HAL
open science

Assistance à la construction et à la comparaison de techniques de diagnostic des connaissances

Sébastien Lallé

To cite this version:

Sébastien Lallé. Assistance à la construction et à la comparaison de techniques de diagnostic des connaissances. Autre [cs.OH]. Université de Grenoble, 2013. Français. NNT : 2013GRENM042 . tel-01135183

HAL Id: tel-01135183

https://theses.hal.science/tel-01135183v1

Submitted on 24 Mar 2015

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

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

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

« Sébastien LALLÉ »

Thèse dirigée par **« Vanda LUENGO »** et
codirigée par **« Nathalie GUIN »**

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

Assistance à la construction et la comparaison de techniques de diagnostic des connaissances

Thèse soutenue publiquement le **« 11 décembre 2013 »**,
devant le jury composé de :

Mme Marie-Christine ROUSSET
Professeure de l'université de Grenoble, Présidente
M. Michel DESMARAIS
Professeur de l'École polytechnique de Montréal, Rapporteur
M. Jean-Marc LABAT
Professeur de l'université Pierre et Marie Curie de Paris, Rapporteur
M. Sebastián VENTURA
Professeur de l'université de Cordoue, Membre
M. Nicolas DELESTRE
Maître de Conférences de l'INSA de Rouen, Membre

Remerciements

Mes premiers remerciements sont pour les deux personnes qui ont durant trois ans accompagné, soutenu, guidé, enrichi, bonifié et pouponné mon travail de thèse, me permettant au-delà du résultat la meilleure entrée dans le monde de la recherche qui soit. Je parle naturellement de mes deux directrices de thèse.

Durant ma thèse, j'ai pu côtoyer deux laboratoires de recherche, et je tiens à mentionner les doctorants et les permanents de l'équipe METAH du LIG à Grenoble et de l'équipe SILEX du LIRIS à Lyon pour l'excellent cadre de recherche qu'ils ont su créer, mêlant habillement sérieux et détente.

Je remercie de même Jack Mostow et son équipe pour l'accueil qu'ils m'ont fait à Pittsburgh durant nos six mois de collaboration, l'ouverture d'esprit dont ils ont su faire preuve et les forts nombreuses petites suggestions ou contributions qui ont renforcé le sérieux de cette thèse.

J'ai enfin une pensée pour mes proches, famille et amis, qui n'ont toujours pas compris le sujet de ma thèse mais qui ont contribué à ce que tout se passe bien durant ces trois années.

Résumé

Cette thèse aborde la thématique de la comparaison et de la construction de diagnostics des connaissances dans les Environnements Informatiques pour l'Apprentissage Humain (EIAH). Ces diagnostics sont utilisés pour déterminer si les apprenants maîtrisent ou non les connaissances ou conceptions du domaine d'apprentissage (par exemple math au collège) à partir des traces collectées par l'EIAH. Bien que ces diagnostics soient récurrents dans les EIAH, ils sont fortement liés au domaine et ne sont que peu formalisés, si bien qu'il n'existe pas de méthode de comparaison pour les positionner entre eux et les valider. Pour la même raison, utiliser un diagnostic dans deux domaines différents implique souvent de le redévelopper en partie ou en totalité, sans réelle réutilisation. Pourtant, pouvoir comparer et réutiliser des diagnostics apporterait aux concepteurs d'EIAH plus de rigueur pour le choix, l'évaluation et le développement de ces diagnostics.

Nous proposons une méthode d'assistance à la construction et à la comparaison de diagnostics des connaissances, réifiée dans une première plateforme, en se basant sur une formalisation du diagnostic des connaissances en EIAH que nous avons défini et sur l'utilisation de traces d'apprenant. L'assistance à la construction se fait via un algorithme d'apprentissage semi-automatique, guidé par le concepteur du diagnostic grâce à une ontologie décrivant les traces et les connaissances du domaine d'apprentissage. L'assistance à la comparaison se fait par application d'un ensemble de critères de comparaison (statistiques ou spécifiques aux EIAH) sur les résultats des différents diagnostics construits. La principale contribution au domaine est la généricité de notre méthode, applicable à un ensemble de diagnostics différents pour tout domaine d'apprentissage.

Nous évaluons notre travail à travers trois expérimentations. La première porte sur l'application de la méthode à trois domaines différents (géométrie, lecture, chirurgie) en utilisant des jeux de traces en validation croisée pour construire et appliquer les critères de comparaison sur cinq diagnostics différents. La seconde expérimentation porte sur la spécification et l'implémentation d'un nouveau critère de comparaison spécifique aux EIAH : la comparaison des diagnostics en fonction de leur impact sur une prise de décision de l'EIAH, le choix d'un type d'aide à donner à l'apprenant. La troisième expérimentation traite de la spécification et de l'ajout d'un nouveau diagnostic dans notre plateforme, en collaborant avec une didacticienne.

Abstract

Comparing and building knowledge diagnostic is a challenge in the field of Technology Enhanced Learning (TEL) systems. Knowledge diagnostic aims to infer the knowledge mastered or not by a student in a given learning domain (like mathematics for high school) using student traces recorded by the TEL system. Knowledge diagnostics are widely used, but they strongly depend on the learning domain and are not well formalized. Thus, there exists no method or tool to build, compare and evaluate different diagnostics applied on a given learning domain. Similarly, using a diagnostic in two different domain usually imply to implementing almost both from scratch. Yet, comparing et reusing knowledge diagnostics

can lead to reduce the engineering cost, to reinforce the evaluation et finally help knowledge diagnostic designers to choose a diagnostic.

We propose a method, refine in a first platform, to assist knowledge diagnostic designers to build et compare knowledge diagnostics, using a new formalization of the diagnostic et student traces. To help building diagnostics, we used a semi-automatic machine learning algorithm, guided by an ontology of the traces and the knowledge designed by the designer. To help comparing diagnostics, we use a set of comparison criteria (either statistical or specific to the field of TEL systems) applied on the results of each diagnostic on a given set of traces. The main contribution is that our method is generic over diagnostics, meaning that very different diagnostics can be built et compared, unlike previous work on this topic.

We evaluated our work though three experiments. The first one was about applying our method on three different domains and set of traces (namely geometry, reading and surgery) to build and compare five different knowledge diagnostics in cross validation. The second experiment was about designing and implementing a new comparison criteria specific to TEL systems: the impact of knowledge diagnostic on a pedagogical decision, the choice of a type of help to give to a student. The last experiment was about designing and adding in our platform a new diagnostic, in collaboration with an expert in didactic.

Sommaire

SECTION I : INTRODUCTION ET ETAT DE L'ART 11

Chapitre 1 : Introduction

I. Concepts généraux 11

II. Motivation de la recherche 12

III. Questions et hypothèses de recherche 13

IV. Cas d'usage 14

1. Cas d'usage 1 : ajout d'un diagnostic des connaissances à un EIAH 14
2. Cas d'usage 2 : comparaison d'un nouveau diagnostic avec l'existant 15

V. Plan du mémoire 15

Chapitre 2 : État de l'art

I. Diagnostic des connaissances 18

1. Problème de la généricité 18
2. Principaux diagnostics génériques des connaissances 19
   A. Knowledge tracing 19
   B. Modèle d'apprenant du Knowledge tracing 21
   C. Constraint-based 24
   D. Control-based 25
   E. Récapitulatif 28

II. Construction d'un diagnostic 28

1. Définition 28
2. Outils auteurs 29
   A. Principe 29
   B. Travaux 29
   C. Caractéristiques et limites 30
3. Apprentissage automatique 31
   A. Principe 31
   B. Travaux 32
   C. Caractéristiques et limites 34
4. Optimisation 36
   A. Principe 36

B. Travaux 36
C. Caractéristiques et limites 37
5. Synthèse des principaux verrous 37
III. Comparaison de diagnostics des connaissances 39
1. Comparaison de diagnostics génériques 40
A. Principe 40
B. Travaux 40
C. Caractéristiques et limites 41
2. Comparaison d'instances de diagnostics génériques 42
A. Principe 42
B. Travaux 43
C. Caractéristiques et limites 45
3. Synthèse des principaux verrous 46
IV. Conclusion 46

SECTION II : CONTRIBUTIONS THÉORIQUES 48
Chapitre 3 : Charactérisation d'une technique de diagnostic
I. Diagnostic comportemental et diagnostic épistémique 49
II. Technique de diagnostic générique 50
III. Modèle de diagnostic générique 51
1. Principe 51
2. Formalisation 51
3. Instanciation au domaine 53
4. Évaluation du champ d'application 53
A. Évaluation empirique 53
B. Évaluation du champ d'application 56
5. Opérationnalisation 57
6. Implémentation 59
A. Définition 59
B. Exemples de transposition 60
C. Compatibilité avec un modèle de diagnostic 60
IV. Implications liées à la formalisation 62
V. Conclusion 63

SECTION III : CONTRIBUTIONS À LA CONSTRUCTION ET À LA COMPARAISON 66

Chapitre 4 : Assistance à la construction de techniques de diagnostic des connaissances

I. Aperçu 66

II. Modélisation des traces et des connaissances..... 67

1. CREAM-C..... 68

2. Conception de l'ontologie des connaissances..... 69

3. Association entre l'ontologie et les traces 70

III. Algorithme semi-automatique d'instanciation des techniques 70

1. Définition du problème algorithmique..... 70

A. Définition et propriétés du problème d'instanciation d'une technique de diagnostic 70

B. Heuristique de recherche locale pour résoudre l'explosion combinatoire..... 72

2. Algorithme d'instanciation d'une technique de diagnostic 74

A. Notations 74

B. Principe et exemple illustratif 74

C. Recherche locale pour l'association entre le modèle de diagnostic et les traces (étape 1) 76

D. Instanciation et implémentation (étape 2) 79

E. Apprentissage des paramètres (étape 3) 83

F. Librairies 85

IV. Exemple d'application de l'algorithme..... 86

1. Control-based 87

2. Knowledge tracing..... 91

3. Constraint-based 93

4. Ontologie détaillée 95

V. Caractéristiques et limites..... 97

1. Propriétés de l'algorithme d'apprentissage 97

2. Limites 97

VI. Retour sur les scénarios..... 98

1. Premier cas d'usage..... 98

2. Second cas d'usage..... 98

VII. Conclusion 99

Chapitre 5 : Assistance à la comparaison de techniques de diagnostic des connaissances

I. Aperçu 102

II. Critères de comparaisons 102

1. Définition 102

2. Caractéristiques des critères de comparaison 103

A. Nature du calcul 103

B. Type du résultat 104

C. But 105

D. Documentation d'un critère 105

3. Interprétation des critères de comparaison 105

4. Exemples de critères 106

A. Précision de la prédiction 106

B. Racine de l'erreur quadratique moyenne 106

C. Akaike Information Criterion 107

D. Bayesian Information Criterion 107

E. Area Under the Curve ROC 107

F. Temps d'exécution 108

G. Complexité de la technique 108

H. Entropie de Shannon 109

I. Corrélation avec un diagnostic externe 109

J. Sensibilité aux données manquantes 110

K. Inférence immédiate 110

L. Impact sur le choix d'un type de rétroaction 110

5. Description d'un critère : impact sur le choix d'un type d'aide 111

A. Principe 111

B. Prérequis 111

C. Algorithme 111

III. Exemples 113

1. Précision de la prédiction 114

2. Corrélation avec un diagnostic externe 115

3. Impact sur le choix d'un type d'aide 116

IV. Retour sur les cas d'usage 117

1. Premier cas d'usage 117
2. Second cas d'usage 117
V. Conclusion 118

Chapitre 6 : Présentation de PlaCID

I. Introduction 120
II. Aperçu de la plateforme 121
1. Schéma général 121
2. Bases de modèles de diagnostic, d'implémentations et de critères 124
3. Construction de techniques de diagnostic 124
4. Comparaison de techniques de diagnostic 124
III. Architecture logicielle 125
1. Technique de diagnostic 125
2. Traces 127
3. Critères de comparaison 128
4. Classe principale 130
5. Interfaces 130
IV. Utilisation de la plateforme 132
V. Limites de la plateforme 135
VI. Conclusion 135

SECTION IV : EVALUATIONS ET CONCLUSION 137

Chapitre 7 : Evaluations

I. Méthodologie d'évaluation 137
II. Bases de traces d'apprenants 139
1. Geometry Tutor 139
2. Reading Tutor 140
3. TELEOS 142
4. APLUSIX 143
5. Collecte et volumétrie 144
III. Première expérimentation : évaluation des méthodes d'assistance 145
1. Méthodologie 145

2. Évaluation sur l'application des critères de comparaison 147
3. Évaluation de la méthode de construction 153
4. Retour sur les questions de recherche 157
IV. Seconde expérimentation : développement d'un critère de comparaison spécifique aux EIAH 158
1. Méthodologie 158
2. Application du critère de comparaison 159
3. Résultats du critère de comparaison 160
4. Retour sur les questions de recherche 163
V. Troisième expérimentation : développement d'une nouvelle technique de diagnostic 163
1. Méthodologie 163
2. Modèle de diagnostic Praxéologie 164
3. Retour sur les questions de recherche 167
VI. Conclusion 167

Chapitre 8 : Conclusion et perspectives

I. Bilan des contributions 170
II. Retour sur les questions de recherches 171
1. Rappel des questions de recherche et des principaux verrous 171
2. Réponses aux questions de recherche 172
3. Positionnement par rapport à la littérature 175
III. Prises de position par rapport aux contributions et limites 176
1. Rôle du concepteur et coût d'utilisation 176
A. Compétences du concepteur 176
B. Coût d'utilisation de PlaCID 179
2. Confiance dans les résultats et possibles biais 180
A. Confiance dans les techniques construites 180
B. Validité des résultats des critères de comparaison 181
3. Sur le diagnostic des connaissances 181
A. Prise de position induite par notre formalisation 181
B. Processus de construction et d'évaluation des techniques de diagnostic des connaissances 182
C. Positionnement par rapport à la communauté EIAH 183

IV. Perspectives 183
1. Prolongement des contributions 184
A. Utilisabilité de PlaCID 184
B. Expérimentations sur l'utilité 185
C. Introduction d'interactions et de paramètres dans l'algorithme de construction et dans le calcul du score biaisé 186
D. Collecte et partage de critères et critères EIAH 187
2. Transposition de nos questions de recherche dans d'autres domaines des EIAH ... 189
A. Collaboration, émotions, jeux sérieux et modèles ouverts 190
B. Conception d'EIAH 191

Chapitre 9 : Bibliographie 193

Annexes 203

Chapitre 1 : Introduction

Sommaire

I. Concepts généraux 11

II. Motivation de la recherche 12

III. Questions et hypothèses de recherche 13

IV. Cas d'usage 14

1. Cas d'usage 1 : ajout d'un diagnostic des connaissances à un EIAH 14
2. Cas d'usage 2 : comparaison d'un nouveau diagnostic avec l'existant 15

V. Plan du mémoire 15

I. Concepts généraux

Cette thèse se situe dans le domaine de recherche ayant pour objet d'études les EIAH : Environnements Informatiques pour l'Apprentissage Humain. Il s'agit d'environnements destinés à favoriser l'apprentissage dans un domaine précis, par exemple la géométrie en classe de quatrième. Les EIAH sont extrêmement variés ; dans ce travail, nous nous intéressons aux situations d'apprentissage individuelles, sans interaction ou collaboration entre apprenants.

Dans ce cadre, le diagnostic des connaissances (*knowledge diagnostic*) est un processus informatique qui vise à déterminer l'état des connaissances d'un apprenant (maîtrisées ou non) à partir de ses interactions avec l'EIAH. Il y a donc trois notions principales : les interactions avec l'EIAH (entrée du diagnostic), le diagnostic des connaissances (le processus), et l'état des connaissances (le résultat).

Les interactions de l'apprenant avec l'EIAH sont collectées par l'EIAH dans les traces. Le contenu des traces dépend donc de chaque EIAH, notamment car il existe très peu de standards pour décrire des traces d'EIAH. Généralement, on trouve dans ces traces l'identifiant de l'apprenant, les actions ou étapes de résolution des problèmes réalisées par l'apprenant, les réponses de l'apprenant, et une information sur le temps.

Le diagnostic des connaissances prend en entrée ces traces pour inférer ou déduire l'état des connaissances de l'apprenant. Selon les diagnostics, il est possible de diagnostiquer les connaissances de l'apprenant trace par trace, c'est-à-dire à chaque action ou étape réalisée par un apprenant, ou bien à la suite d'une séquence d'actions ou d'étapes (typiquement, à la fin d'un exercice). Le diagnostic des connaissances est donc un processus réalisé tout au long de l'interaction de l'apprenant avec l'EIAH, puisque les connaissances de l'apprenant évoluent au cours du temps. Pour concevoir un tel diagnostic, il faut pouvoir représenter les

connaissances, et développer un processus informatique capable de lire les traces de l'apprenant et d'effectuer une inférence ou une déduction. Nous présenterons plus en détails quelques-unes des nombreuses approches existantes pour faire un tel diagnostic dans l'état de l'art.

L'état des connaissances de l'apprenant est le résultat du diagnostic, souvent appelé « profil de l'apprenant » ou « modèle de l'apprenant » (*student model* en anglais). Il n'existe là encore pas de standard pour leur représentation, mais ils se présentent le plus souvent comme une liste de connaissances, pouvant être indépendantes entre elles ou non ; à chaque connaissance est associé le niveau de maîtrise de l'apprenant, par exemple « su » ou « non su », éventuellement avec un degré de confiance. Des exemples plus précis et complets seront donnés par la suite.

La création d'un diagnostic des connaissances pour un EIAH est une tâche pluridisciplinaire qui requiert : un travail didactique afin d'identifier les connaissances du domaine à diagnostiquer, et un travail informatique pour l'implémentation et l'intégration dans l'EIAH.

II. Motivation de la recherche

Le problème du diagnostic des connaissances est la question de sa fidélité en regard des connaissances réellement maîtrisées par l'apprenant ; on parle souvent de performance ou de précision du diagnostic. Comme montré par Self (Self, 1990), il est impossible de prouver formellement en l'EIAH que le résultat d'un diagnostic des connaissances est correct, par impossibilité de mesurer réellement l'état d'une connaissance chez un apprenant. Un diagnostic est donc nécessairement approximatif, et vise à maximiser la performance du modèle d'apprenant qui en résulte. Toutefois, Self montre également que malgré cette imprécision, le diagnostic des connaissances reste utile pour de nombreuses tâches ou de nombreux autres processus de l'EIAH : choisir une aide ou un indice à donner à l'apprenant, choisir le prochain exercice à lui donner afin de renforcer une connaissance ou de déstabiliser l'apprenant, personnaliser la présentation des problèmes et des indices EIAH … Un diagnostic des connaissances s'évalue donc par ses performances ou son utilité.

Un second problème est donc de pouvoir mesurer soit la performance, soit l'utilité d'un diagnostic des connaissances. Or, comme une expérimentation de Gong et al. le suggère, nous ne savons actuellement pas quand et pourquoi un diagnostic est plus performant ou plus utile qu'un autre. De nombreux facteurs influent de façon indéterminée sur le diagnostic, comme : le domaine, les exercices, les apprenants, la qualité et quantité des traces, leur format… De plus, l'utilité d'un diagnostic dépend de l'objectif de son concepteur, c'est-à-dire de la façon dont il veut exploiter le résultat du diagnostic dans l'EIAH ou des informations qu'il veut obtenir sur les apprenants ayant utilisé l'EIAH. Le choix d'un diagnostic *a priori* peut donc être ardu et ne pas reposer sur des résultats scientifiques.

Un troisième problème est la dépendance du diagnostic au domaine d'apprentissage. En effet, les connaissances sont différentes selon les domaines, de même que les informations collectées dans les traces. Cette difficulté explique que de nombreux diagnostics des

connaissances soient conçus et implémentés de façon ad hoc pour un EIAH, c'est-à-dire qu'ils dépendent fortement des connaissances du domaine, des traces de l'EIAH et de l'architecture de l'EIAH. Or, le travail de conception du diagnostic peut être long, complexe et donc coûteux, pour une utilité qui ne peut être mesurée *a priori*. De plus, un diagnostic ad hoc n'est pas réutilisable pour un autre EIAH sans un travail d'adaptation.

Enfin, le diagnostic des connaissances étant un processus informatique, il se pose la question de son implémentation. Différentes implémentations peuvent avoir un impact sur la performance et l'utilité du diagnostic, à mettre en relation avec l'efficacité de chaque implémentation en termes de temps de calcul et de coût de développement. Le dernier problème est donc la dépendance du diagnostic à son implémentation.

Nos motivations à conduire cette recherche nous amènent donc à nous intéresser à deux problèmes :

- Le problème de mesurer la performance et l'utilité d'un diagnostic ainsi que l'efficacité de son implémentation ; ce problème est directement lié au choix du diagnostic à intégrer dans un EIAH ;
- Le problème de construire et réutiliser un diagnostic sans avoir, pour un EIAH donné, de garantie ni sur son utilité, ni sur ses performances, ni sur l'efficacité de son implémentation.

III. Questions et hypothèses de recherche

Les questions de recherches que nous posons à partir des motivations sont les suivantes :

- Comment mesurer les performances et l'utilité de plusieurs diagnostics des connaissances ?
- Comment réduire le coût de conception de diagnostics des connaissances ?
- Comment guider et assister *a priori* le choix d'un diagnostic des connaissances dans un EIAH, pour un domaine d'apprentissage particulier ?

Les verrous importants sont la difficulté de choisir et créer un diagnostic des connaissances *a priori*, c'est-à-dire sans devoir évaluer le diagnostic sur des apprenants en classe lors d'expérimentations. En effet, il est coûteux de développer et implémenter un diagnostic des connaissances s'il se révèle peu performant lors d'expérimentations *a posteriori*. C'est pourquoi une assistance pour le choix d'un diagnostic des connaissances *a priori* pour un EIAH donné peut permettre de réduire ce coût, car comme nous l'avons souligné, un verrou est l'inexistence de garantie sur les performances d'un diagnostic pour un domaine donné. Pour assister le choix d'un diagnostic, il faut donc pouvoir évaluer *a priori* pour l'EIAH considéré les performances d'un ensemble de diagnostics existants dans la littérature.

Nous faisons donc l'hypothèse que les questions de recherche ci-dessus peuvent être adressées par une assistance aux concepteurs de diagnostic des connaissances: assistance pour évaluer la performance de diagnostics des connaissances pour un domaine et/ou un

EIAH donné, pour en choisir un et pour l'implémenter. Pour cela, nous faisons l'hypothèse qu'il faut pouvoir comparer les performances de différents diagnostics des connaissances qui soient le plus possible indépendants du domaine, en prenant en compte les objectifs du concepteur et les spécificités du domaine du concepteur. Une telle comparaison est soumise à deux contraintes. Premièrement, il est nécessaire de construire les différents diagnostics au préalable pour pouvoir les comparer. Deuxièmement, les diagnostics des connaissances fonctionnent en prenant en entrée des traces d'apprenant ; il faut donc collecter ces traces pour pouvoir comparer les résultats de différents diagnostics des connaissances sur ces traces.

Nous pouvons ainsi reformuler plus précisément nos questions de recherche comme suit :

> Comment assister au moyen de traces d'apprenant des concepteurs de diagnostic des connaissances pour la création et la comparaison de différents diagnostics des connaissances de façon indépendante du domaine?

Nous proposons dans ce travail une méthode réifiée dans une première plateforme nommée PlaCID (Platform for Comparing et Instanciating Diagnostics) pour répondre à cette question de recherche.

IV. Cas d'usage

L'utilisateur visé par notre méthode d'assistance est un concepteur de diagnostic des connaissances, ce qui nécessite donc des prérequis dans le domaine. Parmi ces utilisateurs, nous identifions deux profils. Les premiers sont des experts du diagnostic des connaissances, qui s'intéressent à proposer de nouveaux diagnostics, à les améliorer à partir de traces d'apprenant et à les évaluer sur un ou plusieurs domaines. Les seconds sont des utilisateurs des diagnostics des connaissances, qui utilisent les modèles d'apprenant résultants pour les processus de rétroactions, personnalisations, etc.

Le terme « concepteur » est donc une simplification d'usage, mais peut tout à fait désigner plusieurs personnes.

Nous présentons dans cette question des cas d'usage pour les deux grandes problématiques de notre méthode : assistance à la conception et assistance à la comparaison de diagnostic des connaissances.

1. Cas d'usage 1 : ajout d'un diagnostic des connaissances à un EIAH

Léa est une chercheuse en EIAH, mais pas une experte du diagnostic des connaissances. Elle a construit un EIAH destiné à des internes en médecine pour l'apprentissage de la chirurgie. Il s'agit d'un simulateur doté d'une interface 3D sur lequel les apprenants s'exercent à différentes opérations. Le simulateur est capable de tracer les différentes actions réalisées par l'apprenant au cours de la résolution des problèmes, ainsi que de calculer la validité de ces actions en rapport au domaine, à partir de règles expertes. Plusieurs expérimentations

de son simulateur auprès d'apprenants lui ont permis de collecter une base de traces. Toutefois, ses traces sont peu nombreuses car il reste coûteux de mobiliser des internes en médecine. La chirurgie est de plus un domaine parfois mal défini, où un même problème peut être résolu de nombreuses manières.

