour que leur utilisation ne soit pas occasionnelle.

Dans le cadre de notre thèse, nous estimons avoir fait faire un pas significatif au projet Lingot en posant les fondements d'une chaîne logicielle permettant de diversifier les tests diagnostics. La création de banques d'exercices paramétrés est en effet indispensable pour envisager une

utilisation efficace des diagnostics à différents moments de la compétence algébrique des élèves. Dans cette optique, nous avons proposé de développer des classes paramétrées de tâches diagnostiques pour permettre l'adaptation à différents contextes d'un test qui a prouvé son efficacité et sa robustesse. Nous avons développé certaines de ces classes pour montrer la faisabilité de notre approche.

Nous faisons l'hypothèse que les modèles que nous avons proposés et le métamodèle qui les décrit est un premier pas vers un modèle réutilisable pour l'évaluation multicritère de réponses ouvertes. Notre expérience consistant à partir d'exemples d'exercices conçus par des praticiens, puis de généraliser pour définir des classes d'exercices paramétrables, nous semble aussi une expérience réutilisable.

Enfin, dans ce mémoire, nous avons présenté notre point de vue d'informaticiennes, mais notre travail contribue à enrichir le modèle didactique mis au point par les didacticiennes du projet Lingot. Il a permis de systématiser la démarche des didacticiennes de l'équipe et a encouragé et éclairé la mise au point d'exercices de diagnostic pour d'autres niveaux scolaires [Chenevotot et al.2008].

Pour terminer, je voudrais dire que se lancer dans une thèse quand on approche de la retraite est une entreprise un peu folle mais passionnante et je suis partante pour de nouvelles aventures avec l'équipe de Pépite et Lingot.

Bibliographie

1. Bibliographie

[Aho et al. 1991] Aho A., Sethi R., Ullman J., *Compilateurs : Principes techniques et outils*, InterEditions, Paris, 1991.

[Aho et al. 1993] Aho A., Sethi R., Ullman J., Description récursive de motifs, *Concepts fondamentaux de l'informatique*, Dunod, Paris, chapitre 11, pp. 645-698, 1993.

[Ainsworth 2003] Ainsworth S., V., Major N., Grimshaw S., Hayes M., Underwood J., Williams B., Wood D., REDEEM : Simple Intelligent Tutoring System froom usable tools, *Authoring Tools for Advanced Technology Learning Environments*, Tom Murray, Stephen Blessing and Shaaron Ainsworth (Eds.), Kluwer Academic Publishers, pp. 205-232, 2003.

[Aleven et al. 2006] Aleven V., McLaren B. M., Sewall J., Koedinger, K., The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains, *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), Springer-Verlag, Berlin, pp. 61-70, 2006.

[Anderson 1983] Anderson J. R., A general learning theory and its application to the acquisition of proof skills in geometry, *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonell, T. Mitchell (Eds.), 1983.

[Anderson 1986] Anderson J. R., Knowledge compilation: The general learning mechanism, *Machine learning II*. R. Michalski, J. Carbonell, T. Mitchell (Eds.), pp. 289-310, 1986.

[Anderson 1996] Anderson J. R., ACT: A simple theory of complex cognition. *American Psychologist*, vol. 51, n° 4, pp. 355-365,1996.

[Anderson et Pelletier 1991] Anderson, J. R., Pelletier, R., A development system for model-tracing tutors, *Proceedings of the International Conference of the Learning Sciences*, Evanston, pp.1-208, 1991.

[Anderson et al. 1985a] Anderson, J. R., Reiser B. J., The LISP tutor, *Byte*, vol. 10, pp. 159-175, 1985.

[Anderson et al. 1985b] Anderson J. R., Boyle C. F., Yost G., The geometry tutor, *Proceedings of IJCAI*, pp. 1-7, 1985.

[Anderson et al. 1985c] Anderson J. R., Boyle C. F., Reiser B. J., Intelligent tutoring systems, *Science*, vol. 228, pp. 456-462, 1985.

[Anderson et al. 1990] Anderson J. R., Boyle C. F., Corbett A., Lewis M. W., Cognitive modelling and intelligent tutoring, *Artificial Intelligence*, n° 42, pp. 7-49, 1990.

[Anderson et al.1995] Anderson J.R., Corbett A.T., Koedinger K.R., Pelletier R,. Cognitive tutors : Lessons learned, *The Journal of the Learning Sciences*, vol. 4, n° 2, pp. 167-207, 1995.

[Artigue et al. 1998] Artigue M., Lagrange JB, Instrumentation et écologie didactique de calculatrices complexes : Éléments d'analyse à partir d'une expérimentation en classe de première S, *Actes du colloque « calculatrices géométriques et symboliques »*, IREM de Montpellier, mai 1998.

[Artigue et al. 2001] Artigue M., Assude T., Grugeon B., Lenfant A., Teaching and Learning Algebra : approaching complexity trough complementary perspectives, The future of the Teaching and Learning of Algebra, *Proceedings of 12 th ICMI Study Conference*, H. Chick, K. Stacey, J. Vincent, J. Vincent (Eds.), University of Melbourne, Australia, pp. 21-32, December 2001.

[Auzende et al. 2007] Auzende O., Giroire H., Le Calvez F., Propositions d'extensions à IMS-QTI 2.1 pour l'expression de contraintes sur les variables d'exercices mathématiques, *Actes de la conférence EIAH 2007*, T. Nodenot, J. Wallet, E. Fernandes (Eds.), Lausanne, pp. 47-58, juin 2007.

[B.O. 2001] Bulletin officiel du ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche, Programme des lycées, vol. 7, Programme de la classe de seconde générale et technologique, Hors Série N° 2 du 30 août 2001, pp. 31-36.

[B.O. 2005] Bulletin officiel du ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche, Programme des collèges, vol. 2, Programme de mathématiques, Hors Série N° 5 du 25 août 2005, 67 p.

[B.O. 2007] Bulletin officiel du ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche, Mise en œuvre du socle commun de connaissances et de compétences, Hors Série N° 6 du 19 avril 2007, vol. 2, pp. 11-66.

[Baker 2000] Baker M.J. « The roles of models in Artificial Intelligence and Education research : a prospective view » *International Journal of Artificial Intelligence and Education*, vol. 11, pp. 122-143, 2000.

[Balacheff 1994] Balacheff N., «Didactique et intelligence artificielle », *Recherche en Didactique des Mathématiques*, La Pensée Sauvage (Eds.), Grenoble, vol. 14, n° 1-2, pp. 9-42, 1994.

[Baron et Bruillard 1996] Baron G.-L., Bruillard E., L'informatique et ses usagers dans l'éducation, *L'Educateur*, Presses Universitaires de France, Paris, 312 p., 1996.

[Bardini 2003] Bardini C., Le rapport au symbolisme algébrique : une approche didactique et épistémologique, Thèse de doctorat, Université de Paris 7, 2003.

[Bardini et al. 2005] Bardini C., Radford L., Sabena C., Struggling with variables, parameters and indeterminate objects or how to go insane in Mathematics, *Proceedings of the 29th Conference of Psychology of Mathematics Education*, H. L. Chick, J. L. Vincent (Eds.), Melbourne, vol. 2, pp.129-136, 2005.

[Baron et Simonet 1992] Baron M., Simonet P., Génération d'exercices en algèbre, premières approches dans le cadre du projet Aplusix, *Intelligent Tutoring Systems, ITS 1992*, Claude Frasson, Gilles Gauthier, Gordon I. McCalla (Eds.), Montréal, Canada, *Proceedings*, LNCS 608 Springer 1992, 335-342, June 10-12, 1992.

[Bastien et Scapin 2001] Bastien C., Scapin D., *Évaluation des systèmes d'information et critères ergonomiques, Environnements évolués et évaluation de l'IHM, Interaction homme-machine pour les systèmes d'information*, Kolski C. (dir.), Hermès, Paris, vol. 2, pp. 51-75, 2001.

[Beaudouin-Lafon et Mackay 2002] Beaudouin-Lafon M., Mackay W., Prototyping Tools and Techniques, *Human Computer Interaction Handbook*, J.A. Jacko and A. Sears (Eds.), Lawrence Erlbaum Associates, pp. 1006-1031, 2002.

[Beeson 2002] Beeson M., MathXpert: un logiciel pour aider les élèves à apprendre les mathématiques par l'action, *Revue Sciences et Techniques éducatives, Logiciels pour l'apprentissage de l'algèbre*, Hermès, Paris, vol. 9, n° 1-2, pp. 37-62, 2002.

[Bell 2003] Bell B., Supporting Educational SoftWare Design With Kknowledge-Rich Tools, *Authoring Tools for Advanced Technology Learning Environments*, Tom Murray, Stephen Blessing and Shaaron Ainsworth (Eds.), Kluwer Academic Publishers, pp. 341-376, 2003.

[Bennett 2002] Bennett R.E., Inexorable and inevitable : The continuing story of technology and assessment, *Journal of Technology, Learning, and Assessment*, vol. 1, n° 1, 2002.

[Brangier et Barcenilla 2003] Brangier E., Barcenilla X., *Concevoir un produit facile à utiliser : Adapter les technologies à l'homme*, Editions d'organisation, 250 p., 2003.

[Bernard 2006] Bernard J., Réingénierie d'un logiciel de présentation de résultats dans un environnement informatique pour l'apprentissage humain, Mémoire de master sciences et technologies, Option communication Homme/Machine, Université du Maine, 2006.

[Blat et al. 2007] Blat J., Navarrete T., Moghnieh A., Batlle Delgado H., A QTI Management System Implemented for Service Oriented Architectures in Service - Oriented Approaches and Lifelong, *TENCompetence Open Workshop on Service Oriented Approaches and Lifelong Competence Development Infrastructures*, Manchester, pp. 175-182, 11-12 Janvier, 2007[Blessing 1997] Blessing B., A Programming by Demonstration Authoring Tool for Model-Tracing Tutors, *International Journal of Artificial Intelligence in Education*, Murray T., Blessing S. (Eds.), vol. 8, pp. 233-261, 1997.

[Blessing 2003] Blessing B., A Programming by Demonstration Authoring Tool for Model-Tracing Tutors, Tom Murray, Stephen Blessing and Shaaron Ainsworth (Eds.), Kluwer Academic Publishers, pp. 93-119, 2003.

[Bouhineau et Nicaud 2006] Bouhineau D., Nicaud J.-F., Aplusix, un EIAH de l'algèbre, *Environnements informatiques pour l'apprentissage humain*, M. Grandbastien, J.-M. Labat (Eds.), chapitre 15, Hermes-Lavoisier, pp. 333-350, 2006.

[Bouhineau et al. 2001] Bouhineau D., Nicaud J.-F., Pavard X., Sander E., Un micromonde pour aider les élèves à apprendre l'algébre, *Sixièmes journées francophones Environnements Interactifs d'Apprentissage avec Ordinateur EAIO 2001, Sciences et Techniques Educatives*, vol. 8, n° 1-2, pp. 33-47, 2001.

[Bouhineau et al. 2005] Bouhineau D., Bronner A., Chaachoua H., Nicaud J.-F., Patrons d'exercices pour Aplusix, Une étape du développement de l'EIAH occasion d'un travail entre didacticiens et informaticiens, *Actes de la conférence EIAH 2005*, Montpellier, pp. 377-382, mai 2005.

[Bouhineau et al. 2007] Bouhineau D., Chaachoua H., Nicaud J.-F., Viudez C., Introduction de nouvelles représentations dans le micromonde Aplusix, Représentations sous forme mixte Naturelle & Arbre et sous forme graphique d'expressions algébriques, *Actes de la conférence EIAH 2007*, T. Nodenot, J. Wallet, E. Fernandes (Eds.), Lausanne, pp. 389-400, 2007.

[Bourda 2001] Bourda Y., Objets pédagogiques, vous avez dit objets pédagogiques ?, *Cahier GUTenberg*, n° 39-40, pp. 71-79, mai 2001.

[Bourda et Helier 2000] Bourda Y., Helier M., Métadonnées et XML : application aux « Objets pédagogiques », *TICE 2000*, Troyes octobre 2000.

[Bourda et Delestre 2005] Bourda Y., Delestre N., Améliorer l'interopérabilité des profils d'application du LOM, *Revue STICEF*, vol. 12, 2005.

[Bruillard 1997] Bruillard É., *Les machines à enseigner*, Éditions Hermès, Paris, 320 p, 1997.

[Bruillard et al. 2000] Bruillard E., Delozanne E., Leroux P., Delannoy P., Dubourg X., Jacoboni P., Lehuen J., Luzzati D., Teutsch P., Quinze ans de recherche sur les sciences et techniques éducatives au LIUM, Education et informatique, Hommage à Martial Vivet, *Sciences et Techniques éducatives*, Hermès, Paris, vol. 7, n° 1, pp. 87-145, 2000.

[Bruillard et Baron 2006] Bruillard E., Baron G.-L.,. Usages en milieu scolaire : caractérisation, observation et évaluation, *Environnements informatiques pour l'apprentissage humain*, Collection IC2, M. Grandbastien, J.-M. Labat (Eds.), Hermes-Lavoisier, Paris, chapitre 12, pp. 269-284, 2006.

[Burgos et al. 2005] Burgos.D, Arnaud M, Neuhauser P., Koper R., IMS Learning Design : la flexibilité pédagogique au service des besoins de l'e-formation, *revue de l'EPI*, décembre 2005.

[Caroll 2001a] Caroll J.M., Human Computer Interaction in the new Millennium, Caroll J.M. (Eds.), Addison Wesley, 700 p., 2001.

[Caroll 2001b] Carrol J.M. , Chin G., Rosson M. B., Neale D. C., The development of Cooperation : Five years of Particpatory Design in the Virtual School, Caroll J.M. (dir.), *Human Computer Interaction in the new Millennium*, Addison Wesley, pp. 373-396, 2001.

[Cerrulli et al. 2002] Cerrulli M., Mariotti A., L'Algebrista : un micromonde pour l'enseignement et l'apprentissage de algèbre, *Revue Sciences et Techniques éducatives, Logiciels pour l'apprentissage de l'algèbre*, Hermès, Paris, vol. 9, n° 1-2, pp. 149-170, 2002.

[Chaachoua et al. 2005] Chaachoua H., Nicaud J.-F., Bittar M., Détermination automatique des théorèmes-en-acte des élèves en algèbre. Le cas des équations et inéquations de degré 1, *Actes de la conférence EIAH 2005*, Montpellier, pp. 33-44, mai 2005.

[Chaachoua et al. 2007] Chaachoua H.,Crozet M.-C., Bouhineau D., Bittar M., Nicaud J.-F., Description et exploitations des traces du logiciel d'algèbre Aplusix, *Numéro spécial : Analyses des traces d'utilisation dans les EIAH, Revue STICEF*, vol. 14, 2007.

[Chenevotot et Grugeon 2008] Chenevotot F., Grugeon B., Le projet Lingot : Adaptation du diagnostic en algèbre élémentaire, 2008 (article soumis).

[Chi et Vanlehn 2007] Chi M., VanLehn K., Domain-Specific and Domain-Independent Interactive Behaviors in Andes, *Artificial Intelligence in Education*, Amsterdam, Netherlands, R. Luckin, K. R. Koedinger, J. Greer (Eds.), IOS Press, 2007.

[David 2003] David J.P., Modélisation et production d'objets pédagogiques, une approche par objets pédagogiques, *Revue Sciences et Techniques éducatives, Ressources numériques, XML et éducation*, Hors série 2003, Hermès, Paris, pp.69-104, 2003.

[De La Passardiere et Jarraud 2004] De La Passardière B., Jarraud P., ManUeL, un profil d'application de LOM pour C@mpuSciences, *Revue STICEF*, vol. 11, 2004.

[De La Passardiere et Jarraud 2005] De La Passardière B., Jarraud P. ManUeL, LOM et l'indexation de ressources scientifiques, Vers les bonnes pratiques pour l'Université en ligne, *Actes de la conférence EIAH 2005*, Montpellier, pp. 57-68, mai 2005.

[De La Passardiere et Grandbastien 2003] De La Passardière B., GrandBastien M., Présentation de LOM v1.0, standard IEEE, *Revue Sciences et Techniques éducatives, Ressources numériques, XML et éducation*, Hors série 2003, Hermès, Paris, pp.211-218, 2003.

[Delozanne 2006] Delozanne E., Interfaces en EIAH, *Environnements informatiques pour l'apprentissage humain*, Collection IC2, M. Grandbastien, J.-M. Labat (Eds.), Hermes-Lavoisier, Paris, chapitre 10, pp. 223-248, 2006.

[Delozanne et al. 2002a] Delozanne E., Grugeon B. , Artigue M., Rogalski J., Modélisation et mise en œuvre d'environnements informatiques pour la régulation de l'apprentissage, le cas de l'algèbre avec le projet LINGOT, *Réponse à l'appel à Projet Cognitique, École et sciences cognitives : Les apprentissages et leurs dysfonctionnements*, 2002.

[Delozanne et al. 2003] Delozanne E., Prévit D., Grugeon B., Jacoboni P., Scénarios d' utilisation et conception d'un EIAH, le cas du diagnostic dans Pépite, *Colloque Intégration des Technologies à l'Enseignement des Mathématiques*, Reims, juin 2003.

[Delozanne et al. 2003] Delozanne E., Prévit D., Grugeon B., Jacoboni P., Supporting teachers when diagnosing their students in algebra, *Workshop Advanced Technologies for Mathematics Education, Proceedings of Artificial Intelligence in Education*, Sydney, IOS Press, Amsterdam, pp. 461- 470, July 2003.

[Delozanne et al. 2005] Artigues M., Coulange L., Chenevetot F., Delozanne É., Gélis J.-M., Grugeon B., Jacobini P., Normand-Assadi S., Prévit D., Rogalski J., Vincent C., Modélisation et mise en œuvre d'environnements informatiques pour la régulation de l'apprentissage, le cas de l'algèbre avec le projet LINGOT, *Projet Cognitique 2002, École et sciences cognitives: Les apprentissages et leurs dysfonctionnements, rapport de fin de projet*, pp. 91-107, mars 2005.

[Delozanne et al. 2002b] Delozanne E., Grugeon B., Jacoboni P., Analyses de l'activité et IHM pour l'éducation, *Proceedings of IHM'2002, International Conference Proceedings Series, ACM*, Poitiers, France, pp. 25-32, 2002.

[Delozanne et Grugeon 2003] Delozanne E., Grugeon B., EIAH et apprentissage de l'algèbre élémentaire : les projets Pépite et Lingot, *Actes du Séminaire National de Didactique des Mathématiques*, janvier 2003.

[Delozanne et Prévit 2007] Delozanne E., Prévit D., Le projet Lingot et les EIAH en algèbre, Rapport interne, 2007.

[Dershowitz et al.1990] Dershowitz N., Jouannaud J.-P., Rewrite Systems, *Handbook of Theoretical Computer Science*, North-Holland, J. Van Leeuwen, (Eds.), vol. B, chapitre 6, pp. 243-320, 1990.

[Devedzic et Harrer 2005] Devedzic V., HarreR A., Software Patterns in ITS Architectures, *International Journal of Artificial Intelligence in Education*, n° 15, pp. 63-94, 2005.

[Farouk et al. 2007] Farouk M., Réty J.-H., Delozanne É., Grugeon B., Bensimon V, Martin J. C., Stratégies d'utilisation de la direction du regard en situation de communication interpersonnelle enseignant-élève, *Numero Spécial : Les Dimensions émotionnelles de l'interaction dans un EIAH*, *Revue STICEF*, vol. 14, 2007.

[Favre et al. 2006] Favre, J.-M., Estublier, J., Blay-Fornarino, M., *L'ingénierie dirigée par les modèles, au-delà du MDA*. Editions Lavoisier, Hermes Sciences Publications, Paris, 2006.

[Feng et Heffernan 2005] Feng M., Heffernan N., Informing Teachers Live about Student Learning: Reporting in the Assistment System. *The 12th International Conference on Artificial Intelligence in Education* 2005, *Workshop on Usage Analysis in Learning Systems*, Amsterdam, pp. 25-32, 2005.

[Feng et Heffernan 2006] Feng M., Heffernan N., Informing Teachers Live about Student Learning: Reporting in the Assistment System, Technology, Instruction, *Cognition and Learning (TICL)*, vol. 3 n° 1-2, 2006.

[Gelis 1994] Gelis J.-M., Éléments d'un théorie cognitive et computationnelle de l'algèbre. Application au cas de la factorisation d'expressions polynomiales, Thèse de doctorat, Université de Nantes, 260 p., 1994.

[Gounon et al. 2005] Gounon P., Leroux P., Dubourg X., Décrire le comportement des apprenants - Proposition d'une extension du langage de modélisation pédagogique IMS Learning Design, *Actes de la conférence EIAH 2005*, Montpellier, pp. 261-272, mai 2005.

[Grandbastien 1974] Grandbastien M., Un programme qui résout formellement des équations trigonométriques par des procédés heuristiques, Thèse de troisième cycle, Université de Paris 6, juin 1974.

[Grugeon 1995] Grugeon B., Etude des rapports institutionnels et des rapports personnels des élèves à l'algèbre élémentaire dans la transition entre deux cycles d'enseignement : BEP et Première G, Thèse de doctorat, Université Paris 7, 1995.

[Grugeon 1997] Grugeon B., Conception et exploitation d'une structure d'analyse multidimensionnelle en algèbre élémentaire, *Revue de Didactique des Mathématiques*, vol. 17, n°2, pp.167-210, 1997.

[Grugeon et al. 2003] Grugeon B., Coulange L., Larue V., Familles de situations d'interactions en algèbre élémentaire : deux exemples, *Colloque Intégration des Technologies à l'Enseignement des Mathématiques, ITEM 2003*, Reims, juin 2003.

[GHJV95] E. Gamma, E., Helm R., Johnson R., Vlissides J., Design patterns, elements of reusable object-oriented software, Addison-Wesley publishing company,1995.

[Guéraud 2004] Guéraud v., Adam J.-M., Pernin J.-P., Calvary G., David J.-P., L'exploitation d'Objets Pédagogiques Interactifs à distance : le projet Formid, *Revue STICEF*, vol. 11, 2004.

[Hakem et al. 2005] Hakem K., Sander E., Labat J.-M., DIANE (Diagnostic Informatique sur l'Arithmétique au Niveau Elémentaire), *Actes de la conférence EIAH 2005*, Montpellier, pp. 81-92, mai 2005.

[Hefferman et Koedinger 2002] Heffernan N. T., Koedinge K., R., Miss Lindquist : un système fondé sur le dialogue pour apprendre à exprimer algébriquement des énoncés en langage naturel, *Revue Sciences et Techniques éducatives, Logiciels pour l'apprentissage de l'algèbre*, Hermès, Paris, vol. 9, n° 1-2, pp. 11-35, 2002.

[Hibou et Py 2006] Hibou M., Py D., Représentation des connaissances de l'apprenant, *Environnements informatiques pour l'apprentissage humain*, Collection IC2, M. Grandbastien, J.-M. Labat (Eds.), Hermes-Lavoisier, Paris, chapitre 4, pp. 97-116, 2006.

[Hsieh 1999 et al] Hsieh P. Y., Halff H. M., Redfield C. L., Four Easy Pieces: Development Systems for Knowledge-Based Generative Instruction, *International Journal of Artificial Intelligence in Education*, Murray T., Blessing S. (Eds.), vol. 10, 1999, pp. 1-45.

[I.G.E.N. 2007] Inspection Générale de l'Education Nationale, Les livrets de compétences : nouveaux outils pour l'évaluation des acquis, Rapport n° 2007-048, 60 p., juin 2007.

[IEEE 2001] Learning Technology Standards Committee, IEEE Computer Society. Draft Standard for Learning Technology-Learning Technology Systems Architecture (LTSA). IEEE P1484.1/D9, 2001.

[ISO 1999] ISO 13407:1999 Human-centred design processes for interactive systems.

[Jean et al. 1999] Jean S., Delozanne E., Jacoboni P. et Grugeon B., A Diagnosis Based on a Qualitative Model of Competence in Elementary Algebra, *Proceedings of Artificial Intelligence in Education*, Le Mans, S. Lajoie, M. Vivet (Eds.), IOS Press, Amsterdam, pp. 491-498, juillet 1999.

[Jean 2000] Jean S., PEPITE : un système d'assistance au diagnostic de compétences, Thèse de doctorat, Université du Maine, 2000.

[Jean-Daubias 2002] Jean-Daubias S., Un système d'assistance au diagnostic de compétences en algèbre élémentaire, *Revue Sciences et Techniques éducatives, Logiciels pour l'apprentissage de l'algèbre*, Hermès, Paris, vol. 9, n° 1-2, pp.171-200, 2002.

[Joosten-ten Brinke et al. 2007] Joosten-ten Brinke D., van Bruggen J., Hermans H., Burgers J., Giesbers B., Koper R. and Latour I., Modeling assessment for re-use of traditional and new types of assessment, *Computers in Human Behavior*, 23(6), pp. 2721-2741, 2007.

[Jordan et al. 2006] Jordan P. W., Makatchev M., Pappuswamy U., VanLehn K., Albacete P. L., Natural Language Tutorial Dialogue System for Physics, *FLAIRS Conference*, pp. 521-526, 2006.

[Koedinger 2001] Koedinger K. R., Cognitive tutors as modeling tool and instructional model, *Smart Machines in Education: The Coming Revolution in Educational Technology*, Forbus K. D., Feltovich P. J. (Eds.), pp. 145-168, 2001.

[Koedinger et al.1997] Koedinger K., Anderson J., Hadley W., Mark M., Intelligent Tutoring Goes To School in the Big City, *International Journal of Artificial Intelligence in Education*, vol. 8, pp. 30-43, 1997.

[Koedinger et al. 2003] Koedinger K., Aleven V., Heffernan N., Toward a Rapid Developpement Environnement for Cognitive Tutors, *The 11th International Conference on Artificial Intelligence in Education* 2003, vol II, InteractiveEvent, Sydney, pp. 24-26, 2003.

[Koedinger et al. 2004] Koedinger K. , Aleven V., Heffernan. T., McLaren B., Hockenberry M., Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. *In the Proceedings of 7th Annual Intelligent Tutoring Systems Conference*, pp. 162-174, 2004.

[Kolski 2001] Kolski C., *Analyse et conception de l'IHM, Interaction homme-machine pour les systèmes d'information*, C .Kolski (Ed.), Hermès, Paris, vol. 1, 250 p., 2001.

[Koper 2001] Koper R., Modeling units of study from a pedagogical perspective. the pedagogical meta-model behind EML, Open University of the Netherland, 2001.

[Koper et Tattersall 2005] Koper R., Tattersall C, *LearningDesign: A handbook on modellingand delivering networked education and trainin*, Koper R., Tattersall C., (Eds.), Springer-Verlag, Berlin, 415 p., 2005.

[Laforcade 2004] Laforcade P., Méta-modélisation UML pour la conception et la mise en oeuvre de situations-problèmes coopératives, Thèse de doctorat, Université de Pau et des Pays de l'Adour, 345 p., 2004.

[Lagrange 2002] Lagrange J. B., Le projet Casyopée : un environnement de calcul formel pour l'étude de propriétés des fonctions au lycée, *Actes du colloque "Regards sur l'enseignement de l'analyse" de l'IREM de Mulhouse*, Maklouf (Eds.), mars 2002.

[La Passardière et Grandbastien 2003] La Passardière B., Grandbastien M., présentation de LOM v1.0, standard IEEE, *Revue Sciences et Techniques éducatives, Ressources numériques, XML et éducation*, Hors série 2003, Hermès, Paris, pp. 211-218, 2003.

[Larman 2005] Larman C., *UML et les Design patterns*, Editions CampusPress, 660 p.**,** 2003 .

[Laurière 1988] Laurière J.L. , *Intelligence artificielle, Représentation des connaissances*, Eyrolles, 267 p., 1988.

[Lemoyne et al. 2002] Lemoyne G., De Cotret S. R., Coulange L., Des environnements informatisés dédiés à l'étude des conditions d'enseignement et d'apprentissage des mathématiques et à la formation des enseignants, *Actes du Colloque* CIRTA, Acfas, 2002.

[Lenfant 1997] Lenfant A., Étude sur la transposition d'un outil de recherche destiné aux enseignants, Mémoire de DEA de didactique des mathématiques, Université Paris 7, 1997.

[Le Feuvre et al. 2003] Le Feuvre B., Meyrier, X., Heilbronner, L., Lagrange J.B., Casyopée : un logiciel pour l'analyse en lycée, *Actes en ligne du colloque International ITEM*, Lagrange J.B. et al. (Eds.), Reims, Juin 2003.

[Leroux 2006] Leroux P., Micromondes et robotique pédagogique, Représentation des connaissances de l'apprenant, *Environnements informatiques pour l'apprentissage humain*, Collection IC2, M. Grandbastien, J.-M. Labat (Eds.), Hermes-Lavoisier, Paris, chapitre 14, pp. 311-332, 2006.

[Leroux et Jean-Daubias 2006] Leroux P., Jean-Daubias S., EIAH partenaires des acteurs de la situation d'apprentissage, *Environnements informatisés et ressources numériques pour l'apprentissage : conception et usages, regards croisés*, sous la direction de Monique Baron, Dominique Guin, Luc Trouche, Hermès-Lavoisier, chapitre 4, 2006.

[Luengo et al. 2006], Luengo V., Vadcard L., Balacheff N., Les EIAH à la lumière de la didactique. *Environnements informatiques pour l'apprentissage humain*, Collection IC2, M. Grandbastien, J.-M. Labat (Eds.), Hermes-Lavoisier, Paris, chapitre 2, pp. 47-68, 2006.

[Mark et Greer 1993] Mark M.A., Greer J.E., Evaluation Méthodologies for Intelligent Tutoring Systems, *Journal of Artificial Intelligence in Education*, vol. 4, n° 2-3, pp. 129-53, 1993.

[Melis et Ullrich 2003] Melis E., Ullrich C., How to Teach it - Polya-Inspired Scenarios in ActiveMath, *Artificial Intelligence in Education*, U. Hoppe, F. Verdejo, J. Kay (Eds.), pp. 141-147, 2003.

[Melis et al. 1999] Melis E., Leron U., A Proof Presentation Suitable for Teaching Proofs, *9th International Conference on Artificial Intelligence in Education*, pp. 483-490, 1999.

[Melis et al. 2001] Melis E., Andres E., Buedenbender J., Frischauf A. ,Goguadse G., Libbrecht P., Pollet M., Ullrich C., ActiveMath : A Generic and Adaptive Web-Bbased Learning Environment, *International Journal of Artificial Intelligence in Education*, vol. 12, pp. 385-407, 2001.

[Melis et al. 2003] Melis E., Büdenbender J., Andres E., Goguadze G., Libbrecht P., Pollet M., Ullrich C. Knowledge representation and management in ActiveMath, *Annals of Mathematics and Artificial Intelligence*, vol. 38, n°1-3 pp. 47-64, 2003.

[Melis et al. 2004] Melis E., Siekmann J., ActiveMath : An Intelligent Tutoring System for Mathematics, *Artificial Intelligence and Soft Computing - ICAISC 2004, 7th International Conference*, Rutkowski L., Siekmann J., Tadeusiewicz R., Zadeh L. (Eds.), Zakopane, Poland, pp. 91-101, 2004.

[Melis et al. 2007] Melis E., Moormann M., Ullrich C., Goguadze G., LibbrechtP., How ActiveMath Supports Moderate Constructivist Mathematics Teaching, *8th International Conference on Technology in Mathematics Teaching*, Hradec Kralove, 2007.

[Meyer et Baudoin 1984] Meyer B., et Beaudouin C., Méthodes de programmation, *Collection Direction des études et recherches d'Electricité de France (EDF)*, Eyrolles, 688 p., 1984.

[Michel et Rouissi 2003] Michel C., Rouissi S., E-learning : normes et spécifications. Caractérisation des documents numériques avec LOM et IMS-QTI pour l'acquisition et l'évaluation des connaissances, *Document numérique 2003/1-2*, vol. 7, pp. 157-178, 2003.

[Morgan et Ritter 2002] Morgan P., Ritter, S., An experimental study of the effects of Cognitive Tutor

Alegbra I on student knowledge and attitude, 2002.

[Muller et Gaertner 2000] Muller P.-A., Gaertner N., *Modélisation objet avec UML*, Eyrolles, 540 p., 2000.

[Murray 1997] Murray T., Expanding the knowledge acquisition bottleneck for intelligent tutoring system, *International Journal of Artificial Intelligence in Education*, Murray T., Blessing S. (Eds.), vol. 8, pp. 222-232, 1997.

[Murray 1999] Murray T., Authoring Intelligent Tutoring Systems: An analysis of the state of the art, *International Journal of Artificial Intelligence in Education*, Murray T., Blessing S. (Eds.), vol. 10, pp. 98-129, 1999.

[Murray 2003a] Murray T., An overview of intelligent tutoring system authoring tools : updated analysis of the state of the art, *Authoring Tools for Advanced Technology Learning Environments*, Tom Murray, Stephen Blessing and Shaaron Ainsworth (Eds.), Kluwer Academic Publishers, pp. 491-544, 2003.

[Murray 2003b] Murray T., Principles for Pedagogy-Oriented Kknowledge Based Tutor Authoring Systems : Lessons Learned and Design Meta-Model, *Authoring Tools for Advanced Technology Learning Environments*, Tom Murray, Stephen Blessing and Shaaron Ainsworth (Eds.), Kluwer Academic Publishers, pp. 439-466, 2003.

[Murray 2003c] Murray T., EON : Authoring Tools for Content, Instructional Strategy, Student Model, an Interface Design, *Tools for Advanced Technology Learning Environments*, Tom Murray, Stephen Blessing and Shaaron Ainsworth (Eds.), Kluwer Academic Publishers, pp. 309-339, 2003.

[Nguyen-Xuan et al. 2002] Nguyen-Xuan A., Nicaud J.-F., Bastide A., Sander E., Les expérimentations du projet APLUSIX, *Revue Sciences et Techniques éducatives, Logiciels pour l'apprentissage de l'algèbre*, Hermès, Paris, vol. 9, n° 1-2, pp. 63-90, 2002.

[Nicaud 1987] Nicaud J.-F., APLUSIX : un système expert de résolution pédagogique d'exercices d'algèbre, Thèse de doctorat, Université de Paris XI-Orsay , 189 p., 1987.

[Nicaud 1993] Nicaud J.-F., Le projet APLUSIX, *Revue de l'EPI*, n° 72, pp. 213-223, décembre 1993.

[Nicaud 1994] Nicaud J.-F., Modélisation en EIAO, les modèles d'APLUSIX, Recherches en Didactique des Mathématiques, vol. 14, n°1.2, pp. 67-112, 1994.

[Nicaud 2002] Nicaud J.-F., Les expérimentations du projet APLUSIX, Logiciels pour l'apprentisage de l'algèbre, *Sciences et techniques éducatives*, vol. 9, n° 1-2, Hermés, Paris, pp. 63-90, 2002.

[Nicaud 2007] Nicaud J.-F., Natural Editing of Algebraic Expressions, *Mathematical User-Interfaces, Workshop 2007*, Linz, Austria, juin 2007.

[Nicaud et al. 2001] Nicaud J.-F., Bouhineau D., Gelis J.-F., Syntax an Semantics in Agebra, *Proceedings of the 12th ICMI Study Conference: The Future of the Teaching and the Learning of Algebra*, University of Melbourne, 2001.

[Nicaud et al. 2003] Nicaud J.-F., Bouhineau D., Chaachoua H., Huguet T., Bronner A., Computer program for the learning of algebra: description and first experiment, *Eleventh International PEG Conference*, St. Petersburg, Russia, June 2003.

[Nicaud et al. 2004] Nicaud, J.F., Bouhineau, D., Chaachoua H., Mixing microworld and CAS features in building computer systems that help students learn algbra, *International Journal of Computers for Mathematical Learning*, Springer-Verlag, vol. 9, Issue 2, 2004.

[Nicaud et al. 2005a] Nicaud J.-F., Chaachoua H., Bittar M., Bouhineau D. (2005). Student's modelling with a lattice of conceptions in the domain of linear equations and inequations, *Proceedings of « Usage Analysis in Learning Systems » workshop, held in conjunction with the 12th Conference on Artificial Intelligence in Education*, Amsterdam, Netherlands, pp.81-88, 2005.

[Nicaud et al. 2005b] Nicaud J.-F., Gordon M., Bisson G., Renaudie D., Olivera W., Gobeill J., Bénard N., Michelet S., Robinet V., Pellegrino F., Sander E., Tapol S., Bronner A., Dème O., Bittar M., Nicaud J., Balacheff N., Bouhineau D., Chaachoua H., Huguet T., Croset M., Fernandez R., Rapport de fin de projet de l'ACI : Ecole et Sciences Cognitives - Modélisation cognitive d'élèves en algèbre et construction de stratégies d'enseignement dans un contexte technologique, *Cahier du laboratoire Leibniz*, n°123, 140 p., 2005.

[Nicaud et Vivet 1988] Nicaud J.-F., Vivet M., Les tuteurs intelligents, réalisations et tendances de recherche, *Technique et Science Informatique*, vol. 7, n°1, pp. 21-45, 1988.

[Nicaud et al 2002] Nicaud J.F., Delozanne E., Grugeon B., Logiciels pour l'apprentisage de l'algèbre, editorial, *Revue Sciences et Techniques éducatives, Logiciels pour l'apprentissage de l'algèbre*, Hermès, Paris, vol. 9, n° 1-2, pp. 7-10, 2002.

[Nodenot 2006] Nodenot T., Etude du potentiel du langage IMS-LD pour scénariser des situations d'apprentissage : résultats et propositions, *Actes électroniques du colloque Scénarios*, Pernin J-P. et Godinet H. (dir.), pp.83-88, 2006.

[Normand et al. 2005] Normand-Assadi S., Coulange L., Delozanne E. et Grugeon B., "Extraction de pépites de connaissances dans des réponses d'élèves en langage naturel", Atelier "Extraction et Gestion de Connaissances dans les Environnements Informatiques pour l'Apprentissage Humain", *5ème journées Extraction et Gestion de Connaissances 2005*, Paris, pp. 23-25, janvier 2005.

[Nuzzo-Jones et al. 2005] Nuzzo-Jones G., Walonoski J. A., Heffernan N. T., Tom Livak The eXtensible Tutor Architecture: A New Foundation for ITS, *Proceedings of the 12th Artificial Intelligence In Education, Amsterdam*, C.K. Looi, G. McCalla, B. Bredeweg, J. Breuker (Eds.), Amsterdam ISO Press, pp. 902-904, 2005.

[Pernin 2003] Pernin J-P., Objets pédagogiques : unités d'apprentissage, activités et ressources ?, *Revue Sciences et Techniques éducatives, Ressources numériques, XML et éducation*, Hors série 2003, Hermès, Paris, pp.179-210, 2003.

[Pitrat 1966] Pitrat J., Réalisation de programmes de démonstration de théorèmes utilisant des méthodes heuristiques, Thèse d'état, Paris, 1966.

[Prévit 2002] Prévit D., Vers un diagnostic de compétences inspectable par différents types d'utilisateurs, Mémoire de DEA Communication Homme / Machine et Ingénierie Éducative, Université du Maine, septembre 2002.

[Prévit 2003] Différentes modélisations pour prendre en compte les usages dans la conception d'un EIAH, *Annexes aux actes de la conférence EIAH 2003*, Strasbourg, pp. 79-82, avril 2003.

[Prévit et al. 2004] Prévit D., Delozanne É., Grugeon B., Modélisation cognitive en algèbre élémentaire : une conception itérative, *Conférence Technologies de l'Information et de la Communication pour l'Enseignement Supérieur*, Compiègne, pp. 138-145, octobre 2004.

[Prévit et al. 2007] Prévit D., Delozanne É., Grugeon B.,Génération d'exercices de diagnostic de compétences en algèbre, *Actes de la conférence EIAH 2007*, T. Nodenot, J. Wallet, E. Fernandes (Eds.), Lausanne, pp. 545-556, juin 2007.

[Rawlings et al. 2002] Rawlings A., Van Rosmalen P., Koper B., Rodríguez-Artacho M., Lefrere P. Survey of Educational Modelling Languages (EMLs) Version 1, *CEN/ISSS WS/LT Learning Technologies Works*, september 2002.

[Razzaq et al. 2005] Razzaq L., Feng M., Nuzzo-Jones G., Heffernan N., Koedinger K., Junker B., Ritter S.,Knight A., Aniszczyk C., Choksey S., Livak T., Mercado E., Turner T.E., Upalekar R, Walonoski J.A., Macasek. M.A., Rasmussen K.P. (2005). The Assistment Project: Blending Assessment and Assisting. *Proceedings of the 12th Artificial Intelligence In Education, Amsterdam*, C.K. Looi, G. McCalla, B. Bredeweg, J. Breuker (Eds.), Amsterdam ISO Press, pp. 555-562, 2005.

[Rebaï et al 2008] Rebaï I., de La Passardière B., Labat J.-M., To Store and Retrieve Software Components for Interactive Learning Environments: the ECR Repository, *International Journal of Advanced Media and Communication* , vol. 2(1), pp.73-95, 2008.

[Reiser et al. 1985] Reiser B. J., Anderson J. R., Farrell R. G., Dynamic student modelling in an intelligent tutor for LISP programming, *In Proceedings of IJCAI*, pp. 8-14, 1985.

[Ritter et al. 1998] Ritter S., Anderson J. R., Cytrynowicz M., Medvedeva O, Authoring Content in the PAT Algebra Tutor, *Journal of Interactive Media in Education*, vol. 9, 1998.

[Ritter et al. 2007] Ritter S., Anderson J. R., Koedinger K. R., Corbett A., Cognitive Tutor: Applied research in mathematics education, *Psychonomic Bulletin & Review, 14*, pp. 249-255, 2007.

[Rogalski 2005] Rogalski J., Rapport d'activité sur l'axe instrumentation de l'activité des enseignants, Modélisation et mise en œuvre d'environnements informatiques pour la régulation de l'apprentissage, le cas de l'algèbre avec le projet LINGOT, *Projet Cognitique 2002, École et sciences cognitives: Les apprentissages et leurs dysfonctionnements, rapport de fin de projet*, pp. 91-107, mars 2005.

[Roll et al. 2005] Roll, I., Baker, R.S., Aleven, V., McLaren, B.M., & Koedinger, K.R. (2005) Modeling Students' Metacognitive Errors in Two Intelligent Tutoring Systems, *Proceedings of User Modeling* Springer-Verlag, Berlin, pp. 379-388, 2005.

[Rose et al. 2003] Rose C. P., Gaydos A., Hall B. S., Roque A., VanLehn K., Overcomming the Knowledge Engineering Bottleneck for Understanding Student Language Input, *Proceedings of Artificial Intelligence in Education*, 2003.

[Shapiro 2005] Shapiro, J.A. An Algebra Subsystem for Diagnosing Students' Input in a Physics Tutoring System, *International Journal of Artificial Intelligence in Education*, vol. 15, pp. 205-228, 2005.

[Shute 1999] Shute V. J., Torreano L. A., Willis R., E., Exploratory Test of an Automated Knowledge Elicitation and Organization Tool, *International Journal of Artificial Intelligence in Education*, vol. 10, pp. 365-384, 1999.

[Simonin et Carbonell 2006] Simonin J., Carbonell N., Interfaces adaptatives - Adaptation dynamique à l'utilisateur courant, *Interfaces numériques* (ouvrage collectif), Hermès Sciences, chapitre 2, 16 p., 2006.

[Tchounikine 2002] Tchounikine P., Pour une ingénierie des Environnements Informatiques pour l'Apprentissage Humain», *Revue I3 information – interaction – intelligence*, vol. 2 n° 1, pp. 59-95, 2002.

[Tchounikine et al. 2004] Tchounikine P., Baker M, Balacheff N, Baron M, Derycke A, Guin D, Nicaud J-F, Rabardel P, Platon-1 : quelques dimensions pour l'analyse des travaux de recherche en conception d'EIAH, *Rapport de l' Action Spécifique 51 du RTP39 intitulée « Fondements théoriques et méthodologiques de la conception des EIAH »*, Département STIC du CNRS, 2004.

[Tricot et al. 2003] Tricot A., Plegat-Soutjis F., Camps J.-F., Amiel A., Lutz G., Morcillo A., Utilité, utilisabilité, acceptabilité : interpréter les relations entre trois dimensions de l'évaluation des EIAH, *Actes de la conférence EIAH 2003*, C. Desmoulins, P. Marquet, D. Bouhineau (Eds.), pp. 391-402, avril 2003.

[Van Duyne 2003] Van Duyne D. K., Landay J., Hong J., The design of sites : Patterns, Principles and process for crafting a Customer Centered Web experience, Addison-Wesley, 2003.

[Vandebrouck et Cazes, 2005] Vandebrouck, F., Cazes C., Analyse de fichiers de traces d'étudiants : aspects didactiques, *Revue STICEF*, vol. 12, pp. 229-267, 2005.

[VanLehn et Martin 1997] VanLehn K., Martin J., Evaluation of an assessment system based on Bayesian student modelling, *International Journal of Artificial Intelligence in Education*, 8, pp. 179-221, 1997.

[VanLehn et Niu 2001] VanLehn K., Niu Z., Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. *International Journal of Artificial Intelligence in Education*, 12(2), pp. 154-184, 2001.

[VanLehn et al. 2005a] VanLehn K., Lynch C., Schulze K., Shapiro J.A., Shelby R., Taylor L., Treacy D., Weinstein A. Wintersgill M., The Andes physics tutoring system: Lessons learned, *International Journal of Artificial Intelligence and Education*, vol. 15, pp. 147-204, 2005.

[VanLehn et al. 2005b] VanLehn K., Lynch C., Schulze K., Shapiro J. A., Shelby R. H., Taylor L., Treacy D. J., Weinstein A., Wintersgill M. C. The Andes physics tutoring system: Five years of evaluations, *Proceedings of the Artificial Intelligence in Education Conference*, McCalla G. I., Looi C.-K., (Eds.), Amsterdam, 2005.

[VanLehn et al. 2006] VanLehn, K., Graesser, A., Jackson, G., Jordan, P., Olney, A. When are tutorial dialogues more effective than reading?, *Cognitive Science*, pp. 1-60, 2006.

[VanLehn et al. 2007] VanLehn, K., Koedinger, K., Skogsholm, A., Nwaigwe, A., Hausmann, R. What's in a Step? Toward General, Abstract Representations of Tutoring System Log Data. *In User Modeling 2007; In Book Series: Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, pp. 455-459, 2007.

[Vaseux 2003] Vaseux M., Réalisation d'une application client-serveur, multiplateforme et multilingue, PépiJava, Mémoire d'Ingénieur-Maître, IUP-MIME, Université du Maine, Juin 2003.

[Vergnaud 1991]. La théorie des champs conceptuels, *Recherches en Didactique des Mathématiques*, vol. 10/2.3, pp. 133-170, 1991.

[Vincent et al. 2005] Vincent C., Delozanne E., Grugeon B., Gélis J.-M., Rogalski J., Coulange L., Des erreurs aux stéréotypes : Des modèles cognitifs de différents niveaux dans le projet Pépite, *Actes de la conférence EIAH2005*, Montpellier, pp. 297-308, mai 2005.

[Virvou et Moundridou 2000] Virvou M., Moundridou M., A Web-based authoring tool for Algebra-related ITSs, *Journal of International Forum of Educational Technology & Society and IEEE Learning Technology Task Force*, vol. 3, Issue 2, pp. 61-70, 2000.

[Virvou et Moundridou 2000] Virvou M., Moundridou M., Modelling the Instructor in a Web-Based Authoring Tool for Algebra-Related ITSs, *Lecture Notes in Computer Science*, G. Gauthier, C. Frasson, K. Van Lehn (Eds.), "Intelligent Tutoring Systems", ISBN 978-3-540-67655-3, Springer, Berlin, vol. 1839, pp. 635-644, 2000.

[Vivet 1984] Vivet M., Expertise mathématique et informatique : CAMELIA, un logiciel pour raisonner et calculer, Thèse d'état, Université de Paris VI, 1984.

[Waymel 2005] Waymel E., Fouille de données en EIAH : une étude de cas, Mémoire de stage de recherche de Master 2, UFR de Mathématiques et d'informatique de l'université Paris 5, 2005.

[Wenger 1987] Wenger E., *Artificial Intelligence and Tutoring Systems – Computational and Cognitive Approaches to the Communication of Knowledge*, Morgan Kaufmann (Eds.), Los Altos, 486 p., 1987.

[Zapata-Rivera et al.2007] Zapata-Rivera D., Hansen E., Shute V., Underwood J. and Bauer M., Evidence-based Approach to Interacting with Open Student Models, *International Journal of Artificial Intelligence in Education*, vol. 17, pp. 273-303, 2007.

2. Références sur le WEB.

[AIED 2005] http://hcs.science.uva.nl/AIED2005/download.html (consulté en 2007)

[Anderson et al.1995] http://act-r.psy.cmu.edu/papers/Lessons_Learned.html (consulté en 2007)

[Andes] http://www.andes.pitt.edu/ (Consulté en 2008)

[Aplusix] http://aplusix.imag.fr/ (Consulté en 2008)

[Assisment] http://www.assistment.org/ (Consulté en 2008)

[B.O.] http://www.education.gouv.fr/bo/ (consulté en 2007)

[Carnegie Learning] http://www.carnegielearning.com/products.cfm (Consulté en 2008)

[Carton 2007] Carton O., Langages formels, Calculabilité et Complexité, Support de cours : Formation prédoctorale Ecole Normale Supérieur, 2007.

http://www.liafa.jussieu.fr/~carton/Enseignement/Complexite/ENS/support/ (consulté en 2007)

[Cognitive Tutor Authoring Tools] http://ctat.pact.cs.cmu.edu/ (Consulté en 208)

[Gorissen 2006] Gorissen P., Quickscan QTI – 2006, Usability study of QTI for The Digitale Universiteit, 52 p., 2006.

http://www.gorissen.info/Pierre/QTI/Quickscan_QTI_2006.pdf (consulté en 2008)

[IEEE 2002] Learning Objects Metadata Working Group, IEEE 1484.12.1-2002, Draft Standard for Learning Object Metadata, 2002

http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (Consulté en 2008)

[JESS 1997] Friedman-Hill E.F., Jess, The Java Expert System Shell.

http://herzberg.ca.sandia.gov/jess (consulté en 2007)

[GRICEA] http://www.gricea.umontreal.ca/primaire/maths/ (Consulté en 2008)

[IJAIED] International Journal of Artificial Intelligence and Education. http://cbl.leeds.ac.uk/ijaied/

[IMS-LD 2003] IMS Learning Design Information Model.

http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html (consulté en 2007)

[IMS-QTI 2006] IMS Question and Test Interoperability Overview.

http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html (consulté en 2008)

[Jaxe 2007] http://jaxe.sourceforge.net/Jaxe.html (consulté en 2007)

[Jouannaud 2007] Jouannaud J.-P., Informatique théorique : Théorie des langages, Analyse lexicale, Analyse syntaxique, Support de cours. (consulté en 2007)

http://www.lix.polytechnique.fr/~jouannaud/articles/cours-info-theo.pdf (consulté en 2007)

[JTLA] Journal of Technology, Learning, and Assessment. http://www.jtla.org. (consulté en 2008)

[Koper 2001] http://eml.ou.nl/introduction/docs/ped-metamodel.pdf (consulté en 2007)

[Pépite] http://pepite.univ-lemans.fr (Consulté en 2008)

[PISA 2006] http://www.oecd.org/dataoecd/10/45/39777163.pdf (Consulté en 2008)

[RCAT] La Revue Canadienne de l'Apprentissage et de la Technologie. http://www.cjlt.ca (Consulté en 2008)

[STICEF] Sciences et technologies de l'information et de la communiocation pour l'éducation et la formation. http://sticef.org (consulté en 2008)

[Archives EIAH] http://archiveseiah.univ-lemans.fr (consulté en 2007)

Annexes

Annexes A

Modèle conceptuel

A.1. Dimensions et critères d'évaluation locaux aux réponses d'un exercice 293
A.2. Liste des capacités par composantes de la compétence algébrique..... 294
A.3. Description des classes d'exercices 295

A.1. Dimensions et critères d'évaluation locaux aux réponses d'un exercice

(Novembre 2007)

Validité

V0	Absence de réponse	V3	Traitement incorrect
V1	Traitement correct	V ?	Traitement non identifié
V2	Traitement correct partiel ou non attendu		

Statut du signe Égal

- E1 Relation d'équivalence
- E2 Annonce de résultat
- E3 Abréviation (pour succession enchaînée d'égalités)

Utilisation des Lettres

- L1 Utilisation correcte des lettres
- L2 Utilisation des lettres pour leur substituer des valeurs numériques
- L3 Utilisation des lettres pour faire du calcul algébrique avec des règles fausses
- L4 Utilisation des lettres comme étiquettes ou abréviations
- L5 Aucune utilisation des lettres
- L ? Utilisation des lettres non identifiée

Utilisation des règles d'Écriture et de réécriture Algébrique

- EA1 Utilisation correcte des règles de transformation
- EA2 Maîtrise technique fragile
- EA3 Règles de transformation non maîtrisées, mais identification correcte du rôle des opérateurs + et ×
  - EA31 Utilisation inadaptée des parenthèses qui conduit à un résultat correct
  - EA32 Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect
  - EA33 Utilisation de règles de transformation fausses identifiées
  - EA34 Erreurs de signe en cours de calcul
- EA4 Identification incorrecte du rôle des opérateurs + et x
  - EA41 Les règles de transformation utilisées linéarisent les expressions
  - EA42 Les règles de transformation utilisées « assemblent » les termes
- EA5 Écriture sans repérage des blocs
- EA ? Écriture non identifié

Traduction

T1	Traduction correcte	T2	Traduction correcte non attendue
T3	Traduction incorrecte (s'appuyant sur des erreurs récurrentes)		
T4	Traduction abréviative		
T ?	Traduction non identifiée		

Type de Justification

- J0 Pas de justification
- J1 Justification par l'algèbre
- J2 Justification par l'exemple numérique
- J3 Justification de type scolaire
  - J31 Justification reposant sur l'application de règles incorrectes
  - J32 Justification en langage naturel par argumentation
  - J33 Justification s'appuyant sur des formulations d'ordre légal
- J ? Justification non identifiée

Utilisation des règles d'Écriture et de réécriture Numérique

- EN1 Utilisation correcte des règles de transformation
- EN2 Maîtrise technique fragile
- EN3 Règles de transformation non maîtrisées, mais identification correcte du rôle des opérateurs + et ×
  - EN31 Utilisation inadaptée des parenthèses qui conduit à un résultat correct
  - EN32 Utilisation inadaptée des parenthèses qui conduit à un résultat incorrect
  - EN33 Utilisation de règles de transformation fausses identifiées
  - EN34 Erreurs de signe en cours de calcul
- EN4 Identification incorrecte du rôle des opérateurs + et x
  - EN41 Les règles de transformation utilisées linéarisent les expressions
  - EN42 Les règles de transformation utilisées « assemblent » les termes
- EN5 Calcul sans repérage des blocs
- EN ? Calcul non identifié

Connaissances Numériques

N1	Ordre des nombres décimaux	N4	Mobilisation des nombres décimaux
N11	Utilisation correcte de l'ordre	N41	Mobilisation correcte
N12	Utilisation incorrecte de l'ordre	N42	Mobilisation incorrecte
N2	Ordre des nombres négatifs	N5	Mobilisation des nombres rationnels
N21	Utilisation correcte de l'ordre	N51	Mobilisation correcte
N22	Utilisation incorrecte de l'ordre	N52	Mobilisation incorrecte
N3	Mobilisation des nombres relatifs	N6	Mobilisation des nombres irrationnels
N31	Mobilisation correcte	N61	Mobilisation correcte
N32	Mobilisation incorrecte	N62	Mobilisation incorrecte

A.2. Liste des capacités par composantes de la compétence algébrique

Composante : « Effectuer du calcul algébrique » (CA)

Capacités 3ème [d'après le B.O. 2007]

1. Calculer la valeur d'une expression littérale en donnant aux variables des valeurs numériques (entier, décimal, fraction, racine carrée)
2. Tester si une égalité comportant un ou deux nombres indéterminés est vraie lorsqu'on leur attribue des valeurs numériques
3. Réduire une expression littérale à une variable
4. Développer une expression littérale de la forme $(a+b)(c+d)$
5. Factoriser des expressions dans lequel le facteur est apparent, de la forme $(ax+b)(cx+d) + e\,(ax+b)$, $(ax+b)^2 + (ax+b)\,(cx+d)$, a, b, c, d, e et f étant des coefficients numériques (entier, fraction, racine carrée)
6. Connaître les identités remarquables

   $(a-b)(a+b) = a^2-b^2$

   $(a+b)^2 = a^2 + 2ab + b^2$

   $(a-b)^2 = a^2 - 2ab + b^2$
7. Utiliser les identités remarquables dans les deux sens, pour factoriser ou développer, sur des expressions numériques ou littérales simples telles que 101^2 ou $(ax+b)^2 - c^2$, a, b, c étant des coefficients numériques (entier, fraction, racine carrée)
8. Transformer une égalité en une égalité équivalente
9. Reconnaître la structure d'une expression algébrique (somme, produit, carré, différence de deux carrés)
10. Reconnaître différentes écritures d'une même expression et choisir la forme la plus adaptée au travail demandé (forme réduite, factorisée , ..)
11. Résoudre une équation du premier degré (se ramenant à $ax+b = cx+d$) à une inconnue à cœfficients numériques
12. Résoudre une inéquation du premier degré à une inconnue à cœfficients numériques.
13. Résoudre algébriquement un système de deux équations du premier degré à deux inconnues admettant une solution et une seule
14. Résoudre une équation se ramenant à la forme A.B=0, où A et B désignent deux expressions du premier degré de la même variable
15. Déterminer l'expression algébrique d'une fonction linéaire à partir de la donnée d'un nombre non nul et de son image
16. Déterminer l'expression algébrique d'une fonction affine à partir de la donnée de deux nombres et de leurs images

Composante : « Utiliser l'Aalgèbre pour résoudre des problèmes » (UA)

Capacités 3ème [d'après le B.O. 2007]

1. Produire une expression littérale, une formule dans un cadre donné
2. Mettre en équation et résoudre un problème conduisant à une équation ou une inéquation du premier degré à une inconnue, ou un système de deux équations du premier degré
3. Prouver des propriétés sur les nombres entiers, par exemple des propriétés de divisibilité
4. Démontrer des règles de calcul, des propriétés, des identités
5. Exprimer une variable en fonction d'une autre variable dans une formule

Composante : « Traduire d'une représentation dans une autre (algèbre, géométrie, graphique, langage naturel) » (TA)

Capacités 3ème [d'après le B.O. 2007]

1. Traduire une expression algébrique comme le résultat d'un programme de calcul (somme, produit, ..) et vice-versa
2. Traduire une expression algébrique comme un programme de calcul et vice-versa
3. Traduire une expression algébrique comme une longueur de segment, une aire de surface, … et vice-versa
4. Représenter graphiquement une fonction linéaire
5. Représenter graphiquement une fonction affine
6. Lire sur la représentation graphique d'une fonction linéaire l'image d'un nombre donné et l'antécédent d'un nombre donné
7. Lire sur la représentation graphique d'une fonction affine l'image d'un nombre donné et l'antécédent d'un nombre donné
8. Représenter les solutions d'une inéquation du premier degré à une inconnue à cœfficients numériques
9. Donner une interprétation graphique de la solution unique d'un système de deux équations du premier degré à deux inconnues
10. Lire graphiquement le coefficient directeur d'une droite
11. Reconnaître la fonction linéaire dont la courbe représentative est une droite donnée et vice-versa
12. Reconnaître la fonction affine dont la courbe représentative est une droite donnée et vice-versa

A.3. Description des classes d'exercices

CA pour la composante « Effectuer du calcul algébrique »

UA pour la composante « Utiliser l'algèbre pour résoudre des problèmes »

TA pour la composante « Traduire d'une représentation dans une autre »

Le codage est celui qui est spécifié dans l'annexe A1

N pour représenter le numéro de l'exercice du logiciel Pépite, **C** pour représenter les composantes

N	Description	C	Capacités	Codage
1	Reconnaître des égalités numériques vraies	CA	9 Reconnaître la structure d'une expression algébrique	V, EA1 EA31,EA33 EA42
2	Déterminer si une égalité littérale est toujours vérifiée	CA UA	2 Tester si une égalité comportant un ou deux nombres indéterminés est vraie lorsqu'on leur attribue des valeurs numériques 9 Reconnaître la structure d'une expression algébrique 4 Démontrer des règles de calcul, des propriétés	V EA1, EA33 EA41, EA42 L1, L2, L4 J1, J2, J3
3	Associer aire d'un rectangle et expression littérale partie 1 : produire partie 2 : reconnaître	UA TA	1 Produire une expression littérale, une formule dans un cadre donné 3 Traduire une expression algébrique comme une aire d'une surface	V EA1, EA31 EA33, EA41 EA42 T1, T3, T4
4	Déterminer si une égalité littérale est toujours vérifiée	CA UA	2 Tester si une égalité comportant un ou deux nombres indéterminés est vraie lorsqu'on leur attribue des valeurs numériques 9 Reconnaître la structure d'une expression algébrique 4 Démontrer des règles de calcul, des propriétés, des identités	V EA1,EA32, EA33, EA41 EA42 L1, L2, L4 J1, J2, J3
5	Reconnaître des sommes et des produits	TA	1 Traduire une expression algébrique en langue naturelle et vice-versa	V, EA4
6	Déterminer si des expressions algébriques du second degré sont égales	CA	4 Développer une expression littérale de la forme (a+b)(c+d) 7 Utiliser les identités remarquables dans les deux sens	V, EA1 EA2, EA31 EA32, EA33 EA41, EA42
7	Associer droites et fonctions affines	TA	12 Reconnaître la fonction affine dont la courbe représentative est une droite donnée et vice-versa	V, T1, T3, T4
8	Calculer pour un nombre donné, la valeur particulière d'une expression du second degré en sélectionnant l'écriture la plus adaptée	CA	1 Calculer la valeur d'une expression littérale en donnant aux variables des valeurs numériques 3 Réduire une expression littérale à une variable 4 Développer une expression littérale de la forme (a+b)(c+d) 7 Utiliser les identités remarquables dans les deux sens pour factoriser ou développer 10 Reconnaître différentes écritures d'une même expression et choisir la forme la plus adaptée au travail demandé (forme factorisée, forme réduite)	V EA1, EA2 EA31, EA32 EA33, EA41
9	Développer et factoriser une expression du second degré Vérifier qu'un nombre est solution d'une équation	CA	4 Développer une expression littérale de la forme (a+b)(c+d) 5 Factoriser des expressions dans lequel le facteur est apparent 7 Utiliser les identités remarquables dans les deux sens 2 Tester si une égalité comportant un ou deux nombres indéterminés est vraie lorsqu'on leur attribue des valeurs numériques	question 1 V, EA1, EA32, EA33, EA41 question 2 V,EA1, EA31, EA33 EA41, EA42
10	Résoudre un problème linéaire de deux équations à deux inconnues	UA	2 Mettre en équation et résoudre un problème conduisant à un système de deux équations du premier degré	V,T1, T2 T4, L1, L4 EA1, EA31, EA32 EA33 EA41

11	Déterminer des expressions d'un programme de calcul	TA	2 Traduire une expression algébrique comme un programme de calcul et vice-versa	question 1 V, T1, T3, EA1, EA31, EA32, EA33, EA41 question 2 V, T1, T3, EA33, EA41
12	Exprimer la longueur d'un segment	UA TA	1 Produire une expression littérale 3 Traduire une expression algébrique comme une longueur de segment	V, T1, T3
13	Associer aire et expression	TA	3 Traduire une expression algébrique comme une aire d'une surface	V, T1, T3
14	Traduire algébriquement une relation entre deux variables contraintes	TA	1 Traduire une expression algébrique en langue naturelle et vice-versa	V L1, L3, L4 T1, T3, T4
15	Aires de figure composées et équations Partie 1 : produire Partie 2 : produire Partie 3 : mettre en équation et résoudre	UA TA CA UA	1 Produire une expression littérale, une formule dans un cadre donné 3 Traduire une expression algébrique comme une aire d'une surface et vice-versa 11 Résoudre une équation du premier degré à une inconnue à cœfficients numériques 2 Mettre en équation et résoudre un problème conduisant à une équation	question 1 V, T1, T3, T4 question 2 V, T1, T3, T4 question 3 V, L1, L2, L3, L4, L5 EA1, EA31, EA32, EA33, EA41, EA42
16	Preuve et programme de calcul	CA UA TA	3 Réduire une expression littérale à une variable 1 Produire une expression littérale 4 Démontrer des règles de calcul, des propriétés, des identités 1 Traduire un programme en langage naturel en une expression algébrique et	V T1, T2, T3, T4 L1, L2, L3, L5 EA1, EA2, EA31, EA32 EA33, EA42 J1, J2, J3
			vice-versa	
17	Déterminer l'appartenance d'un point à une droite	CA	2 Tester si une égalité comportant un ou deux nombres indéterminés est vraie lorsqu'on leur attribue des valeurs numériques	V
18	Reconnaître la fonction affine dont la courbe représentative est une droite donnée	TA	10 Lire graphiquement le coefficient directeur d'une droite 12 Reconnaître la fonction affine dont la courbe représentative est une droite donnée	V T1, T2, T3
19	Inverser des fonctions affines et linéaires	CA	1 Calculer la valeur d'une expression littérale en donnant aux variables des valeurs numériques 8 Transformer une égalité en une égalité équivalente	question 1 V, L1, L3, EA1 EA31 EA32 EA33, EA42 question 2 V, T1, L1, L3, EA31 EA32 EA33 EA42
20	Résoudre des problèmes affines et linéaires dans différents cadres Partie 1 : lire Partie 2 : produire Partie 3 : résoudre une équation	TA TA CA UA	1 Traduire un texte en langage naturel en une expression algébrique et vice-versa 7 Lire sur la représentation graphique d'une fonction affine l'image d'un nombre donné et l'antécédent d'un nombre donné 1 Traduire un texte en langage naturel en une expression algébrique et vice-versa 11 Résoudre une équation du premier degré à une inconnue à cœfficients numériques 2 Mettre en équation et résoudre un problème conduisant à une équation	question 1 V, T1 question 2 et 3 V, T1, T3, L1, L3 L4 , L5 V T1, L1, L3, EA1, EA
21	Résoudre un système d'équations linéaires	CA	13 Résoudre algébriquement un système de deux équations du premier degré à deux inconnues	V, EA1, EA31, EA32, EA33 EA41
22	Classer des nombres décimaux	CA		N11, N12

Annexes B

Fichiers de tests Pépinière

B.1. Interface de tests de Pépinière 299
B.1.1. Réduction de l'expression (*x*+6)3 -3*x* 299
B.1.2. Réduction de l'expression (4*x*+6)/2-2*x* 299
B.1.3. Réduction de l'expression ((*x*+8)*3-4+*x*)/4+2-*x* 300
B.1.4. Réduction de l'expression ((3*x*-6)*2-3*x*)/3-*x* 300

B.2. Arbre des solutions anticipées pour la réduction de l'expression (*x*+6)*3-3*x* 301
B.2.1. Représentation « fils-aîné-frère-droit » de l'arbre des solutions anticipées 301
B.2.2. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (*x*+6)*3-3*x* 301

B.3. Arbre des solutions anticipées pour la réduction de l'expression (4*x*+6)/2-2*x* 302
B.3.1. Représentation « fils-aîné-frère-droit » de l'arbre des solutions anticipées 302
B.3.2. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (4*x*+6)/2-2*x* 303

B.4. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression ((3*x*-6)*2-3*x*)/3-*x* (exercice créé par une enseignante) 305

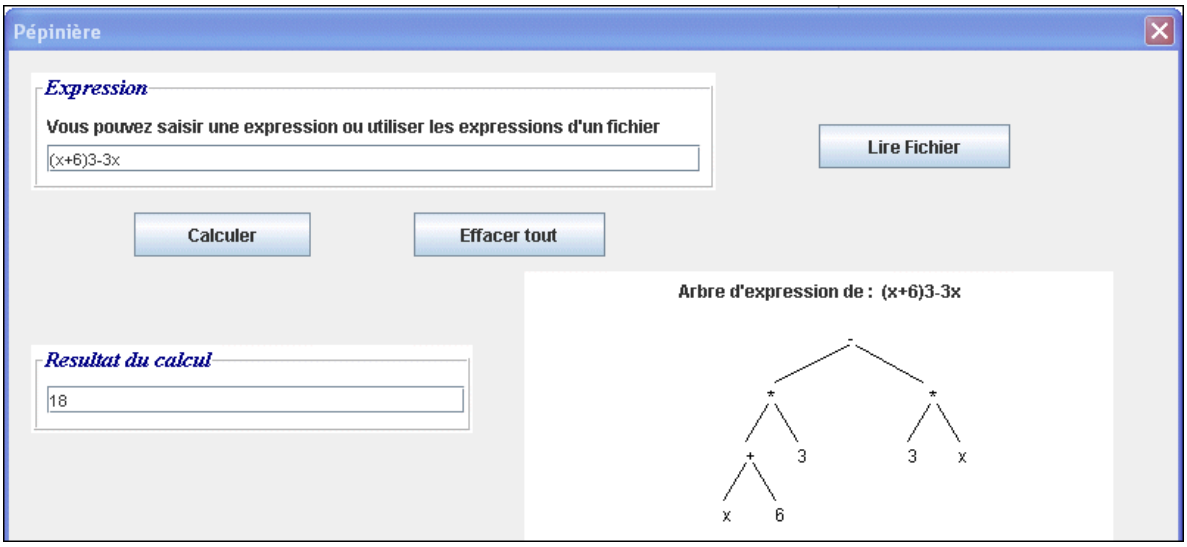
B.5. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (*x*+3x*+4)/4-1 (exercice créé par un enseignant) 306**

B.6. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression ((*x*+8)*3-4+*x*)/4+2-*x* (Pépite originel) 308

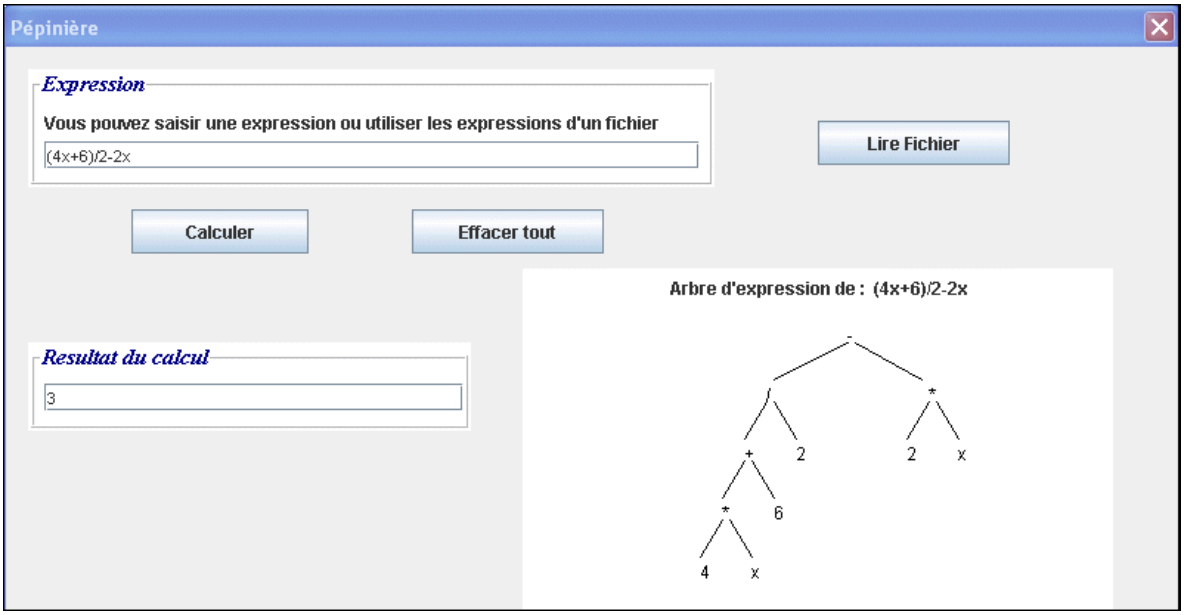
B.7. Transformation de deux arbres en arbres équivalents 316
B.7.1. Exemple d'un exercice de la classe « Preuve et programme de calcul » 316
B.7.2. Exemple d'un exercice de la classe « Déterminer si des expressions algébriques du second degré sont égales » 318

B.1. Interface de tests de Pépinière

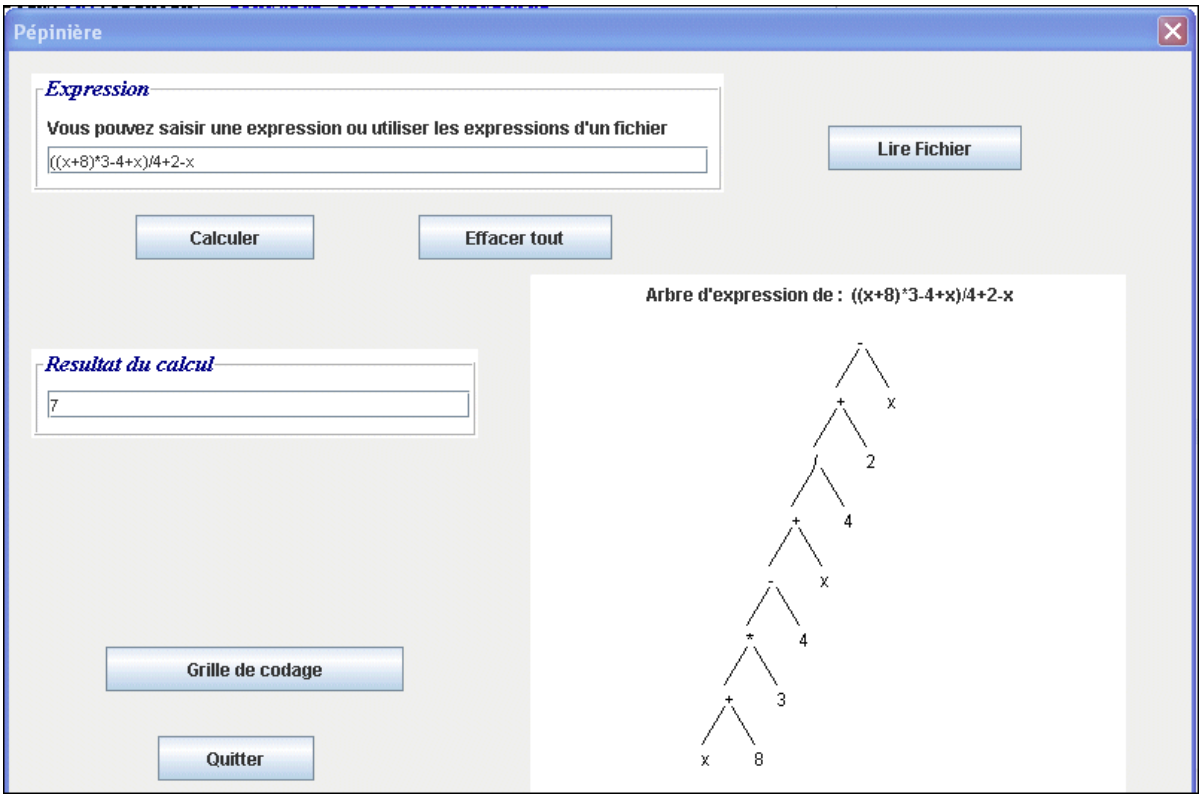
B.1.1. Réduction de l'expression (*x*+6)3 -3*x*



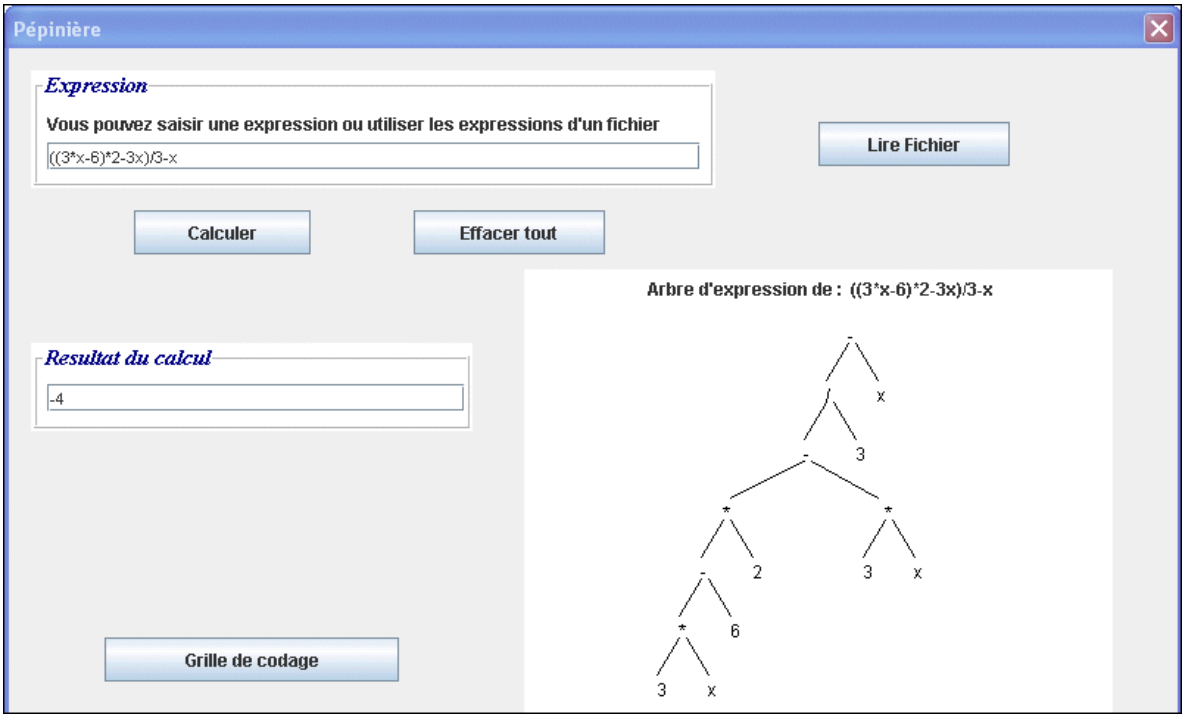
B.1.2. Réduction de l'expression (4*x*+6)/2-2*x*



B.1.3. Réduction de l'expression ((*x*+8)*3-4+*x*)/4+2-*x*

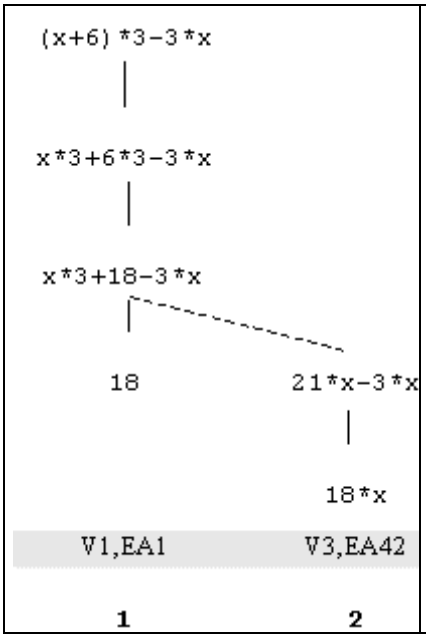


B.1.4. Réduction de l'expression ((3*x*-6)*2-3*x*)/3-*x*



B.2. Arbre des solutions anticipées pour la réduction de l'expression (*x*+6)*3-3*x*

B.2.1. Représentation « fils-aîné-frère-droit » de l'arbre des solutions anticipées



Légende

La figure représente les deux branches de l'arbre des solutions anticipées lorsque l'expression (*x*+6)*3-3*x* est développée, puis réduite.

———— Règles correctes

---------- Règles erronées

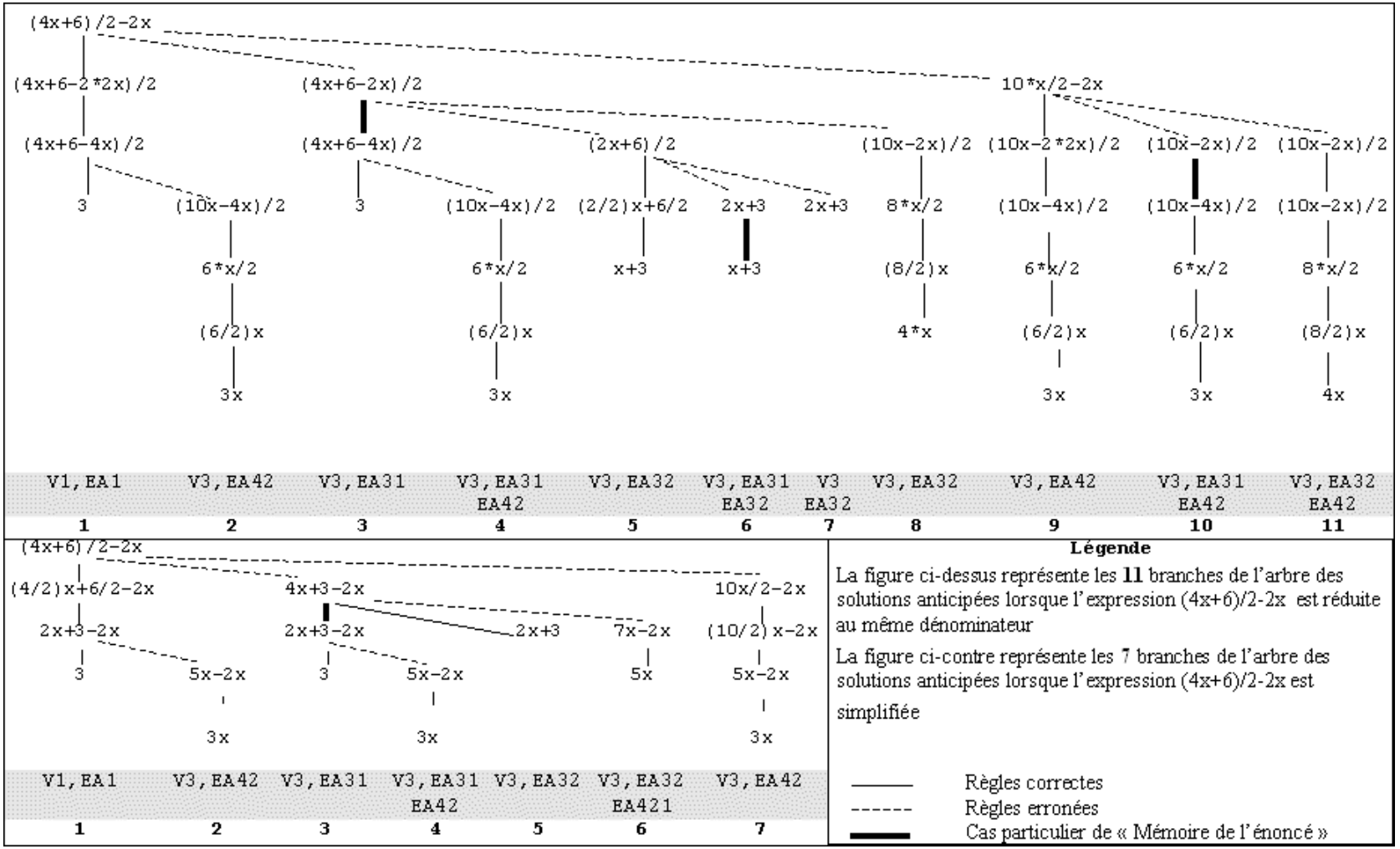
B.2.2. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (*x*+6)*3-3*x*

```
-----------------------------------------------------------------
Arbre des solutions anticipées - Parcours «fils-aîné-frère-droit»
-----------------------------------------------------------------
 Expression          Règle             Type de règle       Codage
-----------------------------------------------------------------
 (x+6)*3-3*x
 x*3+6*3-3*x           (A+B)C ->AC+BC      Règle correcte V1,EA1
 x*3+18-3*x
 18                    AC+BC ->(A+B)C      Règle correcte V1,EA1
-----------------------------------------------------------------
 (x+6)*3-3*x
 x*3+6*3-3*x           (A+B)C ->AC+BC      Règle correcte V1,EA1
 x*3+18-3*x
 21*x-3*x              AC+B ->(A+B)C       Règle erronée  V3,EA42
 18*x                  AC+BC ->(A+B)C      Règle correcte V1,EA1
-----------------------------------------------------------------
```

Une suite d'expressions entre les tirets (`----`) correspond à une branche de l'arbre des solutions construit par Pépinière. Chaque expression est obtenue en appliquant des règles correctes ou erronées, ou un calcul numérique.
Ainsi, pour obtenir le nœud étiqueté par l'expression `x*3+6*3-3*x` à partir du nœud étiqueté par `(x+6)*3-3*x`, la règle appliquée est `(A+B)C --> AC+BC`.
À partir de cet arbre construit par Pépinière, PépiGen mémorise chaque branche (expressions et règles appliquées) et attribue le code associé à la solution représentée par la branche. Le code associé à la branche 1 est V1, EA1 car les règles appliquées sont toutes correctes, le code associé à la deuxième branche est V3, EA42 car une règle de transformation erronée a été appliquée.