Léa souhaite intégrer un diagnostic des connaissances à cet EIAH pour suivre l'acquisition des connaissances des apprenants lors de l'usage de l'EIAH sur plusieurs séances. Ce profil doit servir pour que l'EIAH puisse donner des aides adaptées à chaque apprenant. Elle a donc réalisé une première étude du domaine avec des experts afin d'identifier les connaissances visées par son EIAH.

Léa souhaite utiliser notre plateforme pour choisir parmi différents diagnostics celui qui correspond le mieux à son EIAH, caractérisé par une interface complexe, des traces en faible quantité et un domaine parfois mal défini.

2. Cas d'usage 2 : comparaison d'un nouveau diagnostic avec l'existant

Nadia est une chercheuse dont la thématique de recherche porte sur l'étude des diagnostics des connaissances en EIAH. Elle propose deux nouveaux diagnostics indépendants du domaine ; la différence entre ses deux diagnostics est que l'un est basé sur un système d'inférence à base de règles expertes pour inférer le modèle de l'apprenant, tandis que le second est basé sur de l'inférence bayésienne prenant en compte l'incertitude. Elle souhaite comparer ces deux diagnostics avec ceux existants dans la littérature, afin d'obtenir des résultats d'évaluation publiables. Elle compte pour cela utiliser des traces existantes collectées auparavant au sein de deux EIAH différents, l'un portant sur la géométrie et l'autre sur l'algèbre. Ces deux EIAH sont capables de tracer les différentes étapes de résolution des problèmes réalisées avec succès ou non par les apprenants.

Nadia a effectué une étude au préalable pour identifier les connaissances visées dans ces deux EIAH, puis a implémenté ses deux diagnostics pour ces deux domaines. L'utilisation de notre plateforme doit permettre à Nadia de faire deux types de comparaison. Premièrement, elle souhaite construire différents diagnostics bien connus dans la littérature pour ces deux domaines afin de les comparer aux siens, à l'aide de diverses mesures de performance. Deuxièmement, elle souhaite comparer ses deux diagnostics, c'est-à-dire étudier la meilleure implémentation pour inférer le modèle de l'apprenant entre règles expertes et inférence bayésienne.

V. Plan du mémoire

Ce mémoire est organisé autour des deux thématiques principales de notre question de recherche : la construction et la comparaison de diagnostic des connaissances.

Le chapitre 2 présente les diagnostics des connaissances les plus connus dans la littérature puis traite de l'état de l'art sur le sujet. Il conclut enfin en résumant les principaux verrous ou manques actuels dans la recherche sur les diagnostics des connaissances. Le chapitre 3

s'attache à définir concrètement la notion de « diagnostic des connaissances » au moyen d'un modèle générique. Ce modèle sert de base pour la comparaison et la construction de différents diagnostics des connaissances. Cette partie détaille aussi les implications et les hypothèses faites par ce modèle, et donc par notre travail. Le chapitre 4 porte sur l'assistance à la construction de diagnostics des connaissances au moyen d'un algorithme d'apprentissage semi-automatique, au sens qu'il nécessite un apport de sémantique de la part du concepteur de diagnostics sur les traces. Le chapitre 5 traite de l'assistance à la comparaison de diagnostic des connaissances au moyen de différents critères de comparaison que nous proposons. Nous détaillons également comment ces critères peuvent être utilisés et interprétés, ainsi que la procédure pour définir de nouveaux critères de comparaison. Le chapitre 6 présente PlaCID, une première plateforme qui réifie nos propositions sur l'assistance à la construction et la comparaison de diagnostic des connaissances. Le chapitre 7 porte sur l'évaluation de notre méthode d'assistance. Il s'agit d'une expérimentation basée sur l'utilisation de quatre bases de traces d'apprenants collectées auparavant au sein de quatre EIAH : TELEOS (chirurgie), Reading Tutor (lecture de l'anglais), Geometry Tutor (géométrie des aires) et APLUSIX (algèbre élémentaire). Le chapitre 8 conclut en dressant un bilan des différentes contributions par rapport à l'état de l'art et aborde les limites de nos propositions, puis propose différentes perspectives pour réduire ces limites et, de futures questions de recherche. Le chapitre 9 est un lexique des termes techniques ou scientifiques requis pour la compréhension du mémoire.

Chapitre 2 : État de l'art

Sommaire

I. Diagnostic des connaissances 18
1. Problème de la généricité 18
2. Principaux diagnostics génériques des connaissances 19
A. Knowledge tracing 19
B. Constraint-based 24
C. Control-based 25
D. Récapitulatif 28
II. Construction d'un diagnostic 28
1. Définition 28
2. Outils auteurs 29
A. Principe 29
B. Travaux 29
C. Caractéristiques et limites 30
3. Apprentissage automatique 31
A. Principe 31
B. Travaux 32
C. Caractéristiques et limites 34
4. Optimisation 36
A. Principe 36
B. Travaux 36
C. Caractéristiques et limites 37
5. Synthèse des principaux verrous 37
III. Comparaison de diagnostics des connaissances 39
1. Comparaison de diagnostics génériques 40
A. Principe 40
B. Travaux 40
C. Caractéristiques et limites 41
2. Comparaison d'instances de diagnostics génériques 42
A. Principe 42

B. Travaux 43
C. Caractéristiques et limites 45
3. Synthèse des principaux verrous 46
IV. Conclusion 46

Nous allons présenter dans ce chapitre l'état de l'art sur le diagnostic des connaissances. Nous présenterons dans la section I différents diagnostics des connaissances classiques dans la littérature, qui ont la propriété d'être générique, donc applicable à différents domaines d'apprentissage. Puis nous ferons dans la section II l'état de l'art sur les méthodes et outils pour construire ces diagnostics, c'est-à-dire pour instancier ces diagnostics à un domaine donné, et nous présenterons dans la section III l'état de l'art sur la comparaison de diagnostics des connaissances.

I. Diagnostic des connaissances

1. Problème de la généricité

Nous avons défini dans l'introduction le diagnostic des connaissances comme un processus qui :

- prend en entrée des traces d'apprenants,
- infère ou déduit l'état des connaissances de l'apprenant,
- retourne comme résultat du diagnostic un modèle de l'apprenant.

Il existe de nombreux diagnostics des connaissances différents dans la littérature, que l'on peut séparer en deux grandes familles : les diagnostics dépendants et indépendants du domaine.

Les diagnostics dépendants d'un domaine sont généralement conçus de façon ad hoc pour un EIAH et ne sont pas prévus pour être réutilisés dans un autre EIAH. Parmi ces diagnostics, nous pouvons citer Pépite (Jean, 2000) ou Andes (Vanlehn et al., 2005). Dans ces deux EIAH, le diagnostic des connaissances utilise des éléments spécifiques du domaine (algèbre pour Pépite et physique pour Andes), tels qu'une grille didactique de l'algèbre élémentaire dans Pépite ou les solutions des exercices de physique dans Andes. Nous pouvons écarter ces diagnostics de notre étude car il n'y a que peu d'intérêt d'assister la construction d'un diagnostic qui n'est applicable que dans un seul domaine. Il peut faire sens de comparer ces diagnostics avec des diagnostics génériques sur le domaine considéré, mais là encore, il ne semble pas possible de considérer tous les diagnostics dépendants d'un domaine pour une méthode d'assistance à la comparaison automatique. Nous situons donc notre travail au niveau des diagnostics génériques. Nous reviendrons sur le sujet des diagnostics non génériques ultérieurement afin de montrer plus précisément les difficultés qu'ils posent en rapport à notre question de recherche.

Les diagnostics indépendants du domaine, dont le processus est entièrement générique, sont ceux qui sont applicables sur plusieurs domaines. Cela ne signifie pas qu'ils soient pertinents et utilisables pour la totalité des domaines, mais qu'ils sont formulés de façon générique et peuvent s'appliquer au moins sur un ensemble de domaines, par exemple les domaines scientifiques exacts.

Définition

> Un **diagnostic des connaissances générique** est un diagnostic applicable à plusieurs domaines d'apprentissage.

Depuis au moins les années 1990, ces diagnostics génériques tendent à être de plus en plus utilisés dans le domaine des EIAH. Ils présentent plusieurs avantages : la possibilité de réinvestir l'expérience acquise en instanciant un diagnostic générique à plusieurs domaines, l'existence d'outils largement utilisés en lien avec ce diagnostic (comme des outils auteurs ou des bases de traces), et de nombreuses expérimentations sur différents domaines qui renforcent la validité scientifique du diagnostic. En contrepartie, l'utilisation d'un diagnostic générique présente un coût pour le concepteur, qui doit comprendre sa spécification, ses limites et son utilisation.

2. Principaux diagnostics génériques des connaissances

Nous détaillons ci-dessous quelques-uns des diagnostics génériques les plus répandus (Knowledge tracing, Constriant-based, Control-based), en détaillant dans l'ordre : le principe théorique, le modèle d'apprenant à inférer, les traces considérées et le traitement du diagnostic.

A. Knowledge tracing

Principes théoriques

Le Knowledge tracing (Corbett et Anderson, 1995) dérive à l'origine de la théorie ACT-R d'Anderson (Anderson, 1996, 1983), qui vise à modéliser l'ensemble des processus cognitifs. L'objectif d'ACT-R est ainsi de spécifier l'organisation et le fonctionnement de la mémoire et de l'apprentissage. Le postulat principal part d'une représentation de la mémoire en deux unités distinctes : déclarative et procédurale. La mémoire déclarative se présente sous la forme d'un ensemble de connaissances factuelles ou explicites, tandis que la mémoire procédurale intègre des processus ou schèmes de raisonnements implicites ou tacites. Selon Anderson, l'apprentissage de nouvelles connaissances est procédural : il vise au transfert de la connaissance depuis la mémoire déclarative vers la mémoire procédurale. Une des principales implications de cette hypothèse est que l'apprentissage doit se faire essentiellement par la pratique, dans le but de rendre ce transfert effectif. En d'autres termes, l'apprentissage de connaissances se fait par la résolution répétée de problèmes impliquant la mobilisation de cette connaissance.

La théorie ACT-R a été implémentée dans les EIAH de type « tuteur cognitif » (Anderson et al., 1995) d'abord sous la forme de règles de production modélisant la résolution d'un problème via un enchaînement de buts et de sous-buts. Concrètement, une règle de

production est une règle IF/THEN, la partie IF spécifiant un ensemble de préconditions et la partie THEN le but à atteindre si ces préconditions sont remplies.

Anderson (Anderson, 1983) donne l'exemple de la résolution d'une addition de deux nombres en base 10 :

Règle 1 :
SI le but est de résoudre une addition
ALORS le sous-but est de traiter la colonne 1

Règle 2 :
SI le but est de traiter la colonne i
ET la colonne i existe
ET les chiffres de la colonne sont x1 et x2 avec la retenue x3
ET x1+x2+x3=z
ET z<10
ALORS écrire z dans la colonne i du résultat
ET le sous-but est de traiter la colonne i+1

Règle 3 :
SI le but est de traiter la colonne i
ET la colonne i existe
ET les chiffres de la colonne sont x1 et x2 avec la retenue x3
ET x1+x2+x3=z
ET z>=10
ALORS écrire le chiffre le plus à droite de z dans la colonne i du résultat
ET fixer la retenue de la colonne i+1 à z/10 arrondi à l'inférieur (créer la colonne si non existant)
ET le sous-but est de traiter la colonne i+1

Règle 4 :
SI le but est le traiter la colonne i
ET la colonne i n'existe pas
ALORS
Terminé

L'exemple est simplifié ici pour le cas où il faut créer la colonne si elle n'existe pas en cas de retenue sur la colonne la plus à gauche. Ces règles spécifient chaque étape (but) de résolution du problème qu'un apprenant doit suivre. Si une règle est enfreinte par un apprenant (i.e. la partie IF est satisfaite mais pas le THEN), alors il y a une erreur qui suggère une connaissance absente de la mémoire procédurale de l'apprenant, et le tuteur cognitif doit donner une rétroaction immédiate à l'apprenant pour y pallier.

Les règles de production dans ACT-R modélisent les connaissances du domaine mais ne prennent pas en compte l'apprenant. Le Knowledge tracing vise lui à modéliser l'apprenant, c'est-à-dire à inférer les connaissances du domaine maîtrisées ou non maîtrisées par l'apprenant, en se basant sur le principe d'ACT-R.

Modèle d'apprenant du Knowledge tracing

Le modèle d'apprenant inféré par le Knowledge tracing se compose de trois éléments : les problèmes ou exercices à résoudre (Problem), les étapes de résolution de chaque problème (Step), et les éléments de connaissances au sens d'ACT-R (Knowledge Component ou KC) qui doivent être maîtrisées à chaque étape. Un problème peut être résolu par une suite ordonnée d'étapes, et une étape nécessite une et une seule connaissance. Les connaissances, indépendantes entre elles, peuvent être dans deux états : apprises ou non apprises. Un tel modèle d'apprenant est souvent appelé Cognitive model, ou KC Model, en anglais.

Ci-dessous un exemple en géométrie avec un problème, trois étapes et deux KC différents :

Problem	**Step**	**Knowledge Component (KC)**
Calculer l'aire du parallélogramme ABCD	Longueur d'un côté	PARALLELOGRAM-SIDE (longueur d'un côté)
	Longueur d'un côté adjacent	PARALLELOGRAM-SIDE (longueur d'un côté)
	Calcul d'aire	PARALLELOGRAM-AREA (application de la formule de l'aire d'un parallélogramme)

Traces en entrée

Le Knowledge tracing requiert des traces contenant les données suivantes :

- L'apprenant
- Le problème en cours de résolution
- L'étape courante de résolution du problème
- L'évaluation du résultat produit par l'apprenant à l'étape courante : correct ou incorrect.

Ces traces sont donc temporellement ordonnées en fonction des étapes de résolution des problèmes. Elles impliquent que l'EIAH soit capable d'identifier l'étape courante du problème et d'évaluer à chaque étape le résultat (correct ou non). Les tuteurs cognitifs collectent généralement ces données à partir d'un arbre de résolution associé à chaque problème et dont les nœuds spécifient une étape et la réponse que doit donner l'apprenant à cette étape.

Traitement du diagnostic des connaissances

Le Knowledge tracing se base pour diagnostiquer les connaissances de chaque apprenant à partir de leurs traces sur quatre paramètres (probabilités) liés à chacune des connaissances :

- Pr(L0) : la probabilité de connaître la connaissance *a priori*, avant utilisation de l'EIAH
- Pr(L) : la probabilité d'apprendre la connaissance, qui passe donc de l'état « non apprise » à l'état « apprise » (L pour learn)
- Pr(G) : la probabilité de répondre correctement à une étape d'un problème sans connaître la connaissance (G pour guess)

- Pr(S) : la probabilité de faire une erreur à une étape tout en ayant déjà appris la connaissance (S pour slip)

Ces quatre paramètres doivent être fixés pour chaque connaissance *a priori*, soit par un expert du domaine, soit par apprentissage automatique à partir de traces. Ils sont prérequis. Les connaissances étant indépendantes entre elles dans le Knowledge tracing, ces paramètres sont uniques pour chaque connaissance. À chaque étape de résolution d'un problème, ces paramètres permettent d'inférer la probabilité d'apprendre la connaissance, selon que le résultat de l'étape est correct ou incorrect. Pour une étape quelconque *n,* la probabilité d'apprendre une connaissance $Pr(L_n)$ s'infère via la formule suivante :

- $Pr(L_n \mid Correct_n) = Pr(L_{n-1} \mid Correct_n) + ((1 - Pr(L_{n-1} \mid Correct_n)) * Pr(L)$
- $Pr(L_n \mid Incorrect_n) = Pr(L_{n-1} \mid Incorrect_n) + ((1 - Pr(L_{n-1} \mid Incorrect_n)) * Pr(L)$

Avec :

- $Pr(L_{n-1} \mid Correct_n) =(Pr(L_{n-1}) * (1 - Pr(S)) / (Pr(L_{n-1}) * (1 - Pr(S)) + (1 - Pr(L_{n-1})) * (Pr(G))$
- $Pr(L_{n-1} \mid Incorrect_n) = (Pr(L_{n-1}) * (Pr(S)) / (Pr(L_{n-1}) * Pr(S) + (1 - Pr(L_{n-1})) * (1 - Pr(G))$
- $Correct_n$ et $Incorrect_n$: le résultat de l'étape *n* est correct ou incorrect

$Pr(L_n)$ est la probabilité d'apprendre une connaissance KC à la $n^{ème}$ fois que l'apprenant doit mobiliser KC, sachant si l'apprenant a fait une réponse correcte ou non à cette étape. Le processus étant markovien, $Pr(L_n)$ ne dépend que de $Pr(L_{n-1})$ et utilise le paramètre L (probabilité d'apprendre à une étape quelconque). $Pr(L_{n-1})$ ne dépend que des paramètres S et G, pour prendre en compte l'incertitude quant à la réponse à l'étape *n* (correcte ou non) : l'apprenant peut faire faux tout en maîtrisant la connaissance (*slip*, paramètre S) ou deviner la bonne réponse sans maîtriser la connaissance (*guess*, paramètre G).

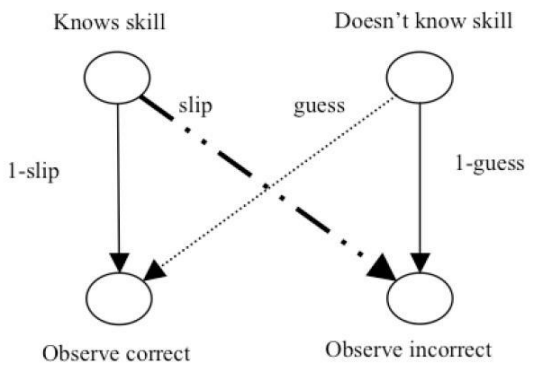


Figure 1 : Schéma illustrant la prédiction des réponses de l'apprenant (correct ou incorrecte) à partir des deux états (*Knows skill* et *doesn't know skill*) d'une connaissance et des paramètres *slip* et *guess* associés à cette connaissance (Beck et Sison, 2006).

Le Knowledge tracing peut facilement être implémenté via un ensemble de modèles de Markov cachés (un par connaissance), avec comme état caché à deux états une connaissance (apprise ou non apprise) et comme état observable le résultat de l'étape

(correct ou incorrect). Il est possible, à partir de l'état d'une connaissance et des paramètres, de prédire le résultat de la prochaine étape de l'apprenant, comme illustré Figure 1.

Variantes pour le traitement

Il existe plusieurs variantes au traitement du diagnostic tel que décrit ci-dessus, basées sur des régressions logistiques. Seul le traitement varie : le principe originel (ACT-R), le modèle d'apprenant à mettre à jour et la nature des traces restent donc identiques.

La première variante, nommée Additive Factor Model ou AFM (Cen et al., 2008), met à jour la probabilité d'apprendre une connaissance (un Knowledge Component) via une courbe de régression logistique définie comme suit :

$$\textbf{AFM:} \quad \ln \frac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj}(\gamma_k N_{ik})$$

Avec :

- i l'apprenant
- j une étape du problème
- k une connaissance
- p_{ij} la probabilité que l'apprenant fera une réponse correcte à l'étape j
- Qune matrice avec Q_{kj} une case de la matrice indiquant si la connaissance k est associée à l'étape j
- N_{ik} le nombre d'étapes de résolution de problèmes passées où l'apprenant i a dû mobiliser la connaissance k
- β_k un coefficient représentant la difficulté de la connaissance k
- γ_k un coefficient représentant l'importance du temps nécessaire pour apprendre la connaissance k
- θ_i un coefficient représentant la facilité d'apprentissage de l'apprenant

Ce diagnostic ne prend pas en compte le nombre d'erreurs passées faites par l'apprenant. Il a donc été raffiné en un second diagnostic nommé Performance Factor Model ou PFM (Pavlik et al., 2009) ou Performance Factor Analysis (PFA), qui se définit comme suit :

$$\textbf{PFM:} \quad \ln \frac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj}(\mu_k S_{ik} + \rho_k F_{ik})$$

Avec à la place de (γ_k N_{ik}) :

- S_{ik} le nombre d'étapes de résolutions de problèmes précédemment effectuées par l'apprenant de façon correcte
- F_{ik} le nombre d'étapes de résolutions de problèmes précédemment effectuées par l'apprenant de façon incorrecte
- μ un coefficient sur le bénéfice des précédents succès sur l'apprentissage
- ρ un coefficient sur le bénéfice des précédents échecs sur l'apprentissage

B. Constraint-based modeling

Principes théoriques

Le Constraint-based modeling (Mitrovic et al., 2007; Ohlsson, 1994) se base sur une théorie nommée *performance errors* (Ohlsson, 1996) qui stipule que l'apprentissage se fait par la confrontation de l'apprenant avec ses erreurs, postulant que l'apprentissage d'une connaissance n'implique pas seulement la mémoire, mais également la mise en pratique de la connaissance. Cette mise en pratique est complexifiée selon les domaines par les nombreuses possibilités de résoudre un problème et les diverses informations à prendre en compte. Cette théorie se base aussi sur une observation pragmatique, qui est qu'il est souvent impossible de modéliser l'ensemble des connaissances et des étapes de résolution des problèmes dans les domaines complexes. La théorie propose donc de modéliser seulement les états des problèmes ayant un intérêt pédagogique car permettant de relever une erreur typique du domaine.

Il ne s'agit ici plus de modéliser les étapes de résolution des problèmes, mais les états dans lesquels des erreurs ayant un intérêt pédagogique peuvent se produire. Sur l'exemple de l'addition, plusieurs états permettent d'identifier s'il y a une erreur dans la résolution du problème faite par l'apprenant :

Contrainte 1 :
Cr : SI plusieurs colonnes n'ont pas été traitées
Cs : ALORS il faut traiter la colonne la plus à droite

Contrainte 2 :
Cr : SI la somme d'une colonne et x1
ET la colonne a une retenue x2
Cs : ALORS le résultat de la colonne est x1+X2

Contrainte 3 :
Cr : SI le résultat x1 d'une colonne est supérieur à dix
Cs : ALORS il faut propager une retenue égale à x1/10 arrondie à l'inférieur

Ces contraintes identifient les états du problème où une erreur importante peut subvenir. Elles ne traitent en revanche pas du plan de résolution du problème.

Modèle d'apprenant du Constraint-based

Dans les EIAH, le Constraint-based (Mitrovic et al., 2007; Ohlsson, 1994) est un diagnostic des connaissances permettant d'implémenter cette théorie. Les connaissances requises dans un état d'un problème sont décrites par des contraintes. Une contrainte est déclarative ; elle peut être écrite comme une règle IF/THEN composé de deux clauses Cr et Cs :

- IF Cr est vrai
- THEN Cs doit être vrai

Cr (pour relevance condition) spécifie l'état du problème dans lequel la contrainte s'applique, et Cs (pour satisfaction condition) spécifie ce que doit avoir obtenu l'apprenant

dans cet état, i.e. l'état dans lequel doit être le problème quand Cr est vrai. Si une contrainte est enfreinte, elle indique que l'apprenant a fait une erreur, et donc une opportunité d'apprentissage.

Le Constraint-based modélise donc les connaissances comme un ensemble de contraintes. Le modèle de chaque apprenant est la liste des contraintes avec pour chacune : le nombre de fois où elle a été utilisée (i.e. les états où Cr était vraie) et le nombre de fois où l'utilisation était correcte, c'est-à-dire que les conditions Cr et Cs étaient vraies.

Ci-dessous un exemple en géométrie :

Contrainte 1 : Cr: SI l'angle b1 de la base d'un triangle isocèle est connu ET l'apprenant a calculé la taille de l'autre angle de la base Cs: ALORS b1=b2

Traces en entrée

Le Constraint-based requiert dans les traces des apprenants des données sur :

- L'identifiant de l'apprenant
- Le problème en cours
- L'état du problème avant intervention de l'apprenant
- L'état du problème après intervention de l'apprenant

Traitement du diagnostic des connaissances

L'inférence dans le Constraint-based lors de l'intervention d'un apprenant consiste à parcourir l'ensemble C des contraintes :

Pour toutes les contraintes C_i=(Cr_i, Cs_i) de C Si Cr_i est vrai et Cs_i est vrai, alors C_i est satisfaite Sinon Si Cr_i est vrai et Cs_i est faux, alors C_i est enfreinte Fin

C. Control-based

Principes théoriques

Le Control-based (Minh Chieu et al., 2010) est basé sur le modèle didactique cKc (Balacheff et Gaudin, 2002; Balacheff, 1995). Dans ce modèle, les connaissances du domaine, appelées en fait conceptions, sont astreintes à un domaine de validité et dépendent de la situation d'apprentissage. cKc modélise donc les conceptions des apprenants et les caractéristiques de la situation qui permet l'apprentissage de ces conceptions. Une conception se définit comme un quadruplet :

- P : un ensemble de problèmes
- O : un ensemble d'opérateurs
- R : un système de représentation
- C : un ensemble de contrôles

L'ensemble des problèmes est celui que la conception permet de résoudre. Il s'agit de variables caractérisant les problèmes. Les opérateurs sont les productions ou comportements possibles pour la résolution des problèmes. Le système de représentation permet l'expression des problèmes et des opérateurs. Enfin, les contrôles permettent de valider la décision de l'emploi d'un opérateur et l'état du problème résultant.