B.3. Arbre des solutions anticipées pour la réduction de l'expression (4*x*+6)/2-2*x*

B.3.1. Représentation « fils-aîné-frère-droit » de l'arbre des solutions anticipées



B.3.2. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (4*x*+6)/2-2*x*

```
-------------------------------------------------------------------
Arbre des solutions anticipées - Parcours «fils-aîné-frère-droit»
-------------------------------------------------------------------

 Expression          Règle             Type de règle       Codage
-------------------------------------------------------------------

 (4*x+6)/2-2*x
 (4*x+6-2*2*x)/2        (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (4*x+6-4*x)/2
 3                      AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*2*x)/2        (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (4*x+6-4*x)/2
 (10*x-4*x)/2           AC+B ->(A+B)C       Règle erronée  V3,EA42
 6*x/2                  AC+BC ->(A+B)C      Règle correcte V1,EA1
 (6/2)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 3*x
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*x)/2          (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (4*x+6-4*x)/2      1                       Mémoire énoncé V3,EA31
 3                      AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*x)/2          (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (4*x+6-4*x)/2                              Mémoire énoncé V3,EA31
 (10*x-4*x)/2           AC+B ->(A+B)C       Règle erronée  V3,EA42
 6*x/2                  AC+BC ->(A+B)C      Règle correcte V1,EA1
 (6/2)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 3*x
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*x)/2          (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (2*x+6)/2              AC+BC ->(A+B)C      Règle correcte V1,EA1
 (2/2)*x+6/2            (A+B)/C ->A/C+B/C   Règle correcte V1,EA1
 x+3
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*x)/2          (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (2*x+6)/2              AC+BC ->(A+B)C      Règle correcte V1,EA1
 2*x+3                  (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 x+3                                        Mémoire énoncé V3,EA31
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*x)/2          (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (2*x+6)/2              AC+BC ->(A+B)C      Règle correcte V1,EA1
 2*x+3                  (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4*x+6-2*x)/2          (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (10*x-2*x)/2           AC+B ->(A+B)C       Règle erronée  V3,EA42
 8*x/2                  AC+BC ->(A+B)C      Règle correcte V1,EA1
 (8/2)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 4*x
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 10*x/2-2*x             AC+B ->(A+B)C       Règle erronée  V3,EA42
 (10*x-2*2*x)/2         (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (10*x-4*x)/2
 6*x/2                  AC+BC ->(A+B)C      Règle correcte V1,EA1
 (6/2)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 3*x
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 10*x/2-2*x             AC+B ->(A+B)C       Règle erronée  V3,EA42
 (10*x-2*x)/2           (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (10*x-4*x)/2                               Mémoire énoncé V3,EA31
 6*x/2                  AC+BC ->(A+B)C      Règle correcte V1,EA1
 (6/2)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 3*x
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 10*x/2-2*x             AC+B ->(A+B)C       Règle erronée  V3,EA42
 (10*x-2*x)/2           (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 8*x/2                  AC+BC ->(A+B)C      Règle correcte V1,EA1
 (8/2)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 4*x
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 (4/2)*x+6/2-2*x        (A+B)/C ->A/C+B/C   Règle correcte V1,EA1
 2*x+3-2*x
 3                      AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
```

[1] On remarque que l'élève a fait une erreur de parenthèse (EA32) mais en fait il garde le sens de l'expression ; sa transformation est alors codée EA31 et non EA32

```
 (4/2)*x+6/2-2*x        (A+B)/C ->A/C+B/C   Règle correcte V1,EA1
 2*x+3-2*x
 5*x-2*x                AC+B ->(A+B)C       Règle erronée  V3,EA42
 3*x                    AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 4*x+3-2*x              (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 2*x+3-2*x                                  Mémoire énoncé V3,EA31
 3                      AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 4*x+3-2*x              (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 2*x+3-2*x                                  Mémoire énoncé V3,EA31
 5*x-2*x                AC+B ->(A+B)C       Règle erronée  V3,EA42
 3*x                    AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 4*x+3-2*x              (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 2*x+3                  AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 4*x+3-2*x              (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 7*x-2*x                AC+B ->(A+B)C       Règle erronée  V3,EA42
 5*x                    AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 (4*x+6)/2-2*x
 10*x/2-2*x             AC+B ->(A+B)C       Règle erronée  V3,EA42
 (10/2)*x-2*x           (AB)/C ->(A/C)B     Règle correcte V1,EA1
 5*x-2*x
 3*x                    AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
```

Chaque expression est obtenue en appliquant des règles correctes ou erronées, ou un calcul numérique.
Ainsi, pour obtenir le nœud étiqueté par l'expression `x*3+6*3-3*x` à partir du nœud étiqueté par `(x+6)*3-3*x,` la règle appliquée est : `(A+B)C --> AC+BC`.
Le code associé à cette branche est V1, EA1 car les règles appliquées sont toutes correctes, le code associé à la deuxième branche est V3, EA42 car il y a application d'une règle de transformation erronée qui « assemble les termes » et qui est codées EA42.

B.4. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression ((3*x*-6)*2-3*x*)/3-*x* (exercice créé par une enseignante)

```
-------------------------------------------------------------------
Arbre des solutions anticipées - Parcours «fils-aîné-frère-droit»
-------------------------------------------------------------------

 Expression          Règle             Type de règle       Codage
-------------------------------------------------------------------

 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 (3*x-12-3*x)/3         (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 -4                     AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 (3*x-12-x)/3           (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (3*x-12-3*x)/3                             Mémoire énoncé V3,EA31
 -4                     AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 (3*x-12-x)/3           (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (2*x-12)/3             AC+BC ->(A+B)C      Règle correcte V1,EA1
 (2/3)*x-12/3                               Règle correcte V1,EA1
 (2/3)*x-4
-------------------------------------------------------------------
 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 (3*x-12-x)/3           (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (2*x-12)/3             AC+BC ->(A+B)C      Règle correcte V1,EA1
 2*x-4                  (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
 (2/3)*x-4                                  Mémoire énoncé V3,EA31
-------------------------------------------------------------------
```

```
((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 (3*x-12-x)/3           (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (2*x-12)/3             AC+BC ->(A+B)C      Règle correcte V1,EA1
 2*x-4                  (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
-------------------------------------------------------------------
 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 (3/3)*x-12/3-x                             Règle correcte V1,EA1
 x-4-x
 -4                     AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 3*x-4-x                (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
 x-4-x                                      Mémoire énoncé V3,EA31
 -4                     AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
 ((3*x-6)*2-3*x)/3-x
 (3*x*2-6*2-3*x)/3-x    (A-B)C ->AC-BC      Règle correcte V1,EA1
 (6*x-12-3*x)/3-x
 (3*x-12)/3-x           AC+BC ->(A+B)C      Règle correcte V1,EA1
 3*x-4-x                (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
 2*x-4                  AC+BC ->(A+B)C      Règle correcte V1,EA1
-------------------------------------------------------------------
```

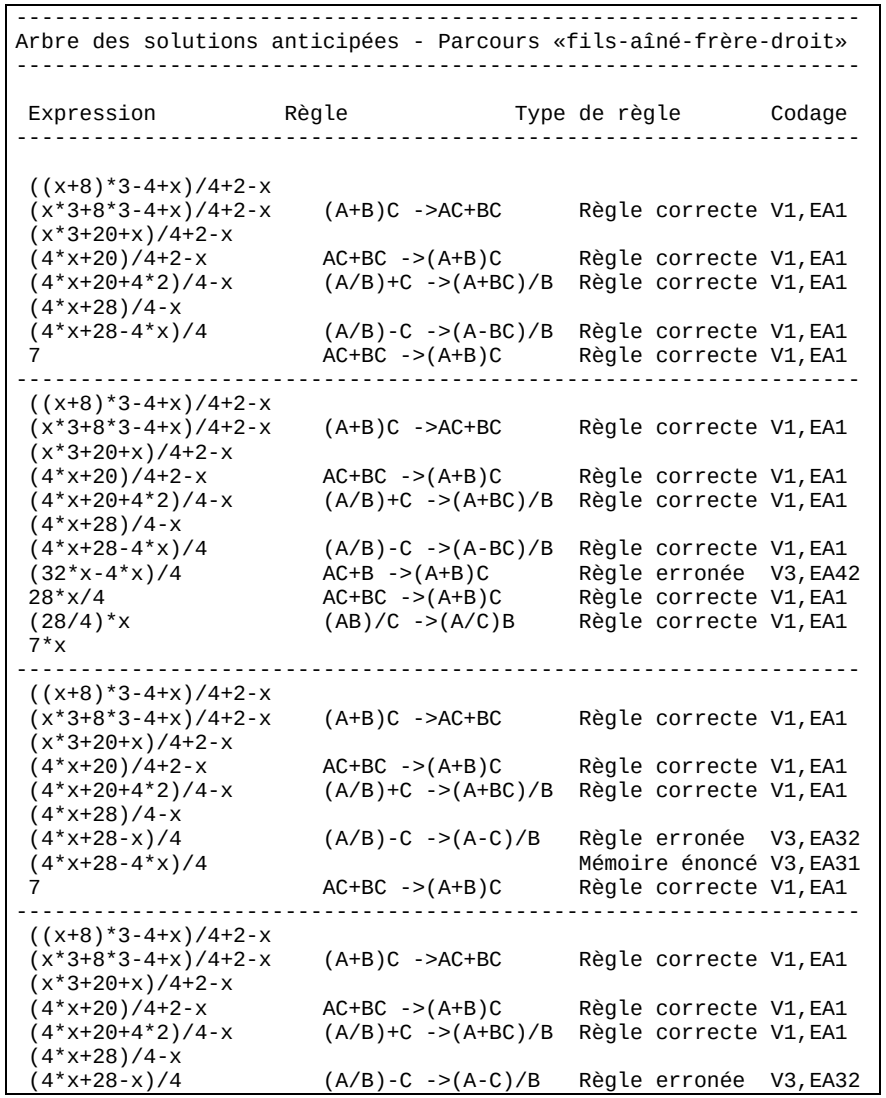
B.5. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression (*x*+3x*+4)/4-1 (exercice créé par un enseignant)**

```
-------------------------------------------------------------------
Arbre des solutions anticipées - Parcours «fils-aîné-frère-droit»
-------------------------------------------------------------------

 Expression          Règle             Type de règle       Codage
-------------------------------------------------------------------

 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+4-4*1)/4          (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (4*x)/4
 (4/4)*x                (A+B)/C ->A/C+B/C   Règle correcte V1,EA1
 x
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+4-4*1)/4          (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (4*x)/4
 4*x                    (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 x                                          Mémoire énoncé V3,EA31
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+4-4*1)/4          (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (4*x)/4
 4*x                    (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+4-4*1)/4          (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 4*x/4                  AC+B ->(A+B)C       Règle erronée  V3,EA42
 (4/4)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 x
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+3)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (4*x)/4                                    Mémoire énoncé V3,EA31
 (4/4)*x                (A+B)/C ->A/C+B/C   Règle correcte V1,EA1
 x
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+3)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (4*x)/4                                    Mémoire énoncé V3,EA31
 4*x                    (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 x                      (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4*x+3)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 4*x/4                  AC+B ->(A+B)C       Règle erronée  V3,EA42
 (4/4)*x                (AB)/C ->(A/C)B     Règle correcte V1,EA1
 x
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-4*1)/4            (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (8*x-4)/4
 (8/4)*x-4/4                                Règle correcte V1,EA1
 2*x-1
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-4*1)/4            (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (8*x-4)/4
 8*x-1                  (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
 2*x-1                                      Mémoire énoncé V3,EA31
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-4*1)/4            (A/B)-C ->(A-BC)/B  Règle correcte V1,EA1
 (8*x-4)/4
 8*x-1                  (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
-------------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-1)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (8*x-4)/4                                  Mémoire énoncé V3,EA31
 (8/4)*x-4/4                                Règle correcte V1,EA1
 2*x-1
-------------------------------------------------------------------
```

```
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-1)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (8*x-4)/4                                  Mémoire énoncé V3,EA31
 8*x-1                  (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
 2*x-1                                      Mémoire énoncé V3,EA31
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-1)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (8*x-4)/4                                  Mémoire énoncé V3,EA31
 8*x-1                  (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-1)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 (8/4)*x-1/4                                Règle correcte V1,EA1
 2*x-1/4
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-1)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 8*x-1/4                (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
 2*x-1/4                                    Mémoire énoncé V3,EA31
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8*x-1)/4              (A/B)-C ->(A-C)/B   Règle erronée  V3,EA32
 8*x-1/4                (A-B)/C ->A-(B/C)   Règle erronée  V3,EA32
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 (4/4)*x+4/4-1          (A+B)/C ->A/C+B/C   Règle correcte V1,EA1
 x+0
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 4*x+0                  (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 x+0                                        Mémoire énoncé V3,EA31
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 4*x+0                  (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
-----------------------------------------------------------------
 (x+3*x+4)/4-1
 (4*x+4)/4-1            AC+BC ->(A+B)C      Règle correcte V1,EA1
 4*x+0                  (A+B)/C ->A+(B/C)   Règle erronée  V3,EA32
 4*x                    AC+B ->(A+B)C       Règle erronée  V3,EA42
-----------------------------------------------------------------
```

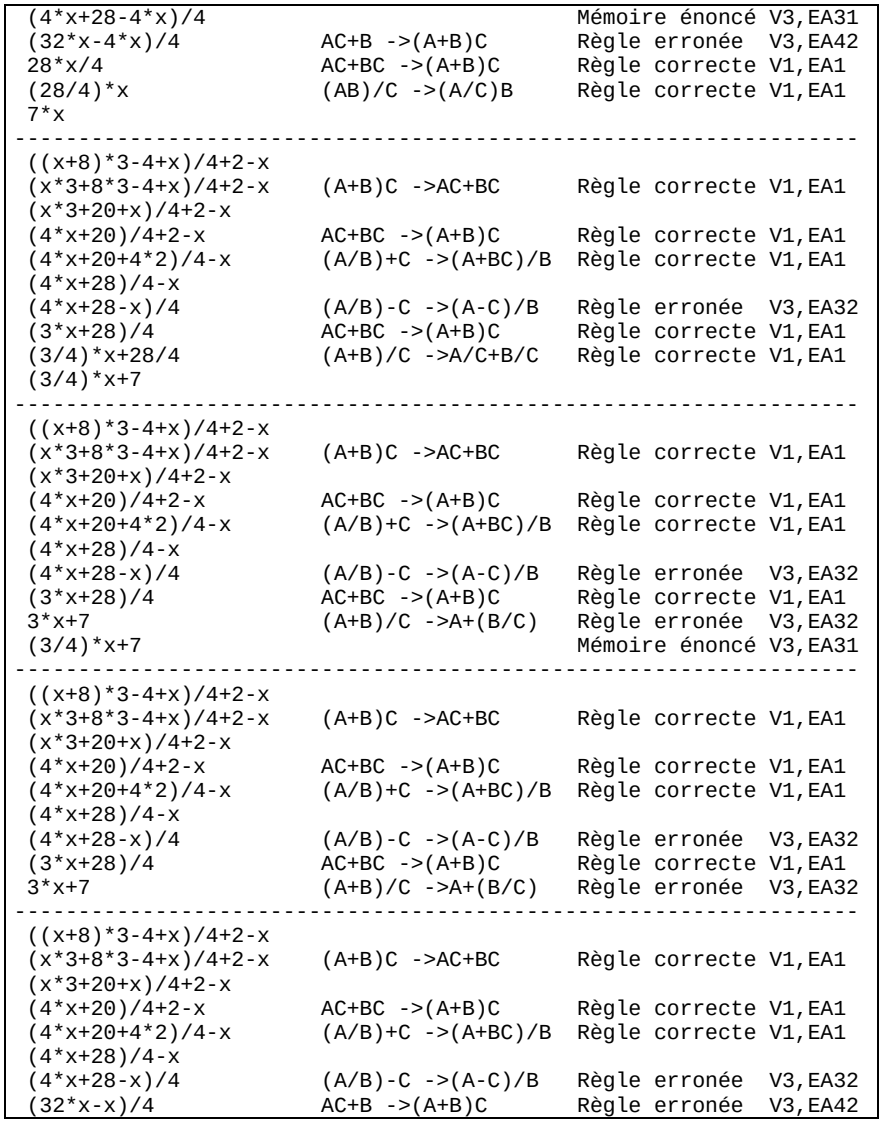
```
 (x+3*x+4)/4-1
 8*x/4-1                AC+B ->(A+B)C       Règle erronée  V3,EA42
 (8/4)*x-1              (AB)/C ->(A/C)B     Règle correcte V1,EA1
 2*x-1
-----------------------------------------------------------------
```

B.6. Arbre des solutions anticipées produit par Pépinière pour la réduction de l'expression ((*x*+8)*3-4+*x*)/4+2-*x* (Pépite originel)



Arbre des solutions anticipées - Parcours «fils-aîné-frère-droit»

Expression	Règle	Type de règle	Codage
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte	V1,EA1
7	AC+BC ->(A+B)C	Règle correcte	V1,EA1
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte	V1,EA1
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée	V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte	V1,EA1
7*x			
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée	V3,EA32
(4*x+28-4*x)/4		Mémoire énoncé	V3,EA31
7	AC+BC ->(A+B)C	Règle correcte	V1,EA1
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée	V3,EA32



(4*x+28-4*x)/4		Mémoire énoncé	V3,EA31
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée	V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte	V1,EA1
7*x			
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée	V3,EA32
(3*x+28)/4	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(3/4)*x+28/4	(A+B)/C ->A/C+B/C	Règle correcte	V1,EA1
(3/4)*x+7			
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée	V3,EA32
(3*x+28)/4	AC+BC ->(A+B)C	Règle correcte	V1,EA1
3*x+7	(A+B)/C ->A+(B/C)	Règle erronée	V3,EA32
(3/4)*x+7		Mémoire énoncé	V3,EA31
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée	V3,EA32
(3*x+28)/4	AC+BC ->(A+B)C	Règle correcte	V1,EA1
3*x+7	(A+B)/C ->A+(B/C)	Règle erronée	V3,EA32
((x+8)*3-4+x)/4+2-x			
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte	V1,EA1
(x*3+20+x)/4+2-x			
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte	V1,EA1
(4*x+20+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte	V1,EA1
(4*x+28)/4-x			
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée	V3,EA32
(32*x-x)/4	AC+B ->(A+B)C	Règle erronée	V3,EA42

31*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(31/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
7	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
7*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(4*x+28-4*x)/4		Mémoire énoncé V3,EA31
7	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(4*x+28-4*x)/4		Mémoire énoncé V3,EA31
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
7*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1

(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(3*x+28)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(3/4)*x+28/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
(3/4)*x+7		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(3*x+28)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
3*x+7	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
(3/4)*x+7		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(3*x+28)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
3*x+7	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+28)/4-x		Mémoire énoncé V3,EA31
(4*x+28-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(32*x-x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
31*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(31/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
11/2	AC+BC ->(A+B)C	Règle correcte V1,EA1

((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(26*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
22*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(22/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
(11/2)*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(4*x+22-4*x)/4		Mémoire énoncé V3,EA31
11/2	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(4*x+22-4*x)/4		Mémoire énoncé V3,EA31
(26*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
22*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(22/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
(11/2)*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(3*x+22)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(3/4)*x+22/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
(3/4)*x+11/2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(3*x+22)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
3*x+11/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
(3/4)*x+11/2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(3*x+22)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
3*x+11/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4*x+22)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(4*x+22-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(26*x-x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
25*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(25/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(20/4)*x+8/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
5*x+2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
5*x+2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		

(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
7*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(20/4)*x+8/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
5*x+2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
5*x+2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1

(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
7*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(23/4)*x+8/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
(23/4)*x+2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
23*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
(23/4)*x+2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		

(x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
23*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+4*2)/4-x	(A/B)+C ->(A+BC)/B	Règle correcte V1,EA1
(24*x+8)/4-x		
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(32*x-x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
31*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(31/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(20/4)*x+8/4 5*x+2	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
5*x+2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1

(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(28/4)*x 7*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(20/4)*x+8/4 5*x+2	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
5*x+2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1

(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(20*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+8-4*x)/4		Mémoire énoncé V3,EA31
(32*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
28*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(28/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
7*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(23/4)*x+8/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
(23/4)*x+2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
23*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
(23/4)*x+2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		

(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+8)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
23*x+2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+8)/4-x		Mémoire énoncé V3,EA31
(24*x+8-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(32*x-x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
31*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(31/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(20/4)*x+2/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
5*x+1/2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+1/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
5*x+1/2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1

(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(20*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+1/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4	(A/B)-C ->(A-BC)/B	Règle correcte V1,EA1
(26*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
22*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(22/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
(11/2)*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4		Mémoire énoncé V3,EA31
(20*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(20/4)*x+2/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
5*x+1/2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4		Mémoire énoncé V3,EA31
(20*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+1/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
5*x+1/2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4		Mémoire énoncé V3,EA31

(20*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
20*x+1/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(24*x+2-4*x)/4		Mémoire énoncé V3,EA31
(26*x-4*x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
22*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(22/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
(11/2)*x		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(23/4)*x+2/4	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
(23/4)*x+1/2		
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
23*x+1/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
(23/4)*x+1/2		Mémoire énoncé V3,EA31
((x+8)*3-4+x)/4+2-x		
(x*3+8*3-4+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(x*3+20+x)/4+2-x		
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(23*x+2)/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
23*x+1/2	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
((x+8)*3-4+x)/4+2-x		

(x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24*x+2)/4-x	(A/B)+C ->(A+C)/B	Règle erronée V3,EA32
(24*x+2-x)/4	(A/B)-C ->(A-C)/B	Règle erronée V3,EA32
(26*x-x)/4	AC+B ->(A+B)C	Règle erronée V3,EA42
25*x/4	AC+BC ->(A+B)C	Règle correcte V1,EA1
(25/4)*x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(4/4)*x+20/4+2-x	(A+B)/C ->A/C+B/C	Règle correcte V1,EA1
x+7-x 7	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
4*x+7-x	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
x+7-x		Mémoire énoncé V3,EA31
7	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
4*x+7-x	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
3*x+7	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(4*x+20)/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
4*x+7-x	(A+B)/C ->A+(B/C)	Règle erronée V3,EA32
11*x-x	AC+B ->(A+B)C	Règle erronée V3,EA42
10*x	AC+BC ->(A+B)C	Règle correcte V1,EA1
((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24/4)*x+2-x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
6*x+2-x 5*x+2	AC+BC ->(A+B)C	Règle correcte V1,EA1

((x+8)*3-4+x)/4+2-x (x*3+8*3-4+x)/4+2-x (x*3+20+x)/4+2-x	(A+B)C ->AC+BC	Règle correcte V1,EA1
(23*x+x)/4+2-x	AC+B ->(A+B)C	Règle erronée V3,EA42
24*x/4+2-x	AC+BC ->(A+B)C	Règle correcte V1,EA1
(24/4)*x+2-x	(AB)/C ->(A/C)B	Règle correcte V1,EA1
6*x+2-x 8*x-x	AC+B ->(A+B)C	Règle erronée V3,EA42
7*x	AC+BC ->(A+B)C	Règle correcte V1,EA1

B.7. Transformation de deux arbres en arbres équivalents

B.7.1. Exemple d'un exercice de la classe « Preuve et programme de calcul »

La figure 1 donne un exemple de réponse mettant en jeu des expressions algébriques. L'expression correcte anticipée correspondant au programme de calcul est (E1).

Réponse attendue : (E1) ((x+8)*3-4+x)/4+2-x

Réponse de l'élève : (E2) [3(x+8) -4+x]/[4]+2-x

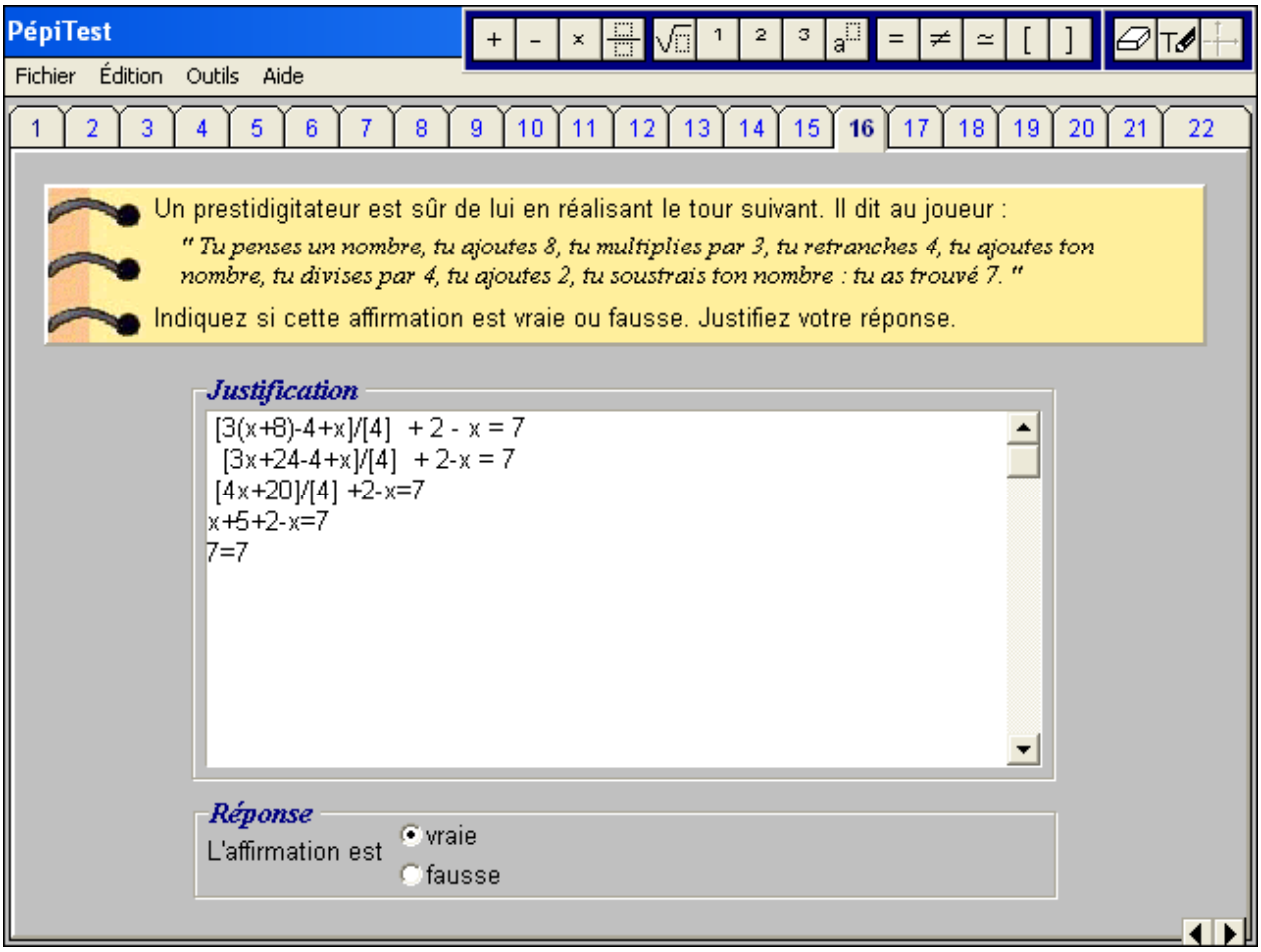


Figure 1 : Réponse de Florence à l'exercice du prestidigitateur

Les deux arbres obtenus ne sont pas équivalents (Figure 2), cependant l'expression (E1) montre que cet élève traduit le programme de calcul par l'expression globale parenthésée attendue (Codée V1) en utilisant le langage algébrique pour justifier l'affirmation (codé J1). En appliquant les procédures de transformation définies dans le chapitre 6 à la section 6.2. Pépinière vérifie que les arbres sont équivalents. Il remplace les soustractions des nœuds ① et ⑤ par des additions et affecte le signe moins aux nœuds impliqués dans la formation puis ordonne les deux fils du nœud ⑦. Dans l'arbre d'expression (E1), le nœud ⑥ ayant un fils droit de poids supérieur à celui du fils gauche, les nœuds ⑦ et ⑧ sont échangés, et les nœuds ① et ② puis ④ et ⑤ sont ordonnés. Nous transformons ainsi les deux expressions en un arbre qui est celui de la Figure 3.

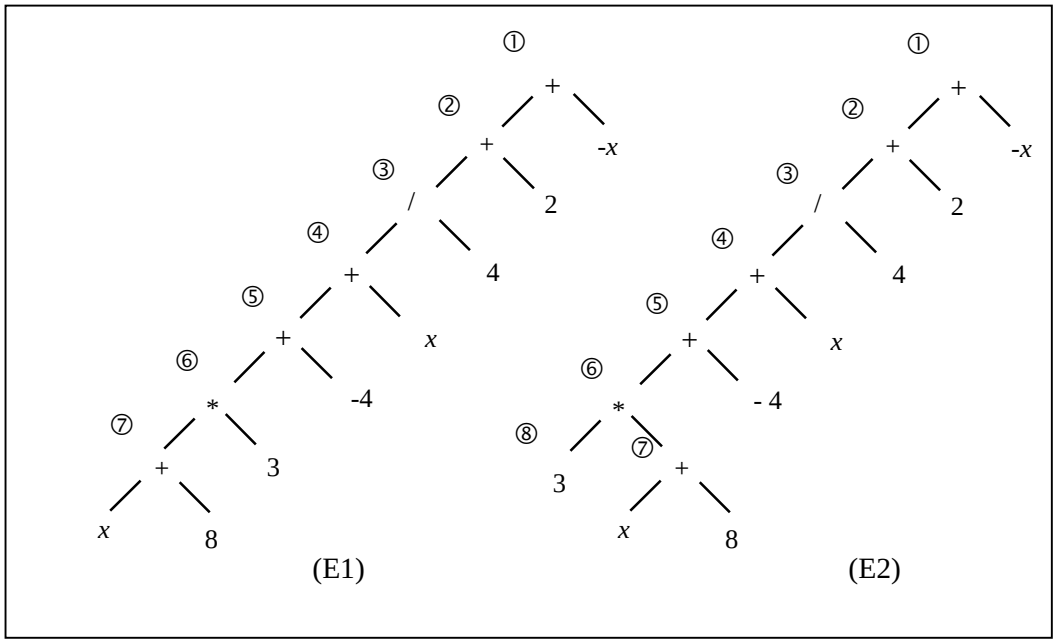


Figure 2 : Arbre d'expression de ((x+8)*3-4+x)/4+2-x et [3(x+8) -4+x]/[4]+2-x

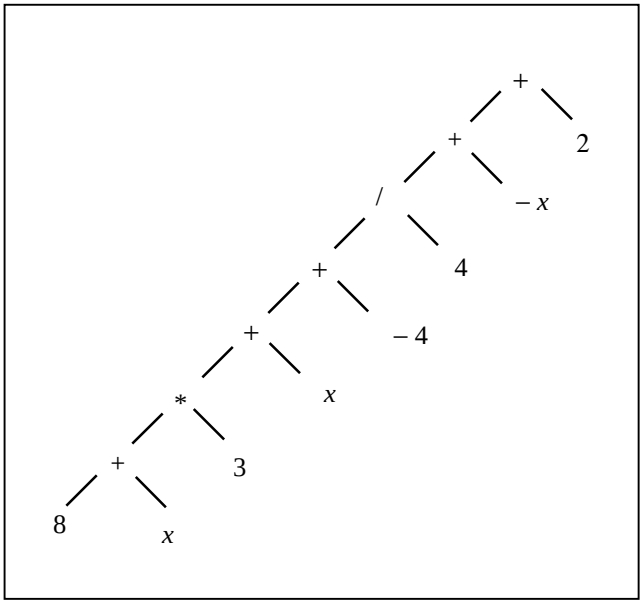


Figure 3 : Arbre d'expression obtenu après transformation de (E1) et (E2)

B.7.2. Exemple d'un exercice de la classe « Déterminer si des expressions algébriques du second degré sont égales »

La

figure **4** donne un exemple de justification qui utilise des calculs algébriques pour répondre à la question posée.

Expression proposée : (E3) $-2\ (x\text{-}1)^2 + 8$

Justification correcte anticipée : (E4) $-2\ (x^2 - 2x + 1) + 8$

Réponse de l'élève : (E5) $-2\ (x^2 + 1 - 2x) + 8$

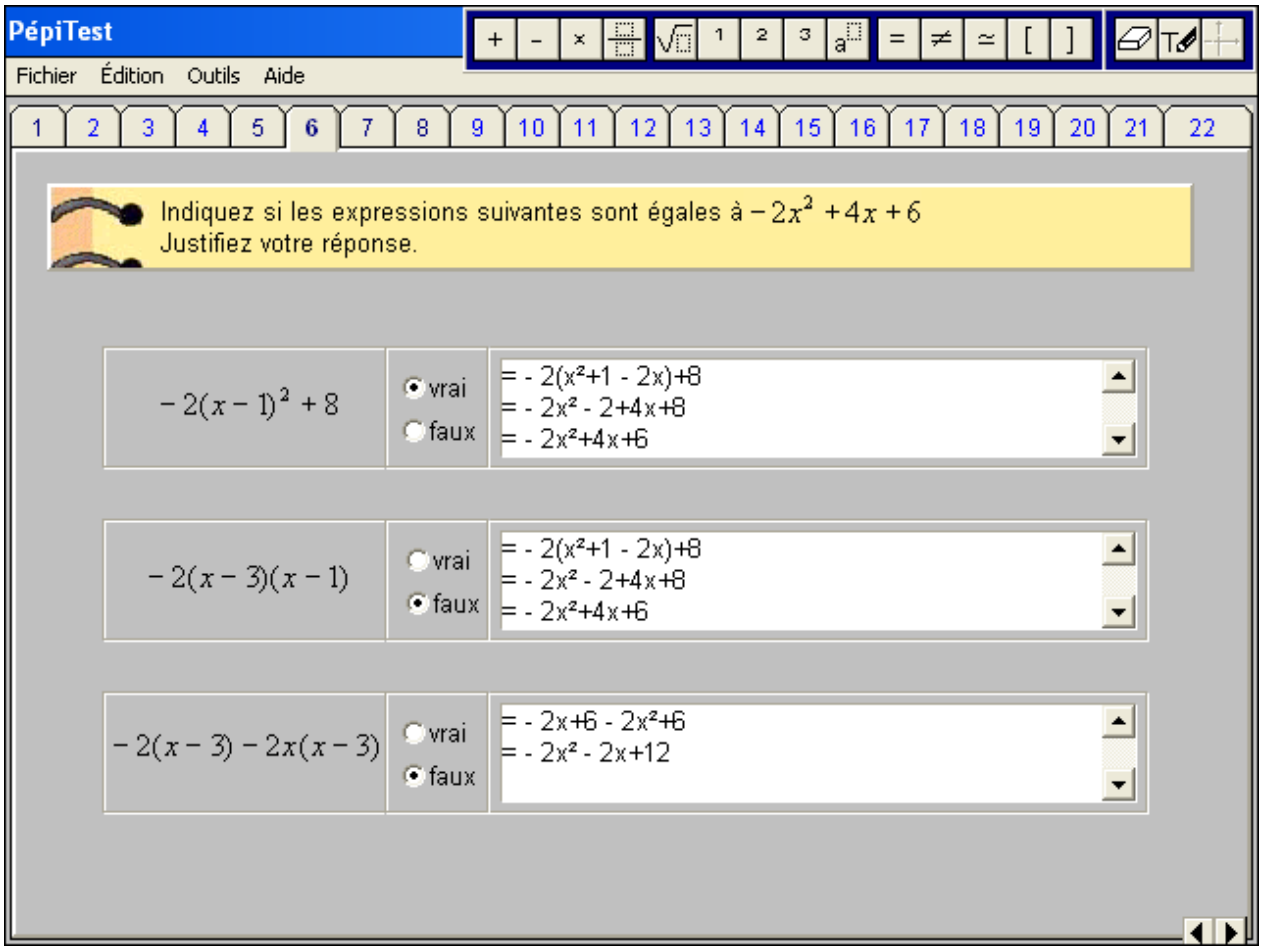


Figure 4 : Réponse de Anne-Laure

Les deux arbres obtenus pour les expression (E4) et (E5) ne sont pas équivalents (**Figure 5**), cependant l'expression (E5) montre que cet élève fait une utilisation correcte du calcul algébrique pour développer l'expression $(x - 1)^2$. En appliquant les procédures de transformation définies dans la section 6.2. du chapitre 6, Pépinière vérifie que les arbres sont équivalents. Il remplace les

soustraction du nœud ⑤de l'arbre (E4) et du nœud ④ de l'arbre (E5) par une addition en appliquant les règles de qui calculent l'opposé d'une différence puis le nœud ② ayant un fils droit de poids supérieur à celui du fils gauche, Pépinière échange les nœuds ③ et ④, enfin il ordonne les nœuds ④ et ⑤ dans l'expression (E5). Pépinière transforme ainsi les deux arbres d'expression en un arbre qui est celui de la Figure 6.

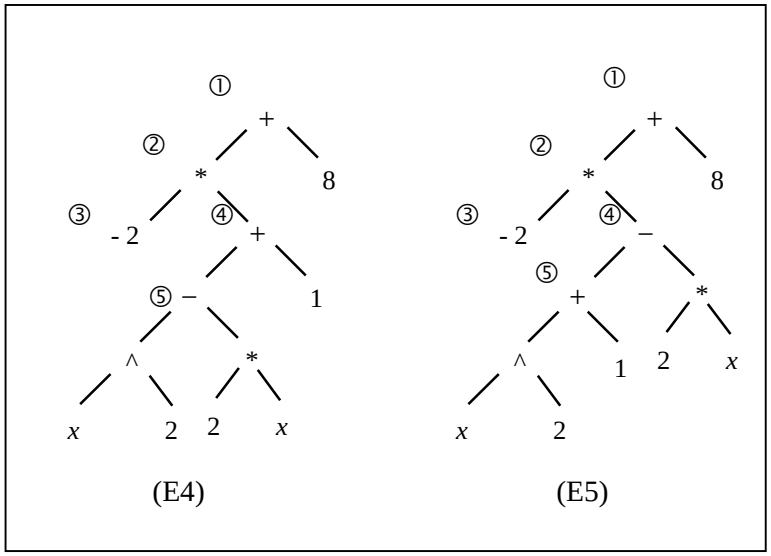


Figure 5 : Arbre d'expression de -2 (x^2 – 2x + 1) + 8 et -2 (x^2 +1 – 2x) + 8

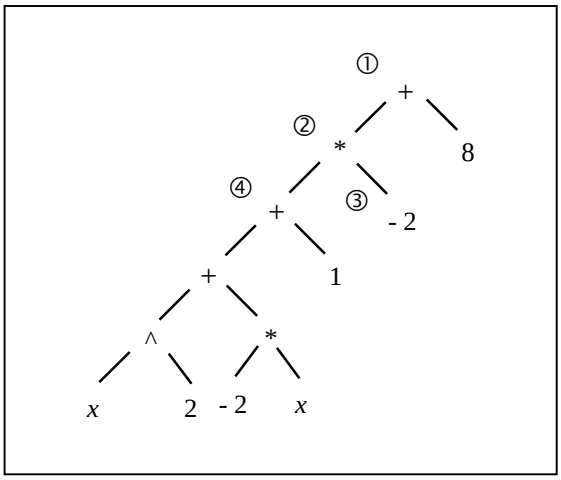


Figure 6 : Arbre obtenu après transformation de (E4) et (E5)

Annexes C

Modèles et tests de PépiGen

C.1. Grammaire de la palette de l'exercice « Preuve et programme de calcul » 323
C.1.1. description des états 323
C.1.2. Représentation de l'automate 325

C.2. Règles de réécriture correctes et erronées 326
C.2.1. Représentation de la structure du fichier XML des régles de réécriture 326
C.2.2. Schéma du fichier XML des régles de réécriture 326
C.2.3. Tableau des régles de réécriture utilisées dans le processus d'unification 327
C.2.4. Fichier XML des règles 329

C.3. Copies d'écrans de PépiGen – Programme de calcul : (x+6)3 -3x 333
C.3.1. Saisie d'un énoncé 333
C.3.2. Analyse des réponses : « Solutions correctes » 333
C.3.3. Analyse des réponses : « Solutions correctes non attendues » 334
C.3.4. Analyse des réponses : « Solutions partielles » 334
C.3.5. Analyse des réponses : « Solutions incorrectes » 335

C.4. Copies d'écrans de PépiGen – Programme de calcul ((3*x*-6)*2-3*x*)/3-*x* 335
C.4.1. Saisie d'un énoncé 335
C.4.2. Analyse des réponses : « Solutions correctes » 336
C.4.3. Analyse des réponses : « Solutions correctes non attendues » 336
C.4.4. Analyse des réponses : « Solutions incorrectes » 337

C.5. Copies d'écrans de PépiGen – Programme de calcul (*x*+3x*+4)/4-1 ..... 337**
C.5.1. Saisie d'un énoncé 337
C.5.2. Analyse des réponses : « Solutions incorrectes » 338

C.6. Copies d'écrans de PépiGen – Programme de calcul ((*x*+8)*3-4+*x*)/4+2-*x* 338
C.6.1. Saisie d'un énoncé 338
C.6.2. Analyse des réponses : « Solutions Correctes » 339
C.6.3. Analyse des réponses : « Solutions correctes non attendues » 339
C.6.4. Analyse des réponses : « Solutions incorrectes » 340

C.7. Schéma XML d'une classe d'exercice 340
C.7.1. Structure de « ClasseExercice » 340
C.7.2. Schéma XML de « ClasseExercice » 341
C.7.3. Structure de « IndexationDidactique » 341
C.7.4. Schéma XML de « IndexationDidactique » 342
C.7.5. Schéma XML de « GenerationEnonce » 342
C.7.5.1. Schéma XML correspondant au type « ContenuType » 342
C.7.5.2. Structure de « GenerationEnonce » 344
C.7.5.3. Schéma XML de « GenerationEnonce » 345
C.7.6. Structure de « InterfaceAbstraite » 346

C.7.7. Schéma XML de « InterfaceAbstraite » 346
C.7.7.1. Structure de « QuestionSimple » 348
C.7.7.2. Schéma XML de « QuestionSimple » 349
C.7.7.3. Schéma XML de « ChoixMultiple 350
C.7.7.4. Schéma XML de « ChoixExclusif » 350
C.7.8. Classe « GenerationGrilleCodage » 351
C.7.8.1. Structure de «GenerationGrilleCodage » 351
C.7.8.2. Schéma XML de « GenerationGrilleCodage » 352
C.7.8.3. Schéma XML de « ReponseQuestionFermee » 352
C.7.8.4. Structure de « Solutions » 353
C.7.8.5. Schéma XML de « Solutions » 354
C.7.8.6. Structure de « SolutionType » 355
C.7.8.7. Schema XML de « SolutionType » 355

C.8. Structure de « ModelesClasseExercices » 355

C.9. Fichier XML contenant deux modèles d'exercices 356

C.10. Base d'exercices 362
C.10.1. Structure de la base d'exercices 362
C.10.2. Schéma XMLde la base d'exercices 362
C.10.3. Structure de « Exercices » 362
C.10.4. Schéma XMLde « Exercice » 363
C.10.5. Structure de Solutions 364

C.11. Clones générés par PépiGen -Programme de calcul : (x+6)3 -3x 365
C.11.1. Programme de calcul : ((x*3 - 6)*2 - 3*x)/3 - x 368
C.11.2. Programme de calcul : (x + 3 * x + 4)/ 4 –1 373
C.11.3. Programme de calcul : (x * 4 + 6)/ 2 - 2 * x 381

C.12. Le diagnostiqueur : Tests 388

C.1. Grammaire de la palette de l'exercice « Preuve et programme de calcul »

C.1.1. description des états

L'automate d'états finis est {K, V_T, f, D, F} avec :
- K : ensemble des états
- V_T : vocabulaire terminal
- f : application de K * V_T dans K de telle sorte que f(k_i , a_i) $\rightarrow$ k_j avec k_i , k_j $\in$ K et a_i V_T
- D : état initial

F : ensemble d'états terminaux.

K = { , état-1, état-2, …….état-39, état-40, état-41}
V_T = { +, -, *, / , tu-prends-un-nombre, double, triple, fois, nombre, nombre-de-départ, résultat, tu-as-trouvé, Que-constates-tu-?, le, ce, ton, de, du, au, à, par, virgule, point, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
D = { état-0}
F = {état-40, état-41}

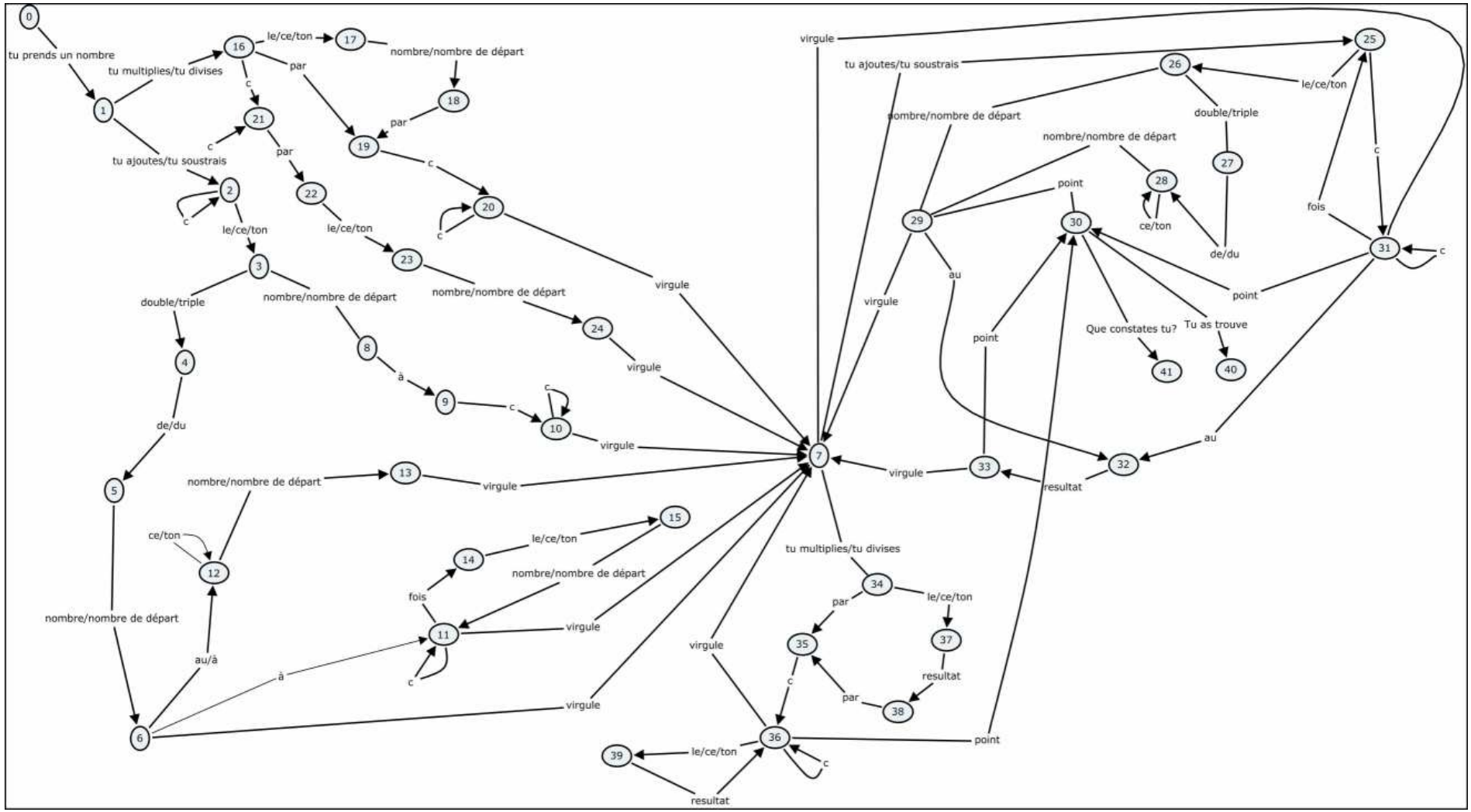
Pour décrire la grammaire nous remplacons les éléments terminaux 0,1,2,3,4,5,6,7,8,9 par *chiffre*, tu ajoutes est remplacé par +, tu soustrais par -, tu multiplies par * et tu divises par /.

f (état-0, tu-prends-un-nombre) = état-1
f(état-1, +) = f(état-1, -) = état-3
f(état-1, *) = f(état-1, /) = état-70
f(état-2, le) = f(état-2, ce) = f(état-2, ton) = état-3
etc.

La grammaire correspondante est :
état-0 $\rightarrow$ tu-prends-un-nombre état 1
état-1 $\rightarrow$ + état-2 | - état-2 | * état-16 | / état-16
état-2 $\rightarrow$ le état-3 | ce état-3 | ton état-3 | *chiffre* état-11
état-3 $\rightarrow$ nombre état-8| nombre-de-départ état-8 | double état-4 | triple état-4
état-4 $\rightarrow$ de état-5 | du état-5
état-5 $\rightarrow$ ce état-5 | ton état-5 | nombre état-6 | nombre-de-départ état-6
état-6 $\rightarrow$ au état-12 | à état-12 | virgule état-7
état-7 $\rightarrow$ + état-25 | - état-25 | * état-34 | / état-34
état-8 $\rightarrow$ à état-9
état-9 $\rightarrow$ *chiffre* état-10
état-10 $\rightarrow$ *chiffre* état-10 | virgule état-7
état-11 $\rightarrow$ *chiffre* état-11 | au état-12 | à état-12 | fois état-14 | virgule état-7
état-12 $\rightarrow$ ce état-12 | ton état-12 | nombre état-13 | nombre-de-départ état-13
état-13 $\rightarrow$ virgule état-7
état-14 $\rightarrow$ le état-15 | ce état-15 | ton état-15
état-15 $\rightarrow$ nombre état-11 | nombre-de-départ état-11
état-16 $\rightarrow$ le état-17 | ce état-17 | ton état-17 | par état-19 | *chiffre* état-21
état-17 $\rightarrow$ nombre état-18 | nombre-de-départ état-18
état-18 $\rightarrow$ par état-19
état-19 $\rightarrow$ *chiffre* état-20
état-20 $\rightarrow$ *chiffre* état-20 | virgule état-7
etat-21 $\rightarrow$ *chiffre* état-21 | par état-22

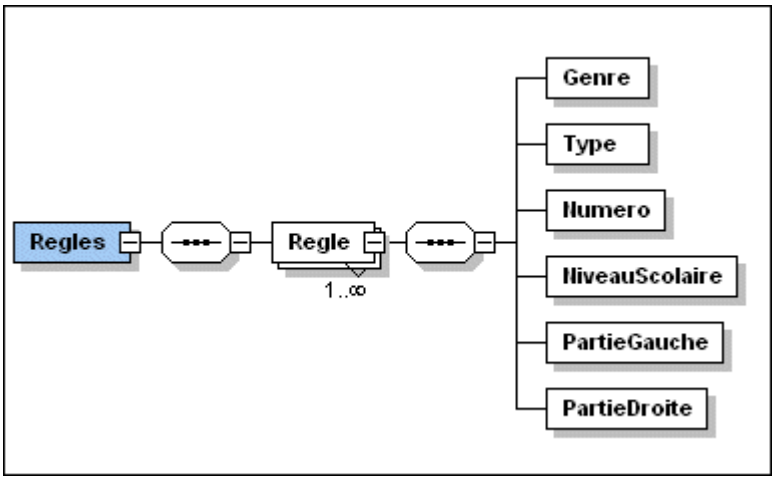
etat-22 → le état-23 | ce état-23 | ton état-23 |
etat-23 → nombre état-24 | nombre-de-départ état-24
etat-24 → virgule état-7
état-25 → le état-26 | ce état-26 | ton état-26 | *chiffre* état-31
état-26 → nombre état-29 | nombre-de-départ état-29 | double état-27 | triple état-27
état-27 → de état-28 | du état-28
état-28 → ce état-28 | ton état-28 | nombre état-29 | nombre-de-départ état-29
état-29 → au état-32 | virgule état-7 | point état-30
état-30 → Tu-as-trouvé | Que-constates-tu-? (états finals 40 et 41)
état-31 → *chiffre* état-31 | au état-32 | virgule état-7 | fois état-25 | point état-30
état-32 → résultat état-33
état-33 → virgule état-7 | point état-30
état-34 → le état-37 | ce état-37 | ton état-37 | par état-35
état-35 → *chiffre* état-36
état-36 → *chiffre* état-36 | le état-39 | ce état-39 | ton état-39 | virgule état-7 | point état-30
état-37 → résultat état-38
état-38 → par état-35
état-39 → résultat état-36

C.1.2. Représentation de l'automate



C.2. Règles de réécriture correctes et erronées

C.2.1. Représentation de la structure du fichier XML des régles de réécriture



C.2.2. Schéma du fichier XML des régles de réécriture

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com) by dominique  -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="Regles">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Regle" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Genre"/>
              <xs:element name="Type"/>
              <xs:element name="Numero"/>
              <xs:element name="NiveauScolaire"/>
              <xs:element name="PartieGauche"/>
              <xs:element name="PartieDroite"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

C.2.3. Tableau des régles de réécriture utilisées dans le processus d'unification

T : type de règles
N : numéro de règle
V, F : genre des régles

T	N	Règles correctes « V »	N	Règles erronées « F »
1 : Développements				
1	1	(A+B) (C+D) → AC+BC+AD+BD		
	2	(A-B) (C+D) → AC - BC+AD - BD		
	3	(A+B) (C-D) → AC+BC - AD - BD		
	4	(A-B) (C-D) → AC - BC - AD+BD		
	5	A (B+C) → AB + AC	5	A (B + C) → AB + C
	6	A (B-C) → AB – AC	6	A (B–C) → AB – C
	7	(B+C) A → BA + CA		
	8	(B-C) A → BA - CA		
	9	$(A+B)^2 \rightarrow A^2 + 2\,AB + B^2$	8	$(A+B)^2 \rightarrow A^2 + AB + B^2$
			9	$(A+B)^2 \rightarrow A^2 + B^2$
	10	$(A-B)^2 \rightarrow A^2 - 2\,AB + B^2$	10	$(A-B)^2 \rightarrow A^2 + B^2$
	11	$(A-B)(A+B) \rightarrow A^2 - B^2$	11	$(A-B)(A+B) \rightarrow A^2 + B^2$
	12	$(A+B)(A-B) \rightarrow A^2 - B^2$		
2 : Simplification des fractions				
2	13	(A+B)/C → (A/C) + (B/C)	13	(A+B)/C → A + (B/C)
	14	(A-B)/C → (A/C) - (B/C)	14	(A-B)/C → A - (B/C)
	15	(AB)/C → (A/C) B		

T	N	Règles correctes « V »	N	Règles erronées « F »
3 : Réduction au même dénominateur				
3	16	(A/B)+(C/D) → (AD+BC)/(BD)	15	(A/B)+(C/D)→(A+C)/(B+D)
			16	(A/B)+(C/D)→(A+C)/(BD)
	17	(A/B)-(C/D) → (AD-BC)/(BD)	17	(A/B)-(C/D)→ (A-C)/(B-D)
	18	(A/B)+C → (A+BC)/B	18	(A/B)+C → (A+C)/B
	19	(A/B)-C → (A-BC)/B	19	(A/B)-C → (A-C)/B
	20	(A/B)(C/D) → (AC)/(BD)		
	21	A(B/C) → (AB)/C	21	A(B/C) → (AB)/(AC)
4 : Identités remarquables				
4	22	$A^2 + 2\,AB + B^2 \rightarrow (A+B)^2$		
	23	$A^2 - 2\,AB + B^2 \rightarrow (A-B)^2$		
	24	$A^2 - B^2 \rightarrow (A+B)\,(A-B)$		
	25	AB + AC → A(B+C)		
5 : Puissances				
5	26	$A^m \times A^n \rightarrow A^{m+n}$	26	$A^m \times A^n \rightarrow A^{m*n}$
	27	$(A^m)^n \rightarrow A^{m*n}$	27	$(A^m)^n \rightarrow A^{m+n}$
	28	$(AB)^n \rightarrow A^n * B^n$	28	$(AB)^n \rightarrow A\,B^n$
	29	$(A/B)^n \rightarrow A^n / B^n$	29	$(A/B)^n \rightarrow A^n / B$
			30	$A^n = n\,A$
			31	$A^m + A^n \rightarrow A^{m+n}$

T : type de règles
N : numéro de règle
V, F : genre des régles

T	N	Règles correctes « V »	N	Règles erronées « F »
6 : Réduction d'un polynôme				
6		A et B sont des nombres, C une puissance d'une variable		A et B sont des nombres, C une puissance d'une variable
		addition de momômes semblables	31	AC+B $\rightarrow$ (A+B) C
				exemple : 3*x*+24 = 27*x*
	31	AC+BC $\rightarrow$ (A+B) C	32	A+BC $\rightarrow$ (A+B) C
				exemple : 3+24x = 27*x*
			33	AC-C $\rightarrow$ A-1
				exemple : 8x – x = 7 ou 23x + x = 24
			34	AC-C $\rightarrow$ A
				exemple : 8x – x = 8 ou 23x + x = 23
			35	AB + B $\rightarrow$ AB^2
				exemple :3x+x = $3x^2$
7 : Propriétés particulières				
7	36	A $\rightarrow$ 1 * A élément neutre		
	37	A + 0 $\rightarrow$ A		
	38	A * 0 $\rightarrow$ 0 élément absorbant		
	39	1 * A $\rightarrow$ A		

C.2.4. Fichier XML des règles

```
<?xml version="1.0" encoding="UTF-8"?>
<Regles >
<Regle>
<Genre>V</Genre>
  <Type>1</Type>
  <Numero>1</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a+b)*(c+d)</PartieGauche>
  <PartieDroite>a*c+a*d+b*c+b*d</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>2</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a-b)*(c+d)</PartieGauche>
  <PartieDroite>a*c+a*d-b*c-b*d</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>3</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a+b)*(c-d)</PartieGauche>
  <PartieDroite>a*c-a*d+b*c-b*d</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>4</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a-b)*(c-d)</PartieGauche>
  <PartieDroite>a*c-a*d-b*c+b*d</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>5</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a*(b+c)</PartieGauche>
  <PartieDroite>a*b+a*c</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>5</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a*(b+c)</PartieGauche>
  <PartieDroite>a*b+c</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>6</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a*(b-c)</PartieGauche>
  <PartieDroite>a*b-a*c</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>6</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a*(b-c)</PartieGauche>
  <PartieDroite>a*b-c</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>7</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(b+c)*a</PartieGauche>
  <PartieDroite>b*a+c*a</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>8</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(b-c)*a</PartieGauche>
  <PartieDroite>b*a-c*a</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>9</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a+b)^2</PartieGauche>
  <PartieDroite>a^2+2*a*b+b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>8</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a+b)^2</PartieGauche>
  <PartieDroite>a^2+ab+b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>9</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a+b)^2</PartieGauche>
  <PartieDroite>a^2+b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
```

```
  <Numero>10</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a-b)^2</PartieGauche>
  <PartieDroite>a^2-2*a*b+b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>10</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a-b)^2</PartieGauche>
  <PartieDroite>a^2-b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>11</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a-b)*(a+b)</PartieGauche>
  <PartieDroite>a^2-b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>11</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a-b)*(a+b)</PartieGauche>
  <PartieDroite>a^2+b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>1</Type>
  <Numero>12</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a+b)*(a-b)</PartieGauche>
  <PartieDroite>a^2-b^2</PartieDroite>
</Regle>
<Regle>
```

```
  <Genre>F</Genre>
  <Type>1</Type>
  <Numero>12</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a+b)*(a-b)</PartieGauche>
  <PartieDroite>a^2+b^2</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>2</Type>
  <Numero>13</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a+b)/c</PartieGauche>
  <PartieDroite>(a/c)+(b/c)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>2</Type>
  <Numero>13</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a+b)/c</PartieGauche>
  <PartieDroite>a+(b/c)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>2</Type>
  <Numero>14</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a-b)/c</PartieGauche>
  <PartieDroite>(a/c)-(b/c)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>2</Type>
  <Numero>14</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a-b)/c</PartieGauche>
  <PartieDroite>a-(b/c)</PartieDroite>
```

```
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>2</Type>
  <Numero>15</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a*b)/c</PartieGauche>
  <PartieDroite>(a/c)*b</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
  <Numero>15</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b+c/d</PartieGauche>
  <PartieDroite>(a+c)/(b+d)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>3</Type>
  <Numero>16</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b+c/d</PartieGauche>
  <PartieDroite>(a*d+b*c)/(b*d)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
  <Numero>16</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b+c/d</PartieGauche>
  <PartieDroite>(a+c)/(b*d)</PartieDroite>
</Regle>
<Regle>
<Genre>V</Genre>
  <Type>3</Type>
  <Numero>17</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
```

```
  <PartieGauche>a/b-c/d</PartieGauche>
  <PartieDroite>(a*d-+b*c)/(b*d)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
  <Numero>17</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b-c/d</PartieGauche>
  <PartieDroite>(a-c)/(b-d)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>3</Type>
  <Numero>18</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b+c</PartieGauche>
  <PartieDroite>(a+b*c)/b</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
  <Numero>18</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b+c</PartieGauche>
  <PartieDroite>(a+c)/b</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>3</Type>
  <Numero>19</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b-c</PartieGauche>
  <PartieDroite>(a-b*c)/b</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
```

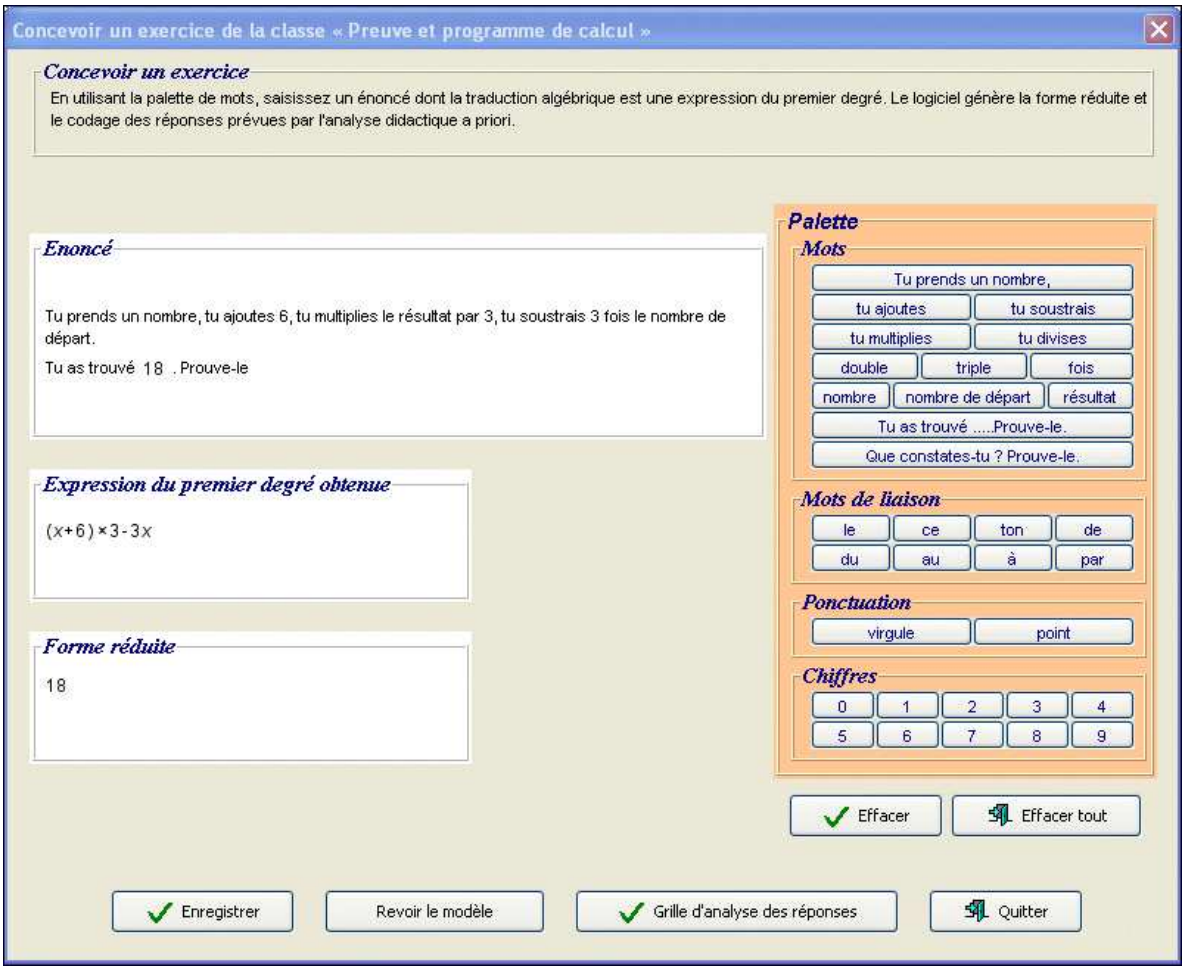
```
  <Numero>19</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a/b-c</PartieGauche>
  <PartieDroite>(a-c)/b</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
  <Numero>20</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>(a/b)*(c/d)</PartieGauche>
  <PartieDroite>(a*c)/(b*d)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>3</Type>
  <Numero>21</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a*(b/c)</PartieGauche>
  <PartieDroite>(a*b)/c</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>3</Type>
  <Numero>21</Numero>
  <NiveauScolaire>4,3,2</NiveauScolaire>
  <PartieGauche>a*(b/c)</PartieGauche>
  <PartieDroite>(a*b)/(a*c)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>4</Type>
  <Numero>22</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>a^2+2*a*b+b^2</PartieGauche>
  <PartieDroite>(a+b)^2</PartieDroite>
</Regle>
<Regle>
```

```
  <Genre>V</Genre>
  <Type>4</Type>
  <Numero>23</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>a^2-2*a*b+b^2</PartieGauche>
  <PartieDroite>(a-b)^2</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>4</Type>
  <Numero>24</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>a^2-b^2</PartieGauche>
  <PartieDroite>(a+b)*(b-c)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>4</Type>
  <Numero>25</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a*b)+(a*c)</PartieGauche>
  <PartieDroite>a*(b+c)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>5</Type>
  <Numero>26</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a^m)*(a^n)</PartieGauche>
  <PartieDroite>a^(m+n)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>5</Type>
  <Numero>26</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a^m)*(a^n)</PartieGauche>
  <PartieDroite>a^(m*n)</PartieDroite>
```

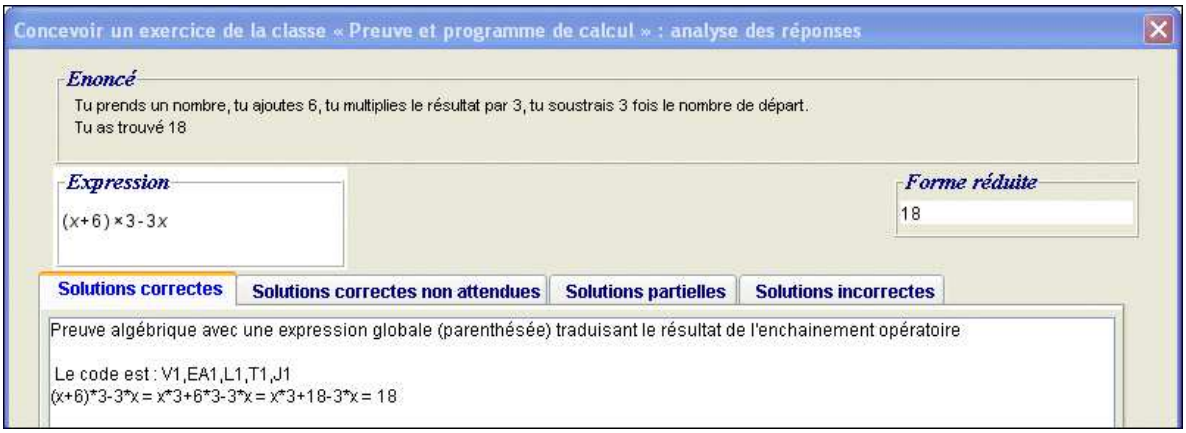
```
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>5</Type>
  <Numero>27</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a^m)^n</PartieGauche>
  <PartieDroite>a^(m*n)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>5</Type>
  <Numero>27</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a^m)^n</PartieGauche>
  <PartieDroite>a^(m+n)</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>5</Type>
  <Numero>28</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a*b)^n</PartieGauche>
  <PartieDroite>(a^n)*(b^n)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>5</Type>
  <Numero>28</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a*b)^n</PartieGauche>
  <PartieDroite>a*b^n</PartieDroite>
</Regle>
<Regle>
  <Genre>V</Genre>
  <Type>5</Type>
  <Numero>29</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a/b)^n</PartieGauche>
  <PartieDroite>(a^n)/(b^n)</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>5</Type>
  <Numero>29</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>(a/b)^n</PartieGauche>
  <PartieDroite>(a^n)/b</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>5</Type>
  <Numero>30</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>a^n</PartieGauche>
  <PartieDroite>n*a</PartieDroite>
</Regle>
<Regle>
  <Genre>F</Genre>
  <Type>5</Type>
  <Numero>310</Numero>
  <NiveauScolaire>3,2</NiveauScolaire>
  <PartieGauche>a^n+a^m</PartieGauche>
  <PartieDroite>a^(n+m)</PartieDroite>
</Regle>
</Regles>
```

C.3. Copies d'écrans de PépiGen – Programme de calcul : (x+6)3 -3x

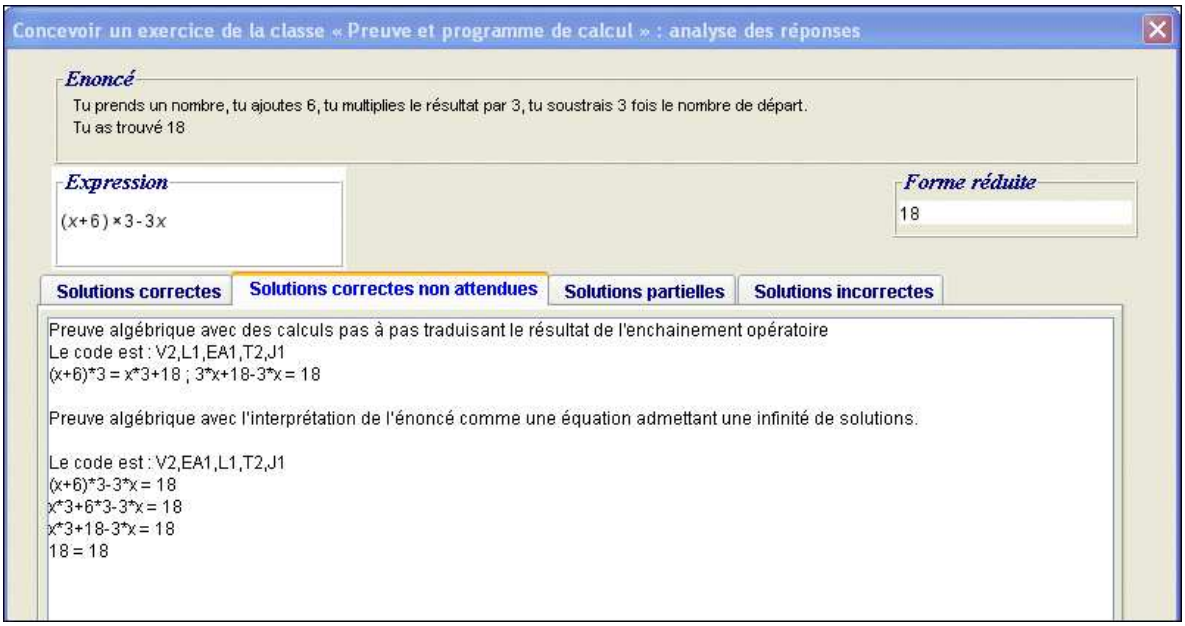
C.3.1. Saisie d'un énoncé



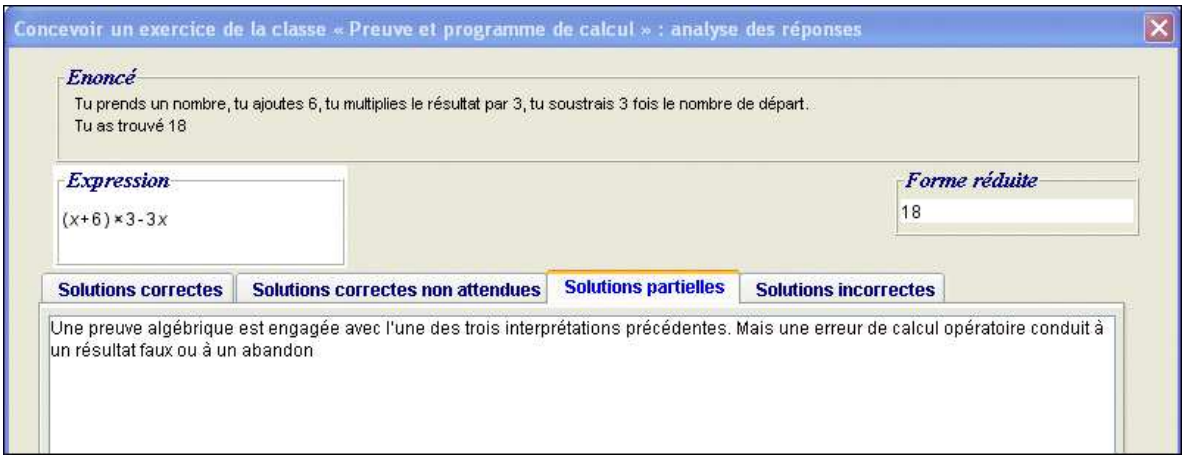
C.3.2. Analyse des réponses : « Solutions correctes »



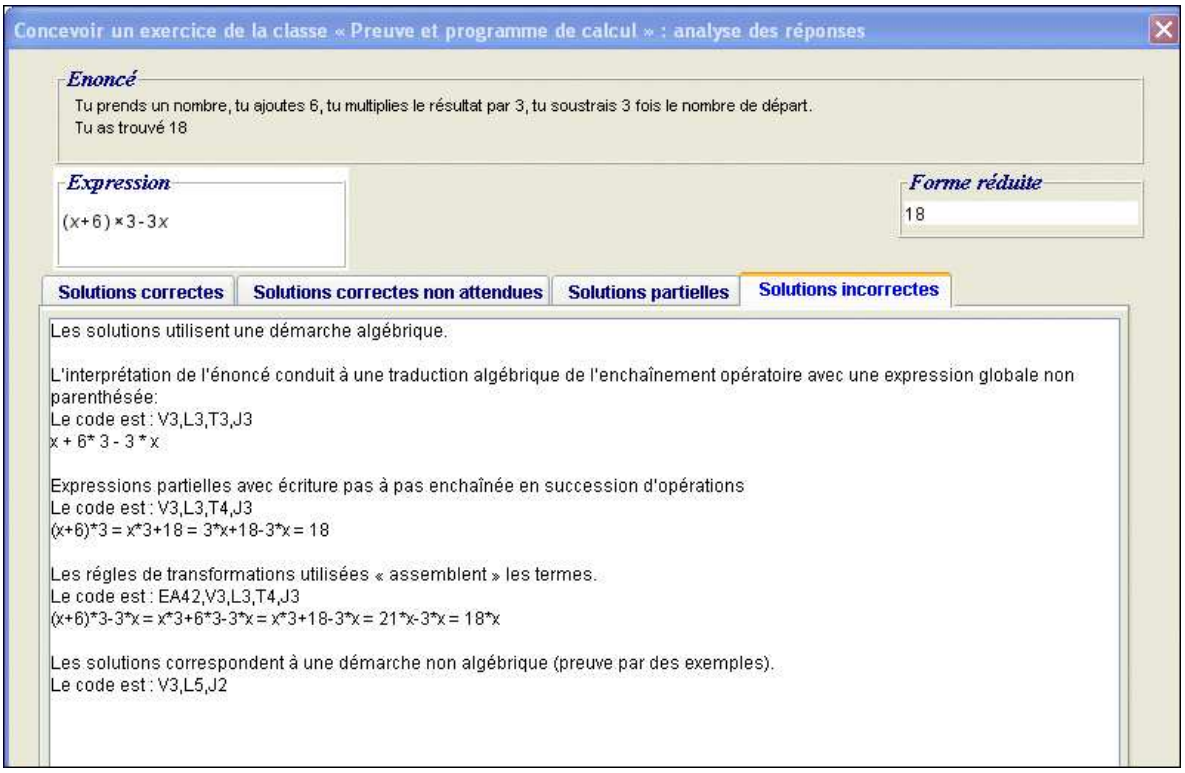
C.3.3. Analyse des réponses : « Solutions correctes non attendues »



C.3.4. Analyse des réponses : « Solutions partielles »

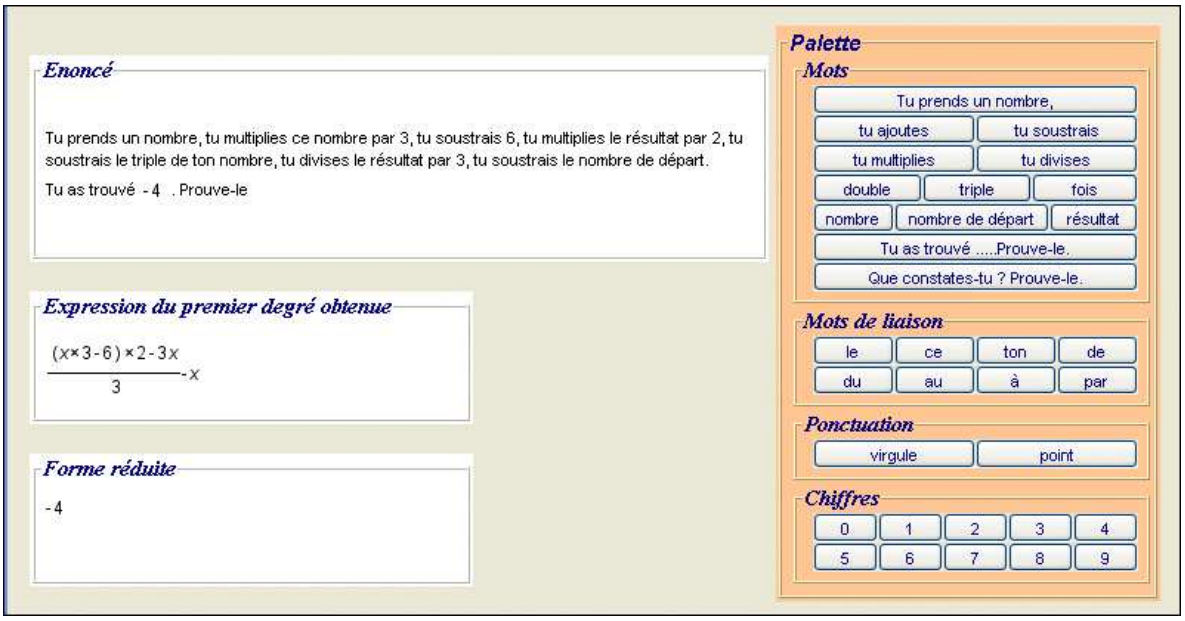


C.3.5. Analyse des réponses : « Solutions incorrectes »