Balacheff donne un exemple de deux conceptions sur l'addition en base 10 :

Conception 1 : addition « décimale » P : tout problème d'addition à deux ou plusieurs nombres entiers O : algorithme (usuel) de l'addition R : écriture décimale des nombres C : contrôle pas-à-pas de l'algorithme Conception 2 : addition avec les doigts P : problèmes du type addition de deux nombres entiers inférieurs à 10 O : prononcer les nombres tout en déployant les doigts successivement R : geste du déploiement des doigts, comptage oral à partir d'un nombre donné C : contrôle auro-visuel de la correspondance entre le nombre de doigts déployés et les nombres énumérés, le contrôle de l'énoncé de la comptine

Le Control-based se base sur ce modèle didactique en représentant les connaissances comme l'ensemble des contrôles C, et l'ensemble des problèmes est redéfini comme l'ensemble des variables didactiques caractérisant les problèmes considérés dans l'EIAH. Le concept de « conception » n'est pas utilisé, car un seul quadruplet (P,O,R,C) est représenté.

Modèle d'apprenant du Control-based

Le modèle d'apprenant inféré par le Control-based est un 5-uplet (P,O,R,C,VS) comme défini ci-dessus, sachant qu'un opérateur est lié à un ou plusieurs problèmes, et un contrôle est lié à un ou plusieurs problèmes, un opérateur et un ou plusieurs registres de représentations. VS, pour variables de situation, indique si l'usage de l'opérateur est valide pour le problème (généralement de façon binaire : correct ou non). Un contrôle peut être dans trois états :

- Mis en jeu de façon valide
- Mis en jeu de façon invalide dans la situation courante
- Non mis en jeu de façon invalide (i.e. non mis en jeu alors qu'il aurait dû l'être)

Pour chaque contrôle c de C, ces états sont modélisés par une probabilité, respectivement Pr(c=Valide), Pr(c=Invalide), Pr(c=Absent), dont la somme fait 1. Le but du Control-based est de mettre à jour ces trois probabilités afin d'inférer la maîtrise du contrôle (de la connaissance) par l'apprenant.

Ci-après un exemple sur la géométrie :

Problème	Opérateur	Registres	Contrôles	Variables de situation
La hauteur est donnée sur la figure	Calcul de l'aire d'un triangle	-Figure	La hauteur est perpendiculaire à la base	Hauteur_Perpendiculaire= Correct
		-Figure -Cours	La hauteur est un côté d'un triangle rectangle si la base n'est pas l'hypoténuse	Selection_hauteur=correct
		-Formule	Aire=base*hauteur /2	Calcul_formule=correct

Traces en entrée

Le Control-based requiert dans les traces des apprenants :

- L'identifiant de l'apprenant
- Les valeurs des variables didactiques de l'ensemble P
- L'opérateur utilisé par l'apprenant
- Le registre de représentation courant
- La valeur des variables de situations (évaluation de l'usage de l'opérateur correct ou incorrect)

Traitement du diagnostic des connaissances

Le Control-based est implémenté comme un réseau bayésien dynamique, avec un nœud par variable des ensembles P, R, O et C. Le but du réseau est de calculer les probabilités pour chaque nœud « contrôle » c au temps t :

- $Pr(c_t=Valide \mid pa(c))$
- $Pr(c_t=Invalide \mid pa(c))$
- $Pr(c_t=Absent \mid pa(c))$

Où pa(c) sont les parents de c dans le réseau, c'est-à-dire : les problèmes, opérateurs et registres liés au contrôle, ainsi que $Pr(c_{t-1})$. En effet, le réseau étant dynamique, les probabilités au temps t dépendent de celles au temps t-1.

L'inférence dans le réseau bayésien dynamique se fait à partir de deux éléments (Naïm, 2007) :

- Une distribution *a priori* des probabilités (c'est-à-dire au temps 0, avant que l'apprenant n'utilise l'EIAH)
- La mise à jour des probabilités au temps t en fonction du temps t-1, grâce à la probabilité jointe du réseau. Soit G=(V,E) le réseau avec V l'ensemble des nœuds et E l'ensemble des arêtes, la probabilité de distribution jointe du réseau est :

$$P(V_t \mid V_{t-1}) = \prod_{x \in V} P(x_t \mid \mathrm{pa}(x_t))$$

D. Récapitulatif

La Table 1 résume les diagnostics présentés plus haut selon les caractéristiques suivantes :

- La forme du résultat, c'est-à-dire le modèle de l'apprenant
- Ce qui doit être tracé dans les traces par l'EIAH
- La nature du traitement
- La prise en compte de l'incertitude dans le traitement du diagnostic

	Knowledge tracing	AFM / PFM	Constraint-based	Control-based
Modèle d'apprenant	Cognitive model	Cognitive Model	Modèle de contrainte	Modèle cKc simplifié
Éléments tracés	Étapes de résolution des problèmes	Étapes de résolution des problèmes	États du problème	Opérateurs utilisés
Traitement	Inférence bayésienne	Régression logistique	Système à base de règles	Inférence bayésienne
Incertitude	Oui	Oui	Non	Oui

Table 1 : Récapitulatif des différents diagnostics des connaissances présentés en fonction de leur modèle d'apprenant, des informations requises dans les traces, du traitement effectué pour inférer le modèle de l'apprenant, et de la prise en compte ou non de l'incertitude lors du traitement.

II. Construction d'un diagnostic

1. Définition

La **construction** d'un diagnostic des connaissances est la modélisation et l'implémentation des trois composants du diagnostic (traces, traitement, résultat) pour un domaine d'apprentissage et/ou un EIAH donné. Dans le cas de diagnostic générique, on parle **d'instanciation** ou **d'application** du diagnostic générique au domaine considéré. En détail, la construction d'un diagnostic permet d'obtenir un composant informatique qui est capable de :

- prendre en entrée les traces d'apprenants, dont le format est imposé par le diagnostic, et qui sont collectées par l'EIAH ou bien via un prototype, un simulateur, des expérimentations…
- réaliser le traitement du diagnostic en fonction de l'implémentation du diagnostic retourner comme résultat un modèle d'apprenant, là encore tel que défini pour ce diagnostic

En revanche, la construction du diagnostic exclut le travail d'identification des connaissances du domaine, de conception des exercices à résoudre ou le suivi de l'activité de l'apprenant et de l'état des exercices dans l'EIAH. Ces travaux relèvent en effet de la didactique ou de l'ingénierie de l'EIAH, et sont un prérequis à la construction d'un diagnostic. Le diagnostic

traite lui de l'inférence ou de la déduction de l'état des connaissances de l'apprenant à partir de ses traces collectées par l'EIAH.

La façon la plus simple de construire un diagnostic est de réaliser le travail de modélisation et d'implémentation manuellement à partir de zéro. Cependant, la diversité des diagnostics et la spécificité de leurs implémentations (souvent issues de l'intelligence artificielle) rendent cette tâche complexe pour les non-spécialistes, et répétitive si l'on travaille sur plusieurs domaines. De plus, la plupart des diagnostics génériques requièrent de fixer des paramètres (cf. les exemples du Knowledge tracing, d'AFM ou du Control-based plus haut) qui jouent un rôle important sur la qualité du diagnostic (Baker et al., 2008; Beck, 2007). Il existe donc des outils ou méthodes facilitant la construction de diagnostics sur un domaine, que nous regroupons en trois grandes catégories : les outils auteurs, les algorithmes d'apprentissage automatique et les méthodes d'optimisation.

2. Outils auteurs

A. Principe

Les outils auteurs sont des environnements informatiques qui permettent de construire tout ou partie d'un EIAH avec un coût réduit ou nul en programmation. Ils permettent notamment de concevoir graphiquement l'interface de l'EIAH, les exercices, les aides, et parfois un diagnostic des connaissances (Murray, 1999; Murray et al., 2003). Du fait de nos questions de recherche, nous nous intéressons uniquement aux outils auteurs permettant la construction d'un diagnostic des connaissances.

B. Travaux

Eon (Murray, 2003) permet la construction d'EIAH complets. Le concepteur peut modéliser les concepts du domaine d'apprentissage et les relations entre concepts via une ontologie. En utilisant le vocabulaire de l'ontologie, il peut écrire manuellement des règles de type règles expertes pour le diagnostic des connaissances, toujours au moyen d'une interface. Les EIAH créés par Eon se chargent ensuite de tracer les réponses de l'apprenant et d'inférer le modèle de l'apprenant à partir des règles.

CTAT (Aleven et al., 2006) permet de construire des diagnostics à base de règles de production, dérivés d'ACT-R. Le concepteur doit d'abord modéliser l'interface de l'EIAH et un graphe orienté des chemins (ou plans) de résolutions des exercices pas à pas.Il peut ensuite associer à chaque étape de résolution une règle de production permettant de diagnostiquer une connaissance. Le code des règles est écrit manuellement par le concepteur.

SimStudent (Matsuda et al., 2007, 2005) est une extension de CTAT qui permet, au lieu d'écrire le code des règles de production, de les apprendre par l'exemple. Le concepteur résout plusieurs problèmes du domaine en indiquant à chaque étape la connaissance qu'il a mobilisée. SimStudent, après quelques exemples de résolution, est capable de construire automatiquement des règles de production associées à chaque connaissance.

Le Cognitive Model SDK (Blessing et Gilbert, 2008) permet de construire les mêmes diagnostics que CTAT, mais pas un EIAH complet. À la différence de CTAT, l'interface, les exercices et le traçage de l'activité de l'apprenant doivent donc être développés à part. Le concepteur doit modéliser une taxonomie des objets ou concepts du domaine, puis utiliser cette taxonomie pour écrire manuellement des règles de prédiction. En sortie, le concepteur obtient le code source du diagnostic des connaissances, qu'il peut intégrer, moyennant un travail de programmation, dans un EIAH en cours de construction, ou tout type de logiciel en général.

ASPIRE (Mitrovic et al., 2008, 2006) est similaire à CTAT, excepté qu'il permet de construire des diagnostics de type Constraint-based. Après la conception de l'interface et des exercices, les concepts du domaine et les états des exercices sont décrits par une ontologie. Le concepteur peut ensuite écrire le code des contraintes, une contrainte représentant une connaissance. ASPIRE est de plus capable de générer automatiquement des contraintes à partir de l'ontologie.

C. Caractéristiques et limites

Les outils auteurs ont l'avantage de réduire le coût en programmation. Toutefois, du point de vue du diagnostic des connaissances, la plupart des outils auteurs requièrent toujours un travail manuel (écriture manuelle de règles dans CTAT, ASPIRE, EON, Cognitive Model SDK) qui ne peut être réalisé que par un expert du diagnostic. Seuls SimStudent et, de façon limitée, ASPIRE permettent d'assister le travail de conception.

Du point de vue de l'architecture logicielle, un outil auteur peut être faiblement ou fortement couplé. Cela dépend du degré d'indépendance entre la conception du diagnostic et les autres composants d'un EIAH (interface, modèle du domaine…) : si le diagnostic dépend des autres composants il sera fortement couplé. Les outils auteurs faiblement couplés permettent de construire seulement un diagnostic des connaissances, tandis que les outils fortement couplés permettent de construire d'autres composants de l'EIAH, voire un EIAH complet.

Les outils fortement couplés présentent des limites fortes pour notre problématique. Un important travail de conception est requis au préalable pour modéliser l'interface et les concepts du domaine, sans qu'il soit possible d'utiliser d'autres interfaces existantes ou d'intégrer le diagnostic dans un EIAH existant, mais ne possédant pas de composant pour le diagnostic. C'est une limite importante notamment lorsque le concepteur souhaite utiliser une interface de haut niveau, utilisant par exemple la 3D, des dispositifs haptiques, un environnement de type jeu sérieux, etc. De plus, il n'est pas possible d'enrichir un EIAH existant (il faut tout reconstruire) ou de changer d'outil auteur en cours de conception, ces derniers n'étant pas compatibles entre eux. Cette non-compatibilité impose également de ne construire qu'un diagnostic (celui supporté par l'outil auteur), sans possibilité d'en développer plusieurs, dérivés de théories différentes. Au niveau développement logiciel, tous les composants sont interconnectés et interdépendants, rendant la maintenance difficile. Ce problème peut être évité par une approche modulaire (ou une approche faiblement couplée sur le diagnostic, comme défini plus haut), c'est-à-dire la conception

d'un diagnostic indépendant, qui peut être intégré dans un EIAH ou un logiciel existant, comme le Cognitive Model SDK. En échange, le travail d'intégration doit être réalisé par un développeur, et le traçage de l'activité de l'apprenant doit collecter les éléments nécessaires au diagnostic.

Concernant les diagnostics pouvant être construits, nous pouvons remarquer que chaque outil auteur impose un diagnostic : Model tracing pour CTAT, SimStudent et Cognitive Model SDK, Constraint-based pour ASPIRE, et système expert pour Eon. Ces outils restent ainsi donc liés à un diagnostic. Cela limite la possibilité de construire plusieurs diagnostics différents en fonction de l'EIAH et du domaine, car il faut rapprendre à utiliser un nouvel outil auteur sans pouvoir réinvestir l'expérience passée. De plus, en cours de conception, il est coûteux de changer de diagnostic, car cela implique de recommencer le travail sur un nouvel outil auteur depuis le début, dans le cas des outils auteurs fortement couplés sur le diagnostic. La démarche de conception logicielle itérative est donc limitée, car il faut choisir le diagnostic dès le début de la conception de l'EIAH. Enfin, dans l'optique de notre question de recherche, il n'est pas possible de construire des diagnostics différents pour les comparer sur un domaine.

Il manque donc d'outils permettant de répondre à nos questions de recherche, c'est-à-dire :

- soit des outils auteurs permettant la construction de plusieurs diagnostics des connaissances,
- soit une collection d'outils auteurs faiblement couplés sur le diagnostic, permettant chacun de construire un diagnostic différent, et ayant un fonctionnement suffisamment similaire pour pouvoir réinvestir l'expérience d'utilisation.

3. Apprentissage automatique

A. Principe

Le principe de l'apprentissage automatique est de construire un diagnostic des connaissances pour un domaine via un algorithme d'apprentissage automatique. Ces algorithmes ont besoin en entrée de traces d'apprenants précédemment collectées : c'est à partir de ces traces qu'ils vont construire le diagnostic, en faisant l'hypothèse que les traces contiennent suffisamment d'information sur les connaissances du domaine et le modèle d'apprenant à inférer (Arroyo et al., 2004; Jonsson et al., 2005; Py, 1998). C'est pourquoi on parle d'extraction d'un diagnostic à partir d'une base de traces d'apprenant, appelée base d'apprentissage. En sortie, il retourne le code source du diagnostic extrait des traces. Modulo un travail de programmation, ce code source peut être intégré à un EIAH existant.

En intelligence artificielle, les algorithmes d'apprentissage automatique se découpent en deux catégories : supervisés et non supervisés (Amershi et Conati, 2007). Les algorithmes supervisés supposent que les traces sont étiquetées par des labels qui donnent une information experte sur la trace. Pour le diagnostic des connaissances, étiqueter une trace d'un apprenant, par exemple une réponse de l'apprenant à un problème, signifierait par exemple de demander à un expert s'il pense que l'apprenant maîtrise les connaissances

requises pour résoudre le problème en fonction de sa réponse. Ce type de label est très coûteux à obtenir, c'est pourquoi les algorithmes d'apprentissage de diagnostic des connaissances sont le plus souvent non supervisés (Sison et Shimura, 1998).

Plus formellement, le problème est le suivant : étant donné une base de traces d'apprenants et un diagnostic générique à instancier au domaine, quelle est la meilleure instanciation qu'il est possible d'extraire des traces ? Pour résoudre ce problème, il faut spécifier le type de traces d'apprenants que l'algorithme d'apprentissage peut traiter, le diagnostic générique à instancier, et une fonction d'évaluation d'une instanciation. Précisons qu'il ne faut pas confondre la base de traces d'apprentissage, qui sert à construire le diagnostic, et les futures traces des apprenants qui utiliseront l'EIAH et seront l'entrée du diagnostic construit.

B. Travaux

L'apprentissage automatique a fait l'objet de nombreux travaux à partir de la fin des années 1990. Comme le nombre d'instanciations d'un diagnostic à partir d'une base d'apprentissage est souvent au moins exponentiel en fonction de la taille de la base, ces travaux ont recours à des heuristiques, c'est-à-dire des algorithmes qui peuvent fournir une solution (ici un diagnostic instancié au domaine) en un temps polynomial mais sans preuve de sa qualité.

Une première classe d'algorithmes d'apprentissage sont les algorithmes dits de couverture (Russell et al., 2003) : l'objectif est d'extraire un ensemble de règles qui permettent de classifier les réponses de l'apprenant selon qu'elles ont été correctes ou incorrectes dans la base d'apprentissage. ACM (Langley et Ohlsson, 1984) vise à construire automatiquement un diagnostic, qui se présente comme une base de règles de production. L'apprentissage de ces règles se fait par identification des étapes de résolution des problèmes. Les actions ou opérations impliquées dans les étapes correctes sont labellisées comme positives et les autres comme négatives, puis ACM infère les chemins de résolution possible des problèmes. Tous les chemins positifs (constitués uniquement d'étapes positives) sont transformés en règles de production (SI/SINON). Ces règles de production fonctionnent de façon similaire à celles d'ACT-R décrit plus haut.

Une deuxième classe d'algorithmes sont les méthodes statistiques, probabilistes ou logistiques. Mayo et Mitrovic (Mayo et Mitrovic, 2001) utilisent une heuristique pour apprendre un diagnostic représenté sous la forme d'un réseau bayésien. Étant donnée une base de traces où une trace est une collection de variables sur l'état du problème, l'action effectuée par l'apprenant et le résultat (correct ou incorrect) de cette action, l'heuristique calcule la dépendance entre chacune des variables via une mesure statistique (un score). L'heuristique vise à maximiser ce score. Il en résulte un graphe de dépendances entre les variables qui forme un réseau bayésien. La seconde étape est d'apprendre les tables de probabilité du réseau en fonction des traces ; ces tables de probabilités vont permettre au réseau de réaliser le traitement du diagnostic. Mayo et Mitrovic utilisent pour cela une seconde heuristique, qui est un comptage dans la base des traces : par exemple, la probabilité Pr(x=vrai | y=faux) entre deux variables x et y du réseau bayésien est égale au nombre de fois où x=vrai et y=faux sur l'ensemble de la base d'apprentissage. Il en résulte un réseau bayésien qui pourra inférer le modèle de l'apprenant à partir de nouvelles traces.

Beck et Woolf (Beck et Woolf, 1998; Beck et al., 2000) utilisent plusieurs méthodes d'apprentissage pour construire un diagnostic générique. Le diagnostic appris est basé sur un modèle linéaire qui se fonde sur la relation entre connaissances, évaluation des réponses de l'apprenant et temps de résolution des problèmes, tandis que le modèle de l'apprenant inféré par le diagnostic se présente sous la forme d'une collection de connaissances associées à des problèmes (un problème requiert une ou plusieurs connaissances). Pour construire par apprentissage le diagnostic, Beck et Woolf (Beck et Woolf, 1998) expérimentent d'abord un réseau de neurones ayant pour prérequis la taxonomie des connaissances du domaine. Puis dans la plateforme ADVISOR (Beck et al., 2000), ils utilisent comme méthodes les moindres carrés, CART (Classification et Regression Trees) et SMART (Multivariate Edaptive Regression Splines, sans taxonomie *a priori* des connaissances.

Gonzalès-Brenes et Mostow (Gonzales-Brenes et Mostow, 2012) proposent eux un algorithme permettant de construire un diagnostic bien connu dans la littérature, le Knowledge tracing (cf. plus haut). Ils utilisent pour cela une variation où plusieurs connaissances peuvent être associées à une étape de résolution d'un problème, et où le traitement du diagnostic est réalisé par un réseau bayésien. Pour construire ce réseau, ils utilisent d'abord un algorithme d'apprentissage exact exponentiel, qui restreint le nombre de connaissances possibles à prendre en compte à environ 4. Pour pouvoir prendre en compte plus de connaissances, Gonzalès-Brenes et Mostow utilisent ensuite une heuristique basée sur une méthode de type Monte-Carlo par chaînes de Markov (*Markov chain Monte Carlo*) (Gonzalez-Brenes et Mostow, 2013). Ces méthodes imposent deux prérequis : le nombre de connaissances à diagnostiquer doit être fourni *a priori*, et les traces doivent contenir les étapes de résolution de problèmes suivies par l'apprenant avec leur résultat à chaque étape (correct ou incorrect). L'algorithme apprend ensuite le modèle de l'apprenant à inférer (les associations entre connaissances, étapes et problèmes, cf. la définition du Knowledge tracing plus haut) et les paramètres requis pour le traitement du diagnostic par le Knowledge tracing (notamment la probabilité d'apprendre une connaissance).

Desmarais (Desmarais, 2011; Desmarais et al., 2006) a étudié la construction automatique d'un diagnostic générique nommé « l'Item to Item Structure », où le modèle de l'apprenant associe les items deux à deux selon si le premier est un prérequis du second. Un item (ou knowledge item) est un élément observable dans les traces, comme les questions d'un exercice, qui représente une partie de l'état des connaissances (*knowledge state*) d'un apprenant. La construction du diagnostic vise donc à associer les items du domaine qui sont liés par une relation de prérequis. Puis, dans un second temps, ces items peuvent être explicitement associés aux connaissances ou concepts implicites, qui ne sont pas directement observables dans les traces. Pour apprendre les relations entre items (l'Item to item structure), Desmarais utilise deux familles d'algorithmes d'apprentissage. Pour la première famille, il redéfinit le modèle de l'apprenant comme un réseau bayésien (donc comme un graphe), et utilise des heuristiques d'apprentissage basées sur des scores. Le score est une mesure permettant de comparer diverses associations entre connaissances et items afin de garder la meilleure (i.e. le meilleur score). Le but est de maximiser (ou minimiser) le score, ici des mesures statistiques. Pour la seconde famille, Desmarais définit le

modèle comme une structure d'ordre partiel, et utilise cette fois un test statistique vérifiant si une association entre connaissances et items vérifie des contraintes prédéfinies. Notons que Desmarais ne prend pas en compte la dimension temporelle. Pour apprendre les relations entre items et connaissances implicites, Desmarais se base sur les travaux de Vomlel (Vomlel, 2004), qui utilise un réseau bayésien. Pour apprendre ce réseau bayésien, Vomlel utilise l'algorithme d'apprentissage PC (Neapolitan, 2004) qui cherche les dépendances les plus fortes entre les variables dans les traces. Son approche est toutefois semi-automatique car un expert doit intervenir après l'algorithme pour ajouter des informations expertes et des contraintes sur les arrêtes du réseau bayésien.

C. Caractéristiques et limites

Tout comme les outils auteurs, les algorithmes d'apprentissage automatique permettent de réduire le coût en programmation et le temps de construction d'un diagnostic des connaissances, car ils retournent en résultat le code source d'un diagnostic instancié pour le domaine et les traces du concepteur, et intégrable dans un EIAH existant ou en cours de construction.

Les algorithmes d'apprentissage posent la question de la modélisation et de la prise en compte des connaissances. En effet, les connaissances ne sont par définition pas directement observables dans les traces d'apprenant. Il faut donc qu'elles soient fournies au préalable, ou bien qu'elles soient inférées (déduites) par l'algorithme d'apprentissage. Dans le premier cas, il faut donc un travail humain *a priori*, ce qui en font en réalité des algorithmes dits semi-automatiques, comme SimStudent (présenté parmi les outils auteurs), ACM, Beck et Woolf. Dans le second cas, l'approche est bien complètement automatique, comme ADVISOR, Gonzales-Brenes et Mostow et Desmarais. Le problème qui se pose dans ces dernières approches est la validité et le sens des connaissances inférées : les algorithmes optimisent une mesure mathématique sans fournir d'information sur la sémantique. Les connaissances inférées sont donc difficiles à interpréter par un humain et n'ont pas forcément de sens du point de vue du domaine. Beck (Beck, 2007) a même montré que les algorithmes d'apprentissage donnent parfois des résultats illogiques du point de vue du diagnostic des connaissances ou de la didactique, par exemple une connaissance qui est toujours apprise par tous les apprenants dès le premier exercice. Découlant de ce problème d'interprétabilité vient le problème de l'utilité : si les connaissances, et donc le modèle d'apprenant inféré par le diagnostic, ne font pas sens pour le concepteur humain, il ne pourra pas l'utiliser dans l'EIAH pour d'autres tâches comme fournir des rétroactions à l'apprenant.

Du point de vue de nos questions de recherches, la non-interprétabilité du résultat pose problème, car réduisant fortement les cas d'utilisation du diagnostic construit. Un diagnostic non interprétable n'est par exemple pas utilisable pour fournir des rétroactions épistémiques à l'apprenant (c'est-à-dire des rétroactions prenant en compte l'état des connaissances de l'apprenant), ni pour adapter l'EIAH en fonction des connaissances de l'apprenant, ni pour choisir le prochain exercice ou le parcours pédagogique…

Pour pallier ce problème, Mayo et Mitrovic ont préféré ne pas prendre en compte les connaissances dans leur algorithme d'apprentissage, modélisant le modèle de l'apprenant seulement comme des éléments observables sur son activité. Cela n'est toutefois pas applicable à des diagnostics qui requièrent la modélisation des connaissances (comme le Knowledge tracing, Constraint-based, Control-based). Une autre solution est donc la prise en compte d'information experte *a priori*, donc d'informations sémantiques, en utilisant des algorithmes semi-automatiques. C'est le principe de SimStudent, bien que l'interprétabilité ne soit pas garantie.