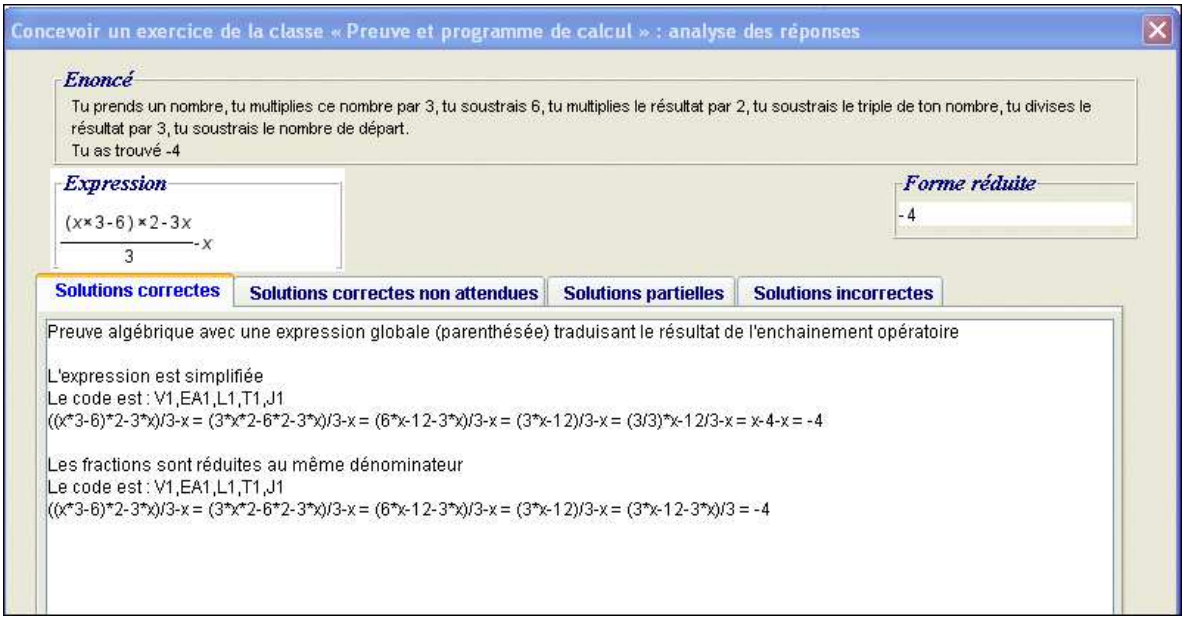


C.4. Copies d'écrans de PépiGen – Programme de calcul ((3*x*-6)*2-3*x*)/3-*x*

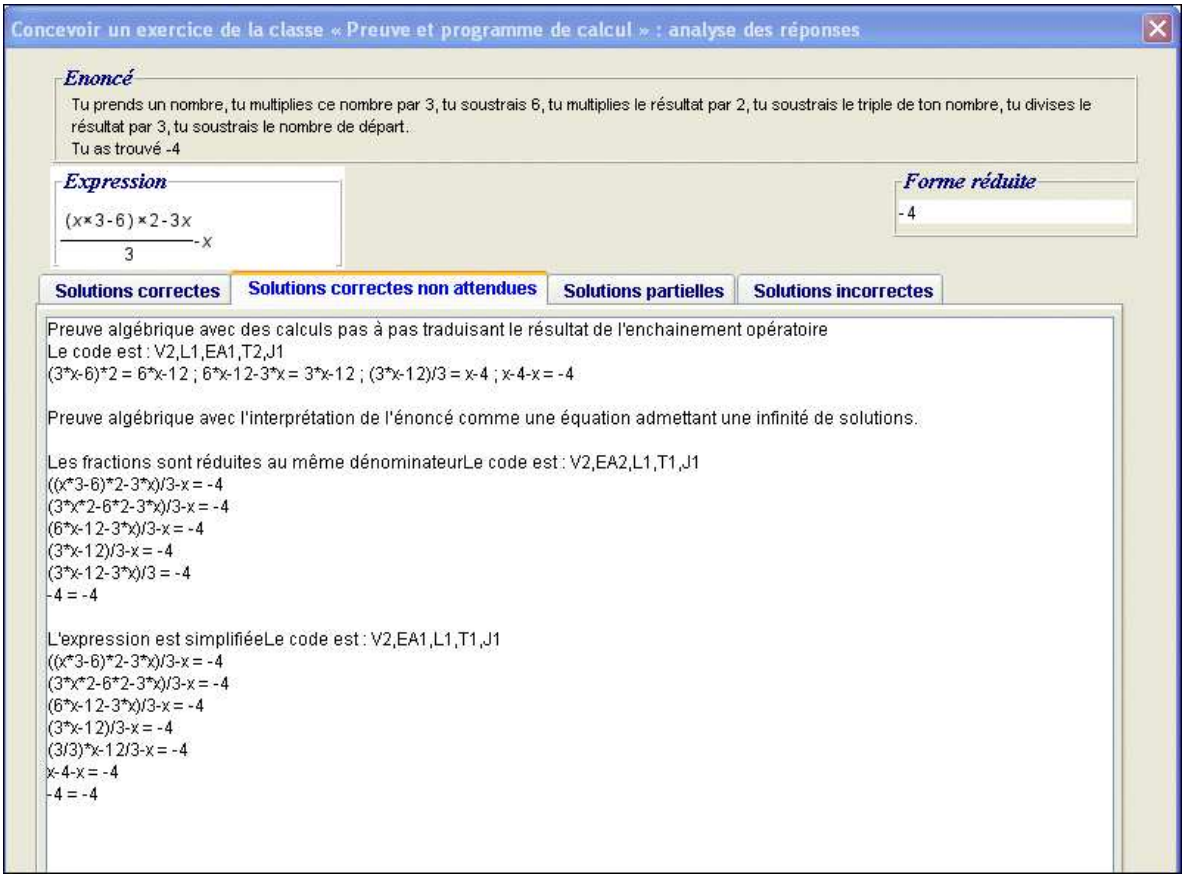
C.4.1. Saisie d'un énoncé



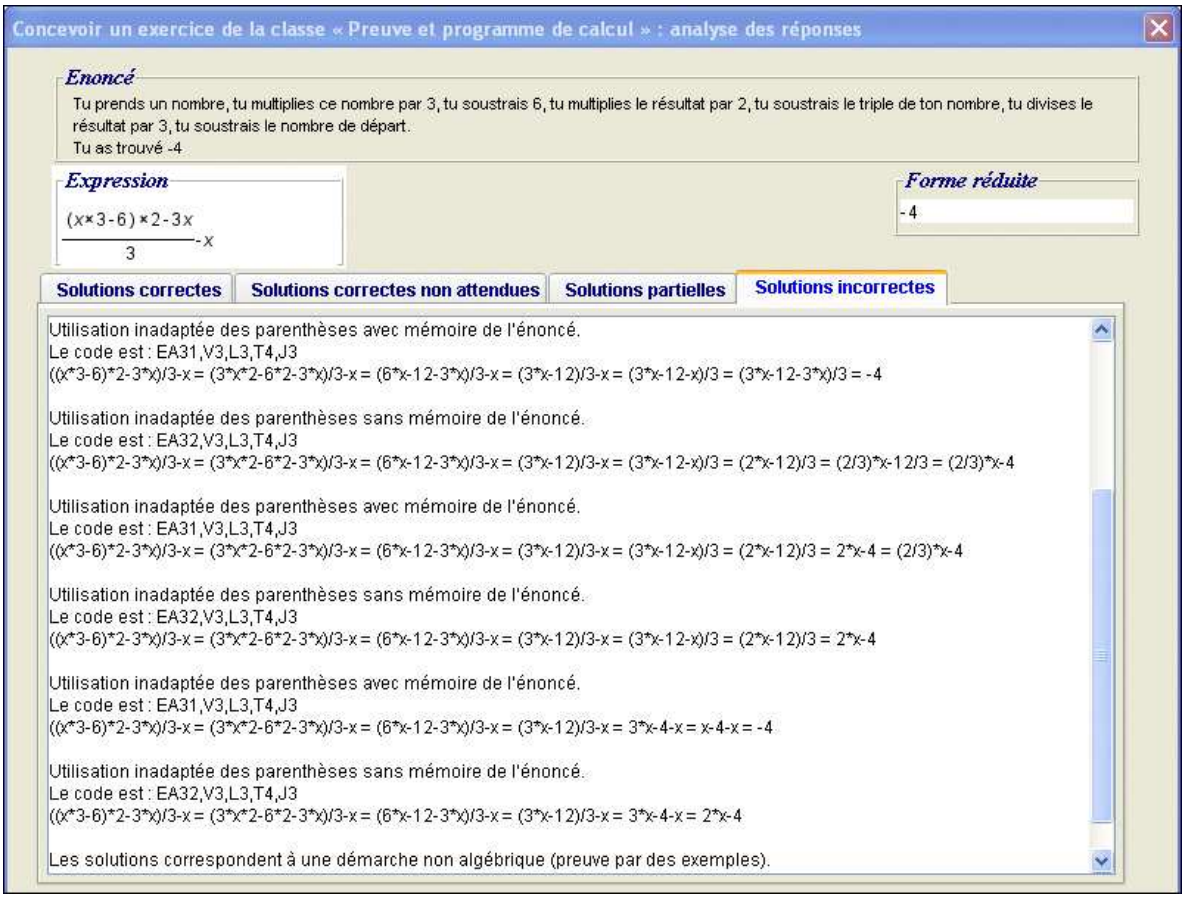
C.4.2. Analyse des réponses : « Solutions correctes »



C.4.3. Analyse des réponses : « Solutions correctes non attendues »



C.4.4. Analyse des réponses : « Solutions incorrectes »



C.5. Copies d'écrans de PépiGen – Programme de calcul (*x*+3**x*+4)/4-1

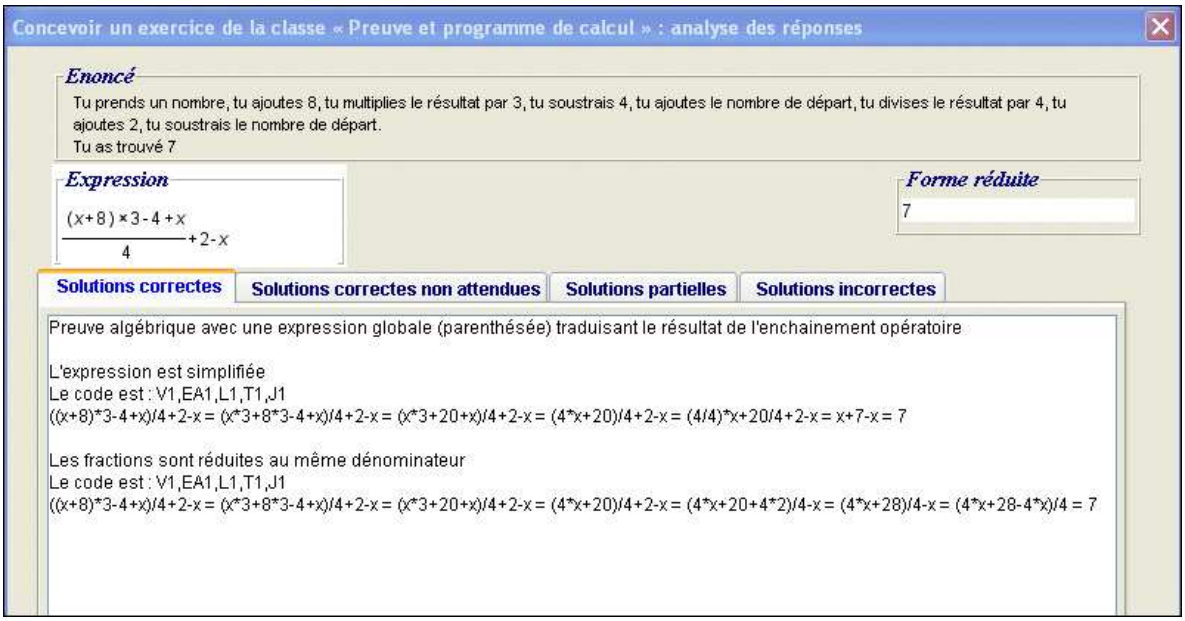
C.5.1. Saisie d'un énoncé

C.5.2. Analyse des réponses : « Solutions incorrectes »

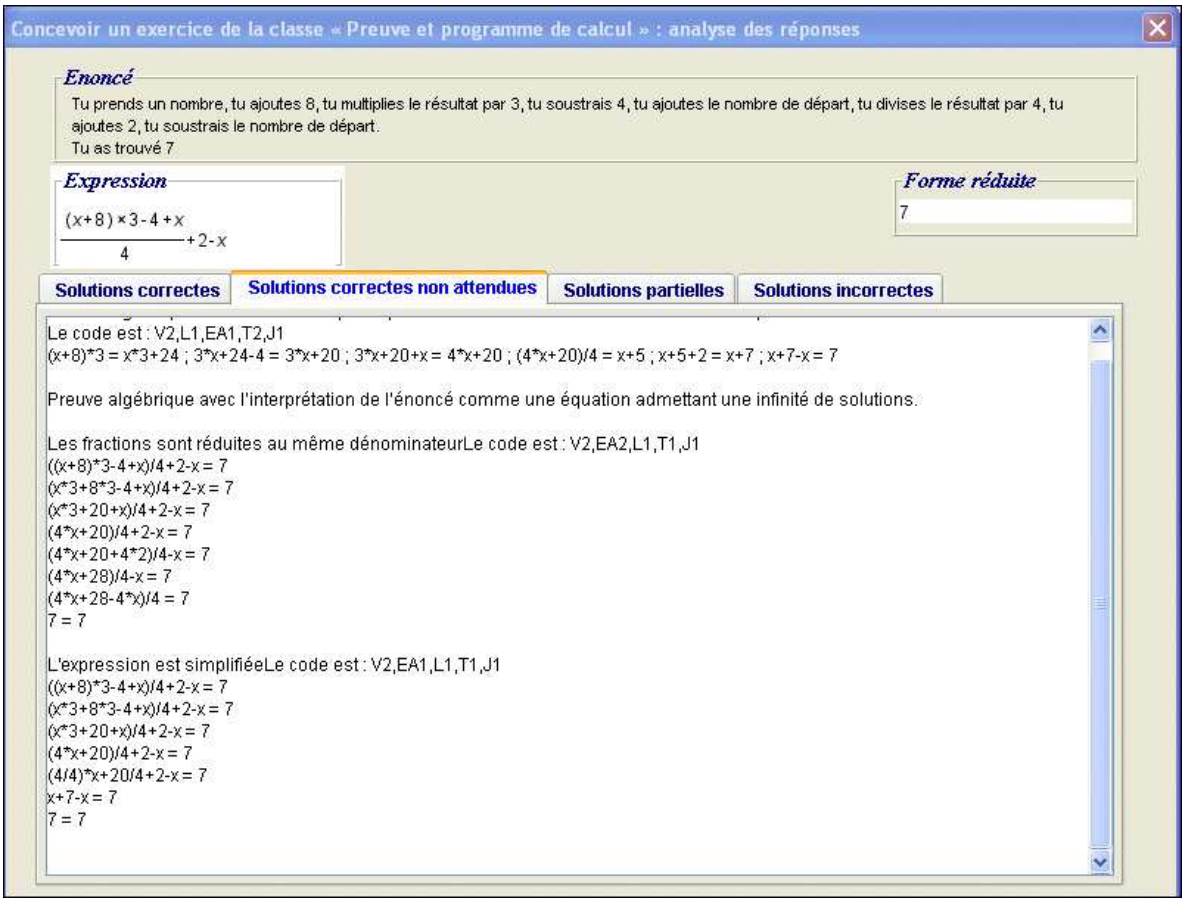
C.6. Copies d'écrans de PépiGen – Programme de calcul ((*x*+8)*3-4+*x*)/4+2-*x*

C.6.1. Saisie d'un énoncé

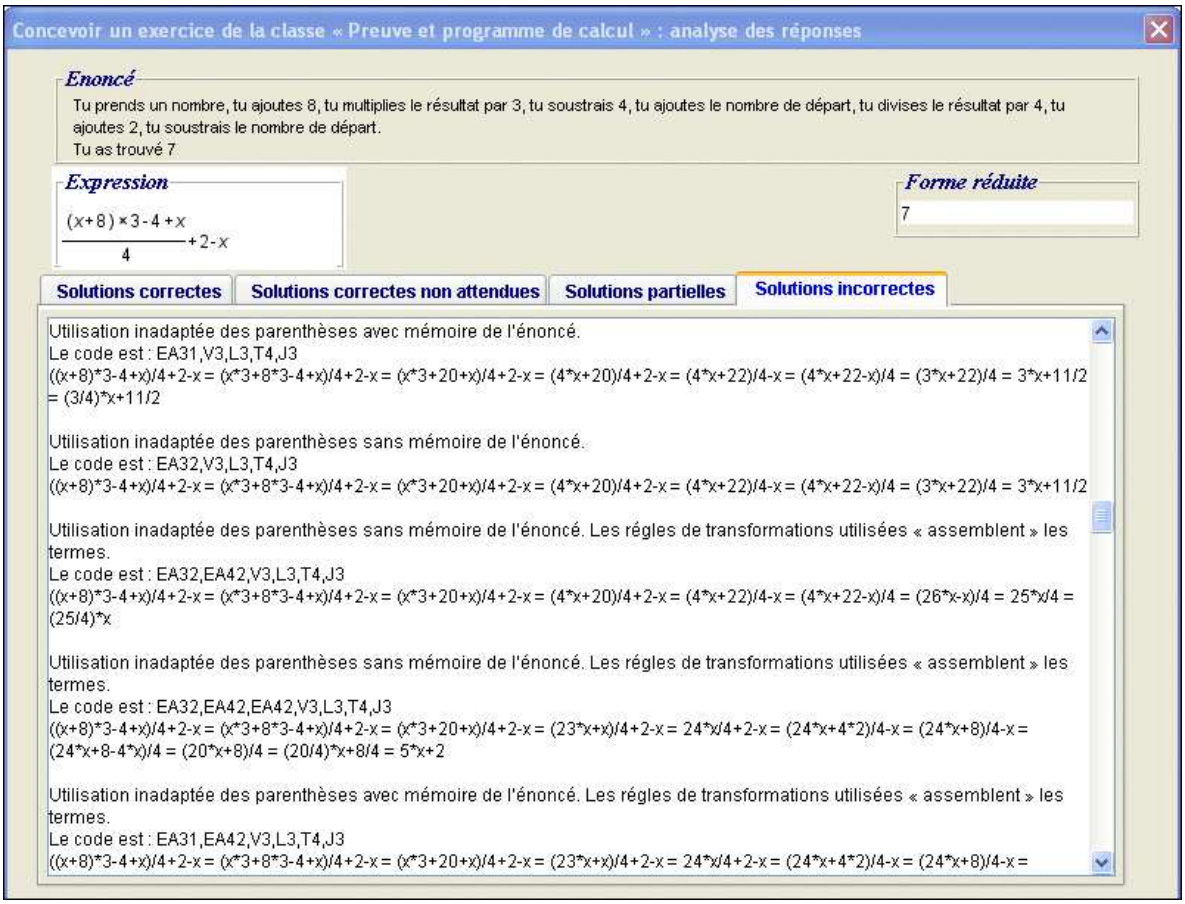
C.6.2. Analyse des réponses : « Solutions Correctes »



C.6.3. Analyse des réponses : « Solutions correctes non attendues »



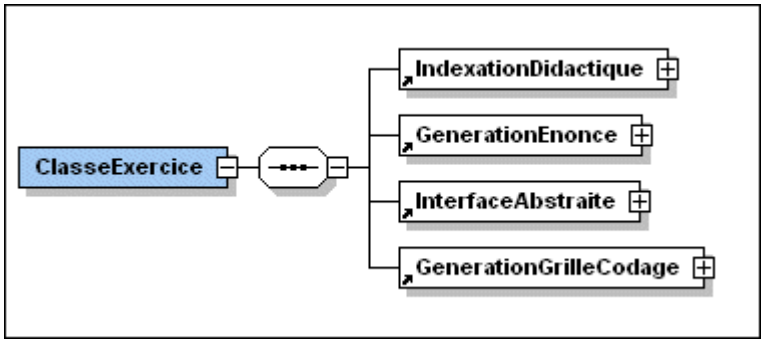
C.6.4. Analyse des réponses : « Solutions incorrectes »



C.7. Schéma XML d'une classe d'exercice

La section C.7. présente l'implémentation du modèle conceptuel d'une classe d'exercices dans un schéma XML. Pour plus de lisibilité, ce schéma XML est détaillé selon les principales classes qui le compose. Pour chacune de ces classes nous présentons la structure du fichier XML, puis le schéma XML correspondant.

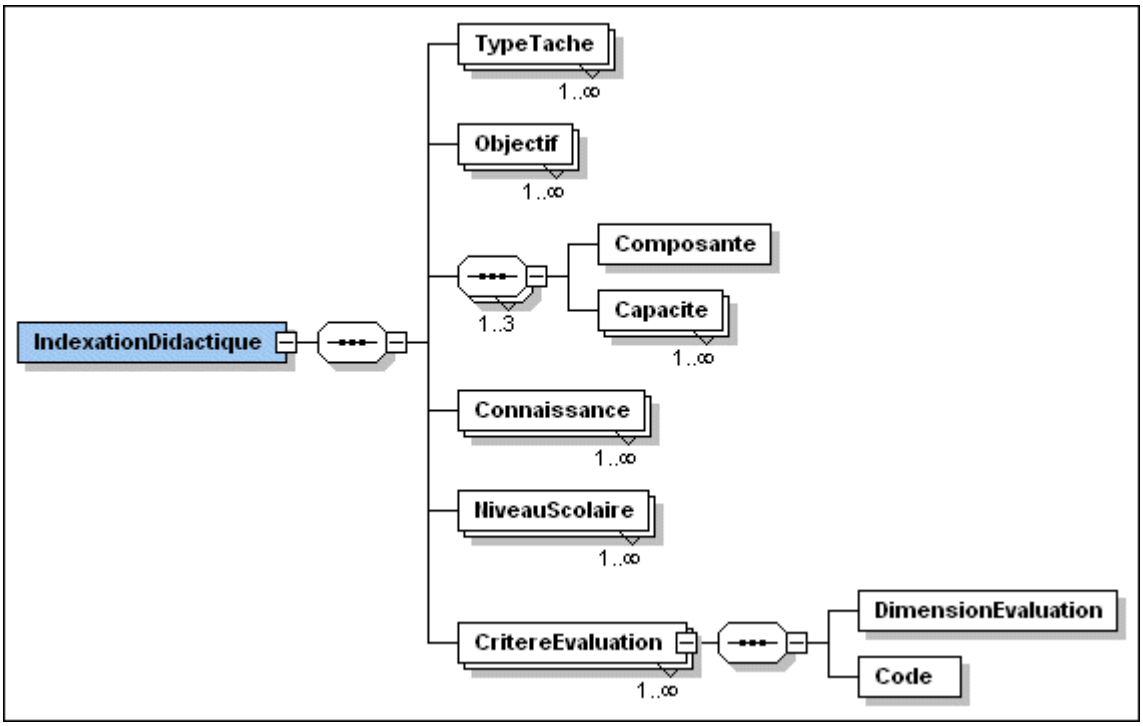
C.7.1. Structure de « ClasseExercice »



C.7.2. Schéma XML de « ClasseExercice »

```
<xs:element name="ClasseExercice">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="IndexationDidactique"/>
      <xs:element ref="GenerationEnonce"/>
      <xs:element ref="InterfaceAbstraite"/>
      <xs:element ref="GenerationGrilleCodage"/>
    </xs:sequence>
    <xs:attribute name="Nom" type="xs:string" use="required"/>
    <xs:attribute name="Numero" type="xs:int" use="required"/>
  </xs:complexType>
</xs:element>
```

C.7.3. Structure de « IndexationDidactique »



C.7.4. Schéma XML de « IndexationDidactique »

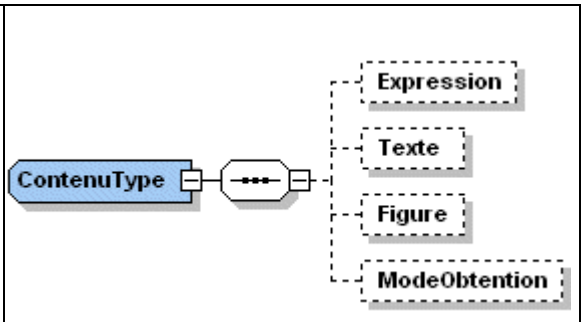
```
<xs:element name="IndexationDidactique">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="TypeTache" maxOccurs="unbounded"/>
      <xs:element name="Objectif" maxOccurs="unbounded"/>
      <xs:sequence maxOccurs="3">
        <xs:element name="Composante"/>
        <xs:element name="Capacite" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:element name="Connaissance" maxOccurs="unbounded"/>
      <xs:element name="NiveauScolaire" maxOccurs="unbounded"/>
      <xs:element name="CritereEvaluation" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="DimensionEvaluation"/>
            <xs:element name="Code"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

C.7.5. Schéma XML de « GenerationEnonce »

Le modèle conceptuel de la génération des énoncés comporte une classe abstraite **Contenu** dont les sous-classes concrètes sont **ContenuSimple** et **ContenuComplexe**. Cette classe abstraite modélise le contenu d'un paramètre, d'un invariant mais aussi d'un énoncé. Nous traduisons cette généralisation par la création d'un type « ContenuType » dans le schéma XML.

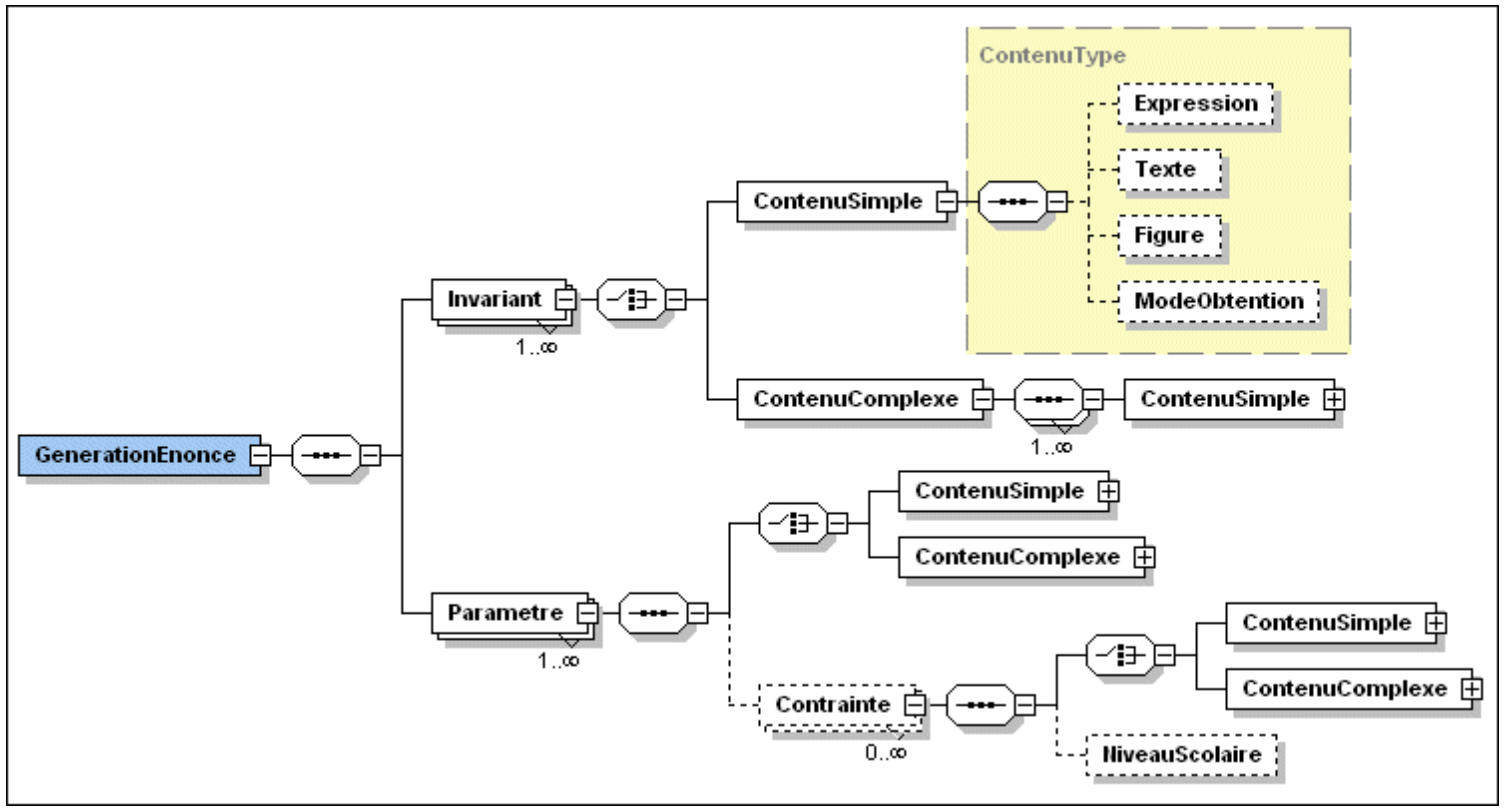
C.7.5.1. Schéma XML correspondant au type « ContenuType »

```
<xs:complexType name="ContenuType">
  <xs:sequence>
    <xs:element name="Expression" minOccurs="0"/>
    <xs:element name="Texte" minOccurs="0"/>
    <xs:element name="Figure" minOccurs="0"/>
    <xs:element name="ModeObtention" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```



C.7.5.2. Structure de « GenerationEnonce »

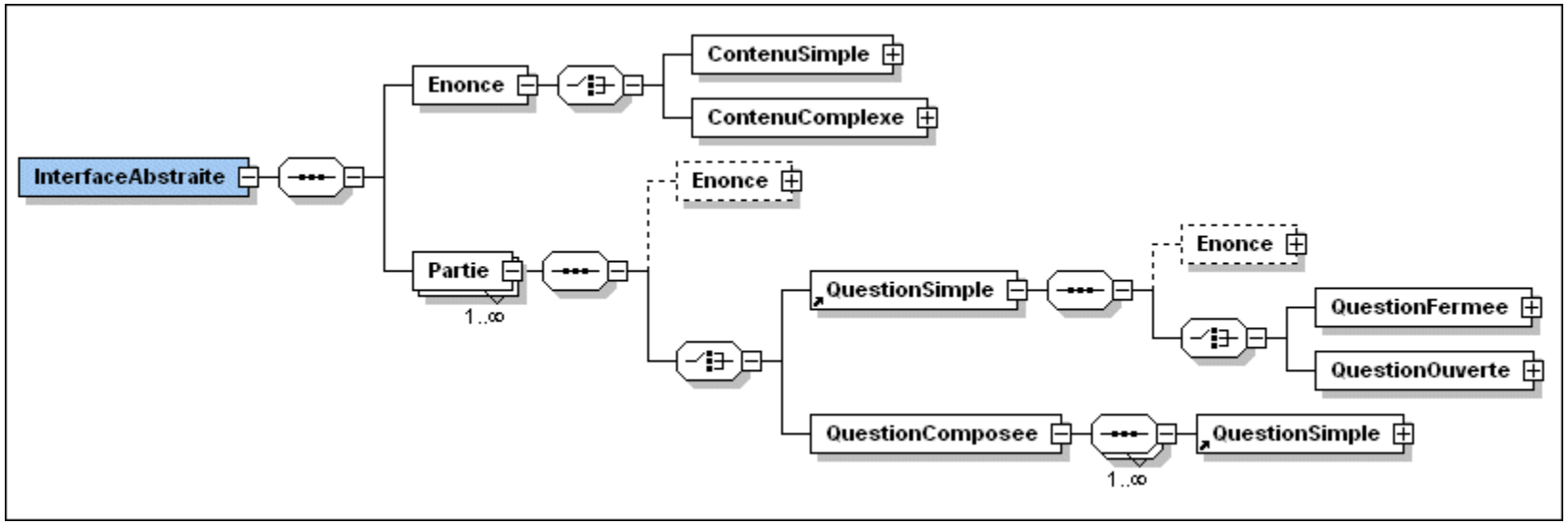
Un contenu simple (**ContenuSimple**) est de type « ContenuType ». Poure cette struture, nous n'avons pas développé toutes les classes. Nous avons seulement développé les deux branches de la classe **Invariant** pour montrer commment été structuré les classes **ContenuSimple** et **ContenuComplexe.**



C.7.5.3. Schéma XML de « GenerationEnonce »

```
<xs:element name="GenerationEnonce">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Invariant" maxOccurs="unbounded">
        <xs:complexType>
          <xs:choice>
            <xs:element name="ContenuSimple" type="ContenuType"/>
            <xs:element name="ContenuComplexe">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="ContenuSimple" type="ContenuType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
          <xs:attribute name="identificateur" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="Parametre" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:choice>
              <xs:element name="ContenuSimple" type="ContenuType"/>
              <xs:element name="ContenuComplexe">
                <xs:complexType>
                  <xs:sequence maxOccurs="unbounded">
                    <xs:element name="ContenuSimple" type="ContenuType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:choice>
            <xs:element name="Contrainte" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:choice>
                    <xs:element name="ContenuSimple" type="ContenuType"/>
                    <xs:element name="ContenuComplexe">
                      <xs:complexType>
                        <xs:sequence maxOccurs="unbounded">
                          <xs:element name="ContenuSimple" type="ContenuType"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:choice>
                  <xs:element name="NiveauScolaire" minOccurs="0"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="Identificateur" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>  </xs:element>
```

C.7.6. Structure de « InterfaceAbstraite »



C.7.7. Schéma XML de « InterfaceAbstraite »

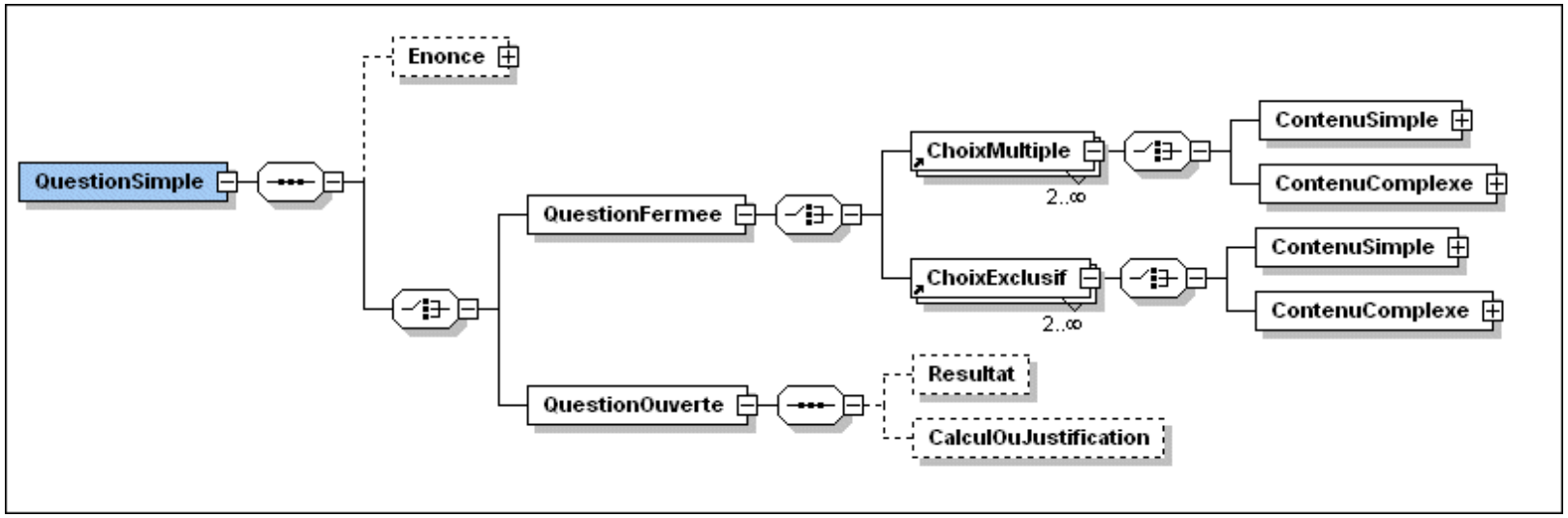
```
<xs:element name="InterfaceAbstraite">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Enonce">
        <xs:complexType>
          <xs:choice>
            <xs:element name="ContenuSimple">
              <xs:complexType>
                <xs:complexContent>
                  <xs:extension base="ContenuType"/>
                </xs:complexContent>
              </xs:complexType>
            </xs:element>
            <xs:element name="ContenuComplexe">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="ContenuSimple">
                    <xs:complexType>
                      <xs:complexContent>
                        <xs:extension base="ContenuType"/>
                      </xs:complexContent>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
```

```
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
    <xs:element name="Partie" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Enonce" minOccurs="0">
            <xs:complexType>
              <xs:choice>
                <xs:element name="ContenuSimple">
                  <xs:complexType>
                    <xs:complexContent>
                      <xs:extension base="ContenuType"/>
                    </xs:complexContent>
                  </xs:complexType>
                </xs:element>
                <xs:element name="ContenuComplexe">
                  <xs:complexType>
                    <xs:sequence maxOccurs="unbounded">
                      <xs:element name="ContenuSimple">
                        <xs:complexType>
                          <xs:complexContent>
                            <xs:extension base="ContenuType"/>
                          </xs:complexContent>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:choice>
            </xs:complexType>
          </xs:element>
          <xs:choice>
            <xs:element ref="QuestionSimple"/>
            <xs:element name="QuestionComposee">
              <xs:complexType>
```

```
                <xs:sequence maxOccurs="unbounded">
                  <xs:element ref="QuestionSimple"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

Dans la section C.7.7.1. nous détaillons la structure de **QuestionSimple** et dans la section C.7.7.2. le schéa XML correspondant.

C.7.7.1. Structure de « QuestionSimple »



La structure de l'arbre et le schéma XML de la classe QuestionSimple font apparaître les classes **ChoixMultiple** et **ChoixExclusif** que nous détaillons dans les sections C.7.2. et C.7.3.

C.7.7.2. Schéma XML de « QuestionSimple »

```
<xs:element name="QuestionSimple">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="Enonce" minOccurs="0">
         <xs:complexType>
           <xs:choice>
             <xs:element name="ContenuSimple">
               <xs:complexType>
                 <xs:complexContent>
                   <xs:extension base="ContenuType"/>
                 </xs:complexContent>
               </xs:complexType>
             </xs:element>
             <xs:element name="ContenuComplexe">
               <xs:complexType>
                 <xs:sequence maxOccurs="unbounded">
                   <xs:element name="ContenuSimple">
                     <xs:complexType>
                       <xs:complexContent>
                         <xs:extension base="ContenuType"/>
                       </xs:complexContent>
                     </xs:complexType>
                   </xs:element>
                 </xs:sequence>
               </xs:complexType>
             </xs:element>
           </xs:choice>
         </xs:complexType>
       </xs:element>
       <xs:choice>
         <xs:element name="QuestionFermee">
           <xs:complexType>
             <xs:choice>
               <xs:element ref="ChoixMultiple" minOccurs="2"
maxOccurs="unbounded"/>
               <xs:element ref="ChoixExclusif" minOccurs="2"
maxOccurs="unbounded"/>
             </xs:choice>
           </xs:complexType>
         </xs:element>
         <xs:element name="QuestionOuverte">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="Resultat" minOccurs="0">
                 <xs:complexType>
                   <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
                 </xs:complexType>
               </xs:element>
               <xs:element name="CalculOuJustification" minOccurs="0">
                 <xs:complexType>
                   <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
                 </xs:complexType>
               </xs:element>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
       </xs:choice>
     </xs:sequence>
     <xs:attribute name="Identificateur" type="xs:string" use="required"/>
   </xs:complexType>
 </xs:element>
```

C.7.7.3. Schéma XML de « ChoixMultiple

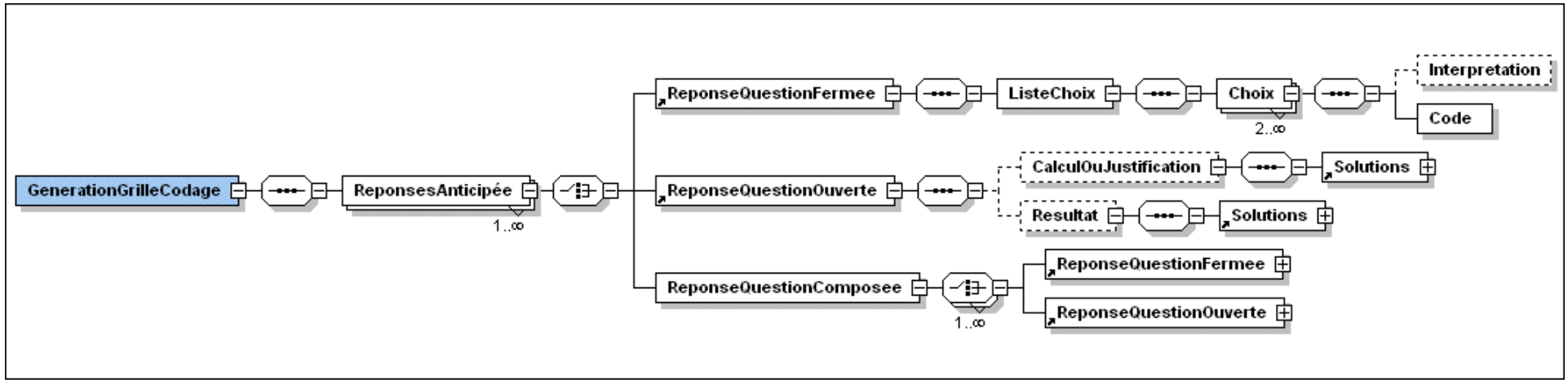
```
<xs:element name="ChoixMultiple">
  <xs:complexType>
    <xs:choice>
      <xs:element name="ContenuSimple">
        <xs:complexType>
          <xs:complexContent>
            <xs:extension base="ContenuType"/>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="ContenuComplexe">
        <xs:complexType>
          <xs:sequence maxOccurs="unbounded">
            <xs:element name="ContenuSimple">
              <xs:complexType>
                <xs:complexContent>
                  <xs:extension base="ContenuType"/>
                </xs:complexContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:choice>
    <xs:attribute name="Identificateur" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