Dans les travaux ci-dessus, les diagnostics construits sont génériques, mais le plus souvent ad hoc. Seuls Gonzales-Brenes et Mostow construisent un diagnostic bien connu, le Knowledge tracing, c'est-à-dire que le modèle de l'apprenant inféré par le diagnostic et les paramètres du diagnostic satisfont la formalisation du diagnostic telle que publiée dans la littérature. Pour nos questions de recherche, il est intéressant de construire des diagnostics génériques bien établis dans la littérature, car leur modélisation est plus répandue et divers outils externes les supportent. Par exemple, pour le Knowledge tracing, il existe la plateforme Datashop (Koedinger et al., 2010) qui collecte et stocke un grand nombre de traces d'apprenants compatibles, et divers outils ont été construits à partir du Knowledge tracing comme l'algorithme de Baker (Baker et al., 2004) visant à identifier les apprenants qui jouent avec l'EIAH (*gaming the system*) au lieu de travailler sérieusement.

Comme pour les outils auteurs, chaque algorithme ne permet d'instancier au domaine qu'un seul diagnostic générique. Leurs résultats sont difficilement comparables car biaisés par leur spécificité, les éléments ou traces pris en compte dans l'apprentissage et les diagnostics construits étant très différents. Du point de vue de nos questions de recherche, la comparaison des diagnostics construits doit être le moins biaisée possible par l'algorithme d'apprentissage afin que la comparaison porte sur les performances du diagnostic et non les performances de l'algorithme d'apprentissage.

Les algorithmes d'apprentissage requièrent également une base de traces d'apprenant. Ces traces peuvent être collectées au préalable via des expérimentations, un EIAH existant, un EIAH en cours de construction capable de tracer l'apprenant... Chaque algorithme d'apprentissage impose un format de traces plus ou moins libre ; par exemple, ADVISOR prend en compte tout type de traces tandis que Gonzales-Brenes et Mostow imposent d'avoir dans les traces les étapes de résolution des problèmes ainsi que l'évaluation de l'apprenant à chaque étape (correct ou incorrect). Moins un algorithme impose de contraintes, plus il est réutilisable et applicable à différents domaines. Outre les traces, certains algorithmes imposent de spécifier d'autres éléments, par exemple le nombre de connaissances pour Gonzales-Brenes et Mostow.

Il se pose enfin la question de la qualité des diagnostics construits automatiquement. La plupart des travaux font des évaluations statistiques. Seul Beck (Beck, 2007) propose une mesure de l'interprétabilité des paramètres d'un diagnostic appris, mais qui s'avère peu concluante selon lui.

4. Optimisation

A. Principe

La troisième et dernière méthode d'assistance à la construction de diagnostics sont les méthodes d'optimisation, c'est-à-dire les méthodes qui visent à améliorer ou raffiner un diagnostic existant. À la différence des outils auteurs et de l'apprentissage automatique, il faut donc un diagnostic *a priori*, qui va être amélioré, soit manuellement, soit automatiquement.

B. Travaux

Datashop (Koedinger et al., 2010) est une plateforme de collecte de traces d'apprenants qui est capable de diagnostiquer les connaissances des apprenants à partir de ces traces en utilisant le diagnostic AFM. Le modèle de l'apprenant à inférer doit être fourni par le concepteur. Datashop permet d'assister l'exploration des traces et la modification du modèle de l'apprenant. Il permet également d'évaluer ce modèle via des critères statistiques. Koedinger et Matsuda (Koedinger et Stamper, 2010; Koedinger et al., 2013, 2012; Stamper et Koedinger, 2011) ont montré dans plusieurs publications comment un expert en diagnostic peut améliorer un diagnostic donné en étant assisté par Datashop. Leur méthode consiste à modifier le diagnostic, puis à l'évaluer statistiquement dans Datashop en utilisant les traces d'apprenant, et à itérer sur ces deux phases. Datashop permet de parcourir les résultats du diagnostic pour découvrir les associations entre les étapes de résolution des problèmes (*step*) et les connaissances (*KC*) qui pénalisent l'évaluation statistique. En modifiant ces associations, le diagnostic doit obtenir une meilleure évaluation statistique basée sur les traces, jusqu'à ce qu'il ne soit plus possible pour un expert de découvrir dans Datashop des associations.

Il existe également des méthodes d'optimisation automatique, à partir de traces d'apprenants. L'une des plus connues est LFA (*Learning Factor Analysis*) (Cen et al., 2006) qui permet d'améliorer un diagnostic *Knowledge tracing*. LFA fonctionne à partir d'un ensemble d'opérateurs définis *a priori* qui vont modifier le diagnostic, ainsi que d'une mesure d'amélioration. Des opérateurs simples consistent par exemple à regrouper ou dissocier des connaissances. La mesure d'amélioration permet de comparer deux diagnostics : dans LFA, il s'agit d'un test statistique. L'algorithme fonctionne selon le principe suivant :

Soit un diagnostic initial D, une base de traces d'apprenants, une mesure d'ordre > entre deux diagnostics et un ensemble d'opérateurs O
Appliquer en boucle les opérateurs o de O sur D ou obtenir D'
Si D'>D, alors D=D' et rappeler la procédure récursivement
Sinon, retourner D

Intuitivement, LFA est une heuristique qui va explorer l'arbre de tous les diagnostics qui améliorent le diagnostic initial. L'interprétabilité du diagnostic résultant de LFA dépend donc fortement des opérateurs appliqués.

DEBUGGY (Burton, 1981) utilise une heuristique d'optimisation pour améliorer un diagnostic de type librairie d'erreurs, où le modèle de l'apprenant est une collection de « bugs » (terme regroupant les erreurs, connaissances non sues, misconceptions) large révélant une connaissance non apprise ou non comprise. À partir d'une base prédéfinie de bugs (fournie par exemple par un expert), DEBUGGY crée un sous-ensemble de bugs qui expliquent les réponses des apprenants dans la base d'apprentissage. Puis, il sélectionne parmi ces erreurs celles expliquant le plus de réponses d'apprenants. Le diagnostic construit est donc un sous-ensemble du diagnostic de l'expert, qui en conserve la sémantique et l'interprétabilité. ASSERT (Baffes et Mooney, 1996) a un fonctionnement similaire à DEBUGGY, excepté que le diagnostic d'origine doit être le modèle de l'expert (nommé le modèle « idéal ») représenté sous forme de règles. Ces règles sont ensuite modifiées ou supprimées en fonction des traces d'apprenants, de façon à inférer les règles qui permettent de diagnostiquer les connaissances des apprenants. ASSERT se base sur le principe que le modèle de l'apprenant est un sous-ensemble du modèle de l'expert.

C. Caractéristiques et limites

Ces méthodes d'optimisation se rapprochent des algorithmes d'apprentissage semi-supervisés, car elles nécessitent un travail humain pour fournir le diagnostic de départ. Tout comme les outils auteurs et les algorithmes d'apprentissage, les méthodes de la littérature ne permettent d'améliorer qu'un seul diagnostic générique.

5. Synthèse des principaux verrous

L'assistance à la construction de diagnostics pose les problèmes du coût de construction, du réinvestissement de l'expérience de la part du concepteur pour de futurs domaines, de l'interprétabilité et de l'utilité du diagnostic construit. Parmi les outils auteurs, plusieurs sont fortement couplés sur le diagnostic des connaissances et imposent généralement de construire un EIAH complet. De plus, le diagnostic est essentiellement construit manuellement par le concepteur, notamment via des règles expertes ou des règles de production, avec peu d'assistance automatique ou semi-automatique. La Figure 2 résume les travaux de la littérature selon ces deux aspects. Nous remarquons qu'il n'existe pas d'outils auteurs basés sur une assistance automatique, et qu'un seul outil semi-automatique est faiblement couplé sur le diagnostic. Il manque plus généralement d'outils auteur pour assister la construction de diagnostics, indépendamment de la construction d'un EIAH complet qui est une tâche extrêmement vaste.

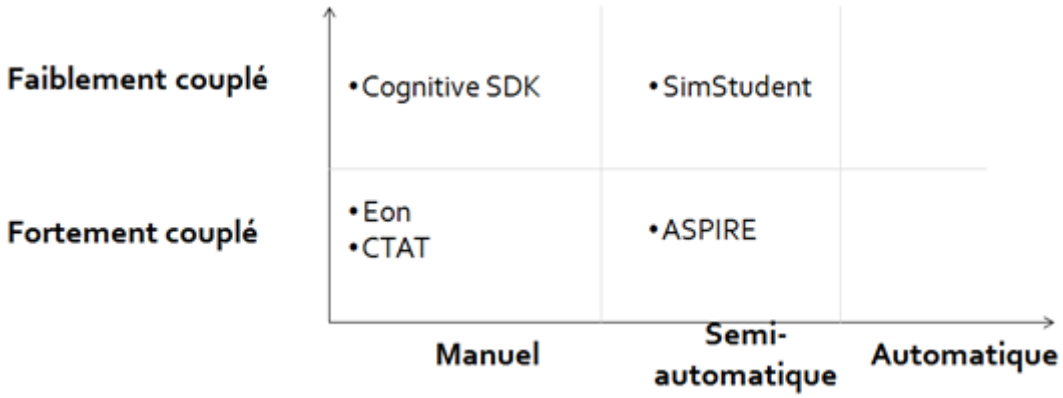


Figure 2 : Positionnement des outils auteurs présentés en fonction de leur couplage sur le diagnostic des connaissances et de leur approche pour la construction du diagnostic, de manuel à automatique.

Concernant les algorithmes d'apprentissage automatique, divers travaux proposent des méthodes dont la réutilisabilité dépend des contraintes imposées par le diagnostic : contraintes sur les traces pouvant être prises en compte et sur les informations expertes qui doivent être spécifiées *a priori*. Le principal verrou est la question de l'interprétabilité, dont découle la question de l'utilité et plus généralement de la compréhension du diagnostic construit. Il n'existe pas de diagnostic automatique incluant des mécanismes ou outils pour renforcer l'interprétabilité du résultat. Un second verrou est la construction de diagnostics connus et publiés : en effet, les diagnostics génériques imposent le plus souvent des contraintes (par exemple, dans le Knowledge tracing, une seule connaissance est associée à chaque étape de résolution des problèmes). Plutôt que de prendre en compte ces contraintes, la plupart des travaux que nous avons étudiés proposent la construction d'un diagnostic ad hoc, et non la construction d'un diagnostic connu dans la littérature qui impose d'en respecter les contraintes. Enfin, les algorithmes se caractérisent par les contraintes qu'ils imposent sur les traces, sur les informations requises au préalable et sur les domaines d'apprentissage pour lesquels ils fonctionnent. Plus les contraintes sont fortes, moins l'algorithme est réutilisable pour différents travaux. La Figure 3 résume les travaux étudiés en fonction de leur réutilisabilité et de leur interprétabilité. SimStudent figure entre crochets car il s'agit en réalité d'une méthode semi-automatique. Nous le faisons toutefois figurer dans le graphe pour illustrer le fait qu'une approche semi-automatique permet de renforcer l'interprétabilité par rapport aux algorithmes automatiques.

Les méthodes semi-automatiques d'optimisation permettent de pallier plusieurs des problèmes posés par les outils auteurs (peu d'outils auteurs faiblement couplés sur le diagnostic) et l'apprentissage automatique (la difficulté pour interpréter les résultats et les contraintes imposées sur les traces). En revanche, ils nécessitent un diagnostic, parfois expert, coûteux à construire. Il n'existe pas de travaux étudiant l'impact du diagnostic initial sur l'optimisation, crucial en algorithmique traditionnelle.

Figure 3 : Positionnement des différents algorithmes d'apprentissage automatique de diagnostics des connaissances en fonction de l'interprétabilité et du diagnostic construit et de la réutilisabilité de l'algorithme.

Enfin, le dernier verrou commun à toutes les méthodes d'apprentissage est la prise en compte de plusieurs diagnostics : il n'existe pas d'outils ou de méthodes permettant d'assister la construction de diagnostics génériques différents, par exemple Knowledge tracing, Constraint-based et Control-based. Tous les travaux présentés imposent un et un seul diagnostic, parfois ad hoc.

III. Comparaison de diagnostics des connaissances

La comparaison de diagnostics génériques des connaissances peut se définir de deux façons différentes :

- La comparaison de diagnostics génériques non instanciés, c'est-à-dire la comparaison de la spécification, des avantages et des limites du diagnostic générique tel qu'il est défini. Les diagnostics n'étant pas instanciés, il est impossible dans ce cas de comparer empiriquement leurs résultats à partir de traces d'apprenants dans un domaine.
- La comparaison de diagnostics génériques instanciés à un même domaine (donc d'une instance). La comparaison se rapproche alors de benchmark en algorithmique, s'il est possible d'évaluer les performances des différents diagnostics empiriquement, à partir d'un jeu de test (ici des traces d'apprenants, qui sont nécessaires pour obtenir les résultats des diagnostics).

Les travaux comparant des diagnostics différents apparaissent dans les années 2000. Mitrovic, Koedinger et Martin (Mitrovic et al., 2003) soulignent le besoin d'une telle comparaison, dans le but de renforcer la validité scientifique de l'évaluation des diagnostics, mais décrivent aussi la difficulté d'une telle comparaison. Cette difficulté naît des divergences profondes entre les diagnostics, qui rendent difficile une comparaison sans biais. Nous présentons ci-dessous une revue de la littérature sur le sujet pour les deux types de comparaison définis ci-dessus : comparaison de différents diagnostics ou d'instances de différents diagnostics.

1. Comparaison de diagnostics génériques

A. Principe

La comparaison de diagnostics génériques, donc non instanciés à un domaine, peut se définir comme une analyse comparative des caractéristiques, des modèles théoriques sous-jacents et du modèle de l'apprenant inféré. Plus généralement, il s'agit de comparer les approches des diagnostics.

B. Travaux

Mitrovic, Koedinger et Martin (Mitrovic et al., 2003) sont les auteurs d'un des premiers travaux comparant deux diagnostics : Model tracing et Constraint-based. Ces deux diagnostics dérivent de deux modèles théoriques sous-jacents : ACT-R (Anderson, 1983) et Performance Error (Ohlsson, 1996). Dans leur étude, ils comparent les deux approches selon les caractéristiques suivantes :

- La représentation des connaissances, donc comment sont représentées les connaissances dans le modèle de l'apprenant.
- La fidélité cognitive du modèle de l'apprenant inféré, c'est-à-dire le degré de simplification qu'impose le modèle de l'apprenant par rapport à la réalité (ici, la réalité désigne l'objet modélisé, donc les connaissances des apprenants).
- Ce qui est évalué lors de la résolution des problèmes par les apprenants.
- La possibilité de prendre en compte plusieurs stratégies de résolution de problèmes.
- La façon dont sont évaluées les solutions des apprenants aux problèmes.
- Le type de rétroaction qu'il est possible de donner à l'apprenant à partir du résultat du diagnostic.
- La possibilité de donner des indices aux apprenants sur la résolution des problèmes.
- La façon dont on évalue si un problème est terminé.
- L'action par défaut du diagnostic si la stratégie de résolution du problème par l'apprenant est inconnue.
- La possibilité de représenter des « bugs » (erreurs).
- Le coût de construction.

On note que certaines caractéristiques sont liées à la didactique (comme la fidélité cognitive), à l'informatique (le coût de construction ou l'évaluation du résultat), à l'utilisation du résultat du diagnostic (rétroaction, indices), ou aux choix de modélisation de l'apprenant (représentation des connaissances, stratégies des apprenants). Le résultat de cette analyse comparative est donné dans le tableau de la Figure 4.

Property	Model Tracing	Constraint-Based Modeling
Knowledge representation	Production rules (procedural)	Constraints (declarative)
Cognitive fidelity	Tends to be higher	Tends to be lower
What is evaluated	Action	Problem state
Problem solving strategy	Implemented ones	Flexible to any strategy
Solutions	Tend to be computed, but can be stored	One correct solution stored, but can be computed
Feedback	Tends to be immediate, but can be delayed	Tends to be delayed, but can be immediate
Problem-solving hints	Yes	Only on missing elements, but not strategy
Problem solved	'Done' productions	No violated constraints
Diagnosis if no match	Solution is incorrect	Solution is correct
Bugs represented	Yes	No
Implementation effort	Tends to be harder, but can be made easier with loss of other advantages	Tends to be easier, but can be made harder to gain other advantages

Figure 4 : Tableau des résultats de l'analyse comparative des diagnostics génériques des connaissances Model tracing et Constraint-based effectuée par Mitrovic et al.

Les auteurs ont également réalisé une étude de cas en instanciant chacun des diagnostics pour le même domaine, la modélisation de bases de données en informatique. Ils indiquent informellement les différences du point de vue du concepteur expert en diagnostic. Ils indiquent également que le diagnostic pour le Model tracing se compose de 25 règles de productions correspondant à 23 contraintes pour le Constraint-based.

Kodaganallur, Weitz et Rosenthal (Kodaganallur et al., 2005) ont fait une analyse comparative similaire pour les mêmes diagnostics, Model tracing et Constraint-based. Ils comparent les caractéristiques suivantes :

- Le modèle de l'apprenant, notamment la modélisation des connaissances et des misconceptions.
- Les rétroactions qu'il est possible de donner à partir du modèle de l'apprenant.
- Le coût de construction (modélisation et implémentation).
- Les domaines d'apprentissage pour lesquels les diagnostics sont pertinents ou non, en fonction de la complexité et du nombre de stratégies de résolution des problèmes.

C. Caractéristiques et limites

Ces analyses comparatives des caractéristiques des diagnostics ont la particularité d'être discursives plutôt que numériques. Elles posent la question de l'interprétation faite par les auteurs des caractéristiques du modèle et de leurs biais éventuels. Par exemple, les groupes d'auteurs des deux travaux présentés ci-dessus (Mitrovic et al. et Kodaganallur et al.) ont confronté leurs points de vue contradictoires dans plusieurs articles (Kodaganallur et al., 2006; Mitrovic et Ohlsson, 2006). De plus, il n'émerge pas de ces travaux une méthodologie de comparaison aisément reproductible.

Ces analyses ne portent également que sur les caractéristiques génériques des diagnostics, pas sur la comparaison d'instances et sans utilisation de traces. Il ne s'agit donc pas à proprement parler de benchmark en informatique. Cependant, les diagnostics étant fortement dépendants du domaine d'apprentissage et des traces collectées, ces analyses ne sont pas assez fines pour comparer les diagnostics domaine par domaine. Elles ne permettent pas non plus de comparer le comportement des diagnostics pour des cas particuliers, par exemple sur deux groupes d'apprenants aux caractéristiques différentes.

2. Comparaison d'instances de diagnostics génériques

A. Principe

Divers travaux s'intéressent à comparer des instances de diagnostics, c'est-à-dire plusieurs diagnostics instanciés pour un même domaine. Plus précisément, il s'agit de comparer les résultats des instances de diagnostics (les modèles d'apprenants inférés) pour un même domaine avec les mêmes traces d'apprenant en entrée. Ce type de comparaison nécessite l'emploi de mesures de comparaisons numériques ou symboliques, que l'on appellera par la suite critère de comparaison. Il est donc requis de construire (instancier) au préalable les diagnostics pour le domaine du concepteur.

Comparer des instances nécessite des traces d'apprenant, qui doivent être, pour éviter tout biais, différentes des traces utilisées pour éventuellement construire les diagnostics et/ou fixer leurs paramètres. Cette méthode de séparation des traces, une partie pour la construction (le jeu d'apprentissage) et une partie pour l'évaluation (le jeu de test), se nomme validation croisée (*cross validation*) (Cornuéjols et Miclet, 2011). Une variante de la validation croisée est la validation croisée par k sous-ensembles (*k-folds cross validation*) (Cornuéjols et Miclet, 2011) qui consiste à : séparer les traces en *k* sous-ensembles, puis répéter k fois en boucle la méthode de validation croisée de sorte que chaque sous-ensemble k représente une fois le jeu de test (les *k-1* autres sous-ensembles étant agrégés pour le jeu d'apprentissage). Le résultat final est obtenu en moyennant les *k* résultats obtenus.

Prenons comme exemple deux diagnostics, d1 et d2, instanciés pour un domaine, un critère de comparaison *fm* et un ensemble de traces temporelles T. L'algorithme général de comparaison en validation croisée serait :

Séparer T aléatoirement en deux sous-ensembles Tapprentissage et Ttest
Fixer les paramètres de d1 et d2 avec Tapprentissage

Diagnostiquer les connaissances de chaque apprenant :
1. Avec comme traces d'entrée Ttest, comme traitement du diagnostic d1 et comme modèle d'apprenant inféré MA1
2. Avec comme traces d'entrée Ttest, comme traitement du diagnostic d2 et comme modèle d'apprenant inféré MA2

Comparer MA1 et MA2 avec la mesure *fm*

L'élément principal de la comparaison est la mesure *fm* (le critère de comparaison), qui doit pouvoir prendre en entrée le résultat de deux diagnostics, et donner en sortie le résultat de la mesure. Nous allons étudier par la suite différents critères de comparaison utilisés dans la littérature pour la comparaison d'instances de diagnostics des connaissances.

B. Travaux

Nous avons vu plus haut que le traitement d'un diagnostic générique peut être réalisé de plusieurs façons, par exemple, AFM et PFM sont des traitements alternatifs aux règles d'inférence bayésienne classiques du Knowledge tracing. Dans la littérature, une grande partie des travaux s'attachent à comparer différents traitements pour un diagnostic donné. Parmi les variantes du Knowledge tracing, soulignons notamment les comparaisons entre :

- Knowledge tracing et AFM (algèbre) (Cen et al., 2008)
- Knowledge tracing et PFM (physique, algèbre, géométrie, fractions en mathématiques) (Gong et al., 2011, 2010; Pardos et Heffernan, 2010; Pavlik et al., 2009)
- AFM, PFM et IFM (physique) (Chi et al., 2011)
- Knowledge tracing, Conjunctive Knowledge tracing et LR-DBN (algèbre et apprentissage de l'anglais) (Xu et Mostow, 2012)
- Knowledge tracing et The Student Skill Model (mathématiques) (Wang et Heffernan, 2012)

Tous ces auteurs réalisent leurs comparaisons au moyen de tests statistiques et prédictifs. En premier lieu, la précision de prédiction est ***le taux de bonne prédiction du diagnostic de la prochaine réponse de l'apprenant*** (correct ou incorrect). Le principe est le suivant : le modèle d'un apprenant inféré à un temps *t* par le Knowledge tracing ou l'une de ses variantes fournit pour chaque connaissance la probabilité qu'un apprenant ait appris cette connaissance. Au temps t+1, c'est-à-dire à la prochaine étape de résolution d'un problème ou au prochain problème traité par l'apprenant, il est possible de calculer la probabilité que l'apprenant réponde correctement ou incorrectement, en fonction des probabilités de son modèle inféré au temps t et de la connaissance courante requise. On définit que le diagnostic fait une bonne prédiction si la probabilité de répondre correctement ou non correspond à ce qui a été observé dans les traces. Ce calcul, appliqué pour toutes les traces de tous les apprenants, donne la précision de la prédiction. Du point de vue du diagnostic des connaissances, l'hypothèse qui est faite est que plus un diagnostic prédit correctement la validité des réponses d'un apprenant, plus le modèle de l'apprenant inféré est précis et valide. Le modèle de l'apprenant est vu comme un « modèle prédictif ». En statistique, il existe de nombreuses autres mesures de précision de prédiction pour les modèles prédictifs. Les travaux listés ci-dessus utilisent en particulier la racine de l'erreur quadratique moyenne (RMSE pour *Root-mean-squared error*, une mesure de précision attribuant un poids différent aux possibles erreurs de prédiction) (Armstrong et Collopy, 1992), l'aire sous la courbe ROC (AUC pour *Area under the curve*, qui donne la probabilité que le diagnostic puisse identifier correctement deux réponses quelconques d'un apprenant, l'une correcte et l'autre incorrecte) (Hanley et McNeil, 1983), le *Bayesian information criterion* et l'*Akaike*

information criterion (BIC et AIC, deux mesures pénalisant la précision, plus précisément la fonction de vraisemblance, par la complexité du diagnostic en terme de nombre de variables) (Akaike, 1974; Schwarz, 1978), ou encore le coefficient de détermination (R^2, qui estime le gain de précision par rapport à un prédicteur trivial donnant toujours pour prédiction la moyenne observée dans les traces) (Draper et al., 1966). Toutes ces mesures de précision permettent de comparer et positionner les différents diagnostics entre eux. À titre d'exemple, nous donnons Figure 5 un tableau de comparaison extrait de Chi et al., utilisant pour critères de comparaison BIC et RMSE.

	{Primary-Justify, Count}	
Model	BIC	10-fold RMSE
AFM-Tell	9037	4.460E-01
AFM+Tell	9117	4.470E-01
PFM	8474	4.235E-01
IFM	8347	4.217E-01

Figure 5 : Exemple de résultats de comparaison entre quatre diagnostics (AFM-Tell, AFM+Tell, PFM, IFM) via deux mesures statistiques (BIC et RMSE). **(Chi et al., 2011)**.

Ces mesures de précision ont également été utilisées pour comparer des variantes de diagnostics basés sur l'Item Response Theory (Lee et al., 2008).

D'autres mesures de comparaison ont été ponctuellement étudiées. Pour le Constraint-based, Martin et al. (Martin et al., 2005) ont utilisé des courbes d'apprentissage, c'est-à-dire une courbe modélisant l'évolution de l'apprentissage de l'apprenant en fonction de l'expérience accumulée. Le plus souvent, l'évolution de l'apprentissage se mesure par le taux d'erreurs ou le temps de résolution des exercices, tandis que l'expérience accumulée est représentée par les opportunités d'apprentissage (i.e. le nombre de problèmes ou d'étapes de problèmes résolus nécessitant la mobilisation d'une connaissance). L'idée de Martin et al. est de mesurer, via le test statistique R^2, le diagnostic qui ***prédit le mieux la courbe d'apprentissage***. Toutefois, les auteurs notent que cette approche est soumise à plusieurs biais, notamment sur la définition de la courbe d'apprentissage, son échelle et ses paramètres. Ils utilisent de plus des jeux de traces différents pour chaque diagnostic. Le et Pinwart (Le et Pinkwart, 2012) ont également comparé deux variantes du diagnostic Constraint-based, en utilisant comme mesure ***la précision de prédiction*** (comme défini plus haut, le taux de bonne prédiction) et ***la couverture*** du diagnostic (le taux absolu de prédiction, i.e. le pourcentage de traces d'apprenants pour lequel le diagnostic est capable de faire une prédiction).

Ces mesures de précision de prédiction sont le critère de comparaison le plus commun. Beck (Beck, 2007) s'est intéressé à mesurer ***la plausibilité des paramètres*** du Knowledge tracing instancié pour le domaine de la géométrie des aires. Beck se base sur l'utilisation de paramètres *a priori* pour privilégier des paramètres adaptés pour le plus d'apprenants possible, mais sans obtenir de résultats significatifs.

Une autre approche pour la comparaison est de ***comparer les résultats de plusieurs instances d'un même diagnostic***. Il s'agit ici de faire varier les traces en entrée ou le modèle de l'apprenant à inférer. La plateforme Datashop (Koedinger et al., 2010; Stamper et Koedinger, 2011) assiste ce genre de comparaisons, car il est possible de filtrer les traces en entrée dans la base de traces d'apprenants (par exemple en filtrant par apprenant, par problème, par niveau…) et de modifier le modèle de l'apprenant. Toutefois, Datashop ne propose qu'un seul diagnostic générique : AFM. À titre d'illustration, nous donnons sur la Figure 6 un tableau de comparaison simplifié extrait de Datashop pour le domaine de la géométrie des aires où le modèle de l'apprenant inféré varie ; la colonne de gauche indique le nom des modèles de l'apprenant utilisés. On note que Datashop propose comme critères de comparaison AIC, BIC et RMSE.

	Model Name	AIC	BIC	RMSE (unstratified)
	LFASearchAICWholeModel3	4989.31	5610.39	0.396639
	LFASearchModel1.context-single	4990.76	5624.93	0.399035
	LFASearchAICWholeModel2	4992.65	5600.67	0.398564
	LFASearchAICWholeModel1	5001.42	5596.35	0.399845
	LFASearchAICWholeModel0	5001.46	5609.47	0.39889
	LFASearchModel1	5027.65	5557.21	0.399224

Figure 6 : Capture d'écran de Datashop montrant trois critères statistiques (AIC, BIC, RMSE) permettant de comparer les modèles de l'apprenant indiqués dans la colonne de gauche.

C. Caractéristiques et limites

La comparaison d'instances de diagnostic via des mesures statistiques a permis de renforcer leur évaluation, notamment car la méthode de comparaison est reproductible et numérique (Desmarais et Baker, 2012). L'un des principaux résultats qui émergent de ces travaux est que les résultats sont souvent contradictoires en fonction des domaines et des traces. Savoir quand et pourquoi une instance de diagnostic est plus performante qu'une autre selon un critère de comparaison donné est une question non résolue (Gong et al., 2011).

Une limite qui émerge de la littérature est que la totalité des travaux ne porte que sur la comparaison d'instances de variantes d'un même diagnostic. Par exemple, pour la majorité des travaux, il s'agit de comparer le Knowledge tracing avec ses nombreuses variantes telles que AFM et PFM, de sorte que les traces d'apprenants et les modèles d'apprenants à inférer soient identiques. Il n'existe dans la littérature aucun travail allant vers une comparaison d'instances de diagnostics différents, par exemple Knowledge tracing et Constraint-based.

Une seconde limite est la nature des critères de comparaison. La littérature porte presque exclusivement sur l'utilisation de mesures statistiques (précision de la prédiction en premier lieu, qui est la mesure la plus importante pour un modèle prédictif). Toutefois, une instance de diagnostic des connaissances peut être évaluée autrement que comme un système

prédictif, par exemple via son impact sur les rétroactions fournies par l'EIAH. Beck (Beck et Xiong, 2013; Beck, 2007) a dans ses publications illustré les limites de la précision de la prédiction, notamment en termes de plausibilité du modèle inféré et de quantité d'informations apportées par le modèle. Par ailleurs, on peut noter que l'essentiel des travaux cités plus haut ont conduit leurs expérimentations sur des domaines scientifiques bien définis : calculer la précision de prédiction requiert en effet de collecter dans les traces l'évaluation de la prochaine réponse de l'apprenant (par exemple correct ou incorrect), ce qui est couramment fait dans les EIAH portant sur des domaines scientifiques, mais plus complexe sur d'autres domaines (Fournier-Viger et al., 2012).

3. Synthèse des principaux verrous

L'assistance à la comparaison de diagnostics des connaissances pose les problèmes du type de mesures utilisées pour de telles comparaisons, et de leur application générique.

Le premier verrou est le nombre, la pertinence et la variété des mesures de comparaison proposées dans la littérature. Nous avons noté plusieurs mesures de comparaison issues du monde de la statistique, mais il n'existe pas de travaux permettant de comparer d'autres aspects des diagnostics des connaissances, tels que leur utilité ou leur impact sur l'EIAH. De plus, il n'existe pas de plateforme ou d'outil permettant de tester différentes mesures. Au contraire, les mesures doivent être implémentées par le concepteur, de même que les diagnostics à comparer.

Le second verrou commun à tous les travaux de comparaison utilisant des traces d'apprenants est la prise en compte de plusieurs diagnostics : il n'existe pas d'outils ou de méthodes permettant d'assister la comparaison de diagnostics génériques différents, par exemple Knowledge tracing, Constraint-based et Control-based. Tous les travaux présentés comparant des instances de diagnostics imposent un et un seul diagnostic, parfois ad hoc. Seules les analyses comparatives des caractéristiques des diagnostics proposent la prise en compte de diagnostics différents, mais sans comparaison numérique utilisant des traces d'apprenants. De plus, ces analyses sont insuffisantes car chaque diagnostic peut être plus ou moins performant selon le domaine d'apprentissage et les traces, comme montré par Gong et al. (Gong et al., 2011), et comme elles relèvent du point de vue de leurs auteurs, nous avons constaté que les travaux peuvent être contradictoires. Il y a donc actuellement un verrou qui est le mélange des deux approches : comparaison à partir de traces d'apprenants, donc calculable et reproductible, mais sur des diagnostics différents, donc de façon indépendante de chaque diagnostic.

IV. Conclusion

La recherche sur les diagnostics des connaissances présente diverses pistes pour leur construction (outils auteurs, apprentissage automatique, optimisation) et leur comparaison (analyse comparative experte, calcul de mesures statistiques). Ces travaux apparaissent pour certains diagnostics, dont le Knowledge tracing, matures, solidement évalués et apportant de nombreuses pistes à explorer. Toutefois, ils sont souvent très pointus, car ne prennent

comme objet d'étude qu'un et un seul diagnostic des connaissances, ou au plus des variantes d'un même diagnostic. Il en ressort un cloisonnement des contributions et des pistes à un seul diagnostic, qui correspond souvent à une communauté de chercheurs.

Pour répondre à notre question de recherche, l'assistance à la construction et à la comparaison de plusieurs diagnostics différents, il est donc nécessaire de généraliser ces travaux afin de les appliquer à différents diagnostics, c'est-à-dire à différents types de traces, à différents modèles d'apprenant et à différents traitements de diagnostic.

Une telle généralisation aurait pour bénéfice de renforcer la recherche sur les diagnostics des connaissances en permettant l'évaluation des travaux de façon plus large, sur plusieurs diagnostics très différents. Il s'agit également d'un moyen de résoudre le problème identifié notamment par Gong et al. : s'il est impossible de savoir *a priori* si un diagnostic sera performant et utile pour un EIAH sur un domaine donné, alors il faut mener une étude comparative pour chaque EIAH. Toutefois, le coût d'une telle étude rend nécessaire la proposition d'outils réutilisables et de méthodes d'assistance pour construire et comparer des instances de diagnostics des connaissances.

SECTION II : CONTRIBUTIONS THEORIQUES

Cette section contient le chapitre 3 sur nos contributions théoriques pour permettre d'étudier nos questions de recherche : l'assistance à la construction et à la comparaison de diagnostic des connaissances. Nous proposons une définition et une formalisation de la notion de diagnostic des connaissances générique, que nous nommons « technique de diagnostic générique ». Cette formalisation permettra par la suite de proposer une méthode d'assistance à la construction et à la comparaison de techniques de diagnostic.

Chapitre 3 : Caractérisation d'une technique de diagnostic

Sommaire

I. Diagnostic comportemental et diagnostic épistémique 49

II. Technique de diagnostic générique 50

III. Modèle de diagnostic générique 51

1. Principe 51
2. Formalisation 51
3. Instanciation au domaine 53
4. Évaluation du champ d'application 53
   A. Evalidation empirique 53
   B. Formalisation du champ d'application 56
5. Opérationnalisation 57
6. Implémentation 59
   A. Définition 59
   B. Exemples de transposition 60
   C. Compatibilité avec un modèle de diagnostic 60

IV. Implications liées à la formalisation 62

V. Conclusion 63

Dans ce chapitre, nous allons présenter nos contributions théoriques. Dans la partie I, nus introduisons les notions de diagnostic comportemental et de diagnostic épistémique. Dans la partie II, nous définissons la notion de technique de diagnostic comme un couple formé par un modèle de diagnostic et une implémentation. Dans la partie III, nous formalisons les

notions de modèles de diagnostic et d'implémentation. Dans la partie IV, nous revenons sur les choix que nous avons faits et les implications de ces choix pour la suite de notre travail.

I. Diagnostic comportemental et diagnostic épistémique

Il convient de définir plus spécifiquement la notion de diagnostic des connaissances, en vue d'étudier la comparaison et la construction de diagnostics. Pour Self (Self, 1994), le diagnostic des connaissances est un outil qui renseigne sur l'objet modélisé, i.e. l'apprenant. La problématique de la comparaison et de la réutilisation de diagnostics pose des difficultés liées à la dépendance au domaine, ainsi qu'à la prise en compte des réponses des apprenants dans les traces, qui peuvent être de formats très variés.

Dans la littérature, plusieurs auteurs définissent plusieurs niveaux de diagnostic au sein d'un EIAH, généralement deux. Wenger (Wenger, 1987) distingue ainsi les niveaux comportementaux et épistémiques. Le niveau comportemental (*behavioural level*) est un diagnostic effectué seulement sur les réponses de l'apprenant (est-ce juste, faux, cohérent, incohérent, etc.), mais sans réflexion sur le raisonnement de l'apprenant et sur son acquisition des connaissances dans le temps. Ce point relève du niveau épistémique (*epistemic*, ou *knowledge level*) où le diagnostic repose sur la reconstruction des raisonnements corrects ou erronés de l'apprenant afin de déduire les connaissances du domaine qu'il maîtrise ou non. Balacheff (Balacheff, 1994) fait une distinction similaire dans la didactique des mathématiques en modélisant ces niveaux épistémiques et comportementaux. Anderson (Anderson, 1983) adopte également cette hiérarchie à deux niveaux dans son modèle cognitif ACT-R (cf. l'état de l'art) : il identifie ce qui relève de la perception et ce qui relève de la mémoire, donc des connaissances. D'un point de vue plus informatique, Sison et Shimura (Sison et Shimura, 1998), dans leur étude sur l'apprentissage automatique de diagnostics des connaissances de type librairie d'erreur, font une distinction similaire avec le niveau d'interprétation de l'activité des apprenants, et le niveau d'interprétation de leurs raisonnements (i.e. de leurs connaissances). Dans leurs travaux, un modèle du domaine qui nécessite des connaissances approfondies du domaine étudié est requis pour le premier niveau (l'interprétation de l'activité).

Une étude des EIAH incluant un diagnostic des connaissances montre que cette distinction se retrouve généralement sous des formes plus ou moins équivalentes. Dans une première catégorie d'EIAH, le diagnostic du comportement est une évaluation binaire (correcte ou incorrecte). Cette évaluation est réalisée grâce à des règles expertes dans APLUSIX (Nicaud et al., 2006), grâce à des graphes représentant les plans de résolution des problèmes dans les tuteurs cognitifs (tels Geometry Tutor, Algebra Tutor, LISP Tutor) (Aleven et al., 2006; Anderson, 1990; Anderson et al., 1995), ou encore grâce à un outil de reconnaissance vocale dans le Reading Tutor (Mostow et Aist, 2001). Dans une seconde catégorie, le diagnostic du comportement est une évaluation non binaire, selon une taxonomie prédéfinie. C'est le cas de TELEOS (Luengo et al., 2011) qui évalue chaque action d'un apprenant comme correcte, incorrecte, incorrecte mais en remédiation, incorrecte en s'aggravant. Enfin, dans une troisième catégorie, le diagnostic du comportement est souvent ad hoc en fonction du

domaine. C'est le cas du SQL Tutor (Mitrović, 1998) dont les contraintes utilisent des mots-clés du langage SQL (par exemple : la clase HAVING doit être précédée de la clause GROUP BY). Ici le diagnostic du comportement est très lié au domaine d'apprentissage.

Définitions

> **Diagnostic du comportement :** processus qui prend en entrée des traces des apprenants collectées par l'EIAH, appelées productions de l'apprenant. Il retourne en sortie des traces enrichies par ajout d'une ou plusieurs variables dont la sémantique est prédéfinie. Dans le cas le plus simple, il s'agit de l'évaluation d'une production de l'apprenant, correcte ou incorrecte.

Nous affinons la définition du diagnostic des connaissances en le restreignant au niveau épistémique :

> **Diagnostic des connaissances :** processus qui prend en entrée des traces d'apprenants enrichies par un diagnostic du comportement et retourne le modèle de l'apprenant. Cette définition reste identique à celle donnée en introduction, hormis pour les traces en entrée.

Le schéma ci-dessous (Figure 7) résume le processus formé par les deux niveaux de diagnostic et les traces en entrée et sorti de ces deux niveaux. Les traces d'apprenants sont lues par le diagnostic du comportement pour obtenir des traces enrichies, comme défini ci-dessus. Ces traces enrichies sont lues en entrée du diagnostic des connaissances pour inférer le modèle de l'apprenant.

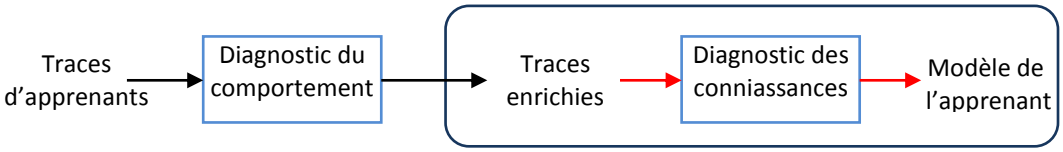


Figure 7 : Schéma du processus allant des traces d'apprenants collectées par l'EIAH au modèle de l'apprenant résultant du diagnostic des connaissances, avec les étapes intermédiaires du diagnostic du comportement qui donnent les traces enrichies. La partie dans le cercle bleu correspond à la partie du processus qui peut être générique.

II. Technique de diagnostic générique

Dans les efforts de catégorisation des diagnostics des connaissances qui ont été faits, notamment par Woolf (Woolf, 2008), se distinguent deux notions : en premier lieu, un modèle du diagnostic permettant de représenter le raisonnement sur les connaissances (par exemple, Knowledge tracing, Constraint-based, librairie d'erreurs) ; ces modèles résultent déjà d'un choix quant à la stratégie d'apprentissage. Ensuite viennent les implémentations de ces modèles, qui permettent d'inférer le modèle de l'apprenant en fonction de ses traces ; majoritairement, il s'agit de méthodes d'IA comme les règles de production, le raisonnement probabiliste et bayésien, la reconnaissance de plan, la logique... Le

développement d'un outil de diagnostic des connaissances résulte en effet d'un double choix de modélisation et d'implémentation.

Définition

> **Technique de diagnostic générique :** une technique de diagnostic générique est l'association d'un modèle de diagnostic générique et d'une implémentation. Elle prend en entrée les traces d'un apprenant enrichies par un diagnostic du comportement et infère le modèle de l'apprenant défini par le modèle de diagnostic.

Nous considérons uniquement les techniques qui ont la propriété d'être génériques. Cette propriété de réutilisation nous impose de prendre en compte les implémentations, ce qui n'a pas été étudié dans les travaux existants.

Par la suite, nous allons définir les notions de modèle de diagnostic générique et d'implémentation, qui forment donc une technique générique de diagnostic.

III. Modèle de diagnostic générique

1. Principe

Un modèle de diagnostic générique spécifie les éléments requis pour inférer le modèle de l'apprenant. Nous proposons dans ce chapitre une formalisation d'un modèle de diagnostic générique, qui doit permettre d'assister la construction et la comparaison de techniques de diagnostics différentes. Les buts visés par une telle formalisation sont de limiter les biais de comparaison et de permettre la construction de techniques de diagnostic sans devoir multiplier les outils ou les méthodes pour chacune des techniques. Le modèle de diagnostic permet enfin de fixer les limites du travail en permettant d'identifier les techniques et les modèles d'apprenants pour lesquels nous pouvons assister la construction et la comparaison.

2. Formalisation

Pour définir un modèle de diagnostic générique, nous proposons une formalisation F_{MD} de ces modèles.

Soit F_{MD} le 7-uplet {O, K, OtoO, OtoK, KtoK, C, BD} avec :

- O un ensemble de types de variables observables
- K un ensemble de types de variables de connaissances (disjoint de O)
- OtoO un ensemble de relations orientées o1…on->o1'…o,' (o1…on, o1'…on' ∈ O) de n éléments de O vers n' éléments de O
- OtoK un ensemble de relations orientées o1…on->k1…km (o1…on ∈ O, k1…km ∈ K) de n éléments de O vers m éléments de K
- KtoK un ensemble de relations orientées k1…kn->k1'…kn' (k1…kn, k1'…kn' ∈ K) de n éléments de K vers n' éléments de K
- C un ensemble de contraintes
- BD ⊂ O un sous-ensemble de O de types de variables résultats d'un processus de diagnostic du comportement

F_{MD} peut donc être vu comme un graphe sémantique orienté. Nous allons définir chaque élément de F_{MD} ci-dessous, avec des exemples.

Définitions

Variable observable : une variable est observable si elle peut être directement ou indirectement collectée par un EIAH dans les traces de l'apprenant. Sont observables dans les EIAH les interactions de l'apprenant avec l'EIAH et l'état des exercices qui en résultent.
Exemples : les actions réalisées par l'apprenant, les étapes du problème en cours, les variables caractérisant les problèmes, les rétroactions données par l'EIAH, l'identifiant de l'apprenant…

Variables de connaissance : une variable de connaissance est une variable cachée, dont la valeur est non observable dans les traces, qui est incluse dans le modèle de l'apprenant. Une variable de connaissance se fonde sur la didactique et la psychologie.
Ex**emples :** compétences, savoirs, connaissances cognitives, stratégie…

Relation OtoO (OtoK, KtoK) : une fonction binaire qui a pour ensemble de départ O et pour ensemble d'arrivée O (respectivement O et K, K et K). On écrit donc OtoO : O -> O (respectivement OtoK : O->K ; KtoK ; K->K).
Exemples : pour OtoO, la relation entre un problème et les étapes de résolution de ce problème ; pour OtoK, la relation entre les actions des apprenants et les connaissances mobilisées pour chaque action ; pour KtoK, la relation entre une stratégie de résolution de problèmes et les connaissances mobilisées par cette stratégie.

Contrainte :

- Soit un couple {R, D1->D2} avec R une relation et D1->D2 une contrainte sur la cardinalité de la relation R, où D1 est la cardinalité de l'ensemble de départ de R et D2 la cardinalité sur l'ensemble d'arrivée de R.
- Soit un couple {E, D} avec E un ensemble de types de variables et D une contrainte sur la cardinalité de E.

Exemples : pour la relation nommée *StrategyToK* entre une stratégie de résolution de problèmes et les connaissances mobilisées par cette stratégie : la cardinalité de son ensemble de départ est 1 et la cardinalité de son ensemble d'arrivée est supérieure ou égale à 1 (signifiant qu'une et une seule stratégie est reliée à au moins une connaissance). Cette contrainte se note { *StrategyToK*, 1->1..*}.

Le modèle fait ainsi l'hypothèse que le traitement du diagnostic des connaissances repose sur les liens entre variables observables et variables de connaissances.

Définition

Un **modèle de diagnostic générique** est une instance de F_{MD}.

3. Instanciation au domaine

Un modèle de diagnostic est par définition générique (cf. ci-dessus), signifiant qu'il peut être appliqué pour diagnostiquer les connaissances d'apprenants pour des domaines d'apprentissage variés (algèbre, chimie, français…).

Définition

> Une **instance de modèle de diagnostic générique** est un diagnostic générique appliqué à un domaine d'apprentissage. On dit que le modèle de diagnostic est instancié au domaine d'apprentissage.

Une telle instanciation requiert une bonne connaissance du domaine et de la modélisation des connaissances. Dans les EIAH, elle est généralement effectuée par ou en collaboration avec des didacticiens ou des psychologues.

4. Évaluation du champ d'application

A. Évaluation empirique

Premièrement, de façon empirique, nous pouvons démontrer que plusieurs modèles de diagnostics génériques publiés dans la littérature sont des instances de la formalisation F_{MD}. Pour chaque modèle de diagnostic générique, nous rappelons succinctement son principe, donnons l'instance de F_{MD} et la représentation sous forme de graphe orienté pondéré de l'instance. On pourra se référer à l'état de l'art pour les détails sur chaque diagnostic.

Knowledge tracing

Le Knowledge tracing (Corbett et Anderson, 1995) représente les problèmes comme une succession d'étapes, chaque étape étant liée à une et une seule connaissance nommée Knowledge Component. Le diagnostic comportemental, nommé Outcome, est binaire et exprime si l'apprenant a fait une réponse ou une action correcte ou incorrecte à chaque étape.

- O = { Problem ; Step ; Outcome }
- K = { Knowledge Component }
- OtoO = { PtoS : Problem -> Step }
- OtoK = { StoK : Step -> Knowledge Component ; OtoK : Outcome -> Knowledge Component}
- KtoK = {}
- C = { C1: Card(StoK)=*->1 ; C2: Card(OtoK)=1->1 }
- BD = { Outcome }

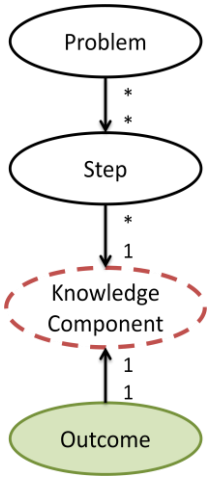


Figure 8 : Représentation graphique du modèle de diagnostic Knowledge tracing. Légende : cercle continu noir=observable, cercle pointillé rouge=connaissance, cercle continu avec fond vert=résultat du diagnostic comportemental.

Constraint-based

Le Constraint-based (Ohlsson, 1994) représente les connaissances comme des contraintes (Cr, Cs). Cr décrit les situations où la contrainte est applicable et Cs l'état du problème qui doit être obtenu si Cr est vrai. Le diagnostic comportemental est inclus dans Cr.

- O = { Problem, Cr }
- K = { Cs }
- OtoO = { fPtoCr : Problem -> Cr }
- OtoK = { gCrtoCs : Cr -> Cs }
- KtoK = {}
- C = { C1: Card(gCrtoCs) = 1->1}
- BD = { Cr }

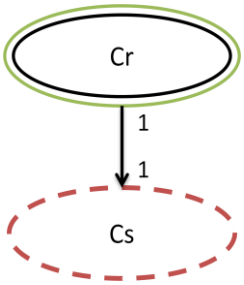


Figure 9 : Représentation graphique du modèle de diagnostic Constraint-based. Légende : cercle continu noir=observable, cercle pointillé rouge=connaissance, cercle continu avec fond vert=résultat du diagnostic comportemental.