C.7.7.4. Schéma XML de « ChoixExclusif »

```
<xs:element name="ChoixExclusif">
  <xs:complexType>
    <xs:choice>
      <xs:element name="ContenuSimple">
        <xs:complexType>
          <xs:complexContent>
            <xs:extension base="ContenuType"/>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="ContenuComplexe">
        <xs:complexType>
          <xs:sequence maxOccurs="unbounded">
            <xs:element name="ContenuSimple">
              <xs:complexType>
                <xs:complexContent>
                  <xs:extension base="ContenuType"/>
                </xs:complexContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:choice>
    <xs:attribute name="Identificateur" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

C.7.8. Classe « GenerationGrilleCodage »

C.7.8.1. Structure de «GenerationGrilleCodage »



La structure de l'arbre et le schéma XML de la classe **GenerationGrilleCodage** font apparaître les classes **ReponseQuestionFermee**, **ReponseQuestionOuverte** et **Solutions** que nous détaillons dans les sections C.8.2. et C.8.3. et C.8.4

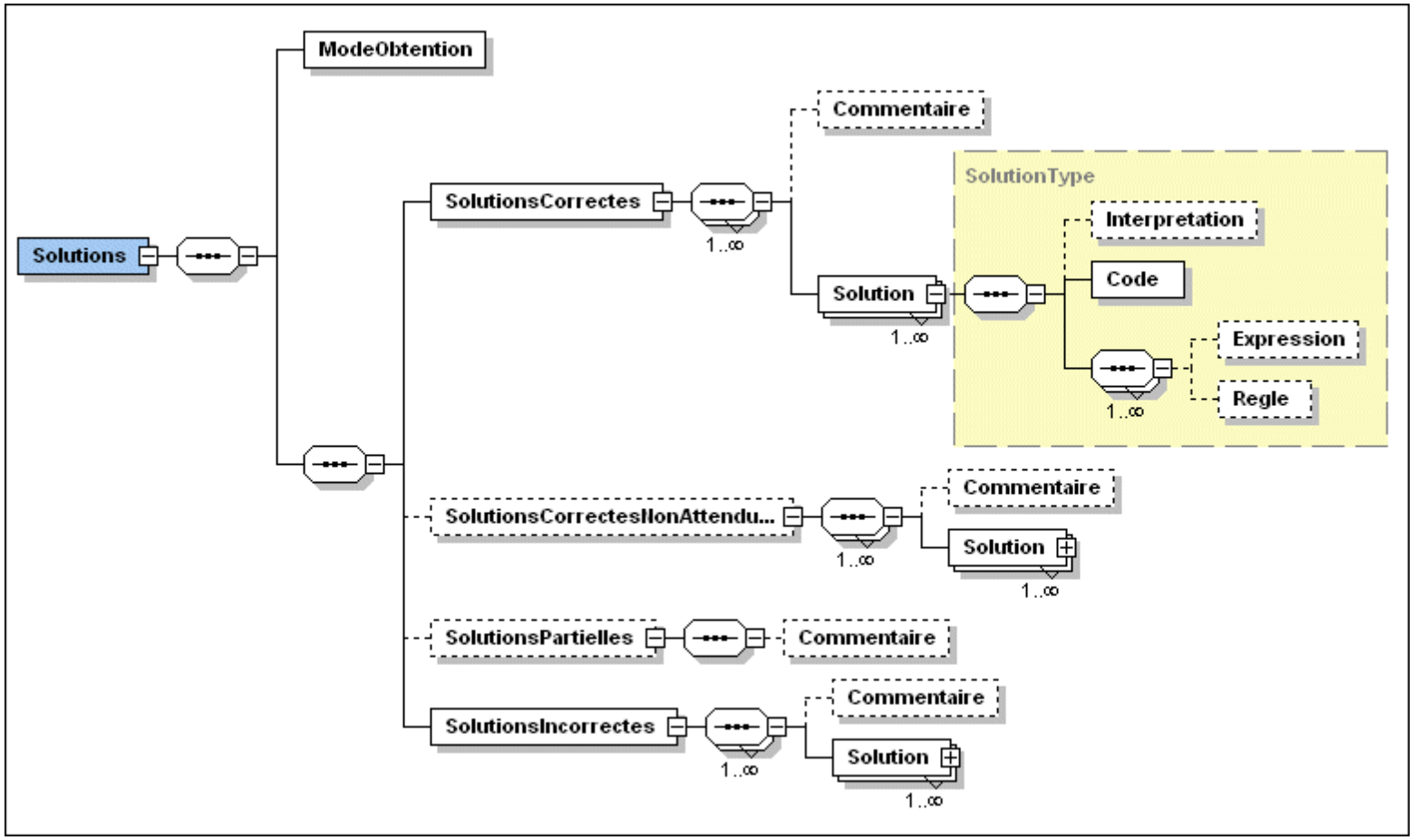
C.7.8.2. Schéma XML de « GenerationGrilleCodage »

```
<xs:element name="GenerationGrilleCodage">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ReponsesAnticipée" maxOccurs="unbounded">
        <xs:complexType>
          <xs:choice>
            <xs:element ref="ReponseQuestionFermee"/>
            <xs:element ref="ReponseQuestionOuverte"/>
            <xs:element name="ReponseQuestionComposee">
              <xs:complexType>
                <xs:choice maxOccurs="unbounded">
                  <xs:element ref="ReponseQuestionFermee"/>
                  <xs:element ref="ReponseQuestionOuverte"/>
                </xs:choice>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

C.7.8.3. Schéma XML de « ReponseQuestionFermee »

```
<xs:element name="ReponseQuestionFermee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ListeChoix">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Choix" minOccurs="2" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Interpretation" minOccurs="0"/>
                  <xs:element name="Code"/>
                </xs:sequence>
                <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Identificateur" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

C.7.8.4. Structure de « Solutions »

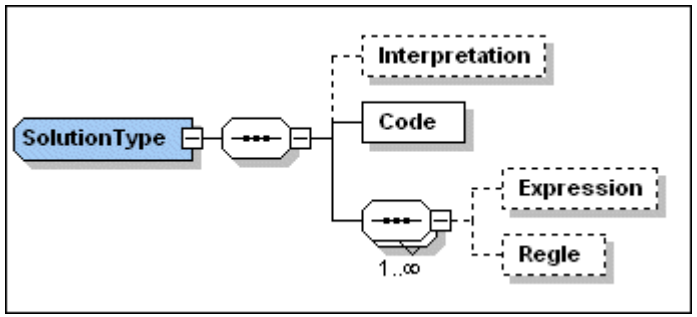


C.7.8.5. Schéma XML de « Solutions »

```
<xs:element name="Solutions">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="ModeObtention"/>
       <xs:sequence>
         <xs:element name="SolutionsCorrectes">
           <xs:complexType>
             <xs:sequence maxOccurs="unbounded">
               <xs:element name="Commentaire" minOccurs="0"/>
               <xs:element name="Solution" type="SolutionType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="SolutionsCorrectesNonAttendues" minOccurs="0">
           <xs:complexType>
             <xs:sequence maxOccurs="unbounded">
               <xs:element name="Commentaire" minOccurs="0"/>
               <xs:element name="Solution" type="SolutionType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="SolutionsPartielles" minOccurs="0">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="Commentaire" minOccurs="0"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="SolutionsIncorrectes">
           <xs:complexType>
             <xs:sequence maxOccurs="unbounded">
               <xs:element name="Commentaire" minOccurs="0"/>
               <xs:element name="Solution" type="SolutionType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
       </xs:sequence>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
```

C.7.8.6. Structure de « SolutionType »

Le modèle conceptuel de la génération des énoncés comporte une classe abstraite **Solution** dont les sous-classes concrètes sont **ContenuSimple** et **ContenuComplexe**. Cette classe abstraite modélise le contenu d'un paramètre, d'un invariant mais aussi d'un énoncé. Nous traduisons cette généralisation par la création d'un type « ContenuType » dans le schéma XML.

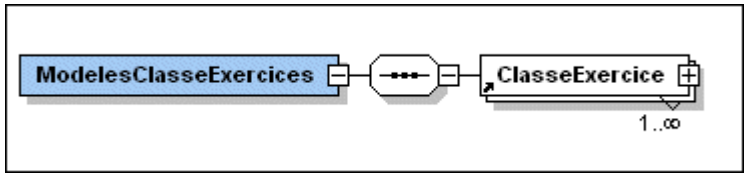


C.7.8.7. Schema XML de « SolutionType »

```
<xs:complexType name="SolutionType">
   <xs:sequence>
     <xs:element name="Interpretation" minOccurs="0"/>
     <xs:element name="Code"/>
     <xs:sequence maxOccurs="unbounded">
       <xs:element name="Expression" minOccurs="0"/>
       <xs:element name="Regle" minOccurs="0"/>
     </xs:sequence>
   </xs:sequence>
 </xs:complexType>
```

C.8. Structure de « ModelesClasseExercices »

Les instances de **ClasseExercice** sont stockées dans un fichier XML dont la structure de l'arbre est représentée ci-dessous.



Le fichier XML de la section C.9. contient deux modèles de classes d'exercices : la classes d'exercices « Correspondance entre aire et expression » et la classe d'exercices « Preuve et programme de calucl ».

C.9. Fichier XML contenant deux modèles d'exercices

```
<?xml version="1.0" encoding="UTF-8"?>
<ModelesClasseExercices>
  <ClasseExercice Numero="13" Nom="Correspondance entre aire et expression">
    <IndexationDidactique>
      <TypeTache>Calculer l'aire d'un domaine plan</TypeTache>
      <TypeTache>Associer une expression algébrique comme une aire de surface</TypeTache>
      <Objectif>Rechercher si un élève sait associer une expression algébrique à un domaine plan</Objectif>
      <Composante>Traduire algébriquement dans différents cadres et vice-versa</Composante>
      <Capacite>Traduire une expression algébrique comme une aire de surface</Capacite>
      <Connaissance></Connaissance>
      <NiveauScolaire>4,3</NiveauScolaire>
      <CritereEvaluation>
        <DimensionEvaluation>Validité</DimensionEvaluation>
        <Code>V1,V3</Code>
          </CritereEvaluation>
      <CritereEvaluation>
        <DimensionEvaluation>Traduction</DimensionEvaluation>
            <Code>T1,T3</Code>
      </CritereEvaluation>
    </IndexationDidactique>
    <GenerationEnonce>
      <Invariant identificateur="i1">
        <ContenuSimple><Figure>F3</Figure></ContenuSimple>
      </Invariant>
      <Parametre Identificateur="p1">
        <ContenuSimple>
          <Figure>F1</Figure>
          <ModeObtention>classe13.genere(F1)</ModeObtention>
        </ContenuSimple>
      </Parametre>
      <Parametre Identificateur="p2">
        <ContenuSimple>
          <Figure>F2</Figure>
          <ModeObtention>classe13.genere(F2)</ModeObtention>
          </ContenuSimple>
          <Contrainte>
            <ContenuSimple><Texte>inclus(F1,F2)</Texte></ContenuSimple>
          </Contrainte>
          <Contrainte>
            <ContenuSimple><Texte>inclus(F2,F3)</Texte></ContenuSimple>
          </Contrainte>
        </Parametre>
        <Parametre Identificateur="p3">
    <ContenuSimple><Expression>classe13.genere(l1)</Expression>
    </ContenuSimple>
        </Parametre>
        <Parametre Identificateur="p4">
          <ContenuSimple><Expression>classe13.genere(l2)</Expression>
          </ContenuSimple>
          <Contrainte>
            <ContenuSimple><Texte>précéde(l1,l2)</Texte></ContenuSimple>
          </Contrainte>
        </Parametre>
        <Parametre Identificateur="p5">

    <ContenuSimple><Expression>classe13.genere(c)</Expression></ContenuSimple
>
          <Contrainte>
          <ContenuSimple><Expression>1&lt;c&lt;9</Expression></ContenuSimple>
          </Contrainte>
        </Parametre>
        <Parametre Identificateur="p6">
    <ContenuSimple><Expression>classe13.genereAire(F1)</Expression>
</ContenuSimple>
        </Parametre>
      </GenerationEnonce>
      <InterfaceAbstraite>
        <Enonce>
          <ContenuSimple>
            <Expression></Expression>
```

```
      <Figure></Figure>
    </ContenuSimple>
  </Enonce>
  <Partie>
    <QuestionSimple Identificateur="q1">
      <QuestionFermee>
        <ChoixMultiple Identificateur="c1">
          <ContenuSimple><Expression>aire1</Expression></ContenuSimple>
        </ChoixMultiple>
        <ChoixMultiple Identificateur="c2">
          <ContenuSimple><Expression>aire2</Expression></ContenuSimple>
        </ChoixMultiple>
        <ChoixMultiple Identificateur="c3">
          <ContenuSimple><Expression>aire3</Expression></ContenuSimple>
        </ChoixMultiple>
      </QuestionFermee>
    </QuestionSimple>
  </Partie>
</InterfaceAbstraite>
<GenerationGrilleCodage>
  <ReponsesAnticipée>
    <ReponseQuestionFermee Identificateur="q1">
    <ListeChoix>
      <Choix Identificateur="c1">
      <Interpretation>La somme des aires est égale à l'expression</Interpretation>
         <Code>V1T1</Code>
      </Choix>
      <Choix Identificateur="c2">
      <Interpretation>La somme des aires est égale à l'expression</Interpretation>
         <Code>V1T1</Code>
      </Choix>
      <Choix Identificateur="c3">
        <Code>V3T3</Code>
      </Choix>
    </ListeChoix>
    </ReponseQuestionFermee>
  </ReponsesAnticipée>
</GenerationGrilleCodage>
```

```
</ClasseExercice>
<ClasseExercice Numero="16" Nom="Preuve et programme de calcul">
  <IndexationDidactique>
    <TypeTache>Produire une expression littérale à partir d'un programme de calcul et prouver qu'une affirmation est vraie </TypeTache>
    <Objectif>Etudier le type de preuves utilisé (par l'exemple ou preuve algébrique), type de traitement algébrique mobilisé, articulation entre différents cadres, signification du signe égal </Objectif>
    <Composante>Utiliser l'algèbre pour résoudre un problème</Composante>
    <Capacite>Produire une expression littérale, Démontrer des régles de calcul, des propriétés, des identités</Capacite>
    <Composante>Traduire dans différents cadres et vice-versa</Composante>
    <Capacite>Traduire un programme de calcul en langage naturel en une expression algébrique</Capacite>
    <Composante>Effecter du calcul algébrique</Composante>
    <Capacite>Réduire une expression littérale</Capacite>
    <Connaissance></Connaissance>
    <NiveauScolaire>3</NiveauScolaire>
    <CritereEvaluation>
      <DimensionEvaluation>Validité</DimensionEvaluation>
      <Code>V1,V2,V3</Code>
    </CritereEvaluation>
    <CritereEvaluation>
      <DimensionEvaluation>Utilisation des lettres</DimensionEvaluation>
      <Code>L1,L2,L3,L5</Code>
    </CritereEvaluation>
    <CritereEvaluation>
      <DimensionEvaluation>Utilisation des règles d'écriture et de réécriture algébrique</DimensionEvaluation>
      <Code>EA1,EA2,EA31,EA32,EA33,EA41,EA42</Code>
    </CritereEvaluation>
    <CritereEvaluation>
      <DimensionEvaluation>Traduction</DimensionEvaluation>
      <Code>T1,T2,T3,T4</Code>
    </CritereEvaluation>
    <CritereEvaluation>
      <DimensionEvaluation>Type de justification</DimensionEvaluation>
      <Code>J1,J2,J3</Code>
```

```
    </CritereEvaluation>
  </IndexationDidactique>
  <GenerationEnonce>
    <Invariant identificateur="i1">
      <ContenuSimple>
        <Texte>Palette de termes </Texte>
      </ContenuSimple>
    </Invariant>
    <Parametre Identificateur="p1">
      <ContenuSimple>
        <Texte></Texte>
        <ModeObtention>classe16.generer(programme)</ModeObtention>
      </ContenuSimple>
    </Parametre>
    <Parametre Identificateur="p2">
      <ContenuSimple>
        <Expression></Expression>
        <ModeObtention>classe16.generer(expression)</ModeObtention>
      </ContenuSimple>
      <Contrainte>
        <ContenuSimple>
          <Expression>nombreParenthése=1</Expression>
        </ContenuSimple>
        <NiveauScolaire>4</NiveauScolaire>
      </Contrainte>
      <Contrainte>
        <ContenuSimple>
          <Expression>nombreParenthése=2</Expression>
        </ContenuSimple>
        <NiveauScolaire>3</NiveauScolaire>
      </Contrainte>
    </Parametre>
    <Parametre Identificateur="p3">
      <ContenuSimple>
        <Expression></Expression>
        <ModeObtention>classe16.generer(formeRéduite)</ModeObtention>
      </ContenuSimple>
      <Contrainte>
        <ContenuSimple>
          <Expression>estaffine(formeRreduite)</Expression>
        </ContenuSimple>
        <NiveauScolaire>2</NiveauScolaire>
      </Contrainte>
      <Contrainte>
        <ContenuSimple>
          <Expression>estConstante(formeRreduite)</Expression>
        </ContenuSimple>
        <NiveauScolaire>3</NiveauScolaire>
      </Contrainte>
    </Parametre>
  </GenerationEnonce>
  <InterfaceAbstraite>
    <Enonce>
      <ContenuSimple>
        <Texte>p1</Texte>
      </ContenuSimple>
    </Enonce>
    <Partie>
    <QuestionComposee>
      <QuestionSimple Identificateur="q1">
        <QuestionFermee>
        <ChoixExclusif Identificateur="c1">
          <ContenuSimple>
            <Texte>Vrai</Texte>
          </ContenuSimple>
        </ChoixExclusif>
        <ChoixExclusif Identificateur="c2">
          <ContenuSimple>
            <Texte>Faux</Texte>
          </ContenuSimple>
        </ChoixExclusif>
        </QuestionFermee>
      </QuestionSimple>
      <QuestionSimple Identificateur="q2">
        <QuestionOuverte>
          <CalculOuJustification  Identificateur="j1"></CalculOuJustification>
```

```
        </QuestionOuverte>
      </QuestionSimple>
    </QuestionComposee>
    </Partie>
  </InterfaceAbstraite>
  <GenerationGrilleCodage>
    <ReponsesAnticipée>
    <ReponseQuestionComposee>
      <ReponseQuestionFermee Identificateur="q1" >
        <ListeChoix>
          <Choix Identificateur="c1">
            <Code>V1</Code>
          </Choix>
          <Choix Identificateur="c2">
            <Code>V3</Code>
          </Choix>
        </ListeChoix>
      </ReponseQuestionFermee>
      <ReponseQuestionOuverte Identificateur="q2">
        <CalculOuJustification Identificateur="j1">
          <Solutions>
            <ModeObtention>classe16.algorithmediagnostic</ModeObtention>
            <SolutionsCorrectes>
              <Commentaire>Preuve algébrique avec une expression globale (parenthésée) traduisant le résultat de l'enchainement opératoire</Commentaire>
              <Solution>
                <Interpretation>Pour les expressions comportant des fractions on réduit au même dénominateur</Interpretation>
                <Code>V1,L1,EA1,T1,J1</Code>
              </Solution>
              <Solution>
                <Interpretation>Pour les expressions comportant des fractions on simplifie</Interpretation>
                <Code>V1,L1,EA2,T1,J1</Code>
              </Solution>
              <Solution>
                <Interpretation>Pour les expressions ne comportant pas de fractions</Interpretation>
                <Code>V1,L1,EA1,T1,J1</Code>
              </Solution>
            </SolutionsCorrectes>
            <SolutionsCorrectesNonAttendues>
              <Commentaire>Preuve algébrique avec des expressions partielles traduisant pas à pas les résultats intermédiaires de l'enchainement opératoire
              </Commentaire>
              <Solution>
                <Interpretation></Interpretation>
                <Code>V2,L1,EA1,T2,J1</Code>
              </Solution>
              <Commentaire>Preuve algébrique avec l’interprétation de l’énoncé comme une équation admettant une infinité de solutions.
              </Commentaire>
              <Solution>
                <Interpretation></Interpretation>
                <Code>V2,EA1,L1,T1,J1</Code>
              </Solution>
            </SolutionsCorrectesNonAttendues>
            <SolutionsPartielles>
              <Commentaire>Une preuve algébrique est engagée avec l'une des trois interprétations précédentes. Mais une erreur de calcul opératoire conduit à un résultat faux ou à un abandon </Commentaire>
            </SolutionsPartielles>
            <SolutionsIncorrectes>
              <Commentaire>Les solutions utilisent une démarche algébrique</Commentaire>
              <Solution>
                <Interpretation>L’interprétation de l’énoncé conduit à une traduction algébrique de l’enchaînement opératoire avec une expression globale non ésée</Interpretation>
                <Code>V3,L3,T3,J3</Code>
              </Solution>
              <Solution>
                <Interpretation>Expressions partielles avec écriture pas à pas enchaînée en succession d’opérations</Interpretation>
                <Code>V3,L3,T4,J3</Code>
              </Solution>
```

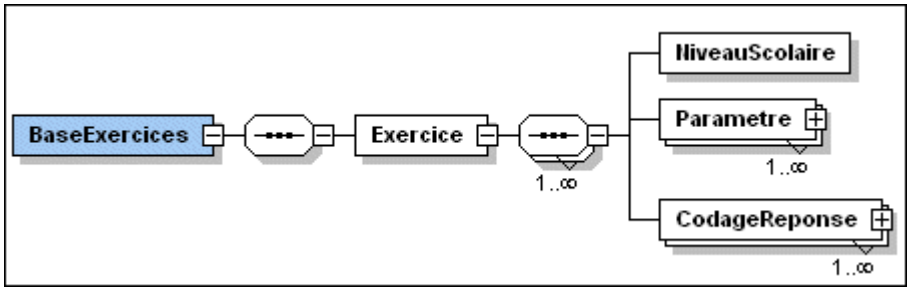
```
            <Commentaire>Il y a manipulation formelle</Commentaire>
            <Solution>
              <Interpretation>avec mémoire de l'énoncé</Interpretation>
              <Code>V3,EA31,L2,T3,J2</Code>
            </Solution>
            <Solution>
              <Interpretation>sans mémoire de l'énoncé</Interpretation>
              <Code>V3,EA32,L2,T3,J2</Code>
            </Solution>
            <Solution>
              <Interpretation>avec erreur d'application d'une régle de calcul</Interpretation>
              <Code>V3,EA33,L2,T3,J2</Code>
            </Solution>
            <Solution>
              <Interpretation>Les régles de transformations utilisées « assemblent » les termes</Interpretation>
              <Code>V3,EA42,L2,T3,J2</Code>
            </Solution>
            <Commentaire>Démarche non algébrique (preuve par l'exemple). Les solutions mettent en jeu des écritures correctes</Commentaire>
            <Solution>
              <Interpretation>expression globale parenthésée</Interpretation>
              <Code>V3,EA1,L5,T1,J2</Code>
            </Solution>
            <Solution>
              <Interpretation>expression partiellee</Interpretation>
              <Code>V3,EA1,L5,T2,J2</Code>
            </Solution>
            <Commentaire>Démarche non algébrique (preuve par l'exemple). Les solutions mettent en jeu des écritures incorrectes</Commentaire>
            <Solution>
              <Interpretation>écriture pas à pas enchainée en succession d'opérations</Interpretation>
              <Code>V3,L5,T4,J2</Code>
            </Solution>
            <Solution>
              <Interpretation>écriture globale non parenthésée</Interpretation>
              <Code>V3,L5,T3,J2</Code>
            </Solution>
            <Solution>
              <Interpretation>avec mémoire</Interpretation>
              <Code>EA31</Code>
            </Solution>
            <Solution>
              <Interpretation>sans mémoire</Interpretation>
              <Code>EA32</Code>
            </Solution>
          </SolutionsIncorrectes>
        </Solutions>
        </CalculOuJustification>
        <Resultat Identificateur="">
          <Solutions>
            <ModeObtention></ModeObtention>
            <SolutionsCorrectes>
              <Commentaire></Commentaire>
              <Solution>
                <Interpretation></Interpretation>
                <Code></Code>
                <Expression></Expression>
                <Regle></Regle>
              </Solution>
            </SolutionsCorrectes>
            <SolutionsCorrectesNonAttendues>
              <Commentaire></Commentaire>
              <Solution>
                <Interpretation></Interpretation>
                <Code></Code>
                <Expression></Expression>
                <Regle></Regle>
              </Solution>
            </SolutionsCorrectesNonAttendues>
            <SolutionsPartielles>
              <Commentaire></Commentaire>
            </SolutionsPartielles>
            <SolutionsIncorrectes>
```

```
            <Commentaire></Commentaire>
            <Solution>
              <Interpretation></Interpretation>
              <Code></Code>
              <Expression></Expression>
              <Regle></Regle>
            </Solution>
          </SolutionsIncorrectes>
        </Solutions>
      </Resultat>
    </ReponseQuestionOuverte>
    </ReponseQuestionComposee>
       </ReponsesAnticipée>
  </GenerationGrilleCodage>
 </ClasseExercice>
</ModelesClasseExercices>
```

C.10. Base d'exercices

Le fichier XML **BaseExercices** contient tous les clones d'exercices.

C.10.1. Structure de la base d'exercices

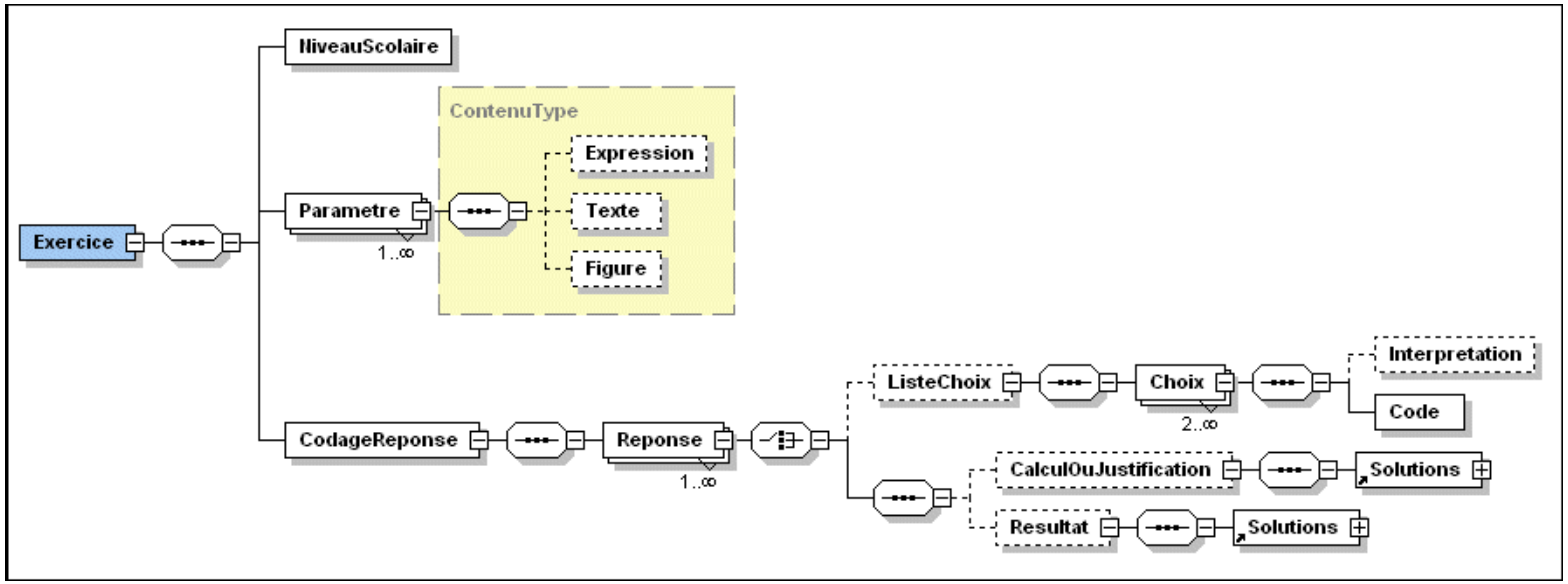


C.10.2. Schéma XMLde la base d'exercices

```
<xs:element name="BaseExercices">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Exercice" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
```

Nous avons utilisé des éléments du schéma XML de **ClasseExercice** en particulier les types **ContenuType** et **SolutionType**

C.10.3. Structure de « Exercices »

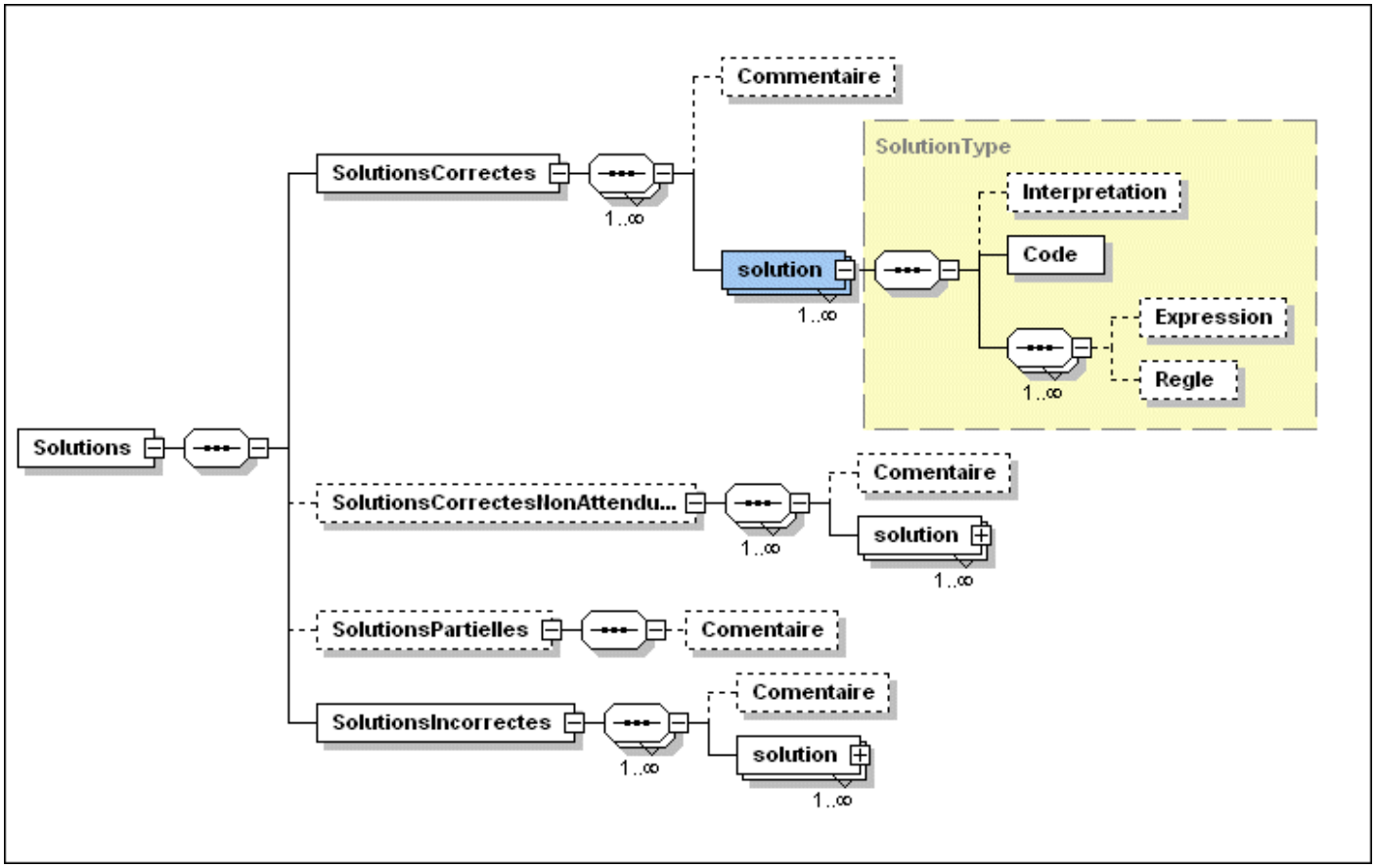


C.10.4. Schéma XMLde « Exercice »

```
   <xs:element name="Exercice">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="NiveauScolaire"/>
       <xs:element name="Parametre" type="ContenuType"
maxOccurs="unbounded"/>
       <xs:element name="CodageReponse">
         <xs:complexType>
           <xs:sequence>
             <xs:element name="Reponse" maxOccurs="unbounded">
               <xs:complexType>
                 <xs:choice>
                   <xs:element name="ListeChoix" minOccurs="0">
                     <xs:complexType>
                       <xs:sequence>
                         <xs:element name="Choix" minOccurs="2"
maxOccurs="unbounded">
                           <xs:complexType>
                             <xs:sequence>
                               <xs:element name="Interpretation" minOccurs="0"/>
                               <xs:element name="Code"/>
                             </xs:sequence>
                             <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
                           </xs:complexType>
                         </xs:element>
                       </xs:sequence>
                     </xs:complexType>
                   </xs:element>
                   <xs:sequence>
                     <xs:element name="CalculOuJustification" minOccurs="0">
                       <xs:complexType>
                         <xs:sequence>
                           <xs:element ref="Solutions"/>
                         </xs:sequence>
                         <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
                       </xs:complexType>
                     </xs:element>
                     <xs:element name="Resultat" minOccurs="0">
                       <xs:complexType>
                         <xs:sequence>
                           <xs:element ref="Solutions"/>
                         </xs:sequence>
                         <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
                       </xs:complexType>
                     </xs:element>
                   </xs:sequence>
                 </xs:choice>
                 <xs:attribute name="Identificateur" type="xs:string"
use="required"/>
               </xs:complexType>
             </xs:element>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
       <xs:attribute name="Nom" type="xs:string" use="required"/>
       <xs:attribute name="Numero" type="xs:int" use="required"/>
       <xs:attribute name="Identificateur" type="xs:string" use="required"/>
     </xs:complexType>
   </xs:element>
```

C.10.5. Structure de Solutions

La structure de l'arbre et le schéma XML de la classe **Exercice** font apparaître la classe **Solutions** que nous détaillons dans cette sections.



Le fichier XML BaseExercices contient tous les clones générés avec PépiGen. Nous présentons dans les sections qui suivent les trois clones d'exercices de la classe « Preuve et programme de calculs » que nous avons extrait de ce fichier pour une meilleure lisibilité.

C.11. Clones générés par PépiGen -Programme de calcul : (x+6)3 -3x

```
<exercices version="date">
   <Exercice Identificateur="18-3-2008-3-51" Nom="Preuve et programme de calcul"
Numero="16">
      <Niveau>4 ième - 3 ième</Niveau>
      <Parametre>
         <Texte>Tu prends un nombre, tu ajoutes 6, tu multiplies le
            résultat par 3, tu soustrais 3 fois le nombre de départ.</Texte>
      </Parametre>
      <Parametre>
         <Expression>(x + 6)* 3 - 3 * x</Expression>
      </Parametre>
      <Parametre>
         <Expression>18</Expression>
      </Parametre>
      <CodageReponse>
         <Reponse Identificateur="r1">
            <ListeChoix>
               <Choix Identificateur="c1">
                  <Code>V1</Code>
               </Choix>
               <Choix Identificateur="c2">
                  <Code>V2</Code>
               </Choix>
            </ListeChoix>
            <CalculOuJustification Identificateur="j1">
               <Solutions>
                  <SolutionsCorrectes>
                     <Commentaire>Preuve algébrique avec une
                        expression globale (parenthésée)
                        traduisant le résultat de
                        l'enchainement opératoire</Commentaire>
                     <Solution>
                        <Code>V1,EA1,L1,T1,J1</Code>
                        <Expression>(x+6)*3-3*x</Expression>
                        <Expression>x*3+6*3-3*x</Expression>
                        <Regle>V,7</Regle>
                        <Expression>x*3+18-3*x</Expression>
                        <Expression>18</Expression>
                        <Regle>V,31</Regle>
                     </Solution>
                  </SolutionsCorrectes>
                  <SolutionsCorrectesNonAttendues>
                     <Commentaire>Preuve algébrique avec des
                        expressions partielles traduisant pas à
                        pas les résultats intermédiaires de
                        l'enchainement opératoire. </Commentaire>
                     <Solution>
                        <Code>V2,L1,EA1,T2,J1</Code>
                        <Expression>(x+6)*3</Expression>
                        <Expression>x*3+18</Expression>
                        <Regle>V,7</Regle>
                        <Expression>3*x+18-3*x</Expression>
                        <Expression>18</Expression>
                        <Regle>V,31</Regle>
                     </Solution>
                     <Commentaire>Preuve algébrique avec
```

```
        l'interprétation de
        l'énoncé comme une équation
        admettant une infinité de solutions.</Commentaire>
      <Solution>
        <Code>V2,EA1,L1,T1,J1</Code>
        <Expression>(x+6)*3-3*x = 18</Expression>
        <Expression>x*3+6*3-3*x = 18</Expression>
        <Regle>V,7</Regle>
        <Expression>x*3+18-3*x = 18</Expression>
        <Expression>18 = 18</Expression>
        <Regle>V,31</Regle>
      </Solution>
    </SolutionsCorrectesNonAttendues>
    <SolutionsPartielles>
      <Commentaire>Une preuve algébrique est
        engagée avec l'une des trois
        interprétations précédentes. Mais une
        erreur de calcul opératoire conduit à un
        résultat faux ou à un abandon </Commentaire>
    </SolutionsPartielles>
    <SolutionsIncorrectes>
      <Commentaire>Les solutions utilisent une
        démarche algébrique.</Commentaire>
      <Solution>
        <Interprétation>L'interprétation
          de l'énoncé conduit à une
          traduction algébrique de
          l'enchaînement opératoire
          avec une expression globale non
          parenthésée: </Interprétation>
        <Code>V3,L3,T3,J3</Code>
        <Expression>x+6*3-3*x</Expression>
      </Solution>
      <Solution>
        <Interprétation>- Expressions partielles
          avec écriture pas à pas enchaînée en
          succession d'opérations</Interprétation>
        <Code>V3,L3,T4,J3</Code>
        <Expression>(x+6)*3</Expression>
        <Expression>x*3+18</Expression>
        <Regle>V,7</Regle>
        <Expression>3*x+18-3*x</Expression>
        <Regle>V,31</Regle>
        <Expression>18</Expression>
      </Solution>
      <Solution>
        <Interprétation> Les régles de
          transformations utilisées «
          assemblent » les termes.</Interprétation>
        <Code>V3,EA42,L3,T3,J3</Code>
        <Expression>(x+6)*3-3*x</Expression>
        <Expression>x*3+6*3-3*x</Expression>
        <Regle>V,7</Regle>
        <Expression>x*3+18-3*x</Expression>
        <Expression>21*x-3*x</Expression>
        <Regle>F,31</Regle>
        <Expression>18*x</Expression>
        <Regle>V,31</Regle>
      </Solution>
      <Commentaire>Les solutions correspondent à
        une démarche non algébrique (preuve par
        des exemples). Les lettres ne sont pas disponibles.</Commentaire>
      <Solution>
        <Code>V3,L5,J2</Code>
```