Control-based

Le Control-based (Minh Chieu et al., 2010) décrit les problèmes, les opérateurs que l'apprenant peut mobiliser pour résoudre chaque problème, les registres de représentations des problèmes et des opérateurs, ainsi que les connaissances, nommées contrôles, que l'apprenant doit mettre en jeu pour valider l'utilisation de chaque opérateur en fonction des problèmes et des registres. Le diagnostic comportemental est représenté par les variables de situations associées à chaque contrôle.

- O = { Problem ; Operator ; Register ; Situation Variable }
- K = { Control }
- OtoO = { PtoOp : Problem -> Operator}
- OtoK = { OptoC : Operator -> Control ; RtoC : Register -> Control ; PtoC : Problem -> Control ; SVtoC : Situation Variable -> Control }
- KtoK = {}
- C = {}
- BD = { Situation Variable }

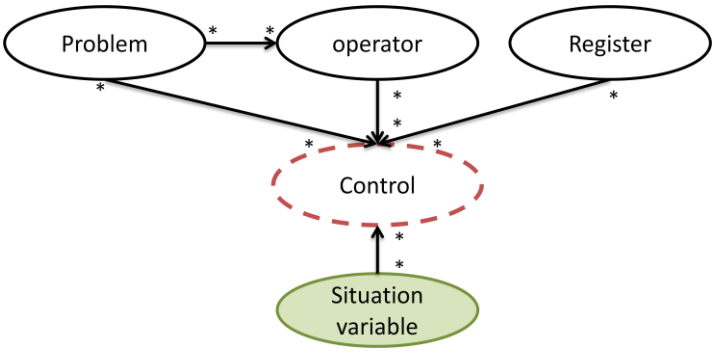


Figure 10 : Représentation graphique du modèle de diagnostic Control-based. Légende : cercle continu noir=observable, cercle pointillé rouge=connaissance, cercle continu avec fond vert=résultat du diagnostic comportemental.

Item response theory

L'Item response theory (Johns et al., 2006) décrit un ensemble de connaissances dite cachées ou latentes. Chaque connaissance est associée à un ou plusieurs *items* (éléments observables du domaine, généralement les questions d'un exercice) indépendants. Le diagnostic comportemental est la réponse de l'apprenant à chaque *item*.

- O = { Item, Response }
- K = { Skill }
- OtoO = { RtoI : Response -> Item }
- OtoK = { ItoS : Item -> Skill }
- KtoK = {}
- C = { C1: Card(ItoS)=*->1 ; C2: Card(RtoI)=1->1 }
- BD = { Response }

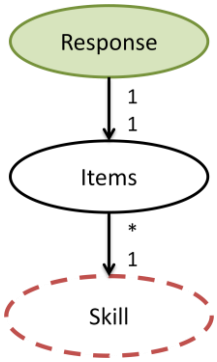


Figure 11 : Représentation graphique du modèle de diagnostic Item response theory. Légende : cercle continu noir=observable, cercle pointillé rouge=connaissance, cercle continu avec fond vert=résultat du diagnostic comportemental.

LR-DBN et Conjunctive Knowledge tracing

LR-DBN (Xu et Mostow, 2011) et le Conjunctive Knowledge tracing (Koedinger et al., 2011) dérivent du Knowledge tracing. La différence est qu'un Knowledge Component, ici appelé Skill, peut être associé à une ou plusieurs sous-connaissances (Subskills).

- O = { Problem ; Step ; Outcome }
- K = { Skill, Subskill }
- OtoO = { PtoS : Problem -> Step ; OtoS : Step -> Outcome }
- OtoK = { StoK : Step -> Skill }
- KtoK = { SubtoS: Subskill -> Skill }
- C = { C1: Card(StoK)=1->1 ; C2: Card(SubtoS)=*->1 ; C3: Card(OtoS)=1->1 }
- BD = { Outcome }

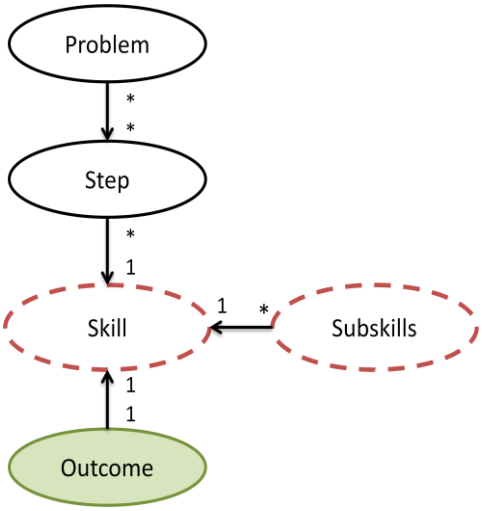


Figure 12 : Représentation graphique des modèles de diagnostic LR-DBN et Cnjunctive Knowledge tracing. Légende : cercle continu noir=observable, cercle pointillé rouge=connaissance, cercle continu avec fond vert=résultat du diagnostic comportemental.

B. Évaluation du champ d'application

Empiriquement, nous avons montré que la formalisation permet de représenter différents diagnostics génériques classiques dans la littérature. Toutefois, nous nous posons la question de la systématisation de ce procédé. Le problème est de pouvoir définir le champ d'application de F_{MD}, i.e. de pouvoir répondre à la question : un diagnostic générique quelconque est-il une instance de notre formalisation ?

Pour ce faire, nous reformulons notre formalisation F_{MD} en logique de description AL :

O ≡ ¬ K
BD ⊑ O
relationOtoK ⊑ ⊤ r
relationKtoO ⊑ ⊤ r
Kobs ≡ K ⊓ relationKtoO
Ok ≡ O ⊓ relationOtoK

Avec O signifiant observables et K signifiant connaissances. Nous nommons cette formulation $F_{MD\text{-}AL}$. La logique de description AL est une généralisation des réseaux sémantiques qui permet de représenter des connaissances et des concepts (Baader, 2003). Or, notre formalisation F_{MD} est, comme nous l'avons noté, proche des réseaux sémantiques. De plus, AL permet de résoudre les problèmes de subsomption, satisfiabilité, équivalence et disjonction avec une complexité polynomiale.

Or, nous pouvons définir que l'ensemble des diagnostics génériques modélisés par notre formulation est l'ensemble des diagnostics génériques formalisés en logique descriptive qui sont subsumés par $F_{MD\text{-}AL}$. Par corollaire, un diagnostic donné est une instance de la formulation F_{MD} s'il est possible d'en donner une formalisation en logique de description qui est subsumée par $F_{MD\text{-}AL}$.

5. Opérationnalisation

La formalisation que nous avons présentée plus haut, qui est en fait un graphe sémantique, peut être opérationnalisée en logique descriptive AL comme montré ci-dessus (l'inférence de AL ayant une complexité polynomiale), ou bien sous forme d'ontologie au format RDF/OWL. Nous avons retenu cette seconde option, du fait du grand nombre d'outils et de librairies qui supportent RDF/OWL.

Le vocabulaire du modèle étant déjà posé, nous donnons directement l'ontologie, formée de :

- trois classes : Observable, Knowledge et Comportement
- trois relations : OtoO, OtoK, KtoK

```
<Ontology xmlns="http://www.w3.org/2002/07/owl#">

<ontology IRI="Ontology1378170810155.owl">

<!-- Déclaration du vocabulaire -->

   <Declaration><Class IRI="#Comportement"/></Declaration>
   <Declaration><Class IRI="#Knowledge"/></Declaration>
   <Declaration><Class IRI="#Observable"/></Declaration>
   <Declaration><ObjectProperty IRI="#KtoK"/></Declaration>
   <Declaration><ObjectProperty IRI="#OtoK"/></Declaration>
   <Declaration><ObjectProperty IRI="#OtoO"/></Declaration>

<!-- Héritage -->

   <SubClassOf>
      <Class IRI="#Comportement"/>
      <Class IRI="#Observable"/>
   </SubClassOf>

<!-- Propriétés -->
```

```
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#KtoK"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#KtoK"/>
      <Class IRI="#Knowledge"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>

  <ObjectPropertyDomain>
    <ObjectProperty IRI="#OtoK"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#OtoK"/>
      <Class IRI="#Observable"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>

  <ObjectPropertyDomain>
    <ObjectProperty IRI="#OtoO"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#OtoO"/>
      <Class IRI="#Observable"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyDomain>

  <ObjectPropertyRange>
    <ObjectProperty IRI="#KtoK"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#KtoK"/>
      <Class IRI="#Knowledge"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>

  <ObjectPropertyRange>
    <ObjectProperty IRI="#OtoK"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#OtoK"/>
      <Class IRI="#Knowledge"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>

  <ObjectPropertyRange>
    <ObjectProperty IRI="#OtoO"/>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#OtoO"/>
      <Class IRI="#Observable"/>
    </ObjectSomeValuesFrom>
  </ObjectPropertyRange>
</Ontology>
```

L'ontologie est composée de trois parties : premièrement la déclaration des variables, via les balises <Declaration></ Declaration>, puis la déclaration des relations d'héritage via les balises <SubClassOf></ SubClassOf>, et enfin la déclaration des relations via les balises <ObjectPropertyRange></ ObjectPropertyRange>.

L'ontologie est de plus capable de modéliser les contraintes des modèles de diagnostics sur les cardinalités des relations via plusieurs opérateurs. Soit une cardinalité c, les opérateurs sont : « il existe » (c>0), « un et un seul » (c=1), max (c<=x), min (c>=x), égal à (c=x), x étant un nombre entier.

6. Implémentation

A. Définition

Nous avons défini plus haut une technique comme l'association entre un modèle de diagnostic et une implémentation. Les sections précédentes s'attachent à définir le modèle de diagnostic, et nous allons à présent définir la notion d'implémentation.

Définition

> L'**implémentation d'un modèle de diagnostic** consiste à **transposer** le modèle de diagnostic instancié au domaine dans un formalisme calculable.

Nous pouvons noter que cette définition induit une limite : toute implémentation ne permet pas d'implémenter tous les modèles de diagnostic. Prenons par exemple le modèle de diagnostic Control-based présenté plus haut : il s'agit d'un graphe qui ne peut être implémenté via un modèle de Markov caché. En effet, un modèle de Markov caché est défini comme suit :

Définition

> Un **modèle de Markov caché** est un automate décrivant un système par le quadruplet {S, O, P, A, B} avec S l'ensemble des états de l'automate, O l'ensemble des symboles d'observations du système, $p_i \in P$ la probabilité que l'état initial de l'automate soit s_i , $a_{ij} \in A$ la probabilité de transition de l'état s_i à s_j et $b_{jk} \in B$ la probabilité d'observer l'observation k dans l'état j. (Baum et Petrie, 1966)

Selon cette définition, un modèle de Markov caché ne peut représenter que deux et seulement deux ensembles de variables (les états et les observations), et les relations entre ces deux ensembles. Le Control-based possède en revanche cinq ensembles de variables et des relations entre ces ensembles, donc ne peut être implémenté via un modèle de Markov caché.

La transposition d'un modèle de diagnostic vers une implémentation requiert les tâches suivantes :

- La déclaration de l'ensemble des variables
- La déclaration de l'ensemble des relations

- La déclaration d'une structure d'adjacence entre les variables et les relations
- La déclaration d'accesseurs pour les cardinalités des ensembles et des relations

B. Exemples de transposition

Les informations nécessaires à la transposition d'un modèle de diagnostic vers une implémentation sont spécifiques à chaque implémentation. Prenons les exemples du réseau bayésien et de la logique du premier ordre.

Un réseau bayésien est un graphe G=(V,E) avec V l'ensemble des sommets, E l'ensemble des arrêtes, ainsi qu'une distribution de probabilités Pr(V) associée aux nœuds du graphe. L'implémentation d'un modèle de diagnostic se fait comme suit :

- les variables sont les nœuds V
- les relations sont les arrêtes E
- la structure d'adjacence est la matrice d'adjacence du graphe
- les accesseurs se basent sur le nombre de parents immédiats de chaque nœud

La logique du premier ordre est composée d'un ensemble de variables (logiques) et d'un ensemble de prédicats. Elle est munie des symboles pour tout, il existe, et, ou, non, implique. L'implémentation d'un modèle de diagnostic se fait comme suit :

- les variables du modèle sont les variables de la logique
- les relations sont les prédicats
- la structure d'adjacence est l'ensemble des formules atomiques, c'est-à-dire de type P(a, b,…n) (P un prédicat, a, b…n des variables)
- les accesseurs se basent sur le comptage des formules atomiques

C. Compatibilité avec un modèle de diagnostic

Nous avons montré que toute implémentation ne permet pas nécessairement d'implémenter un modèle de diagnostic. Dans le paragraphe 6.A ci-dessus, nous avons par exemple montré qu'un modèle de Markov caché ne peut pas implémenter le modèle de diagnostic Control-based.

Propriété

> Soit une technique de diagnostic le couple formé par un modèle de diagnostic MD et une implémentation I. MD et I sont **compatibles** si et seulement si I peut implémenter toute instance du modèle de diagnostic MD.

Il est important que cette propriété soit respectée, car elle permet de garantir que toutes instances entre deux ou plusieurs techniques de diagnostic seront bien comparables.

Il est possible de démontrer par induction qu'une implémentation permet de transposer toute instance d'un modèle de diagnostic donné. Prenons par exemple le cas d'un réseau bayésien.

Définition

Un **réseau bayésien** est un graphe orienté acyclique G=(V, E) avec V l'ensemble des sommets (aussi appelés nœuds) et E l'ensemble des arrêtes de G. À l'ensemble des sommets de V est associé à la distribution de probabilité $P(V) = \prod_{x \epsilon V} Pr(x \mid pa(x))$ où pa(x) est l'ensemble des parents immédiats de x dans V (Pearl, 1988).

Intuitivement, un réseau bayésien permet de représenter un ensemble de variables aléatoires et les relations de probabilités entre ces variables.

Soit un réseau bayésien à deux nœuds *a* et *b* et une arrête entre ces nœuds, ce qui correspond au modèle de diagnostic suivant (cf. les schémas sur la Figure 13) :

M0 = { O = {a} ; K = {b} ; OtoK = {a -> b} }

On veut démontrer que l'ajout d'éléments ou de relations dans O, K et OtoK conserve la transposition en un réseau bayésien, i.e. en un graphe orienté acyclique. Il est possible d'ajouter un élément *c* soit dans l'ensemble O, soit dans l'ensemble K du modèle de diagnostic :

M1 = { O = {a, c} ; K = {b} ; OtoK = {a -> b ; c -> b} }
M2 = { O = {a} ; K = {b, c} ; OtoK = {a -> b ; a -> c} }

M1 et M2 sont tous deux des réseaux bayésiens, c'est-à-dire un graphe orienté sans cycle, avec un nœud et une arrête supplémentaire par rapport à M0. En généralisant, on peut ajouter *n* variables dans O ou *m* variables dans K :

M1t = { O = {a1 ; … ; an} ; K = {b} ; OtoK = { a1 -> b ; … ; an -> b } }
M2t = { O = {a} ; K = {b1 ; … ; bm} ; OtoK = { a -> b1 ; … ; a -> bm }

Par induction, M1 et M2 étant des réseaux bayésiens construits à partir de M0, alors M1t et M2t sont des réseaux bayésiens respectivement construits à partir de M1 et M2.

Généralisons pour Mt avec l'ajout de *n* éléments dans O et de *m* éléments dans K simultanément :

Mt = { O = { a1 ; … ; an } ; K = { b1 ; … ; bm } ; OtoK = { a1 -> b1 ; … ; a1 -> bm ; … an -> bm } }

Par induction, Mt est toujours un graphe orienté acyclique et donc un réseau bayésien. On en conclut que tout modèle de diagnostic avec seulement des relations OtoK peut être transposé en un réseau bayésien. Le même mécanisme de preuve peut être appliqué aux autres relations (KtoK et OtoO) et à d'autres implémentations que les réseaux bayésiens. La formalisation F_{MD} permet donc de vérifier la compatibilité entre une implémentation et un modèle de diagnostic.

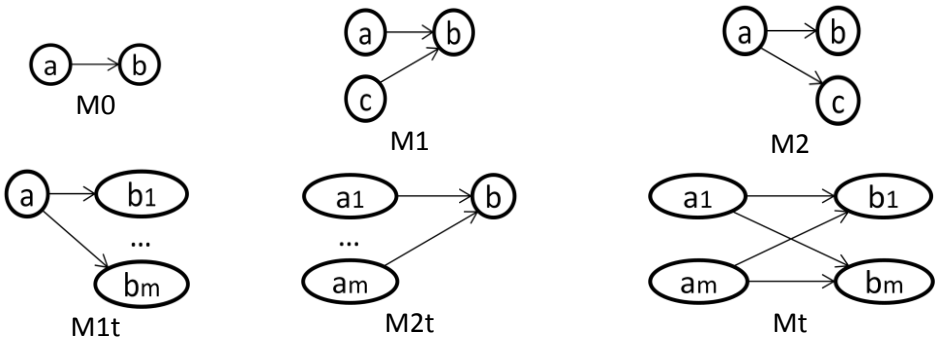


Figure 13 : Représentation des différents réseaux bayésiens associés à chaque modèle de diagnostic décrit plus haut.

D'un point de vue informatique, une implémentation peut être vue comme un code source capable de transposer un modèle de diagnostic (comme détaillé ci-dessus) et d'inférer le modèle de l'apprenant. Dans nos travaux, nous considérerons pour illustration les implémentations suivantes : réseau bayésien dynamique, modèle de Markov caché, régression linéaire, règles IF/THEN, logique floue.

IV. Implications liées à la formalisation

La formalisation d'un modèle de diagnostic générique F_{MD} proposée dans ce chapitre entraîne plusieurs implications.

Premièrement, cette formalisation permet de délimiter le cadre de notre travail en définissant l'ensemble des techniques de diagnostic que nous prenons en compte. Nous n'affirmons pas que toute technique de diagnostic générique publiée soit une instance de notre formalisation, mais nous délimitons un sous-ensemble de techniques pour lesquelles nous pouvons étudier la construction et la comparaison de façon générique.

Deuxièmement, nous faisons un certain nombre d'assertions. Nous estimons qu'un modèle de diagnostic générique associe des variables observables et des variables non observables (connaissances). Nous estimons qu'il ne peut y avoir de relations orientées de l'ensemble des connaissances vers l'ensemble des observables (KtoO), donc qu'il ne peut y avoir de cycle entre les deux ensembles. Nous estimons que la formalisation d'un modèle de diagnostic est équivalente à une ontologie (donc à un graphe sémantique). Nous estimons que tout modèle de diagnostic peut être implémenté, donc est calculable.

Troisièmement, nous avons vu dans l'état de l'art l'inexistence de travaux portant sur la construction ou la comparaison de plusieurs techniques de diagnostic basées sur des approches différentes (par exemple, un diagnostic Knowledge tracing et un diagnostic Constraint-based). Notre formalisation, en permettant justement de représenter différentes approches, a pour but de pallier ce problème. En effet, nous nous attacherons par la suite à définir des outils d'assistance qui se basent sur la formalisation, donc qui doivent fonctionner pour toute technique de diagnostic modélisée par cette formalisation. Cette approche nous donne une base pour répondre aux questions de recherche.

V. Conclusion

Nous avons proposé de définir une technique de diagnostic générique comme l'association d'un modèle de diagnostic générique et d'une implémentation, où le modèle de diagnostic générique est une instance de la formalisation F_{MD} qui peut être représentée comme une ontologie, et où l'implémentation permet de rendre le modèle de diagnostic générique calculable à partir de traces d'apprenants.

Cette notion de technique de diagnostic pose une définition commune pour un ensemble de diagnostics très différents. Cela doit permettre la proposition d'outils d'assistance communs à cet ensemble de diagnostics. De plus, la séparation effectuée entre le diagnostic comportemental et le diagnostic épistémique, reprise de la littérature, permet d'isoler deux problèmes différents que sont le suivi comportemental d'un apprenant dans un domaine donné, et le suivi de l'état de ses connaissances qui peut être fait génériquement.

VanLehn (VanLehn, 1988) note qu'il n'existe pas dans la littérature de formalisation ou de définition très précise du contenu d'un modèle d'apprenant ou d'un diagnostic. La seule approche semblable à la nôtre est celle de Self (Self, 1994) qui propose une approche formelle pour le diagnostic des connaissances (*student modelling* en anglais dans son article). Il propose de se baser sur une des diverses définitions philosophiques de la connaissance pour en dériver une modélisation en logique : un apprenant possède une connaissance 1/ s'il croit en cette connaissance 2/ si la connaissance est vraie (en EIAH, vraie du point de vue du domaine) 3/ si la croyance est justifiée (par exemple, l'apprenant a résolu des exercices en utilisant sa croyance en la connaissance). Dans l'approche de Self, le modèle de l'apprenant est l'ensemble des propositions logiques que l'apprenant croit (par exemple, « l'apprenant croit qu'une masse ne peut pas être négative » en physique), qui est un sous-ensemble de propositions connues par l'EIAH. Le diagnostic vise à mettre à jour (*update*), à partir des interactions de l'apprenant avec l'EIAH représentées par un vecteur de variables, les propositions dans lesquelles l'apprenant croit. La modélisation de Self permet de réaliser ce diagnostic de deux façons :

- en cherchant dans l'ensemble des propositions connues par l'EIAH celles qui sont satisfaites par les entrées de l'apprenant (par exemple, « si l'apprenant calcule masse=100 », alors la proposition « l'apprenant croit qu'une masse ne peut pas être négative » peut être incluse dans le modèle de l'apprenant) ;
- en cherchant dans un second ensemble de propositions erronées (par exemple « l'apprenant croit qu'une masse peut être négative » ou « l'apprenant croit que le poids est indépendant de la masse », toutes deux fausses en physique) celles qui expliquent une erreur de l'apprenant.

Enfin, en cas d'incohérences dans l'ensemble des propositions crues par l'apprenant, le diagnostic inclut des axiomes d'inférence permettant de les résoudre. Ces incohérences correspondent pour Self au temps de l'apprentissage où les croyances de l'apprenant évoluent.

L'objet du travail de Self est de proposer un nouveau diagnostic formalisé de sorte qu'il ne dépende pas d'une seule théorie d'apprentissage ou d'un seul modèle didactique. En prenant pour appui une définition philosophique de la connaissance et non une théorie d'apprentissage, il fait l'hypothèse que son diagnostic est applicable pour tout EIAH et synthétise le fonctionnement de différentes techniques de diagnostic. En revanche, il ne vise pas à formaliser un ensemble de diagnostics génériques différents. En ce sens, son approche fournit une technique de diagnostic supplémentaire, synthétique et formelle, mais ne permet pas de formaliser plusieurs techniques de diagnostic différentes dans le but de proposer des méthodes de comparaison. De plus, sa formalisation est essentiellement basée sur la logique du premier ordre, qui ne prend par exemple pas en compte l'incertitude, là où nous proposons une stricte séparation entre formalisation et implémentation. Notre définition d'une technique de diagnostic se veut être une contribution permettant de traiter de ces questions de recherche (comparer et construire plusieurs techniques de diagnostic différentes).

Dans la suite du mémoire, l'objet central de nos travaux sera la notion de « technique de diagnostic des connaissances » telle que définie dans ce chapitre.

SECTION III : CONTRIBUTIONS POUR LA CONSTRUCTION ET LA COMPARAISON

Cette section contient les chapitres décrivant nos contributions sur l'assistance à la construction et à la comparaison de techniques de diagnostic des connaissances, telle que définie dans la section précédente. Nous décrivons d'abord la méthode pour chacun des deux problèmes, i.e. construction au chapitre 4 et comparaison au chapitre 5, puis présentons une première plateforme réifiant ces méthodes au chapitre 6.

Chapitre 4 : Assistance à la construction de techniques de diagnostic

Sommaire

I. Aperçu 66

II. Modélisation des traces et des connaissances 67

1. CREAM-C 68
2. Conception de l'ontologie des connaissances 69
3. Association entre l'ontologie et les traces 70

III. Algorithme semi-automatique d'instanciation des techniques 70

1. Définition du problème algorithmique 70
   - A. Définition et propriétés du problème d'instanciation d'une technique de diagnostic 70
   - B. Heuristique de recherche locale pour résoudre l'explosion combinatoire 72
2. Algorithme d'instanciation d'une technique de diagnostic 74
   - A. Notations 74
   - B. Principe et exemple illustratif 74
   - C. Recherche locale pour l'association entre le modèle de diagnostic et les traces (étape 1) 76
   - D. Instanciation et implémentation (étape 2) 79
   - E. Apprentissage des paramètres (étape 3) 83
   - F. Librairies 85

IV. Exemple d'application de l'algorithme 86
1. Control-based 87
2. Knowledge tracing 91
3. Constraint-based 93
4. Ontologie détaillée 95
V. Caractéristiques et limites 97
1. Propriétés de l'algorithme d'apprentissage 97
2. Limites 97
VI. Retour sur les scénarios 98
1. Premier cas d'usage 98
2. Second cas d'usage 98
VII. Conclusion 99

I. Aperçu

Dans ce chapitre, nous allons présenter nos contributions pour l'assistance à la construction de techniques de diagnostic des connaissances via un algorithme d'apprentissage semi-automatique. Dans la partie II, nous décrivons l'ontologie permettant l'apport de sémantique par le concepteur pour guider l'algorithme d'apprentissage. La partie III décrit l'algorithme semi-automatique pour la construction des techniques. La partie IV présente quatre exemples complets de construction d'une technique de diagnostic. La partie V revient sur les caractéristiques et les limites de notre proposition, et la partir VI sur les cas d'usage.

Nous avons défini au chapitre 3 la notion de technique générique de diagnostic des connaissances. Le problème traité dans ce chapitre est la construction d'une technique de diagnostic, c'est-à-dire l'instanciation de la technique générique aux traces du concepteur et son implémentation.Pour éviter le travail fastidieux qui consisterait à instancier chaque technique séparément, notre méthode utilise un seul algorithme d'apprentissage capable d'instancier chaque technique de diagnostic (à condition que les techniques soient conformes à notre formalisation chapitre 3). Toutefois, instancier des techniques de diagnostic à partir de traces quelconques n'est pas un problème informatique solvable dans un cas général, car les possibilités d'instanciation sont très grandes, comme nous le démontrerons dans ce chapitre. Un apport de sémantique est nécessaire. De plus, notre objectif est de conserver l'interprétabilité et la plausibilité des techniques construites. Enfin, des connaissances du domaine que le concepteur souhaite diagnostiquer avec les techniques peuvent ne pas être observées dans les traces. Pour pallier à ces problèmes, l'algorithme d'apprentissage que nous proposons est guidé par une ontologie des connaissances, dont les éléments sont liés aux traces pour apporter la sémantique. Pour un domaine donné, représenté par un ensemble de traces, une seule ontologie est réalisée par

le concepteur pour l'ensemble des techniques, car elle est liée aux traces et non à chaque technique de diagnostic.

L'objectif est d'instancier (d'appliquer) un ensemble de techniques de diagnostic conformes à notre formalisation F_{MD} du chapitre 3 au domaine à partir de traces, de sorte que les traces du concepteur et les connaissances de son domaine puissent être prises en compte. Il en résulte des techniques de diagnostic applicables aux traces et/ou à l'EIAH du concepteur. La plateforme assiste chaque concepteur dans cette tâche, via le processus suivant : le concepteur crée une ontologie des connaissances du domaine et la met en correspondance avec ses traces. Puis l'algorithme instancie au domaine chaque technique générique de sa base à partir des traces, de l'ontologie et de la mise en correspondance, et apprend les paramètres de l'implémentation de la technique si besoin (par exemple les tables de probabilités d'un réseau bayésien).

II. Modélisation des traces et des connaissances

Comme expliqué précédemment, le choix d'un algorithme semi-automatique est dû à trois limites de l'apprentissage automatique non supervisé : le manque de données dans les traces, les variables cachées et la difficulté à interpréter et à utiliser les résultats de l'apprentissage. Le concepteur guide l'apprentissage par un apport *a priori* de sémantique experte sur le diagnostic. Cet apport de sémantique doit satisfaire les contraintes suivantes :

- être suffisamment proche de notre formalisation F_{MD} pour pouvoir guider la construction de tous les modèles de diagnostic considérés, qui sont des instances de $F_{MD,}$
- être peu coûteux pour être effectué par le concepteur, puisque l'objectif est de guider l'algorithme de construction semi-automatique et non de construire les techniques manuellement.

La première contrainte requiert que le concepteur puisse modéliser des connaissances et des observables via un modèle aussi expressif que les graphes, puisque F_{MD} peut être représenté comme un graphe. Ce modèle doit être lié aux traces pour leur apporter de la sémantique. D'un point de vue technique, cela peut être un graphe, une ontologie, une carte conceptuelle… La seconde contrainte impose de proposer une modélisation qui permet de définir peu d'éléments et qui est suffisamment explicite pour guider le concepteur.

Dans la littérature, plusieurs travaux proposent la modélisation de connaissances et d'observables via des ontologies ou des graphes acycliques. Toutefois, Gertner intègre la modélisation de buts des exercices et la succession des connaissances et actions à mobiliser pour atteindre les buts, ce qui ne nous intéresse pas ici. Murray intègre une modélisation riche et peu contrainte du domaine et des connaissances, or, nous ne souhaitons pas modéliser tous les concepts d'un domaine mais seulement décrire les observables et connaissances dans les traces et apporter les connaissances manquantes dans les traces. Nkambou intègre une modélisation fine des connaissances et des ressources en lien avec ces connaissances. Leur modèle basé sur une ontologie nommée CREAM-C correspond mieux à

nos contraintes : il permet au concepteur de modéliser des connaissances de façon guidée sous forme de graphe (ontologie).

1. CREAM-C

L'ontologie conçue par le concepteur doit donc étendre CREAM-C (Nkambou et al., 2003, 1997), qui apporte le vocabulaire de base. CREAM-C est une ontologie visant à modéliser les « capacités » d'un domaine. Une capacité se définit comme « une connaissance, acquise ou développée, permettant à une personne de réussir dans l'exercice d'une activité physique ou intellectuelle ». Elle se caractérise principalement par son type, au nombre de neuf : loi, proposition, ensemble de propositions, concept défini, concept concret, règle, règle d'ordre supérieur, stratégie de résolution de problèmes et stratégie d'apprentissage (voir Figure 14).

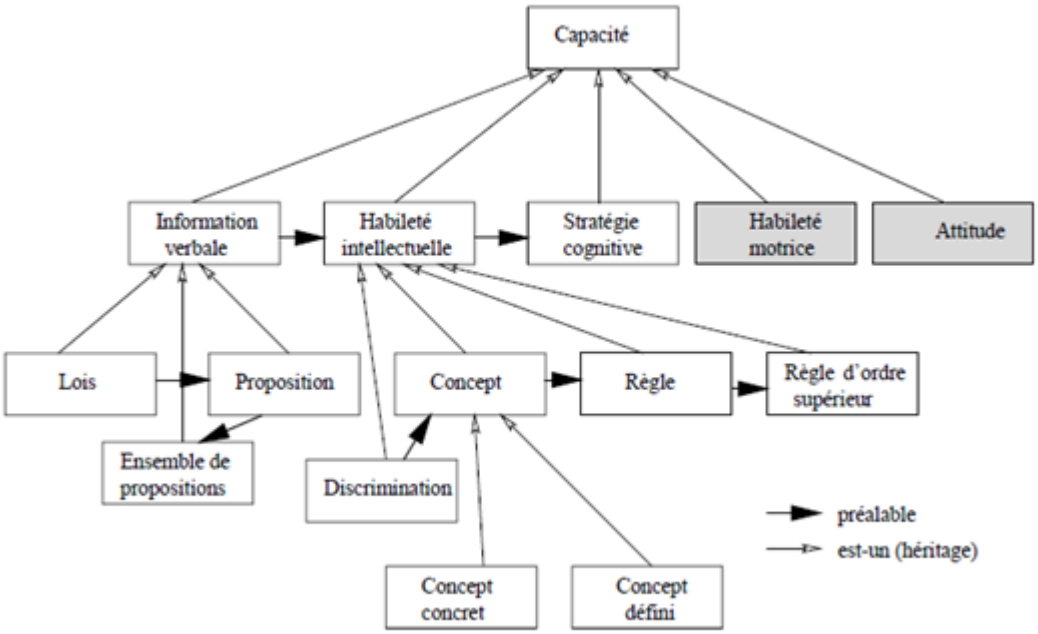


Figure 14 : Taxonomie des capacités utilisée dans CREAM-C (Nkambou et al., 1997).

Les lois et propositions sont « des capacités permettant à l'individu de se représenter la réalité et de communiquer ces représentations à d'autres individus. La possession de ce type de capacité signifie que l'individu peut énoncer, sous forme de proposition, expliquer ou décrire ce qu'il a appris. Par exemple, "énoncer la loi d'Ohm". »

Les concepts, règles et stratégies sont « constitués d'opérations mentales à exécuter pour agir (intérieurement) sur la réalité. Elles permettent de manifester des comportements du type démontrer, prouver, exécuter, reconnaître, identifier, procéder… ».

Les stratégies sont « des capacités utilisées dans les processus cognitifs mis en œuvre pour apprendre (dans ce cas on parle de stratégies d'apprentissage) ou pour résoudre des problèmes (on parle de stratégies de résolution de problèmes). Leur utilisation nécessite un contrôle de la part de l'étudiant en fonction de la situation. »

Quatre types de relations entre les capacités sont possibles : d'analogie (A), de généralisation (G), d'agrégation (Ag) et de déviation (D).

Le modèle des capacités se définit comme le triplet suivant[1] :

- l'ensemble C de capacités,
- la partition (Ci) i∈I de C correspondant aux types de capacités,
- l'ensemble Rc = {A, G, Ag ,D} des relations définies sur les capacités.

2. Conception de l'ontologie des connaissances

L'ontologie des connaissances que doit concevoir le concepteur se compose de deux parties : les capacités (telles que définies ci-dessus au sein de CREAM-C) et les transitions exprimant les informations observables dans les traces. Le terme « transition » est utilisé pour suggérer la transition d'un problème d'un état observable à un autre. Deux niveaux d'ontologies peuvent être construits :

- ontologie de haut niveau, où les classes de l'ontologie représentent les variables présentes dans les traces ;
- ontologie détaillée, où les classes peuvent représenter les variables des traces ainsi que des exemples de valeurs de ces variables dans les traces.

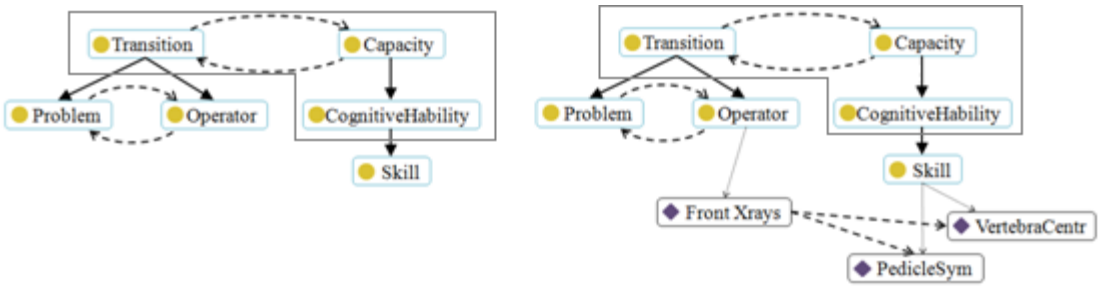


Figure 15 : Deux exemples d'ontologie, de haut niveau à gauche, détaillée à droite. Les ronds désignent les classes de haut niveau et les losanges les classes du niveau détaillé. Les flèches pleines épaisses sont les relations d'héritage, les flèches pointillées les relations objets et les flèches grises fines les relations d'appartenance d'un individu à une classe. La partie dans le cadre gris correspond à l'ontologie par défaut, et les éléments hors du cadre sont les apports du concepteur, qui apportent de la sémantique sur ses traces. « CognitiveHability » correspond au « habilité intellectuelle » sur la Figure 14 selon la traduction de Nkambou et al.

La Figure 15 montre deux exemples d'ontologies, une de haut niveau (gauche) et une détaillée (droite), dans le domaine de la chirurgie orthopédique. Les classes héritent des deux classes Capacity et Transition, exprimant le lien entre éléments observables et connaissances. L'ontologie détaillée montre trois exemples d'individus, un pour la classe Operator (action) et deux pour la classe Skill (connaissance). Les flèches pleines sont les relations d'héritage et les fines flèches grises les relations d'appartenance d'un individu à une classe. Les flèches pointillées expriment les relations objet entre classes ; des relations objet sont proposées par défaut, comme « a une capacité » de Transition à Capacité, et la

[1] Modèle simplifié par rapport à l'original.

relation inverse « a une transition », ou la relation « a pour préalable » de Capacité vers Capacité. Le concepteur peut définir de nouvelles relations pour enrichir la sémantique.

3. Association entre l'ontologie et les traces

Le concepteur crée ensuite une mise en correspondance entre l'ontologie et les traces : une ou plusieurs classes ou individus de l'ontologie sont associés à un ou plusieurs éléments observés dans les traces. Cette mise en correspondance permet de lier aux traces la sémantique apportée par l'ontologie. Plusieurs degrés d'informations expertes sont possibles en fonction : du niveau de l'ontologie (haut niveau ou niveau détaillé) ; de la complétude de l'ontologie ; de la complétude de la mise en correspondance. Notre plateforme ne permet pas d'estimer la quantité d'information experte requise pour instancier les techniques de diagnostic, mais l'approche automatique permet de tester et évaluer plusieurs degrés de complétude de l'ontologie, d'abord de haut niveau puis au niveau détaillé.

Prenons un exemple de traces ci-dessous :

ID Apprenant	Exercice	Phase	Connaissance
St1	Opération L3	Réglage	Pédicule symétrique
St1	Opération L3	Réglage	Disques visibles

Une association entre ces traces et l'ontologie détaillée de la Figure 15 est la suivante :

- La variable *Problem* de l'ontologie est associée à *Exercice* dans les traces.
- La variable *Skill* de l'ontologie est associée à *Connaissance* dans les traces.
- La variable *PedicleSym* de l'ontologie est associée à la valeur *Pédicule symétrique* de la variable *Connaissance* dans les traces.

III. Algorithme semi-automatique d'instanciation des techniques

1. Définition du problème algorithmique

A. Définition et propriétés du problème d'instanciation d'une technique de diagnostic

Nous allons redéfinir le problème de la façon suivante : étant donné une base de traces d'apprenant T et un ensemble de techniques de diagnostic TD = {MD, I} (MD le modèle de diagnostic générique et I l'implémentation) à instancier au domaine du concepteur, le problème est d'instancier le modèle de diagnostic MD, puis de construire l'implémentation I pour MD. Nous considérons T comme une table (une collection de variables), sans contrainte sur son format.

Pour illustration, instancier le modèle du Knowledge tracing pour le domaine du calcul d'aire nécessite d'identifier O et K, que nous avons défini comme O={Problem, Step, Outcome} et K={Knowledge Component} au chapitre 3. Pour le calcul d'aire :

- les Knowledge Component sont les connaissances requises pour calculer des aires (aire d'un rectangle, d'un triangle…) ;
- les étapes (Step) sont les étapes de résolution pour calculer chaque type d'aire, par exemple pour le rectangle, trouver la taille du côté le plus long, puis du côté adjacent, puis appliquer la formule ;
- les problèmes (Problem) sont les exercices, par exemple « calculer l'aire d'un rectangle de longueur X et de largeur Y » ;
- le diagnostic comportemental, ici la variable Outcome, indique si les étapes sont correctement ou incorrectement résolues par les apprenants.

Propriétés

La première propriété à noter est que tous les modèles de diagnostics sont formalisés par un unique modèle associant éléments observables et éléments de connaissances, ainsi que les relations entre ces variables et un ensemble de contraintes. L'algorithme doit donc prendre en compte ces contraintes lors de l'instanciation, afin de ne pas les enfreindre. Par exemple, le Knowledge tracing requiert qu'une et une seule connaissance soit associée à une étape de résolution d'un problème : une instance de Knowledge tracing qui ne respecte pas cette assomption enfreint une contrainte du modèle de diagnostic. L'algorithme d'apprentissage doit donc également être guidé par le modèle de diagnostic et ses contraintes. La seconde propriété est la complexité du problème. Considérons ***m*** le nombre de variables du modèle de diagnostic (MD) et ***n*** le nombre de variables dans les traces (T), avec $m<n$: le nombre d'associations entre les variables de MD et de T est le coefficient binomial : $n! / (m! (n – m)!)$ (si $m>n$ la formule doit être inversée). À noter que des variables des traces peuvent ne pas être utiles pour instancier le modèle de diagnostic, et donc être ignorées. Cette combinatoire ne prend en compte que les variables et pas les relations entre les variables. Nous avons vu qu'un modèle de diagnostic et l'ensemble de ses instances peuvent être définis comme un graphe orienté. Or, l'ensemble des graphes orientés différents qu'il est possible d'obtenir à *m* sommets est $2^{m(m-1)}$. Pour chaque combinaison, il y a donc $2^{m(m-1)}$ instances possibles, correspondant au nombre de graphes différents qu'il est possible d'obtenir en ne modifiant que les arêtes du graphe, donc les relations du modèle de diagnostic. La combinatoire augmente donc à $n!/(m!(n–m)!) * 2^{m(m-1)}$ pour le nombre d'instances possibles d'un modèle de diagnostic à partir des traces. Enfin, le traitement du problème doit être répété pour chaque modèle de diagnostic à instancier dans la plateforme.

Une telle combinatoire explique la difficulté informatique du problème, et le besoin d'élaguer dans ces solutions de manière à trouver une instanciation cohérente du modèle de diagnostic pour le domaine (cohérente en termes d'apprentissage et de modélisation des connaissances). De plus, les traces pouvant être incomplètes pour l'instanciation, il faut un apport externe d'informations (de variables). C'est le rôle de l'ontologie des connaissances du domaine conçue par le concepteur et de son association avec les traces.

Nous pouvons donc redéfinir notre problème de façon plus précise.

Définition

L'**instanciation du modèle de diagnostic MD au domaine** consiste à identifier dans les traces enrichies T et dans l'ontologie des connaissances du domaine O les variables et les domaines des variables du domaine qui vont hériter des variables (Observables et Connaissances) du modèle de diagnostic générique, respectant la spécification et les contraintes du modèle de diagnostic MD.

Enfin, une technique de diagnostic est un couple formé par un modèle de diagnostic et une implémentation. Comme expliqué dans le chapitre 3, les implémentations ont une complexité et un pouvoir expressif différent, une structure différente et des paramètres différents. Or, le résultat de l'algorithme d'apprentissage étant un diagnostic implémenté capable d'inférer le modèle de l'apprenant, l'implémentation va donc influer sur l'apprentissage. D'un point de vue informatique, les algorithmes permettant d'apprendre la structure d'un réseau bayésien ou un ensemble de règles sont différents.

Rappelons qu'il a été défini en section 3 que l'implémentation d'un modèle de diagnostic consiste à transposer le modèle de diagnostic instancié au domaine dans un formalisme calculable.

Définition

L'**instanciation de la technique de diagnostic TD={MD,I} au domaine** consiste à implémenter dans le formalisme calculable I le modèle de diagnostic MD, après que MD ait été instancié au domaine comme défini juste ci-dessus.

B. Heuristique de recherche locale pour résoudre l'explosion combinatoire

Pour répondre au problème posé, notre approche consiste à utiliser une famille d'heuristiques (aussi appelée *métaheuristique*) permettant de construire toute technique de diagnostic générique. La restriction à une seule famille d'heuristique, et si possible à un nombre d'heuristiques le plus restreint, a pour but de limiter les biais liés à la qualité de l'algorithme d'apprentissage. Par exemple, l'usage d'une heuristique connue comme très performante pour l'apprentissage d'un réseau bayésien et d'une heuristique connue comme médiocre pour l'apprentissage de règles de productions induirait un biais lors de la comparaison.

Nous proposons une approche basée sur la méta-stratégie de la recherche locale par score. De façon informelle, un algorithme de recherche locale vise à traiter un problème algorithmique complexe en essayant de trouver la solution optimale de proche en proche. À partir d'une solution initiale (qui peut être obtenue de n'importe quelle manière, y compris aléatoirement), l'algorithme n'explore qu'un sous-ensemble de solutions limité, le voisinage, de manière à trouver une meilleure solution, de remplacer la solution courante par cette nouvelle solution, et d'itérer le processus (Aarts et Lenstra, 2003). L'algorithme tente ainsi de n'explorer qu'une petite partie des solutions. Enfin, un tel algorithme requiert de fixer deux éléments : une fonction d'exploration de l'espace des solutions (i.e. la constitution du

voisinage d'une solution), et une fonction de coût (appelée score) permettant de comparer deux solutions.

Une telle approche est intéressante pour notre problème, car elle permet d'éviter l'explosion combinatoire et elle est applicable à tout type d'implémentation. Or, comme montré ci-dessus, la complexité de notre problème est très élevée, i.e. le nombre de solutions possibles est très grand. Enfin, il nous est possible de guider l'algorithme de recherche locale avec nos informations expertes externes (l'ontologie des connaissances et l'association entre cette ontologie et les traces) et nos contraintes (les contraintes du modèle de diagnostic) en proposant une fonction de coût (score) prenant en compte ces éléments.

Définitions (Aarts et Lenstra, 2003)

Un **problème d'optimisation combinatoire** est défini par un ensemble d'instances du problème et par un problème d'optimisation (soit maximalisation, soit minimalisation).

Une **instance d'un problème d'optimisation combinatoire** est une paire (S, f), où S est l'ensemble des solutions du problème et f est la fonction de coût (aussi appelée score) définie comme $f : S \to \mathbb{R}$. Le problème est de trouver la **solution optimale**, i.e. une solution $s^* \in S$ tel que $f(s^*) <= f(s)$ pour tout $s \in S$. De plus, $f^* = f(s^*)$ dénote le coût optimal, et $S^* = \{i \in S, f(i) = f^*)$ dénote l'ensemble des solutions optimales.

À noter que parler de fonction coût permet de faire abstraction de la nature du problème d'optimisation (optimiser signifie réduire le coût, d'où la définition $f(s^*) <= f(s)$). Du point de vue de l'implémentation, le coût sera évalué différemment selon qu'il s'agisse d'un problème de maximisation ou de minimisation.

Une **fonction de voisinage** est l'association $N : S \to 2^S$, qui définit pour chaque solution $s \in S$ un ensemble N(s) inclus dans S de solutions qui sont proches de s. L'ensemble N(s) est le voisinage de la solution s, et toute solution $j \in N(i)$ est une voisine de i.

Un **algorithme de recherche locale** vise à résoudre un problème d'optimisation combinatoire. Partant d'une solution initiale $s \in S$, l'algorithme procède par exploration du voisinage N(s) de manière à trouver la solution $j^* \in N(i)$ telle que $f(j^*) <= f(j)$ et $f(j^*) <= f(s)$ pour tout $j \in N(i)$. La solution finale d'un algorithme de recherche locale est la solution s^{**} qui est la meilleure solution trouvée par exploration successive des voisinages de chaque solution j trouvée. s^{**} peut être un optimum local, c'est-à-dire la meilleure solution dans un ensemble de voisinages donné, ou la solution optimale s^* du problème.

Pour appliquer une heuristique de recherche locale, nous devons redéfinir notre problème comme un problème d'optimisation combinatoire. C'est ce que nous allons faire dans la partie qui suit.

2. Algorithme d'instanciation d'une technique de diagnostic

A. Notations

Nous rappelons ci-dessous les différentes notations utilisées par la suite :

- T : base de traces d'apprenant enrichies MD : modèle de diagnostic générique
- I : implémentation pour un modèle de diagnostic générique
- O : ontologie des connaissances du domaine conçue par le concepteur
- Map : association entre l'ontologie O et les traces T
- TD : technique de diagnostic générique (TD = {MD,I})
- search() : heuristique de recherche locale
- fS et fS' : scores de l'heuristique de recherche locale (fS' est défini plus bas)
- R : matrice de probabilités pour le calcul du score (définie plus bas)

B. Principe et exemple illustratif

L'algorithme d'instanciation d'une technique de diagnostic TD = {MD,I} aux traces du concepteur fonctionne en trois étapes :

1. recherche de l'instance du modèle de diagnostic (i.e. du sous-ensemble de variables) dans les traces ou classes de l'ontologie maximisant la probabilité de satisfaire *MD*. Par exemple, déterminer quelles variables dans les traces représentent les Knowledge Component pour le Knowledge Tracing. Cette étape est réalisée par une heuristique de recherche locale (définie plus haut comme la combinaison d'un algorithme de recherche de solutions candidates *search()* et d'un score *fS* mesurant la qualité de chaque solution) ;
2. extraction du domaine (i.e. des valeurs possibles) de ces variables, par exemple l'ensemble des Knowledge Components (en géométrie, le domaine de la variable Knowledge Components serait par exemple « Savoir calculer l'aire d'un triangle », « Savoir calculer l'aire d'un rectangle », « Savoir calculer la hauteur d'un triangle »…). Dans l'ontologie, ce sont les classes du niveau détaillé ;
3. apprentissage automatique des paramètres si requis, par exemple les distributions de probabilité d'un réseau bayésien ou les coefficients d'un modèle linéaire.

À la fin de chacune de ces étapes, les résultats suivants sont obtenus :

1. une association entre les variables du modèle de diagnostic MD et les variables des traces et de l'ontologie
2. la technique de diagnostic instanciée au domaine et implémentée avec I
3. la valeur des paramètres de l'implémentation I, uniquement si cette implémentation a des paramètres (par exemple, un réseau bayésien)

Prenons un exemple naïf afin d'illustrer les résultats attendus de ces trois étapes (nous donnerons des exemples concrets et complets plus bas, ainsi que les algorithmes) avant de présenter chaque étape en détail.