```
                    </Solution>
                  </SolutionsIncorrectes>
                </Solutions>
              </CalculOuJustification>
            </Reponse>
          </CodageReponse>
        </Exercice>
</exercices>
```

C.11.1. Programme de calcul : ((x*3 - 6)*2 - 3*x)/3 - x

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<exercices version="date">
  <Exercice Identificateur="18-3-2008-4-50"
    Nom="Preuve et programme de calcul" Numero="16">
    <Niveau>4 ième - 3 ième</Niveau>
    <Parametre>
      <Texte>Tu prends un nombre, tu multiplies ce nombre par 3,
        tu soustrais 6, tu multiplies le résultat par 2, tu
        soustrais le triple de ton nombre, tu divises le
        résultat par 3, tu soustrais le nombre de départ.</Texte>
    </Parametre>
    <Parametre>
      <Expression>((x * 3 - 6)* 2 - 3 * x)/ 3 - x</Expression>
    </Parametre>
    <Parametre>
      <Expression>-4</Expression>
    </Parametre>
    <CodageReponse>
      <Reponse Identificateur="r1">
        <ListeChoix>
          <Choix Identificateur="c1">
            <Code>V1</Code>
          </Choix>
          <Choix Identificateur="c2">
            <Code>V2</Code>
          </Choix>
        </ListeChoix>
        <CalculOuJustification Identificateur="j1">
          <Solutions>
            <SolutionsCorrectes>
              <Commentaire>Preuve algébrique avec une
                expression globale (parenthésée)
                traduisant le résultat de
                l'enchainement opératoire</Commentaire>
              <Solution>
                <Interprétation>Les fractions sont
                  réduites au même dénominateur.</Interprétation>
                <Code>V1,EA2,L1,T1,J1</Code>
                <Expression>((x*3-6)*2-3*x)/3-x</Expression>
                <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
                <Regle>V,8</Regle>
                <Expression>(6*x-12-3*x)/3-x</Expression>
                <Expression>(3*x-12)/3-x</Expression>
                <Regle>V,31</Regle>
                <Expression>(3*x-12-3*x)/3</Expression>
                <Regle>V,19</Regle>
                <Expression>-4</Expression>
                <Regle>V,31</Regle>
              </Solution>
              <Solution>
                <Interprétation>L'expression est simplifiée.</Interprétation>
                <Code>V1,EA1,L1,T1,J1</Code>
                <Expression>((x*3-6)*2-3*x)/3-x</Expression>
                <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
                <Regle>V,8</Regle>
                <Expression>(6*x-12-3*x)/3-x</Expression>
                <Expression>(3*x-12)/3-x</Expression>
                <Regle>V,31</Regle>
                <Expression>(3/3)*x-12/3-x</Expression>
                <Regle>V,14</Regle>
```

```
            <Expression>x-4-x</Expression>
            <Expression>-4</Expression>
            <Regle>V,31</Regle>
        </Solution>
    </SolutionsCorrectes>
    <SolutionsCorrectesNonAttendues>
        <Commentaire>Preuve algébrique avec des
            expressions partielles traduisant pas à
            pas les résultats intermédiaires de
            l'enchainement opératoire. </Commentaire>
        <Solution>
            <Code>V2,L1,EA1,T2,J1</Code>
            <Expression>(3*x-6)*2</Expression>
            <Expression>6*x-12</Expression>
            <Regle>V,8</Regle>
            <Expression>6*x-12-3*x</Expression>
            <Expression>3*x-12</Expression>
            <Regle>V,31</Regle>
            <Expression>(3*x-12)/3</Expression>
            <Expression>x-4</Expression>
            <Regle>V,14</Regle>
            <Expression>x-4-x</Expression>
            <Expression>-4</Expression>
            <Regle>V,31</Regle>
        </Solution>
        <Commentaire>Preuve algébrique avec
            l'interprétation de
            l'énoncé comme une équation
            admettant une infinité de solutions.</Commentaire>
        <Solution>
            <Interprétation>Les fractions sont
                réduites au même dénominateur.</Interprétation>
            <Code>V2,EA2,L1,T1,J1</Code>
            <Expression>((x*3-6)*2-3*x)/3-x = -4</Expression>
            <Expression>(3*x*2-6*2-3*x)/3-x = -4</Expression>
            <Regle>V,8</Regle>
            <Expression>(6*x-12-3*x)/3-x = -4</Expression>
            <Expression>(3*x-12)/3-x = -4</Expression>
            <Regle>V,31</Regle>
            <Expression>(3*x-12-3*x)/3 = -4</Expression>
            <Regle>V,19</Regle>
            <Expression>-4 = -4</Expression>
            <Regle>V,31</Regle>
        </Solution>
        <Solution>
            <Interprétation>L'expression est simplifiée.</Interprétation>
            <Code>V2,EA1,L1,T1,J1</Code>
            <Expression>((x*3-6)*2-3*x)/3-x = -4</Expression>
            <Expression>(3*x*2-6*2-3*x)/3-x = -4</Expression>
            <Regle>V,8</Regle>
            <Expression>(6*x-12-3*x)/3-x = -4</Expression>
            <Expression>(3*x-12)/3-x = -4</Expression>
            <Regle>V,31</Regle>
            <Expression>(3/3)*x-12/3-x = -4</Expression>
            <Regle>V,14</Regle>
            <Expression>x-4-x = -4</Expression>
            <Expression>-4 = -4</Expression>
            <Regle>V,31</Regle>
        </Solution>
    </SolutionsCorrectesNonAttendues>
    <SolutionsPartielles>
        <Commentaire>Une preuve algébrique est
            engagée avec l'une des trois
            interprétations précédentes. Mais une
```

```
        erreur de calcul opératoire conduit à un
        résultat faux ou à un abandon </Commentaire>
    </SolutionsPartielles>
    <SolutionsIncorrectes>
      <Commentaire>Les solutions utilisent une
        démarche algébrique.</Commentaire>
      <Solution>
        <Interprétation>L'interprétation
          de l'énoncé conduit à une
          traduction algébrique de
          l'enchaînement opératoire
          avec une expression globale non
          parenthésée: </Interprétation>
        <Code>V3,L3,T3,J3</Code>
        <Expression>x*3-6*2-3*x/3-x</Expression>
      </Solution>
      <Solution>
        <Interprétation>- Expressions partielles
          avec écriture pas à pas enchaînée en
          succession d'opérations</Interprétation>
        <Code>V3,L3,T4,J3</Code>
        <Expression>(3*x-6)*2</Expression>
        <Expression>6*x-12</Expression>
        <Regle>V,8</Regle>
        <Expression>6*x-12-3*x</Expression>
        <Expression>3*x-12</Expression>
        <Regle>V,31</Regle>
        <Expression>(3*x-12)/3</Expression>
        <Expression>x-4</Expression>
        <Regle>V,14</Regle>
        <Expression>x-4-x</Expression>
        <Expression>-4</Expression>
```

```
        <Regle>V,31</Regle>
      </Solution>
      <Solution>
        <Interprétation>Les fractions sont
          réduites au même dénominateur.
          Utilisation inadaptée des
          parenthèses avec mémoire de l'énoncé.</Interprétation>
        <Code>V3,EA31,L3,T3,J3</Code>
        <Expression>((x*3-6)*2-3*x)/3-x</Expression>
        <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
        <Regle>V,8</Regle>
        <Expression>(6*x-12-3*x)/3-x</Expression>
        <Expression>(3*x-12)/3-x</Expression>
        <Regle>V,31</Regle>
        <Expression>(3*x-12-x)/3</Expression>
        <Regle>F,19</Regle>
        <Expression>(3*x-12-3*x)/3</Expression>
        <Expression>-4</Expression>
        <Regle>V,31</Regle>
      </Solution>
      <Solution>
        <Interprétation>Les fractions sont
          réduites au même dénominateur.
          Utilisation inadaptée des
          parenthèses sans mémoire de l'énoncé.</Interprétation>
        <Code>V3,EA32,L3,T3,J3</Code>
        <Expression>((x*3-6)*2-3*x)/3-x</Expression>
        <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
        <Regle>V,8</Regle>
        <Expression>(6*x-12-3*x)/3-x</Expression>
        <Expression>(3*x-12)/3-x</Expression>
        <Regle>V,31</Regle>
```

```
    <Expression>(3*x-12-x)/3</Expression>
    <Regle>F,19</Regle>
    <Expression>(2*x-12)/3</Expression>
    <Regle>V,31</Regle>
    <Expression>(2/3)*x-12/3</Expression>
    <Regle>V,14</Regle>
    <Expression>(2/3)*x-4</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>((x*3-6)*2-3*x)/3-x</Expression>
    <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
    <Regle>V,8</Regle>
    <Expression>(6*x-12-3*x)/3-x</Expression>
    <Expression>(3*x-12)/3-x</Expression>
    <Regle>V,31</Regle>
    <Expression>(3*x-12-x)/3</Expression>
    <Regle>F,19</Regle>
    <Expression>(2*x-12)/3</Expression>
    <Regle>V,31</Regle>
    <Expression>2*x-4</Expression>
    <Regle>F,14</Regle>
    <Expression>(2/3)*x-4</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses sans mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA32,L3,T3,J3</Code>
    <Expression>((x*3-6)*2-3*x)/3-x</Expression>
    <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
    <Regle>V,8</Regle>
    <Expression>(6*x-12-3*x)/3-x</Expression>
    <Expression>(3*x-12)/3-x</Expression>
    <Regle>V,31</Regle>
    <Expression>(3*x-12-x)/3</Expression>
    <Regle>F,19</Regle>
    <Expression>(2*x-12)/3</Expression>
    <Regle>V,31</Regle>
    <Expression>2*x-4</Expression>
    <Regle>F,14</Regle>
</Solution>
<Solution>
    <Interprétation>L'expression est
        simplifiée. Utilisation inadaptée
        des parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>((x*3-6)*2-3*x)/3-x</Expression>
    <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
    <Regle>V,8</Regle>
    <Expression>(6*x-12-3*x)/3-x</Expression>
    <Expression>(3*x-12)/3-x</Expression>
    <Regle>V,31</Regle>
    <Expression>3*x-4-x</Expression>
    <Regle>F,14</Regle>
    <Expression>x-4-x</Expression>
    <Expression>-4</Expression>
    <Regle>V,31</Regle>
</Solution>
```

```
                    <Solution>
                        <Interprétation>L'expression est
                            simplifiée. Utilisation inadaptée
                            des parenthèses sans mémoire de l'énoncé.</Interprétation>
                        <Code>V3,EA32,L3,T3,J3</Code>
                        <Expression>((x*3-6)*2-3*x)/3-x</Expression>
                        <Expression>(3*x*2-6*2-3*x)/3-x</Expression>
                        <Regle>V,8</Regle>
                        <Expression>(6*x-12-3*x)/3-x</Expression>
                        <Expression>(3*x-12)/3-x</Expression>
                        <Regle>V,31</Regle>
                        <Expression>3*x-4-x</Expression>
                        <Regle>F,14</Regle>
                        <Expression>2*x-4</Expression>
                        <Regle>V,31</Regle>
                    </Solution>
                    <Commentaire>Les solutions correspondent à
                        une démarche non algébrique (preuve par
                        des exemples). Les lettres ne sont pas disponibles.</Commentaire>
                    <Solution>
                        <Code>V3,L5,J2</Code>
                    </Solution>
                </SolutionsIncorrectes>
            </Solutions>
        </CalculOuJustification>
    </Reponse>
  </CodageReponse>
</Exercice>
</exercices>
```

C.11.2. Programme de calcul : (x + 3 * x + 4)/ 4 –1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<exercices version="date">
  <Exercice Identificateur="18-3-2008-4-53"
    Nom="Preuve et programme de calcul" Numero="16">
    <Niveau>4 ième - 3 ième</Niveau>
    <Parametre>
      <Texte>Tu prends un nombre, tu ajoutes le triple de ce
        nombre, tu ajoutes 4, tu divises le résultat par 4, tu
        soustrais 1.</Texte>
    </Parametre>
    <Parametre>
      <Expression>(x + 3 * x + 4)/ 4 - 1</Expression>
    </Parametre>
    <Parametre>
      <Expression>x</Expression>
    </Parametre>
    <CodageReponse>
      <Reponse Identificateur="r1">
        <ListeChoix>
          <Choix Identificateur="c1">
            <Code>V1</Code>
          </Choix>
          <Choix Identificateur="c2">
            <Code>V2</Code>
          </Choix>
        </ListeChoix>
        <CalculOuJustification Identificateur="j1">
          <Solutions>
            <SolutionsCorrectes>
              <Commentaire>Preuve algébrique avec une
                expression globale (parenthésée)
                traduisant le résultat de
                l'enchainement opératoire</Commentaire>
              <Solution>
                <Interprétation>Les fractions sont
                  réduites au même dénominateur.</Interprétation>
                <Code>V1,EA2,L1,T1,J1</Code>
                <Expression>(x+3*x+4)/4-1</Expression>
                <Expression>(4*x+4)/4-1</Expression>
                <Regle>V,31</Regle>
                <Expression>(4*x+4-4*1)/4</Expression>
                <Regle>V,19</Regle>
                <Expression>(4*x)/4</Expression>
                <Expression>(4/4)*x4</Expression>
                <Regle>V,13</Regle>
                <Expression>x</Expression>
              </Solution>
              <Solution>
                <Interprétation>L'expression est simplifiée.</Interprétation>
                <Code>V1,EA1,L1,T1,J1</Code>
                <Expression>(x+3*x+4)/4-1</Expression>
                <Expression>(4*x+4)/4-1</Expression>
                <Regle>V,31</Regle>
                <Expression>(4/4)*x+4/4-1</Expression>
                <Regle>V,13</Regle>
                <Expression>x</Expression>
              </Solution>
            </SolutionsCorrectes>
            <SolutionsCorrectesNonAttendues>
              <Commentaire>Preuve algébrique avec des
```

```
    expressions partielles traduisant pas à
    pas les résultats intermédiaires de
    l'enchainement opératoire. </Commentaire>
  <Solution>
    <Code>V2,L1,EA1,T2,J1</Code>
    <Expression>x+3*x</Expression>
    <Expression>4*x</Expression>
    <Regle>V,13</Regle>
    <Expression>(4*x+4)/4</Expression>
    <Expression>x+1</Expression>
    <Regle>V,13</Regle>
    <Expression>x+1-1</Expression>
    <Expression>x</Expression>
    <Regle>V,13</Regle>
  </Solution>
  <Commentaire>Preuve algébrique avec
    l'interprétation de
    l'énoncé comme une équation
    admettant une infinité de solutions.</Commentaire>
  <Solution>
    <Interprétation>Les fractions sont
      réduites au même dénominateur.</Interprétation>
    <Code>V2,EA2,L1,T1,J1</Code>
    <Expression>(x+3*x+4)/4-1 = x</Expression>
    <Expression>(4*x+4)/4-1 = x</Expression>
    <Regle>V,31</Regle>
    <Expression>(4*x+4-4*1)/4 = x</Expression>
    <Regle>V,19</Regle>
    <Expression>(4*x)/4 = x</Expression>
    <Expression>(4/4)*x = x</Expression>
    <Regle>V,13</Regle>
    <Expression>x = x</Expression>
  </Solution>
  <Solution>
    <Interprétation>L'expression est simplifiée.</Interprétation>
    <Code>V2,EA1,L1,T1,J1</Code>
    <Expression>(x+3*x+4)/4-1 = x</Expression>
    <Expression>(4*x+4)/4-1 = x</Expression>
    <Regle>V,31</Regle>
    <Expression>(4/4)*x+4/4-1 = x</Expression>
    <Regle>V,13</Regle>
    <Expression>x+0 = x</Expression>
  </Solution>
</SolutionsCorrectesNonAttendues>
<SolutionsPartielles>
  <Commentaire>Une preuve algébrique est
    engagée avec l'une des trois
    interprétations précédentes. Mais une
    erreur de calcul opératoire conduit à un
    résultat faux ou à un abandon </Commentaire>
</SolutionsPartielles>
<SolutionsIncorrectes>
  <Commentaire>Les solutions utilisent une
    démarche algébrique.</Commentaire>
  <Solution>
    <Interprétation>L'interprétation
      de l'énoncé conduit à une
      traduction algébrique de
      l'enchaînement opératoire
      avec une expression globale non
      parenthésée: </Interprétation>
    <Code>V3,L3,T3,J3</Code>
    <Expression>x+3*x+4/4-1</Expression>
  </Solution>
```

```
<Solution>
  <Interprétation>- Expressions partielles
    avec écriture pas à pas enchaînée en
    succession d'opérations</Interprétation>
  <Code>V3,L3,T4,J3</Code>
  <Expression>x+3*x</Expression>
  <Expression>4*x</Expression>
  <Regle>V,13</Regle>
  <Expression>(4*x+4)/4</Expression>
  <Expression>x+1</Expression>
  <Regle>V,13</Regle>
  <Expression>x+1-1</Expression>
  <Expression>x</Expression>
  <Regle>V,13</Regle>
</Solution>
<Solution>
  <Interprétation>Les fractions sont
    réduites au même dénominateur.
    Utilisation inadaptée des
    parenthèses avec mémoire de l'énoncé.</Interprétation>
  <Code>V3,EA31,L3,T3,J3</Code>
  <Expression>(x+3*x+4)/4-1</Expression>
  <Expression>(4*x+4)/4-1</Expression>
  <Regle>V,31</Regle>
  <Expression>(4*x+4-4*1)/4</Expression>
  <Regle>V,19</Regle>
  <Expression>(4*x)/4</Expression>
  <Expression>4*x</Expression>
  <Regle>F,13</Regle>
  <Expression>x</Expression>
</Solution>
<Solution>
```

```
  <Interprétation>Les fractions sont
    réduites au même dénominateur.
    Utilisation inadaptée des
    parenthèses sans mémoire de l'énoncé.</Interprétation>
  <Code>V3,EA32,L3,T3,J3</Code>
  <Expression>(x+3*x+4)/4-1</Expression>
  <Expression>(4*x+4)/4-1</Expression>
  <Regle>V,31</Regle>
  <Expression>(4*x+4-4*1)/4</Expression>
  <Regle>V,19</Regle>
  <Expression>(4*x)/4</Expression>
  <Expression>4*x</Expression>
  <Regle>F,13</Regle>
</Solution>
<Solution>
  <Interprétation>Les fractions sont
    réduites au même dénominateur. Les
    régles de transformations utilisées
    « assemblent » les termes.</Interprétation>
  <Code>V3,EA42,L3,T3,J3</Code>
  <Expression>(x+3*x+4)/4-1</Expression>
  <Expression>(4*x+4)/4-1</Expression>
  <Regle>V,31</Regle>
  <Expression>(4*x+4-4*1)/4</Expression>
  <Regle>V,19</Regle>
  <Expression>4*x/4</Expression>
  <Regle>F,31</Regle>
  <Expression>(4/4)*x</Expression>
  <Regle>V,15</Regle>
  <Expression>x</Expression>
</Solution>
<Solution>
```

```
    <Interprétation>Les fractions sont
      réduites au même dénominateur.
      Utilisation inadaptée des
      parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>(4*x+4)/4-1</Expression>
    <Regle>V,31</Regle>
    <Expression>(4*x+3)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(4*x+0)/4</Expression>
    <Expression>(4/4)*x+0/4</Expression>
    <Regle>V,13</Regle>
    <Expression>x+0</Expression>
  </Solution>
  <Solution>
    <Interprétation>Les fractions sont
      réduites au même dénominateur.
      Utilisation inadaptée des
      parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>(4*x+4)/4-1</Expression>
    <Regle>V,31</Regle>
    <Expression>(4*x+3)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(4*x)/4</Expression>
    <Expression>4*x</Expression>
    <Regle>F,13</Regle>
    <Expression>x</Expression>
  </Solution>
  <Solution>
```

```
    <Interprétation>Les fractions sont
      réduites au même dénominateur.
      Utilisation inadaptée des
      parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>(4*x+4)/4-1</Expression>
    <Regle>V,31</Regle>
    <Expression>(4*x+3)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(4*x)/4</Expression>
    <Expression>4*x</Expression>
    <Regle>F,13</Regle>
  </Solution>
  <Solution>
    <Interprétation>Les fractions sont
      réduites au même dénominateur.
      Utilisation inadaptée des
      parenthèses sans mémoire de
      l'énoncé. Les régles de
      transformations utilisées «
      assemblent » les termes.</Interprétation>
    <Code>V3,EA32,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>(4*x+4)/4-1</Expression>
    <Regle>V,31</Regle>
    <Expression>(4*x+3)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>4*x/4</Expression>
    <Regle>F,31</Regle>
    <Expression>(4/4)*x</Expression>
    <Regle>V,15</Regle>
```

```
    <Expression>x</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur. Les
        régles de transformations utilisées
        « assemblent » les termes.</Interprétation>
    <Code>V3,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-4*1)/4</Expression>
    <Regle>V,19</Regle>
    <Expression>(8*x-4)/4</Expression>
    <Expression>(8/4)*x-4/4</Expression>
    <Regle>V,14</Regle>
    <Expression>2*x-1</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-4*1)/4</Expression>
    <Regle>V,19</Regle>
    <Expression>(8*x-4)/4</Expression>
    <Expression>8*x-1</Expression>
    <Regle>F,14</Regle>
    <Expression>2*x-1</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses sans mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA32,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-4*1)/4</Expression>
    <Regle>V,19</Regle>
    <Expression>(8*x-4)/4</Expression>
    <Expression>8*x-1</Expression>
    <Regle>F,14</Regle>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
```

```
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-1)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(8*x-4)/4</Expression>
    <Expression>(8/4)*x-4/4</Expression>
    <Regle>V,14</Regle>
    <Expression>2*x-1</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-1)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(8*x-4)/4</Expression>
    <Expression>8*x-1</Expression>
    <Regle>F,14</Regle>
    <Expression>2*x-1</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-1)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(8*x-4)/4</Expression>
    <Expression>8*x-1</Expression>
    <Regle>F,14</Regle>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses sans mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA32,EA42,L3,T3,J3</Code>
    <Expression>(x+3*x+4)/4-1</Expression>
    <Expression>8*x/4-1</Expression>
    <Regle>F,31</Regle>
    <Expression>(8*x-1)/4</Expression>
    <Regle>F,19</Regle>
    <Expression>(8/4)*x-1/4</Expression>
    <Regle>V,14</Regle>
    <Expression>2*x-1/4</Expression>
```

```
</Solution>
<Solution>
   <Interprétation>Les fractions sont
      réduites au même dénominateur.
      Utilisation inadaptée des
      parenthèses avec mémoire de
      l'énoncé. Les régles de
      transformations utilisées «
      assemblent » les termes.</Interprétation>
   <Code>V3,EA31,EA42,L3,T3,J3</Code>
   <Expression>(x+3*x+4)/4-1</Expression>
   <Expression>8*x/4-1</Expression>
   <Regle>F,31</Regle>
   <Expression>(8*x-1)/4</Expression>
   <Regle>F,19</Regle>
   <Expression>8*x-1/4</Expression>
   <Regle>F,14</Regle>
   <Expression>2*x-1/4</Expression>
</Solution>
<Solution>
   <Interprétation>Les fractions sont
      réduites au même dénominateur.
      Utilisation inadaptée des
      parenthèses sans mémoire de
      l'énoncé. Les régles de
      transformations utilisées «
      assemblent » les termes.</Interprétation>
   <Code>V3,EA32,EA42,L3,T3,J3</Code>
   <Expression>(x+3*x+4)/4-1</Expression>
   <Expression>8*x/4-1</Expression>
   <Regle>F,31</Regle>
   <Expression>(8*x-1)/4</Expression>
   <Regle>F,19</Regle>
   <Expression>8*x-1/4</Expression>
   <Regle>F,14</Regle>
</Solution>
<Solution>
   <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses avec mémoire de l'énoncé.</Interprétation>
   <Code>V3,EA31,L3,T3,J3</Code>
   <Expression>(x+3*x+4)/4-1</Expression>
   <Expression>(4*x+4)/4-1</Expression>
   <Regle>V,31</Regle>
   <Expression>4*x+0</Expression>
   <Regle>F,13</Regle>
   <Expression>x+0</Expression>
</Solution>
<Solution>
   <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses sans mémoire de l'énoncé.</Interprétation>
   <Code>V3,EA32,L3,T3,J3</Code>
   <Expression>(x+3*x+4)/4-1</Expression>
   <Expression>(4*x+4)/4-1</Expression>
   <Regle>V,31</Regle>
   <Expression>4*x+0</Expression>
   <Regle>F,13</Regle>
</Solution>
<Solution>
   <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses sans mémoire de
      l'énoncé. Les régles de
```

```
                transformations utilisées «
                assemblent » les termes.</Interprétation>
            <Code>V3,EA32,EA42,L3,T3,J3</Code>
            <Expression>(x+3*x+4)/4-1</Expression>
            <Expression>(4*x+4)/4-1</Expression>
            <Regle>V,31</Regle>
            <Expression>4*x+0</Expression>
            <Regle>F,13</Regle>
            <Expression>4*x</Expression>
            <Regle>F,31</Regle>
        </Solution>
        <Solution>
            <Interprétation>L'expression est
                simplifiée. Les régles de
                transformations utilisées «
                assemblent » les termes.</Interprétation>
            <Code>V3,EA42,L3,T3,J3</Code>
            <Expression>(x+3*x+4)/4-1</Expression>
            <Expression>8*x/4-1</Expression>
            <Regle>F,31</Regle>
            <Expression>(8/4)*x-1</Expression>
            <Regle>V,15</Regle>
            <Expression>2*x-1</Expression>
        </Solution>
        <Commentaire>Les solutions correspondent à
            une démarche non algébrique (preuve par
            des exemples). Les lettres ne sont pas disponibles.</Commentaire>
        <Solution>
            <Code>V3,L5,J2</Code>
        </Solution>
                </SolutionsIncorrectes>
            </Solutions>
        </CalculOuJustification>
    </Reponse>
</CodageReponse>
</Exercice>
</exercices>
```

C.11.3. Programme de calcul : (x * 4 + 6)/ 2 - 2 * x

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<exercices version="date">
  <Exercice Identificateur="18-3-2008-4-58"
    Nom="Preuve et programme de calcul" Numero="16">
    <Niveau>4 ième - 3 ième</Niveau>
    <Parametre>
      <Texte>Tu prends un nombre, tu multiplies par 4, tu ajoutes
        6, tu divises le résultat par 2, tu soustrais le double
        de ton nombre.</Texte>
    </Parametre>
    <Parametre>
      <Expression>(x * 4 + 6)/ 2 - 2 * x</Expression>
    </Parametre>
    <Parametre>
      <Expression>3</Expression>
    </Parametre>
    <CodageReponse>
      <Reponse Identificateur="r1">
        <ListeChoix>
          <Choix Identificateur="c1">
            <Code>V1</Code>
          </Choix>
          <Choix Identificateur="c2">
            <Code>V2</Code>
          </Choix>
        </ListeChoix>
        <CalculOuJustification Identificateur="j1">
          <Solutions>
            <SolutionsCorrectes>
              <Commentaire>Preuve algébrique avec une
                expression globale (parenthésée)
                traduisant le résultat de
                l'enchainement opératoire</Commentaire>
              <Solution>
                <Interprétation>Les fractions sont
                  réduites au même dénominateur.</Interprétation>
                <Code>V1,EA2,L1,T1,J1</Code>
                <Expression>(x*4+6)/2-2*x</Expression>
                <Expression>(4*x+6-2*2*x)/2</Expression>
                <Regle>V,19</Regle>
                <Expression>(4*x+6-4*x)/2</Expression>
                <Expression>3</Expression>
                <Regle>V,31</Regle>
              </Solution>
              <Solution>
                <Interprétation>L'expression est simplifiée.</Interprétation>
                <Code>V1,EA1,L1,T1,J1</Code>
                <Expression>(x*4+6)/2-2*x</Expression>
                <Expression>(4/2)*x+6/2-2*x</Expression>
                <Regle>V,13</Regle>
                <Expression>2*x+3-2*x</Expression>
                <Expression>3</Expression>
                <Regle>V,31</Regle>
              </Solution>
            </SolutionsCorrectes>
            <SolutionsCorrectesNonAttendues>
              <Commentaire>Preuve algébrique avec des
                expressions partielles traduisant pas à
                pas les résultats intermédiaires de
                l'enchainement opératoire. </Commentaire>
```

```
<Solution>
  <Code>V2,L1,EA1,T2,J1</Code>
  <Expression>(4*x+6)/2</Expression>
  <Expression>2*x+3</Expression>
  <Regle>V,13</Regle>
  <Expression>2*x+3-2*x</Expression>
  <Expression>3</Expression>
  <Regle>V,31</Regle>
</Solution>
<Commentaire>Preuve algébrique avec
  l'interprétation de
  l'énoncé comme une équation
  admettant une infinité de solutions.</Commentaire>
<Solution>
  <Interprétation>Les fractions sont
    réduites au même dénominateur.</Interprétation>
  <Code>V2,EA2,L1,T1,J1</Code>
  <Expression>(x*4+6)/2-2*x = 3</Expression>
  <Expression>(4*x+6-2*2*x)/2 = 3</Expression>
  <Regle>V,19</Regle>
  <Expression>(4*x+6-4*x)/2 = 3</Expression>
  <Expression>3 = 3</Expression>
  <Regle>V,31</Regle>
</Solution>
<Solution>
  <Interprétation>L'expression est simplifiée.</Interprétation>
  <Code>V2,EA1,L1,T1,J1</Code>
  <Expression>(x*4+6)/2-2*x = 3</Expression>
  <Expression>(4/2)*x+6/2-2*x = 3</Expression>
  <Regle>V,13</Regle>
  <Expression>2*x+3-2*x = 3</Expression>
  <Expression>3 = 3</Expression>
  <Regle>V,31</Regle>
</Solution>
</SolutionsCorrectesNonAttendues>
<SolutionsPartielles>
  <Commentaire>Une preuve algébrique est
    engagée avec l'une des trois
    interprétations précédentes. Mais une
    erreur de calcul opératoire conduit à un
    résultat faux ou à un abandon </Commentaire>
</SolutionsPartielles>
<SolutionsIncorrectes>
  <Commentaire>Les solutions utilisent une
    démarche algébrique.</Commentaire>
  <Solution>
    <Interprétation>L'interprétation
      de l'énoncé conduit à une
      traduction algébrique de
      l'enchaînement opératoire
      avec une expression globale non
      parenthésée: </Interprétation>
    <Code>V3,L3,T3,J3</Code>
    <Expression>x*4+6/2-2*x</Expression>
  </Solution>
  <Solution>
    <Interprétation>- Expressions partielles
      avec écriture pas à pas enchaînée en
      succession d'opérations</Interprétation>
    <Code>V3,L3,T4,J3</Code>
    <Expression>(4*x+6)/2</Expression>
    <Expression>2*x+3</Expression>
    <Regle>V,13</Regle>
    <Expression>2*x+3-2*x</Expression>
```

```
    <Expression>3</Expression>
    <Regle>V,31</Regle>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur. Les
        régles de transformations utilisées
        « assemblent » les termes.</Interprétation>
    <Code>V3,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>(4*x+6-2*2*x)/2</Expression>
    <Regle>V,19</Regle>
    <Expression>(4*x+6-4*x)/2</Expression>
    <Expression>(10*x-4*x)/2</Expression>
    <Regle>F,31</Regle>
    <Expression>6*x/2</Expression>
    <Regle>V,31</Regle>
    <Expression>(6/2)*x</Expression>
    <Regle>V,15</Regle>
    <Expression>3*x</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>(4*x+6-2*x)/2</Expression>
    <Regle>F,19</Regle>
    <Expression>(4*x+6-4*x)/2</Expression>
    <Expression>3</Expression>
```

```
    <Regle>V,31</Regle>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>(4*x+6-2*x)/2</Expression>
    <Regle>F,19</Regle>
    <Expression>(4*x+6-4*x)/2</Expression>
    <Expression>(10*x-4*x)/2</Expression>
    <Regle>F,31</Regle>
    <Expression>6*x/2</Expression>
    <Regle>V,31</Regle>
    <Expression>(6/2)*x</Expression>
    <Regle>V,15</Regle>
    <Expression>3*x</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses sans mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA32,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>(4*x+6-2*x)/2</Expression>
    <Regle>F,19</Regle>
```

<Expression>(2*x+6)/2</Expression>
<Regle>V,31</Regle>
<Expression>(2/2)*x+6/2</Expression>
<Regle>V,13</Regle>
<Expression>x+3</Expression>
</Solution>
<Solution>
<Interprétation>Les fractions sont
réduites au même dénominateur.
Utilisation inadaptée des
parenthèses avec mémoire de l'énoncé.</Interprétation>
<Code>V3,EA31,L3,T3,J3</Code>
<Expression>(x*4+6)/2-2*x</Expression>
<Expression>(4*x+6-2*x)/2</Expression>
<Regle>F,19</Regle>
<Expression>(2*x+6)/2</Expression>
<Regle>V,31</Regle>
<Expression>2*x+3</Expression>
<Regle>F,13</Regle>
<Expression>x+3</Expression>
</Solution>
<Solution>
<Interprétation>Les fractions sont réduites au même dénominateur.
Utilisation inadaptée des parenthèses sans mémoire de l'énoncé.</Interprétation>
<Code>V3,EA32,L3,T3,J3</Code>
<Expression>(x*4+6)/2-2*x</Expression>
<Expression>(4*x+6-2*x)/2</Expression>
<Regle>F,19</Regle>
<Expression>(2*x+6)/2</Expression>
<Regle>V,31</Regle>
<Expression>2*x+3</Expression>
<Regle>F,13</Regle>

</Solution>
<Solution>
<Interprétation>Les fractions sont
réduites au même dénominateur.
Utilisation inadaptée des
parenthèses sans mémoire de
l'énoncé. Les régles de
transformations utilisées «
assemblent » les termes.</Interprétation>
<Code>V3,EA32,EA42,L3,T3,J3</Code>
<Expression>(x*4+6)/2-2*x</Expression>
<Expression>(4*x+6-2*x)/2</Expression>
<Regle>F,19</Regle>
<Expression>(10*x-2*x)/2</Expression>
<Regle>F,31</Regle>
<Expression>8*x/2</Expression>
<Regle>V,31</Regle>
<Expression>(8/2)*x</Expression>
<Regle>V,15</Regle>
<Expression>4*x</Expression>
</Solution>
<Solution>
<Interprétation>Les fractions sont
réduites au même dénominateur. Les
régles de transformations utilisées
« assemblent » les termes.</Interprétation>
<Code>V3,EA42,L3,T3,J3</Code>
<Expression>(x*4+6)/2-2*x</Expression>
<Expression>10*x/2-2*x</Expression>
<Regle>F,31</Regle>
<Expression>(10*x-2*2*x)/2</Expression>
<Regle>V,19</Regle>

```
    <Expression>(10*x-4*x)/2</Expression>
    <Expression>6*x/2</Expression>
    <Regle>V,31</Regle>
    <Expression>(6/2)*x</Expression>
    <Regle>V,15</Regle>
    <Expression>3*x</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
        parenthèses avec mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>10*x/2-2*x</Expression>
    <Regle>F,31</Regle>
    <Expression>(10*x-2*x)/2</Expression>
    <Regle>F,19</Regle>
    <Expression>(10*x-4*x)/2</Expression>
    <Expression>6*x/2</Expression>
    <Regle>V,31</Regle>
    <Expression>(6/2)*x</Expression>
    <Regle>V,15</Regle>
    <Expression>3*x</Expression>
</Solution>
<Solution>
    <Interprétation>Les fractions sont
        réduites au même dénominateur.
        Utilisation inadaptée des
```

```
        parenthèses sans mémoire de
        l'énoncé. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA32,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>10*x/2-2*x</Expression>
    <Regle>F,31</Regle>
    <Expression>(10*x-2*x)/2</Expression>
    <Regle>F,19</Regle>
    <Expression>8*x/2</Expression>
    <Regle>V,31</Regle>
    <Expression>(8/2)*x</Expression>
    <Regle>V,15</Regle>
    <Expression>4*x</Expression>
</Solution>
<Solution>
    <Interprétation>L'expression est
        simplifiée. Les régles de
        transformations utilisées «
        assemblent » les termes.</Interprétation>
    <Code>V3,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>(4/2)*x+6/2-2*x</Expression>
    <Regle>V,13</Regle>
    <Expression>2*x+3-2*x</Expression>
    <Expression>5*x-2*x</Expression>
    <Regle>F,31</Regle>
    <Expression>3*x</Expression>
    <Regle>V,31</Regle>
</Solution>
<Solution>
```

```
    <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses avec mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA31,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>4*x+3-2*x</Expression>
    <Regle>F,13</Regle>
    <Expression>2*x+3-2*x</Expression>
    <Expression>3</Expression>
    <Regle>V,31</Regle>
  </Solution>
  <Solution>
    <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses avec mémoire de
      l'énoncé. Les régles de
      transformations utilisées «
      assemblent » les termes.</Interprétation>
    <Code>V3,EA31,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>4*x+3-2*x</Expression>
    <Regle>F,13</Regle>
    <Expression>2*x+3-2*x</Expression>
    <Expression>5*x-2*x</Expression>
    <Regle>F,31</Regle>
    <Expression>3*x</Expression>
    <Regle>V,31</Regle>
  </Solution>
  <Solution>
    <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses sans mémoire de l'énoncé.</Interprétation>
    <Code>V3,EA32,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>4*x+3-2*x</Expression>
    <Regle>F,13</Regle>
    <Expression>2*x+3</Expression>
    <Regle>V,31</Regle>
  </Solution>
  <Solution>
    <Interprétation>L'expression est
      simplifiée. Utilisation inadaptée
      des parenthèses sans mémoire de
      l'énoncé. Les régles de
      transformations utilisées «
      assemblent » les termes.</Interprétation>
    <Code>V3,EA32,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>4*x+3-2*x</Expression>
    <Regle>F,13</Regle>
    <Expression>7*x-2*x</Expression>
    <Regle>F,31</Regle>
    <Expression>5*x</Expression>
    <Regle>V,31</Regle>
  </Solution>
  <Solution>
    <Interprétation>L'expression est
      simplifiée. Les régles de
      transformations utilisées «
      assemblent » les termes.</Interprétation>
    <Code>V3,EA42,L3,T3,J3</Code>
    <Expression>(x*4+6)/2-2*x</Expression>
    <Expression>10*x/2-2*x</Expression>
    <Regle>F,31</Regle>
```

```
                        <Expression>(10/2)*x-2*x</Expression>
                        <Regle>V,15</Regle>
                        <Expression>5*x-2*x</Expression>
                        <Expression>3*x</Expression>
                        <Regle>V,31</Regle>
                    </Solution>
                    <Commentaire>Les solutions correspondent à
                        une démarche non algébrique (preuve par
                        des exemples). Les lettres ne sont pas disponibles.</Commentaire>
                    <Solution>
                        <Code>V3,L5,J2</Code>
                    </Solution>
                </SolutionsIncorrectes>
            </Solutions>
        </CalculOuJustification>
    </Reponse>
  </CodageReponse>
 </Exercice>
</exercices>
```

C.12. Le diagnostiqueur : Tests

Diagnostic d'un corpus de réponses à l'exerecice dit du « Prestidigitateur » de la classe d'exercices « Preuve et programme de calcul ».

Tu prends un nombre, tu ajoutes 8, tu multiplies par 3, tu retranches 4, tu ajoutes ton nombre, tu divises par 4, tu ajoutes 2, tu soustrais ton nombre : tu trouves 7. Prouve-le

Expression algébrique : $((x+8)3-4+x)/4+2-x$

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~
Exercice 16 : Prestidigitateur
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~
--------------------------------
--
Fichier_1  céline

[E16]
10
[(4+8)3)-4+4]/[4] +2 -4
=  [ 36 ]/[ 4] -2
=  9-2
= 7

--------------------------------
--
Fichier_2  Lauren

[E16]
10
[(x+8)×3-4+x]/4+2-x
(3x+24-4+x)/4+2-x
4x+20/4+2-x
x+5+2-x
7

--------------------------------
--
Fichier_3  Cynthia

[E16]
10

--------------------------------
--
Fichier_4  Jennifer

[E16]
10
(x+8)×3-4+x=3x+24-
4+x=3x+20+x=4x+20/4=x+5+2-x=7

--------------------------------
```

```
--
Fichier_5  Yoann

[E16]
10
[ (x+8)×3-4+x]/4+2-x = 7
[4x+20/4]+2-x = 7
0x+5+2 = 7
7 = 7

--------------------------------
--
Fichier_6  Florence

[E16]
10
soit x le nombre choisi
( x+8 × 3-4+x) []/[] 4+2-x=7
(3x+24-4+x) []/[] 4+2-x=7
(4x+20) []/[] 4+2-x=7
x+5+2-x=7
7=7

--------------------------------
--
Fichier_7  christophe

[E16]
01
3(x+8)-4-x []/[] 4+2-x <> 7
x+22 <> 4×7
x <> 28-22
x <> 6

--------------------------------
--
Fichier_8  Khemarak

[E16]
10
Soit 5 un nombre

((5+8)×3-4+5)/4+2-5=7 ?
((13)×3-4+5)/4+2-5=7 ?
(39-4+5)/4+2-5=7 ?
```

```
10+2-5=7 ?
10-3=7 ?
7=7 ? Oui donc cela marche

----------------------------------
--
Fichier_9  VIMY

[E16]
10
soit x le nombre choisi :
[3(x+8)-4+x]/[4]+2-x = 7

----------------------------------
--
Fichier_10  VIRGINIE

[E16]
10
(((x+8) × 3)-4+x)/4+2-x

----------------------------------
--
Fichier_11  THY

[E16]
10
(((x + 8) × 3 - 4) + x)/4 + 2 - x
= 7
soit x = 2
(((2 + 8) × 3 - 4 ) + 2)/4 + 2 - 2
= ((10 × 3 - 4) + 2)/4 = (26+2)/4
= 28/4
= 7

----------------------------------
--
Fichier_12  MARIE

[E16]
10
soit nb = x
[ (x+8)3-4+x]/4+2-9 = 7
pour que x = 7 x doit etre  <> 0
ramplacons x par 7
[(7+8)3-4+7]/4+2-9  = 7
7+8 = 15
15×3 = 45  45-4 = 41
41+7 =48
48/4 = 12

----------------------------------
--
Fichier_13  virginia

[E16]
10
(  1+ 8 × 3 - 4 + 1)/[4] + 2 -1 =
7

----------------------------------
--
Fichier_14  Aurelien

[E16]
10

----------------------------------
--
Fichier_15  Aurélia

[E16]
10
Soit x ce nombre:
[(x+8)X3-4+x]/4+2-x=[3x+24-
4+x]/4+2-x

----------------------------------
--
Fichier_16  Cecile

[E16]
00

----------------------------------
--
Fichier_17  vincent

[E16]
10
soit x le nombre cherché,
[(x+8) × 3-4+x]/[4] +2-x =
[4x+20]/[4] +2-x =
x+5+2-x =
7

----------------------------------
--
Fichier_18  Fahriye

[E16]
00
(x+8)×4-4+x:4+2-x =

----------------------------------
--
Fichier_19  NICOLAS

[E16]
10
3 + 8 = 11
11 × 3 = 33
33 - 4 = 29
29 + 3 = 32
32/4 = 8
8 + 2 = 10
10 - 3 = 7

----------------------------------
--
Fichier_20  Kamel
```

```
[E16]
01

--------------------------------
--
Fichier_21  Samir

[E16]
10
(((((x+8)3-4)+x) /4)+2)-x = 7

--------------------------------
--
Fichier_22  cedric

[E16]
10
c   est   vrai   car   c   est   un
prestidigitateur

--------------------------------
--
Fichier_23  SOUADE

[E16]
10

--------------------------------
--
Fichier_24  johann

[E16]
10
×  + 8 × ³  - 4 +  ×  + 2 = 2 ×  +
8×3 - 4+2  []/[] 5 = 16 ×  + 6 ×
- 2 []/[] 5 = 22 × -2

--------------------------------
--
Fichier_25  Matthieu

[E16]
10
f(x)=((x+8)×3-4+x)/4+2-x
=(3x+24-4+x)/4+2-x
=(4x+20)/4+2-x
=x+5+2-x
=7
Pour n'importe quel nombre x, on
trouve  que  f(x)=7.  L'affirmation
est donc juste.

--------------------------------
--
Fichier_26  alexandre

[E16]
00
{[(x + 8) × 3- 4+ x]
{------------------------------
+ 2} - x = 7
```

```
{
4

--------------------------------
--
Fichier_27  christopher

[E16]
00
soit x le nbre pense

--------------------------------
--
Fichier_28  severine

[E16]
01

--------------------------------
--
Fichier_29  olivia

[E16]
00

--------------------------------
--
Fichier_30  XAVIER

[E16]
10
Prenons le nombre:2
2+8 = 10
10 × 3 = 30
30 - 4 = 26
26+2 = 28
28/4 = 7
7+2 = 9
9 - 2 = 7
Ce  résultat  correspond  bien  au
résultat trouvé dans la phrase du
texte.
7 = 7

--------------------------------
--
Fichier_31  benoit

[E16]
01
[[3[x + 8]] - 4 + x]/[4]  + 2 - x
=  [4x + 4]/[4]  + 2 - x
= x + 1 + 2 - x
= 3

--------------------------------
--
Fichier_32  aurelie

[E16]
10
```

soit 5 le nombre 5 + 8 = 13 13 × 3 = 39 39 - 4 = 35 35 + 5 = 40

Fichier_33 karen

[E16]
10
Prenons pour exemple le nombre 4:

Fichier_34 Cindy

[E16]
00

Fichier_35 Marie

[E16]
00

Fichier_36 Xavier

[E16]
10
car on soustrais le nombre de depart a la fin donc les operations subsidiaires font 7.
8 × 3=24
24-4=20
20 []/[] 4=5
5+2=7
supposons que le chiffre en question soit x,on aura x+7-x donc 7

Fichier_37 LOIC

[E16]
10
[[x+8]3-4+x]/4+2-x=7
[3x+24-4+x]/4+2=7
[4x+20]/4+2=7
x+5+2=7
x+7=7
x=7/7=1

Fichier_38 FLORENT

[E16]
00
x=le nbre
([((3(x+8)-4)+x)]/[4]+2)-x=7
([(3x+8)-4)+x)]/[4]+2)-x=7
([(3x-4+4)+x]/[4]+2)-x=7
([3x+x]/[4]+2)-x)=7
([3/4x+1/4x]/[4]+2)-x=7
([x]/[4]+2)-x=7

Fichier_39 stephanie

[E16]
10
[x+8] × 3 = 3x+24

Fichier_40 Caroline

[E16]
10
Avec le nombre 9 ça marche

Fichier_41 Sabine

[E16]
10
[(X+8)3-4+X]/4+2-X
[3X+24-4+X]/4+2-X
[4X+20]/4+2-X
[4X]/[4] + [20]/[4] + 2-X
X+5+2-X
7

Fichier_42 stephanie

[E16]
00
soit x le nombre
[x+8] × 3 = 3x+24-4 = 3x+20 = 4x+20 = [4x+20]/4 = x+5 = x+5+2 = x+7 = x+7-x = 7

Fichier_43 samantha

[E16]
00
x+8

Fichier_44 David

[E16]

```
00
on appelle x le nombre pensé:
(x+8)×3/4+x/4+2-x

-----------------------------------
--
Fichier_45  laure

[E16]
10
((x + 8) × 3 - 4 + x) / 4 + 2 - x
=( 3x + 24 - 4 + x)/4 + 2 - x
=(4x +20) / 4 + 2 - x
=x + 5 + 2 - x
=7

-----------------------------------
--
Fichier_46  stephane

[E16]
10
si x=le nombre inconnu alors[x +
8] × 3] - 4] + x]:4] + 2] - x]
donc x=7

-----------------------------------
--
Fichier_47  loraine

[E16]
10
[ (x + 8) × 3 - 4 + x]/4 + 2 - x =
7
= (3x + 24 - 4 + x)/4 + 2 - x = 7
=(4x + 20)/4 + 2 - x
= x + 5 + 2 - x
= 7

-----------------------------------
--
Fichier_48  Marine

[E16]
10

-----------------------------------
--
Fichier_49  Chloe

[E16]
00

-----------------------------------
--
Fichier_50  Faustine

[E16]
01
x=le nombre choisi par la personne
(((x+8)3)/4+x)/4+2-
x=((3x+24)/4+x)/4+2-x.

-----------------------------------
--
Fichier_51  Laurent

[E16]
01
Je calcule par étapes,pour x=2:
2+8=10;10×3=30;30-
4=26;26+2=28;28/4=7.
Ce premier exemple parait
justifier cette affirmation:
Pour x=3
3+8=11;11×3=33;33-
4=29;29+3=32;32/4=8;8-3=5
Si mes calcules sont justes,ce
prestigitateur est un charlatan.

-----------------------------------
--
Fichier_52  Mohamed

[E16]
10
Elle est vraie car ,j' ai choisi 5
:
5+8 =13 × 3=39-4=35+5=40
40/4=10 10+2=12 12-5= 7
J'obtient bien 7 .

-----------------------------------
--
Fichier_53  mirella

[E16]
10
effectuons ce probléme prenons le
nombre 4
4+8=12
12×3= 36
36-4=32
32+4=36
36/4=9
9+2=11
10-4=7

-----------------------------------
--
Fichier_54  Balavathany

[E16]
00
x

+8 × 3-4+x:4+2-x

-----------------------------------
--
Fichier_55  raphael
```

```
[E16]
10
LA REPONSE EST OUI PUISQUE SI on
applique CE CALCUL ON TROUVE BIEN
7 EXEMPLE AVEC 5
5  + 8 = 13 × 3 = 39 - 4 = 35 + 5
= 40 /4 = 10 + 2 = 12 - 5 = 7

-----------------------------------
--
Fichier_56  Kamel

[E16]
01
il y a qu'une possibilité  encore
faut-il l'a trouver

-----------------------------------
--
Fichier_57  JOHANNA

[E16]
10
Mon nombre est quatre:

-----------------------------------
--
Fichier_58  Huseyin

[E16]
10
Le nombre auquel j ai pensé est 8.
8+8=16
16 × 3=48
48-4=44
44+8=52
52/4=13
13+2=15
15-8=7
L'affirmation est donc vrai

-----------------------------------
--
Fichier_59  nadia

[E16]
10
(3+8 × 3-4+3)/4+2-3
32/4+2-3
8+2-3
10-3
7

-----------------------------------
--
Fichier_60  NAIM

[E16]
10
3+8=11
11×3=33
33-4=29
29+3=32
32 []/[] 4=8
8+2=10
10-3=7

-----------------------------------
--
Fichier_61  MASTHAN

[E16]
01
2+8=10×3=30-4=26
2+6=8/4=2+2=4

-----------------------------------
--
Fichier_62  myriam

[E16]
00
(3+8 ×3-4+3)/4 +2-3=7

-----------------------------------
--
Fichier_63  KELLY

[E16]
01

-----------------------------------
--
Fichier_64  Ahmed

[E16]
01
2+8=10
10 × 3=30
30 - 4=26
26/4=6,5
6,5+2=8,5
8,5  <> 7

-----------------------------------
--
Fichier_65

[E16]
01
8 + 8 × 3 - 4 + 8/4 + 2 - 8=7
8 + 24 - 4 + 2 + 2 - 8=7
24=7


3 + 8 × 3 - 4 + 3/4 + 2 - 3=7
3+24 - 4 + 0.75 + 2 - 3=7
22.75=7

-----------------------------------
--
Fichier_66  Jean-Luc
```

[E16]
10
Soit x un nombre
le calcul est [3(x+8)-4+x]/4+2-x=7
[3x+24-4+x]/4+2-x=7
[4x+20]/4+8/4-4x/4=7
4x+20+8-4x=7×4
28=28

Fichier_67 Jean-philippe

[E16]
10
5+8×3-4+5/4+2-5=7

Fichier_68 Amandine

[E16]
01

Fichier_69 Céline

[E16]
00

Fichier_70 Cedric

[E16]
10
N=2

2+8=10
10×3=30
30-6=24
24+2=28
28/4=7
7+2=9
9-2=7

Fichier_71 Florence

[E16]
01
2+8=10
10×3=30
30/4=7.5
7.5+2=9.5
9.5/4

--
Fichier_72 Guillaume

[E16]
00
x est le nombre pensé
(x+8)×3 = 3x+8
3x+4+x/4+2

--
Fichier_73 Benjamin

[E16]
10
4 est le nombre auquel je pense
4 + 8 × 3 - 4 + 4 : 4 + 2 - 4 = 7

--
Fichier_74 Baptiste

[E16]
10
on est toujours oblige de soustrere, meme acec 1pour trouver 7

--
Fichier_75 Marwan

[E16]
10
prenons 2
2+8=10
10x3=30
30-4=26
26+2=28
28/4=7
7+2=9
9-2=7

--
Fichier_76 Elizabeth

[E16]
10

--
Fichier_77 Albert

[E16]
00

--
Fichier_78 laure

```
[E16]
01
=a+8
=(a+8) × 3
=a+24-4
=a+20+a+20
=a+5+a+20+2

----------------------------------
--
Fichier_79  Sébastien

[E16]
01
2+8×3-4+2/4+2/2=

----------------------------------
--
Fichier_80  Benjamin

[E16]
10
(12 + 8) × 3
60 -4
56+12
68:4
17+2
19-12
7

----------------------------------
--
Fichier_81  Florence

[E16]
10
[3(x+8)-4+x]/[4]  + 2 - x = 7
[3x+24-4+x]/[4]  + 2-x = 7
[4x+20]/[4] +2-x=7
x+5+2-x=7
7=7

----------------------------------
--
Fichier_82  Guillaume

[E16]
10
5 + 8 = 13
13 × 3 = 39
39 - 4 = 35
35 + 5 = 40
[40]/[4] = 10
10 + 2 = 12
12 - 5 = 7

----------------------------------
--
Fichier_83  béatrice

[E16]
00

----------------------------------
--
Fichier_84  Hélène

[E16]
00
Soit x ce nombre
```

Annexes D

Implémentation du module auteur PépiGen

D.1. Mise en œuvre des cas d'utilisation..... 399

D.2. Diagrammes d'interaction : cas d'utilisation « Créer un exercice »..... 400
D.2.1. Action « Choisir une classe d'exercices » 400
D.2.2. Action « Saisir les paramètres d'un exercice » 401
D.2.3. Action « Voir l'exercice »..... 402
D.2.4. Action « Voir la grille de codage des réponses »..... 403
D.2.5. Action « Enregistrer un exercice »..... 405

D.3. Cas d'utilisation « Composer un test diagnostic » : PépiGenTest 406
D.3.1. Premier niveau d'utilisation : « Choisir un test existant » 407
D.3.2. Deuxième niveau d'utilisation : « Modifier un test existant » 407
D.3.3. Étude des actions « Enregistrer un test » et « Administrer un test »..... 408

D.1. Mise en œuvre des cas d'utilisation

Les cas d'utilisation présentés dans le chapitre 7 à la section 2 décrivent les interactions entre les acteurs externes qui peuvent être des acteurs humains ou des systèmes informatiques et le système logiciel que l'on veut créer. Pour définir les classes logicielles impliquées dans ces interactions, nous avons appliqué de façon méthodique des patterns de conception notamment les trois patterns GRASP (General Responsability Assignment Software Patterns [Larman 2005]) Les patterns permettent de répondre aux questions suivantes : quelles classes logicielles ont la responsabilité de créer une instance (pattern « Créateur »), de gérer les évènements avec des systèmes externes (pattern « Contrôleur »), quelles classes possèdent les informations nécessaires pour effectuer certaines actions (pattern « Expert ») [Gamma et al 2004] ?

Le modèle conceptuel d'une classe d'exercices (Chapitre 5) est mis en œuvre par une classe abstraite ***ClasseExercice***[1]. Cette classe factorise les données et le code commun à chacune des classes d'exercices [Beaudoin-Lafon 1992]. C'est une classe composite qui entretient ainsi une relation de composition avec les classes **InterfaceAbstraite, DonneeIndexation, GenerationEnonce** et **GenerationGrilleCodage.** (Figure 1)

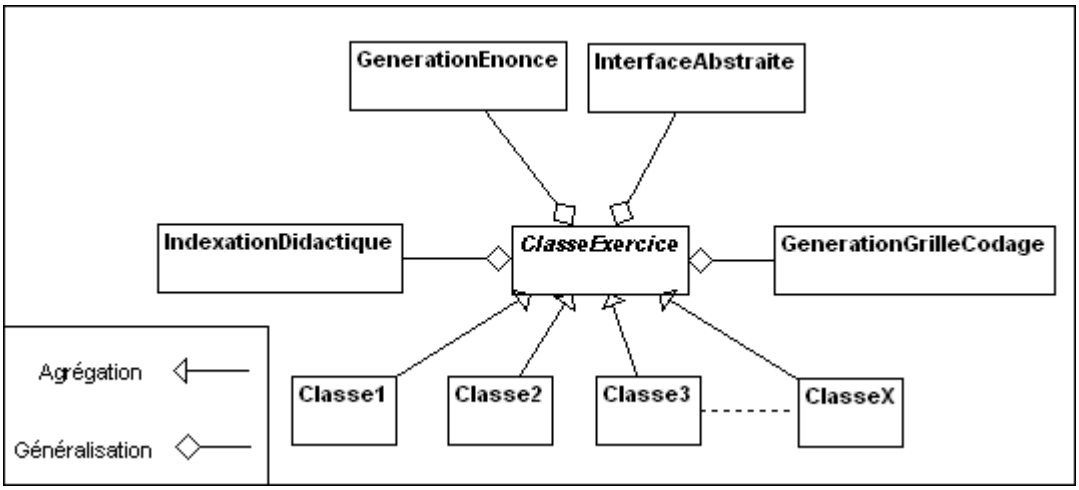


Figure 1 : Diagramme de classe de PépiGen

Chacune des classes d'exercices est une sous-classe de cette classe abstraite et un exercice est donc une instance d'une de ces sous-classes. La fonctionnalité principale de PépiGen est de créer des instances de ces sous-classes.

La modélisation UML propose des diagrammes qui illustrent la mise en oeuvre des cas d'utilisation : les diagrammes d'interaction. A l'aide de ces diagrammes, nous représentons la

[1] Selon les conventions utilisées dans les diagrammes UML, nous écrivons les noms des classes abstraites en italique.

mise en oeuvre du cas d'utilisation « Créer un exercice » en nous centrant sur la description des mécanismes d'instanciation que nous avons entièrement développés. Nous évoquons pour terminer la réalisation du cas d'utilisation de constitution d'un test que nous avons étudié mais que nous n'avons pas implémenté.

D.2. Diagrammes d'interaction : cas d'utilisation « Créer un exercice »

Nous étudions cinq actions de ce cas d'utilisation (Chapitre 7, section 2) qui contribuent à préciser le comportement du système PépiGen : « Choisir une classe d'exercices », « Saisir les paramètres d'un exercice », « Voir l'exercice », « Voir la grille de codage des réponses », « Enregistrer l'exercice ». Le cas d'utilisation « Créer un exercice » débute après le démarrage de l'application PépiGen lorsque l'auteur choisit d'enrichir la base d'exercices.

Pour chacune de ces actions nous présentons le point de vue de l'utilisateur et l'implémentation

D.2.1. Action « Choisir une classe d'exercices »

D.2.1.1. Point de vue de l'utilisateur

Pour l'auteur, il s'agit d'abord de choisir un modèle d'exercices. Il spécifie les paramètres d'indexation à l'aide d'une interface de type tableau de configuration qui propose différents choix : niveau de la classe, composantes de la compétence algébrique à tester, liste d'exercices, ou d'autres critères tels que les capacités ou les connaissances en jeu. Il sélectionne un choix et PépiGen affiche la classe d'exercices choisie **Classe**i qui est représentée par une instance particulière : l'exercice originel de Pépite (Chapitre 5).

D.2.1.2. Implémentation

Pour assurer leur persistance, les instances des classes d'exercices se trouvent sur un support de stockage. Si les classes d'exercices sont peu nombreuses, il est possible de les charger directement en mémoire au démarrage de l'application, sinon elles sont chargées de façon ponctuelle en fonction des besoins. Dans la version actuelle, nous avons fait le choix de les charger au démarrage de l'application. Toutes les tâches d'initialisation qui s'exécutent au démarrage de PépiGen sont décrites dans la section concernant la couche « application » de l'architecture logicielle (Chapitre 7, section 4.3). La Figure 2 présente le diagramme d'interaction de « Choisir une classe d'exercices ». Pour éviter des confusions sur les deux sens du mot interface qui peut être utilisé pour désigner une interface graphique ou pour désigner le comportement visible d'un élément (langage de modélisation UML ou langage java), nous avons

composé le nom des classes logicielles qui décrivent les interfaces graphiques avec le terme Jframe (e.g. **JFrameChoixClasseExercice**). Afin d'augmenter le potentiel de réutilisation du système, le pattern GRASP « Contrôleur » préconise de ne pas attribuer aux objets de l'interface (e.g. l'interface graphique utilisée pour choisir un modèle d'exercice) la responsabilité de traiter des évènements liés au domaine. Nous avons ajouté une classe logicielle, **GestionExercice** qui gère les événements concernant la gestion des modèles. A la réception de l'événement **choisirClasseExercices(selection)**, la classe **GestionExercice** opère l'affichage d'une instance de la classe d'exercice correspondant au choix d'un auteur par l'envoi du message **traiter(selection)** à la liste des représentants des classes d'exercice **Classe1**, **Classe2**, **..ClasseX**. La classe **GestionExercice** procède par itération sur l'ensemble de la liste d'exercices qui constitue une collection. Sur le schéma, les nombres 1 et 2 donnent l'ordre chronologique des évènements.

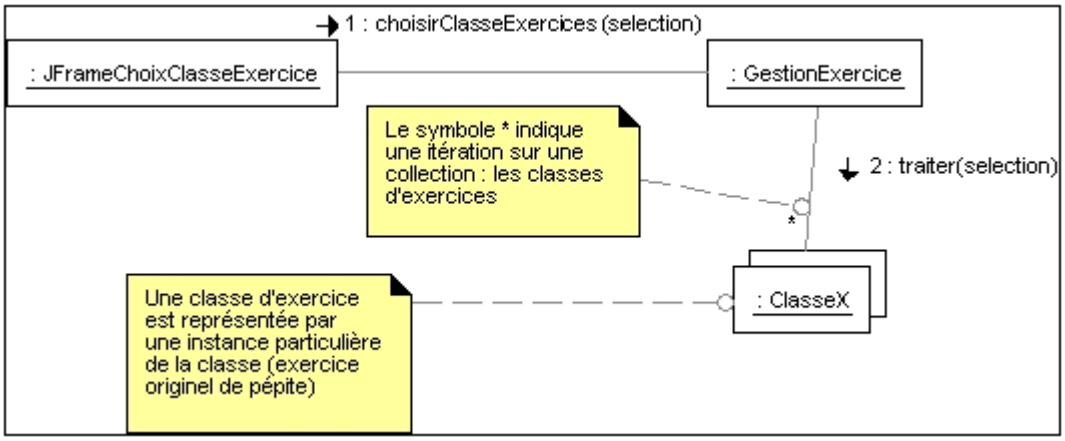


Figure 2 : Diagramme d'interaction « Choisir une classe d'exercices »

D.2.2. Action « Saisir les paramètres d'un exercice »

D.2.2.1. Point de vue utilisateur

La génération des exercices est automatique pour les exercices dont les paramètres sont fortement contraints et assistée pour les autres (Chapitre 5, section 11.2). Selon le mode d'obtention de l'exercice l'auteur saisi ou non les paramètres de l'exercice. Chaque classe d'exercices pour laquelle la génération des clones est assistée a une interface graphique différente pour la saisie des paramètres : ce peut être par exemple un formulaire (e.g. la classe « Déterminer si des expressions algébriques du second degré sont égales »), une interface qui comporte des figures géométriques (e.g. la classe « Associer aire et expression »), une palette de termes (e.g. la classe « Preuve et programme de calcul »). Le chapitre 7 présente à la section 5 un exemple de génération assistée où l'auteur saisit un programme de calcul à l'aide d'une palette de termes.

D.2.2.2. Implémentation

Selon le modèle conceptuel de la génération des énoncés défini dans le chapitre 5 (Chapitre 5, Figure 16), la classe **GenerationEnonce** comporte un ou plusieurs paramètres (classe **Parametre**) et un ou plusieurs invariants (classe **Invariant**). Les paramètres sont saisis par un utilisateur lorsque la génération d'un exercice est assistée.

La Figure 3 présente le diagramme d'interaction de l'action « Saisir les paramètres d'un exercice ». Le pattern GRASP « Créateur » préconise de donner à la classe **GenerationEnonce** la responsabilité de créer la classe **ParametreSaisi** qui joue le rôle de contrôleur pour traiter les évènements entrants liés à l'interface de saisie des paramètres (e.g. **getParametre()**).

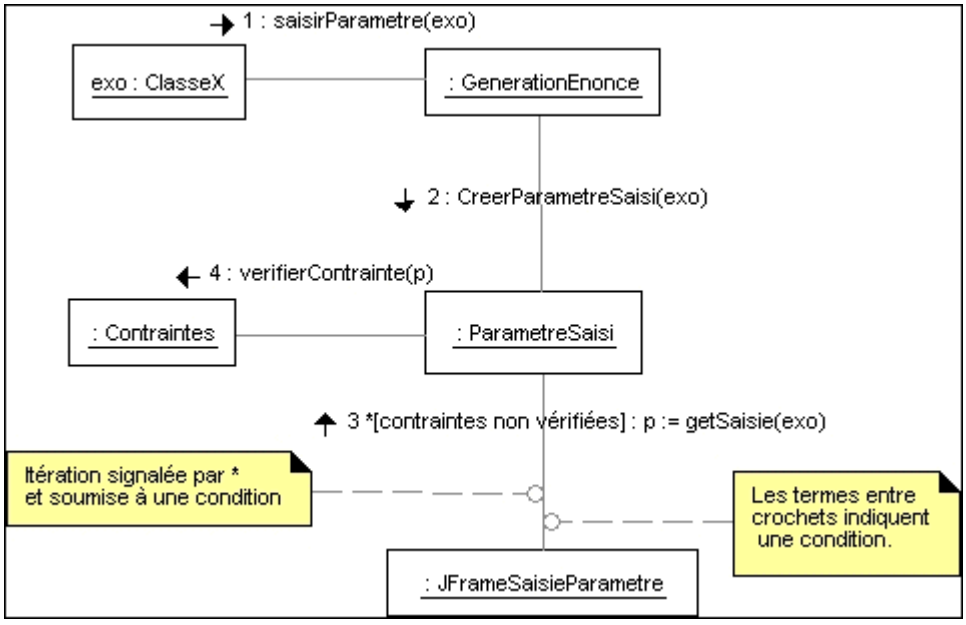


Figure 3 : Diagramme d'interaction « Saisir les paramètres d'un exercice »

D.2.3. Action « Voir l'exercice »

D.2.3.1. Point de vue de l'utilisateur

Pour l'auteur, l'action « Voir l'exercice » consiste à afficher l'interface élève du clone d'exercice obtenu avec les valeurs des paramètres qu'il a saisies (génération assistée des clones) ou avec les valeurs générées de façon automatique par PépiGen (génération automatique des clones).

D.2.3.2. Implémentation

La classe **InterfaceAbstraite** décrit l'organisation des exercices d'une classe d'exercices en parties et questions et les contenus des énoncés associés : les contenus sont constitués d'expressions, de textes ou de figures. Pour tout ou partie de l'énoncé, soit ces éléments sont constants, soit ils sont générés à partir des paramètres et des invariants de la classe. Ils sont donc différents selon le clone d'exercice. Chaque énoncé, que ce soit l'énoncé de l'exercice, d'une partie de l'exercice ou d'une question, est obtenu selon ce processus de génération. Le mode d'obtention des contenus des paramètres (génération automatique ou génération assistée) est connu au moment de l'exécution du programme, aussi nous avons appliqué un pattern Gang of Four (GoF) appelé « Abstract Factory » [GHJV95] pour la classe **GenerationEnonce** ayant la responsabilité de fournir le type d'instance attendu. L'interface élève de ce clone est une instance de la classe **JframeInterfaceExoX** obtenue à partir des éléments de l'instance de la classe **InterfaceAbstraite** (Figure 4).

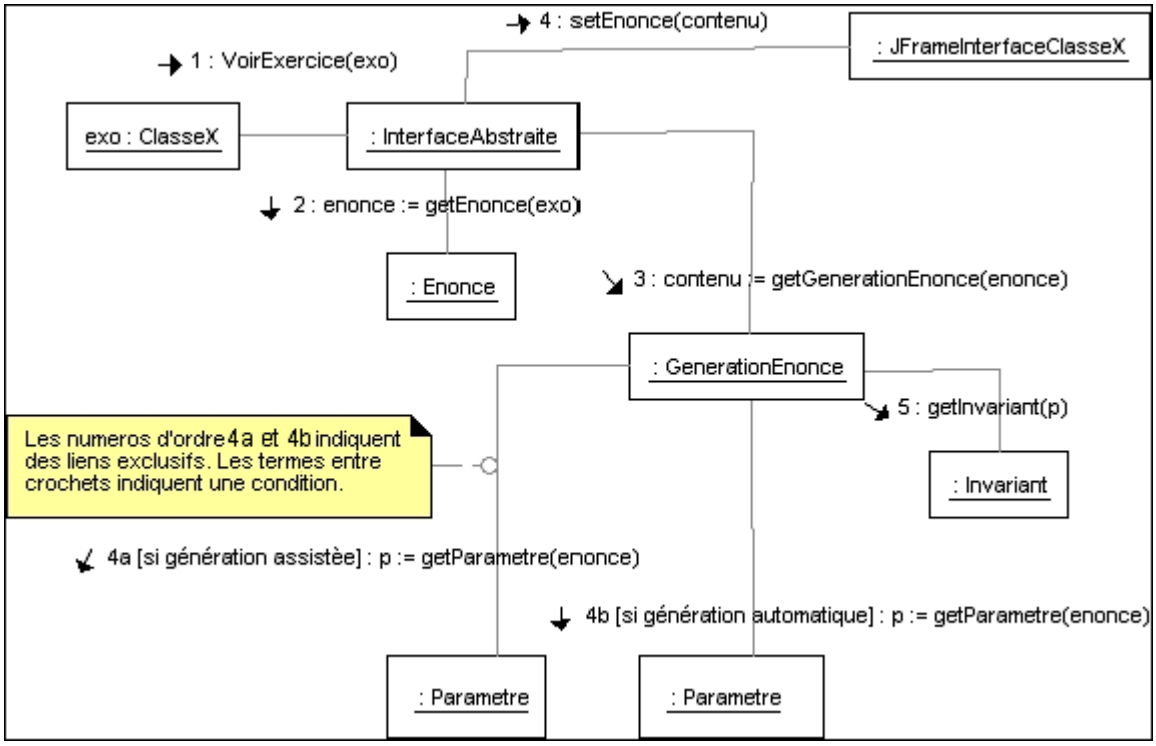


Figure 4 : Diagramme d'interaction « Voir l'exercice »

D.2.4. Action « Voir la grille de codage des réponses »

D.2.4.1. Point de vue utilisateur

Pour l'auteur, l'action « Voir la grille de codage » consiste à afficher, pour le clone d'exercice qu'il a conçu, les réponses anticipées ainsi que les codes associés à ces réponses

(Chapitre Figure 14). Les réponses aux questions fermées se présentent sous la forme de choix, les réponses aux questions ouvertes sous la forme d'un calcul ou d'une justification.

D.2.4.2. Implémentation

Dans le modèle conceptuel d'une classe d'exercices, une réponse anticipée est décrite par une classe abstraite ***ReponseAnticipee*** qui factorise les données et le code commun aux trois types de traitement des réponses : les réponses à des questions fermées (**ReponseQuestionFermee**), les réponses à des questions ouvertes (**ReponseQuestionOuverte**) et les réponses à des questions composées (**ReponseQuestionComposee**) constituées d'une question fermée et d'une question ouverte. (Chapitre 5, Figure 19). Le type de réponses dépend de la classe d'exercices à instancier. Il est connu au moment de l'exécution du programme. Aussi nous avons appliqué le pattern Gof « Fabrique » [GHJV95] à la classe **GenerationGrilleCodage** qui fournit le type d'instance attendue. A chaque type de réponses est associé un code et un commentaire qui constituent la grille de codage et qui sont affichés avec la réponse par la classe **JframeGrilleCodageClasseX** (Figure 5).

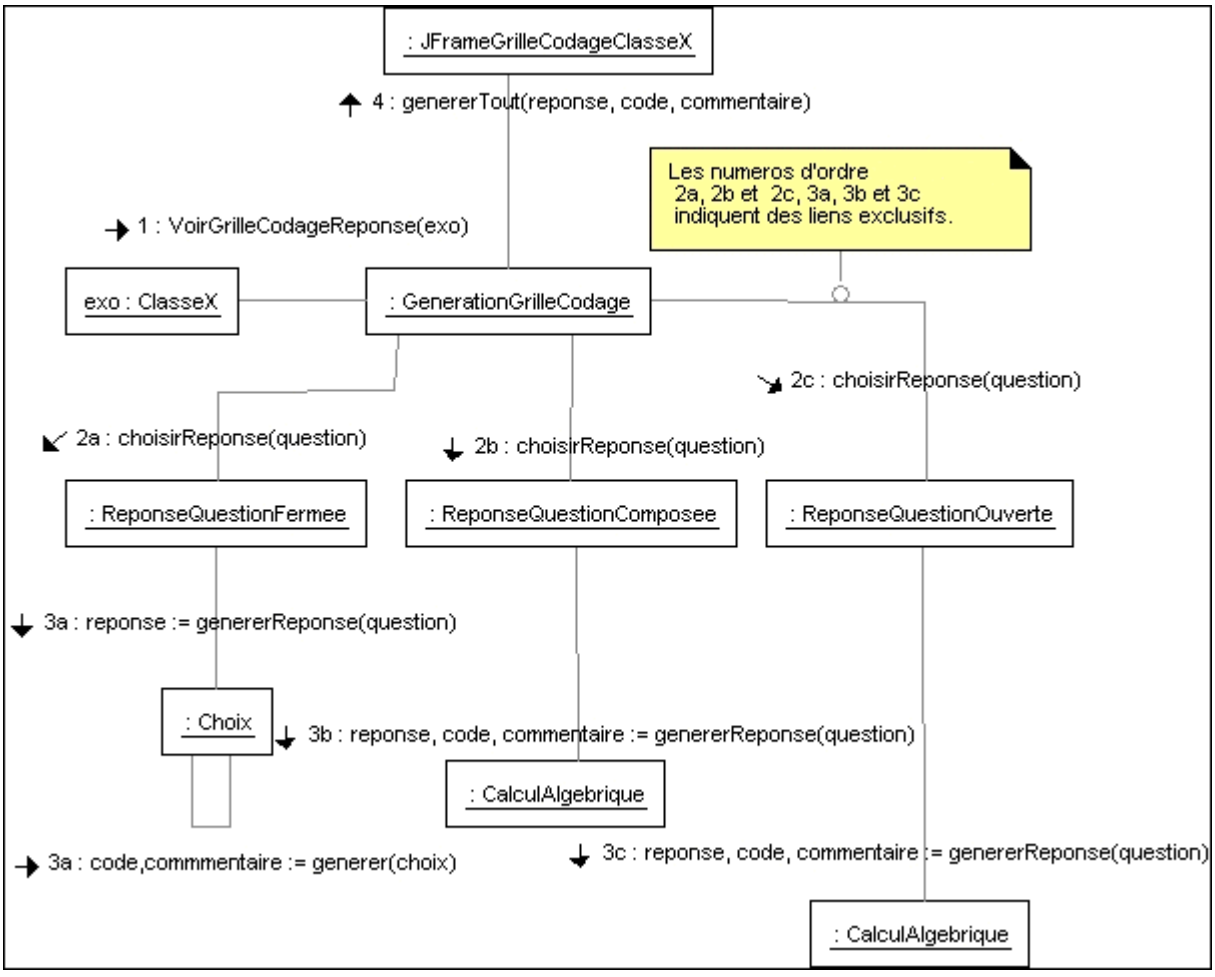


Figure 5 : Diagramme d'interaction « Voir la grille de codage des réponses »

Les réponses anticipées sous la forme d'un calcul, d'une justification ou d'un résultat sont des expressions algébriques générées à partir de l'arbre des solutions anticipées obtenu avec le module Pépinière décrit dans le chapitre 6. Les accès à ce module se font par l'intermédiaire d'une seule classe que nous avons appelé **CalculAlgebrique** selon les préconisations du pattern GoF « Façade » [GHJV95] facilitant ainsi la réutilisation de ce module. Le module Pépinière construit un arbre général des réponses anticipées, du codage et des commentaires associés à la réponse pour les exercices qui mobilisent les capacités de la composante « *Effectuer du calcul algébrique* ». PépiGen complète ce codage en prenant en compte les deux autres composantes de la compétence algébrique : « *Utiliser l'algèbre pour résoudre* » et « *Traduire algébriquement dans différentes représentations* (*géométrique, graphique, numérique*) ». La génération complète de la grille de codage est décrite dans le chapitre 7 section 6.

D.2.5. Action « Enregistrer un exercice »

D.2.5.1. Point de vue de l'utilisateur

Pour l'auteur, l'action « Enregistrer un exercice » consiste à enregistrer les paramètres qu'il a saisis pour concevoir le clone d'un exercice dont la génération est assistée, soit pour enrichir la banque d'exercices (Chapitre 4, scénario 4), soit pour composer un test diagnostic (Chapitre 4, scénario 2).

D.2.5.2. Implémentation

L'action « Enregistrer un exercice » pose le problème de la persistance des objets qui ont été créés. Nous abordons d'abord les questions relatives au choix des classes qui ont la responsabilité de sauvegarder les instances des objets qui ont été créés, puis nous précisons les raisons qui nous ont amenées à sauvegarder dans le même fichier les paramètres saisis, les réponses anticipées et la grille de codage de ces réponses.

Dans les sections précédentes nous nous sommes appuyés essentiellement sur le modèle conceptuel du domaine pour définir les classes logicielles que nous utilisons. Selon le pattern GRASP « Expert » [Larman 2003], ce sont les classes **GenerationEnonce** et **GenerationGrilleCodage** qui possèdent les données qu'il faut sauvegarder, mais cette tâche de sauvegarde nécessite aussi des opérations liées à la manipulation de données XML et à la transformation d'un document XML en un objet Java et réciproquement. Les classes d'exercices sont stockées dans un fichier (**ModelesClasseExercices**), les clones d'exercices dans le fichier (**BanqueExercices**). Selon le pattern GRASP « Fabrication », nous avons créé une nouvelle classe

que nous appelons **StockagePersistant** (Figure 6) qui n'appartient pas au domaine conceptuel et qui se charge des opérations liées aux échanges avec les fichiers contenant les données XML.

En ce qui concerne le stockage des clones d'exercices, deux options sont possibles : la première option consiste à sauvegarder dans le fichier **BanqueExercices** seulement les clones qui sont générés de façon assistée, la deuxième consiste à les sauvegarder tous. Pour les clones d'exercices dont la génération est automatique, nous avons pris la première option et nous les générons de façon aléatoire à la demande de l'auteur, à partir du modèle de l'exercice, quand cela est nécessaire pour constituer un test. Ce choix a pour objet d'éviter d'avoir un grand nombre de clones à stocker. En effet, pour la classe d'exercices « Reconnaître des égalités vraies » nous pouvons générer automatiquement plus de deux mille clones.

Enfin, le fichier **BanqueExercices** qui stocke les clones contient d'une part les paramètres nécessaires à l'interpréteur d'exercices, PépiTest, et, d'autre part les réponses anticipées et la grille de codage utilisés par le diagnostiqueur, PépiDiag. L'interpréteur d'exercices PépiTest et le diagnostiqueur PépiDiag sont décrits dans le chapitre 7, sections 8 et 9.

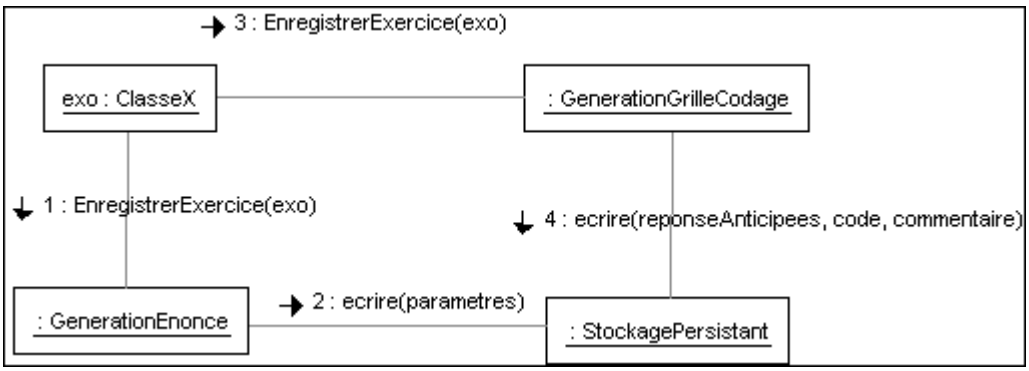


Figure 6 : Diagramme d'interaction « Enregistrer un exercice »

D.3. Cas d'utilisation « Composer un test diagnostic » : PépiGenTest

Pour ce cas d'utilisation, nous étudions plus particulièrement les relations entre les deux modules de SuperPépite, **PépiGenTest** et **PépiTest** (Chapitre 4, figure 1). L'observation du diagramme de ce cas d'utilisation d'un point de vue des interactions entre **PépiGenTest** et **PépiTest** d'une part, **PépiGenTest** et les fichiers **BanqueExercices, BanqueTests²** et **ModelesClasseExercices** d'autre part, fait apparaître trois niveaux d'utilisation de **PépiGenTest**. Un premier niveau est de choisir un test existant et de l'administrer à des élèves en l'état. Un second niveau est de créer un test en modifiant un test prédéfini en changeant l'ordre des

exercices, en supprimant des exercices ou en ajoutant un exercice au test. Un troisième niveau est de modifier un test existant en remplaçant l'un des exercices par un clone choisi dans les clones du fichier **BanqueExercices** ou par un nouveau clone créé avec PepiGen (section D.2).

Nous étudions la mise en œuvre des actions qui correspondent aux deux premiers niveaux en montrant les incidences qu'elles ont sur l'utilisation des fichiers XML et les relations entre les modules **PépiGenTest** et **PépiGen** puis, nous abordons la réalisation des actions « Enregistrer un test existant » et « Administrer un test ».

D.3.1. Premier niveau d'utilisation : « Choisir un test existant »

Ce premier niveau d'utilisation est constitué essentiellement des échanges avec le fichier **BanqueTests** qui contient un ensemble de tests constitués à partir des exercices générés avec **PépiGen**. L'action « Choisir un test existant » nécessite un système d'indexation de la base de tests à partir des données d'indexation didactique du modèle conceptuel d'une classe d'exercices auxquelles devront s'ajouter d'autres informations pour caractériser le test (e.g. date, nom de l'enseignant) et l'implémentation des opérations de bases d'un système de gestion de base de données. La conception de ce module est un des prolongements de ce travail de thèse qui doit être mené avec les didacticiens et les enseignants de l'équipe.

D.3.2. Deuxième niveau d'utilisation : « Modifier un test existant »

Ce niveau d'utilisation conduit à des modifications du test sans conception de nouveaux clones d'exercices. Si le changement de l'ordre des exercices, la suppression ou l'ajout d'un exercice peut se concevoir à partir de la liste des exercices du test et d'une description de ces exercices chargée en mémoire au moment du choix d'un test prédéfini, l'action « Voir l'exercice » nécessite l'accès au fichier **BanqueExercices** et au fichier **ModelesClasseExercices.** En effet, dans la section D.2.5 nous avons décrit la solution que nous avons adoptée pour enregistrer un clone d'exercices dont la génération était assistée. Seuls les paramètres de l'exercice sont enregistrés dans la base d'exercices. Pour voir un exercice nous devons d'abord extraire du fichier **ModelesClasseExercices** les caractéristiques de la classe concrète correspondant au modèle de l'exercice, puis, pour instancier le clone de l'exercice, extraire les paramètres correspondants du fichier **BanqueExercices** (Section D.2.3).

---

[2] Le développement du module **PépiGenTest** n'est pas réalisé actuellement. Dans le cadre de ce travail de thèse nous enregistrons les clones d'exercices ainsi que les différents tests dans deux fichiers XML (**BanqueExercices, BanqueTests**).

Ce niveau montre une utilisation conjointe des fichiers **BanqueTests** et **BanqueExercices** par les deux modules **PepiGen** et **PépiGenTest**. Lors du développement du module **PépiGenTest** l'implémentation des opérations de base d'un système de gestion de base de données pour **BanqueTests** devra prendre en compte la gestion du fichier **BanqueExercices**. C'est un des prolongements de ce travail de thèse qui doit être mené avec les didacticiens et les enseignants de l'équipe.

D.3.3. Étude des actions « Enregistrer un test » et « Administrer un test »

Les clones d'exercices, dont la génération est assistée, sont sauvegardés dans le fichier **BanqueExercices** qui contient pour chaque clone, les paramètres, les réponses anticipées et la grille de codage.

Actuellement, pour l'action « Administrer un test » les instances des exercices du test sont extraites du fichier **BanqueExercices** à partir de la liste ordonnée de références. L'action « Administrer un test » consiste à créer un fichier contenant les exercices auxquels s'ajoutent des données qui caractérisent le test (e.g. date du test, nom de l'enseignant, nom de la classe, information didactiques sur le test).