Prenons par exemple la technique de diagnostic Knowledge tracing implémentée par un réseau bayésien. Pour rappel (cf. chapitre 3), le modèle de diagnostic du Knowledge tracing est : O={Problem, Step, Outcome}, K={Knowledge Component} et BD={Outcome}. Pour les relations : Problem est lié à Step, Step est lié à Konwldge Component, et Outcome est lié à Knowledge Component. Prenons pour cet exemple des traces en géométrie sur des calculs d'aire :

Student	Topic	Area	Student's action	Skill	Correct
Raleigh	Area	Circle	Find radius	RADIUS	Yes
Raleigh	Area	Circle	Apply formula	CIRCLE AREA	Yes

Sur ces traces, la première étape doit fournir une association entre le modèle de diagnostic et les variables des traces et de l'ontologie (nous ne considérons ici que les traces pour conserver la simplicité de l'exemple). Nous pouvons imaginer l'association suivante pour le Knowledge tracing (il s'agit du résultat de l'étape 1) :

Modèle de diagnostic	Traces
Problem	- Topic - Area
Step	Student's action
Outcome	Correct
Knowledge Component	Skill

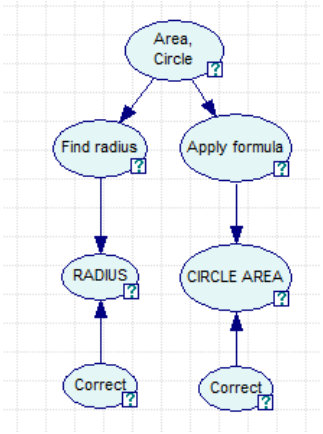
On note que la variable Problem du modèle de diagnostic est associée à deux variables dans les traces (Topic et Area).

L'étape 2 va instancier le modèle de diagnostic au domaine et implémenter le modèle de diagnostic (avec l'implémentation de la technique, ici un réseau bayésien). Il faut donc extraire les valeurs des variables du modèle de diagnostic pour le domaine considéré dans les traces. Ici, le résultat est le suivant :

Modèle de diagnostic	Traces	Valeurs pour le domaine
Problem	- Topic - Area	- {Area, Circle}
Step	Student's action	- Find radius - Apply formula
Outcome	Correct	- Yes
Knowledge Component	Skill	- RADIUS - CIRCLE AREA

Le résultat de l'étape 2 est donc l'implémentation sous forme de réseau bayésien de cette instanciation du modèle de diagnostic au domaine. Ce résultat dépend naturellement entièrement de l'implémentation de la technique à construire (cf. le chapitre 3 au sujet de la transposition d'un modèle de diagnostic vers une implémentation). Sur notre exemple, imaginons que la manière de transposer le modèle de diagnostic vers le réseau bayésien soit le suivant : toutes les valeurs deviennent des nœuds du réseau, sauf la variable Outcome (le

diagnostic comportemental) où l'on garde le nom de la variable (Correct), et les nœuds sont reliés en fonction des relations définies dans le modèle de diagnostic. Pour notre exemple avec un réseau bayésien, on peut donc imaginer le résultat suivant pour l'étape 2 :



Enfin, l'étape 3 vise à apprendre des paramètres si besoin. Là encore, ces paramètres dépendent de l'implémentation ; pour un réseau bayésien, il s'agit des probabilités du réseau. Pour simplifier notre exemple, prenons un seul paramètre comme résultat de l'étape 3 : la probabilité que la variable RADIUS (de type Knowledge Component, donc connaissance) soit sue sachant que l'apprenant est dans l'étape « Find radius » et que Correct=yes est de 0,8.

Après l'étape 3, le résultat final obtenu est donc : une technique de diagnostic instanciée au domaine et implémentée.

Nous allons présenter ci-dessous chacune de ces trois étapes en détail.

C. Recherche locale pour l'association entre le modèle de diagnostic et les traces (étape 1)

Ensemble des solutions

La première étape est l'instanciation du modèle de diagnostic au domaine à partir des traces et de l'ontologie conçue par le concepteur. L'algorithme que nous proposons résout le problème d'optimisation combinatoire défini dans la section III. 1 au moyen d'une approche par recherche locale par score (également définie section III. 1). L'algorithme se compose d'une méthode d'exploration des solutions *search()*, et un score *fS* mesurant la qualité de chaque solution. Avant de présenter la méthode *search()* et le score *fS*, nous devons définir pour notre problème ce qu'est une solution.

Définitions

L'**ensemble S des solutions** est l'ensemble des instances possibles du modèle de diagnostic à partir des traces d'apprenants et de l'ontologie.

Le **voisinage d'une solution** $s \in S$ est l'ensemble des instances du modèle de diagnostic qui ne diffèrent que d'une relation ou d'une variable avec s.

Nous appelons **solution candidate** ou **solution courante** la solution s du problème dont l'algorithme de recherche local est en train d'explorer le voisinage.

Nous disons qu'une solution s est **meilleure** qu'une solution s' si le coût f(s)<f(s').

Dans notre méthode, nous réutilisons un algorithme d'exploration et des scores existants pour chaque implémentation, et proposons un second score unique *fS'* pondérant *fS*. *fS'* prend en compte le modèle de diagnostic *MD* en cours d'apprentissage, ainsi que l'ontologie O et la mise en correspondance de l'ontologie vers les traces Map. Le score *fS'* apporte un biais ou une contrainte sur le score *fS* afin que le modèle statistique appris satisfasse le modèle de diagnostic *MD*, donc soit une instance de la technique générique de diagnostic considérée.

Définition

Une solution candidate **satisfait** le modèle de diagnostic si et seulement si l'ensemble des contraintes du modèle de diagnostic sont satisfaites.

Un exemple typique est la contrainte du Knowledge tracing qui impose qu'une et une seule connaissance soit associée à une étape de résolution d'un exercice. Toute solution candidate qui associerait plusieurs connaissances à une étape ne satisferait pas le modèle.

Score

Comme détaillé plus haut, nous utilisons deux scores visant à évaluer une solution candidate : un score statistique fS et un score de pertinence fS'.

fS est un score statistique ou probabiliste. Ces scores sont liés à une implémentation particulière et visent à évaluer à partir des seules traces la pertinence de la solution courante. L'utilité de ces scores est basée sur l'hypothèse que les traces (ici les traces d'apprenants enrichies) contiennent suffisamment de connaissance pour extraire un modèle (ici le modèle de diagnostic) pertinent. C'est une hypothèse que nous avons posée lors de la définition de nos questions de recherche et qui a été confortée par l'étude de l'état de l'art. Il existe dans la littérature un grand nombre de scores pour l'apprentissage automatique, basés généralement sur la fonction de vraisemblance, l'entropie, la probabilité *a posteriori* (Aarts et Lenstra, 2003).. Ces scores (qui seront présentés ci-après) n'étant pas originaux, nous avons réutilisé des scores classiques dans la littérature et mis à disposition dans des librairies. *A priori*, tout score défini comme une fonction de coût (définie plus haut partie III. 1) en optimisation peut être utilisé dans notre méthode.

Nous proposons ensuite un score de pertinence fS' qui va « biaiser » le score fS en prenant en compte : l'ontologie, l'association entre les traces et l'ontologie, et les contraintes du modèle de diagnostic. Ce score se définit, similairement à fS, comme une fonction de coût associée à un problème d'optimisation combinatoire.

Définition

Soit S l'ensemble des solutions candidates, fS' : S->$\mathbb{R}$.

Nous définissons une matrice *R* dont les lignes sont les variables de *MD* et les colonnes les variables des traces. *R(md,t)* est la probabilité que la variable *t* de *T* soit équivalente à la variable *md* de *MD*. La matrice est initialisée à 1 si *t* a été liée par le concepteur à une classe de l'ontologie *O* (via la mise en correspondance), 0,5 sinon. Ces probabilités servent à calculer le score final *fS(s)*fS'(s)* pour une solution candidate s, puis sont mises à jour à chaque itération de l'algorithme d'exploration. Dans le cas d'une ontologie détaillée, les variables de l'ontologie non associées à une variable des traces sont également ajoutées dans les colonnes de la matrice *R*. Le calcul de f*S'* est défini ci-dessous.

Soit s la solution candidate, et *N=(μ, σ2)* une gaussienne centrée en 1 (*μ = 1*), σ2 étant fixé de sorte que *N(0.5) = 1*, et *N(R(t, md))* le score correspondant à la probabilité R(t, md). Alors le score fS' est l'image de N du produit des *R(t, md)* pour tout *t* de *T* et *md* de *MD* de la solution candidate.

$$fS'(s) = N\left(\prod_t \prod_{md} R(t,\ md)\right)$$

Ce score est calculé pour l'ensemble du voisinage de la solution candidate, et comparé à la solution candidate, de manière à trouver la meilleure solution. Nous modifions toutefois chaque score avec un paramètre *a* calculé de la sorte :

$a \times fS(s_i) \times fS'(0{,}8) = fS(s_j) \times fS'(0{,}5)$ avec s_i la solution ayant le plus mauvais score dans le voisinage et s_j le meilleur score dans le voisinage.

Cette contrainte permet de vérifier la propriété suivante : un score fS' pour une valeur supérieure ou égale à 0,8 sera toujours privilégié par rapport à un score aléatoire (0,5), quel que soit f*S*. Cette propriété vise à privilégier le score fS' lorsque la probabilité de pertinence est suffisamment élevée, et que le score statistique fS donne des résultats contradictoires. Cette contrainte vise donc à traiter les problèmes où les scores fS et fS' divergent en donnant l*a priori*té à fS' si la probabilité de pertinence est forte. Les valeurs 0,8 et 0,5 ont été fixées empiriquement pour cette étude.

Le meilleur score est celui maximisant *a×fS×fS'* et tel que la solution candidate satisfasse *MD* pour une itération de l'algorithme.

La matrice *R* est finalement mise à jour en considérant : le meilleur score f*S**, la satisfaction *Ss* du modèle de diagnostic *MD*, et la compatibilité avec l'ontologie *O*. On cherche donc à inférer Pr(*R(md,t) | R(md,t), fs*, Ss, O)*, que l'on simplifie ainsi :

- Pr(*R(md,t) | R(md,t), fs*)* si *Ss* et *O* sont vrais (MD et O sont satisfaits par la solution)
- Pas d'inférence sinon

Recherche locale dans l'espace des solutions candidates

Étant donné le score, l'heuristique de recherche locale a trois fonctions : générer une solution candidate initiale, générer le voisinage d'une solution candidate, et itérer en sélectionnant dans le voisinage la nouvelle solution candidate (ou terminer l'algorithme). Pour cette première plateforme, nous avons utilisé la solution dite Hill-Climbing (Langley et

al., 1987) qui génère la solution candidate initiale aléatoirement, calcule le voisinage en ne changeant qu'un élément (relation ou variable) de la solution candidate, et itère en sélectionnant toujours dans le voisinage la solution ayant le meilleur score. Comme pour le score fS, nous avons utilisé des librairies existantes implémentant Hill-Climbing.

L'algorithme d'exploration locale est relancé jusqu'à convergence de *R*. Pour deux exécutions successives donnant en résultat deux matrices R1 et R2, la convergence de R est définie comme suit :

- Soit, | r2-r1 | <= δ pour tout r1 ∈ R1, r2 ∈ R2, avec δ une loi de décroissance $\delta_t=0{,}99\delta_{t-1}$ et $\delta_0=1$ (état dit « stable »)
- Soit r2 > 0,9 ∨ r2 < 0,1 pour tout r2 ∈ R2 (état dit « terminal »)

En résultat, la matrice *R* associe à chaque variable du modèle de diagnostic une ou plusieurs variables des traces ayant la plus forte probabilité d'être équivalentes.

Par exemple, prenons une matrice de résultat quelconque, avec md1, md2 ∈ MD et t1, t2, t3 ∈ T :

	t1	t2	t3
md1	0,2	0,1	**0,85**
md2	**0,9**	**0,9**	0,3

Table 2 : Exemple abstrait de matrice R : les lignes sont les variables du modèle de diagnostic (md1, md2) et les colonnes les variables des traces (t1, t2, t3). La case R(md1, t1) est par exemple la probabilité que la variable md1 soit représentée par la variable t1 des traces.

L'association résultante de cette matrice est la suivante :

```
md1->{t3}
md2->{t1, t2}
```

Dans le cas où aucune solution candidate ne satisfait le modèle de diagnostic, ou qu'une variable du modèle de diagnostic ne peut être associée à aucune variable des traces ou de l'ontologie avec une probabilité supérieure à 0.5, la construction de la technique de diagnostic échoue.

D. Instanciation et implémentation (étape 2)

Construction du graphe d'instanciation

À ce stade, nous avons une association entre les variables du modèle de diagnostic et les variables des traces et de l'ontologie. La seconde étape instancie et implémente le modèle de diagnostic au domaine à partir des résultats de la première étape. Par exemple, si le modèle de diagnostic possède une variable *Action* de type *Observable*, l'instanciation de cette variable au domaine de l'addition décimale est l'identification de toutes les actions possibles dans ce domaine, telles que « additionner une colonne » ou « poser une retenue ». Grâce au résultat de l'heuristique de recherche locale, cette instanciation se fait en extrayant les informations contenues dans les traces et l'ontologie. Concrètement, il s'agit

de créer dans le modèle générique de diagnostic MD de nouvelles variables pour chaque valeur possible d'instanciation des variables de MD.

Pour ce faire, l'algorithme de cette seconde étape est le suivant :

1. Duplication : Duplique l'ontologie du modèle de diagnostic, que nous appellerons MD'
2. Extraction des variables : Extrait toutes les valeurs possibles de chaque variable (i.e. leur domaine) de T et de O associée à une variable du modèle de diagnostic. On obtient pour chaque variable de MD' le vecteur du domaine *val={val1…valn}* des valeurs différentes qu'elles peuvent prendre.
3. Création de variables : Pour chaque variable v_{MD} de MD', crée une nouvelle variable héritant de v_{MD} pour chaque valeur du domaine de v_{MD}
4. Création des relations : Extrait toutes les relations entre les variables créées au point 3, et crée la relation équivalente dans MD'
5. Résolution des conflits : Résout les conflits en cas de contraintes de MD' enfreintes lors des points 3 et 4. Il en résulte ce que nous nommons le **graphe d'instanciation** du modèle de diagnostic MD.
6. Implémentation : Transpose MD' selon l'implémentation liée à la technique de diagnostic. Les informations nécessaires à cette transposition sont fournies par l'implémentation, comme détaillé dans le chapitre 3.

Il en résulte la technique de diagnostic instanciée au domaine.

Les étapes 2 à 5 permettent d'obtenir le graphe d'instanciation censé respecter toutes les contraintes du modèle de diagnostic. S'il est impossible d'obtenir un tel graphe, la construction de la technique de diagnostic échoue. La nature de ce graphe dépend des contraintes, et de l'association entre les variables du modèle de diagnostic et les variables des traces et de l'ontologie. Si nous notons cette association *g*, nous pouvons dire que *g* est un morphisme au sens des graphes (Fournier, 2011) entre les deux ensembles de variables MD (modèle de diagnostic) et celles de T (traces) et O (ontologie) :

$\forall(v1, v2) \in MD, (g(v1), g(v2)) \in T \cup O$ avec v1, v2 deux variables de MD reliées par une arête.

Si *g* est injective (toute variable de MD a au plus une image dans T), alors chaque valeur des variables *v* de MD peut être traitée indépendamment lors de la construction du graphe d'instanciation.

Si *g* n'est pas injective, alors il est possible qu'une variable de MD soit liée à plusieurs variables dans les traces ou l'ontologie. Dans ce cas, il est nécessaire de créer des relations entre les valeurs d'une variable *v* de MD.

Prenons un exemple naïf pour le cas non injectif. Soit une base de trace T composée de trois variables T={t1, t2, t3} et un modèle de diagnostic MD composé de deux variables MD={v1, v2}. Posons que v1 est liée à t1 et t2, et v2 est liée à t3.

Sur la base de traces suivantes :

t1	t2	t3
a	c	e
a	d	e
b	d	f

Appliquons l'algorithme à partir de l'étape 2 :

- Étape 2 (extraction des variables) : Le domaine de t1 est {a, b}, et celui de t2 {c, d, e}
- Étape 3 (création des variables) : Une variable est créée dans le modèle de diagnostic MD pour chaque valeur du domaine des variables des traces (Figure 16)
- Étape 4 (création des relations) : Les relations entre les variables sont ajoutées (Figure 17)

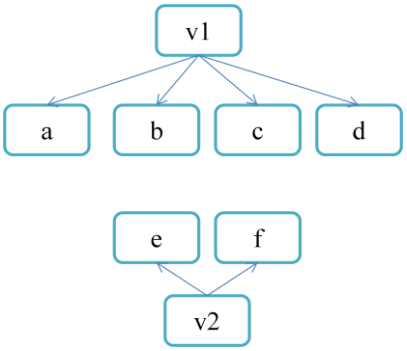


Figure 16 : Graphe d'instanciation au point 3 de l'algorithme.

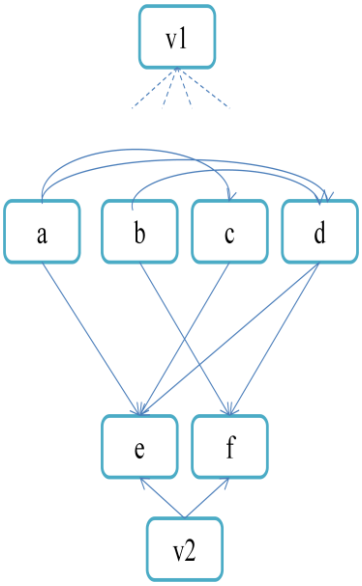


Figure 17 : Graphe d'instanciation au point 4 de l'algorithme, avec les relations entre variables. Les arcs pointillés en haut sont les associations de v1 aux variables a, b, c et d comme sur la Figure 16, ce afin de conserver la lisibilité de la figure.

En théorie, le graphe d'instanciation correspond à la définition d'un hypergraphe (Berge, 1987), c'est-à-dire un graphe où une arête relie un nombre quelconque de sommets. Par exemple le graphe ci-dessus peut être reformulé comme un hypergraphe G=(V, E) de la façon suivante (nous simplifions en ne représentant pas v1 et v2) :

- Sommets : V={ a, b, c, d, e, f}
- Arêtes : E={e1, e2, e3} avec e1={a, c, e}, e2={a, c, e} et e3={b, d, f}

La représentation graphique est donnée ci-dessous Figure 18 (e1 est représenté par la ligne discontinue noire à droite, e2 par la ligne pointillée bleue au centre et e3 par la ligne continue rouge à gauche) :

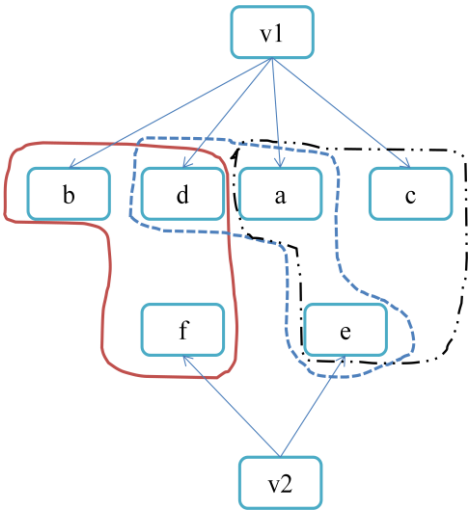


Figure 18 : Graphe d'instanciation complet représenté sous forme d'hypergraphe, avec trois arêtes.

Une telle représentation permet de conserver intuitivement toute l'information contenue dans les traces, ici une table, tout en conservant les propriétés requises dans notre algorithme (le morphisme). Elle évite de plus les confusions : par exemple sur la Figure 17, il est impossible de dissocier les relations *a->e* et *a,d->e*.

Enfin, à l'étape 5 (résolution des conflits), toutes les contraintes du modèle de diagnostic sont évaluées (pour rappel, les contraintes portent sur les cardinalités du modèle de diagnostic, comme défini chapitre 3). Si une contrainte est enfreinte, alors le graphe d'instanciation est incorrect. Par exemple, le Knowledge tracing impose comme contrainte qu'une étape de résolution d'un problème ne peut être liée qu'à une et une seule connaissance. Si le graphe d'instanciation associe une étape à deux connaissances différentes, alors il y a violation des contraintes du modèle de diagnostic Knowledge tracing.

Pour résoudre un problème de contrainte enfreinte, l'étape 5 de l'algorithme fonctionne comme suit. Posons de façon générale qu'une contrainte *c1* est enfreinte par un ensemble de *n* sommets {v1…vn} du graphe d'instanciation (n>0) :

- Si une variable dans les traces correspondant à un des sommets de {v1…vn} a été associée à l'ontologie des traces conçue par le concepteur, alors recherche dans l'ontologie d'une relation permettant de satisfaire la contrainte *c1* sans enfreindre une autre contrainte
- Sinon, test de plusieurs solutions en modifiant sur le graphe d'instanciation : soit en fusionnant deux ou plusieurs sommets, soit en scindant un sommet en au moins deux nouveaux sommets, de façon à satisfaire la contrainte *c1* sans enfreindre une autre contrainte. Le score *fS* est recalculé pour chacune de ces solutions et celle ayant le score maximal est retenue

Nous donnerons plus bas dans la section IV. 2 un exemple de résolution d'une contrainte enfreinte.

Définition

Pour un modèle de diagnostic donné MD, un **graphe d'instanciation** est un hypergraphe GI=(V, E), dont MD est un sous-graphe. Les sommets de V non inclus dans MD sont des variables dépendantes du domaine d'apprentissage (instanciées) et les arêtes E non incluses dans MD sont les relations entre ces variables dépendantes du domaine.

Implémentation de la technique

Rappelons que l'objectif est d'assister un concepteur de diagnostic pour la construction d'une technique de diagnostic, qui est l'association entre un modèle de diagnostic et une implémentation. Le graphe d'instanciation décrit l'instanciation du modèle de diagnostic, i.e. spécifie toutes les variables et relations pour réaliser le diagnostic des connaissances dans le domaine du concepteur. Il reste donc à implémenter la technique, ce que nous avons défini au chapitre 3 comme dépendant de chaque implémentation.

E. Apprentissage des paramètres (étape 3)

Principe

La troisième étape utilise des algorithmes de la littérature pour apprendre les paramètres des techniques, si requis. Ces paramètres sont définis par l'implémentation de la technique : par exemple, les paramètres d'un réseau bayésien sont les tables de probabilités jointes associées à chaque nœud, la structure étant donnée par le modèle de diagnostic MD, et pour une régression, ce sont les coefficients de régression associés à chaque facteur. Certaines implémentations n'ont pas de paramètres (par exemple un système à base de règles IF/THEN).

Ces paramètres ont un impact fort sur l'inférence, donc il s'agit de trouver des paramètres valides. Pour illustrer l'impact des paramètres, prenons l'exemple naïf d'une chaîne de Markov à une variable : *ma_connaissance* abrégée MC, pouvant prendre pour valeur « apprise » et « non apprise ». Les paramètres d'une chaîne de Markov sont les probabilités de transitions entre les états de la variable, du temps *t-1* au temps *t*. Prenons pour l'exemple deux paramétrages différents donnés Table 3.

	MC_t =apprise	MC_t=non apprise
MC_{t-1}= apprise	0,9	0,1
MC_{t-1}=non apprise	0,8	0,2

	MC_t =apprise	MC_t=non apprise
MC_{t-1}= apprise	0,6	0,4
MC_{t-1}=non apprise	0,55	0,45

Table 3 : Exemples de deux distributions de paramètres différentes pour la même chaîne de Markov à deux états : connaissance apprise ou non apprise. Les lignes représentent l'état de la chaîne de Markov au temps *t-1* et les colonnes l'état au temps *t*. Les cases spécifient les probabilités de transition de l'état de la chaîne de Markov du temps *t-1* au temps *t*. Par exemple, la probabilité que la connaissance soit apprise au temps *t* sachant que la connaissance était non apprise au temps *t-1* est de 0,8 sur la distribution de gauche, tandis qu'elle est de 0,55 sur la distribution de droite.

Le calcul de la limite de la loi de probabilité (c'est-à-dire la valeur vers laquelle converge la chaîne de Markov quand *t* tend vers l'infini) donne pour la première table une probabilité d'apprendre la connaissance de 0.89 et pour la seconde 0.58, soit un résultat très différent dû ici seulement aux paramètres.

Algorithme EM

Nous utilisons pour apprendre les paramètres des implémentations de nos techniques de diagnostic l'algorithme EM (Expectation-Maximization)(Dempster et al., 1977) applicable aux modèles probabilistes et aux régressions. L'algorithme EM vise à maximiser la vraisemblance L des paramètres θ, à partir de traces T (définie dans nos travaux comme une collection de variables) :

$$\theta^* = \mathrm{argmax}_\theta \, L(\theta|T)$$

Il présente l'avantage de prendre en compte des traces incomplètes (i.e. la valeur de certaines variables des traces n'a parfois pas été observée dans les traces) ainsi que des paramètres dits « cachés » qui vont permettre de simplifier la maximisation de la vraisemblance. Pour ce faire, l'hypothèse est qu'il existe une base de traces d'apprenant T observée et une hypothétique base de traces (ou latentes, non observées) T2={T, X} cachée. La maximisation de la vraisemblance s'écrit alors :

$$\theta^* = \mathrm{argmax}_\theta \, L(\theta|T, X)$$

Pour résoudre ce problème, l'algorithme EM se présente sous la forme d'une itération sur deux étapes. Soit *t* l'itération en cours, les deux étapes sont :

- L'étape d'estimation (E) de l'espérance mathématique de la vraisemblance en fonction de T, de X et des paramètres θ^{t-1} trouvés à l'itération précédente. Il s'agit d'estimer la vraisemblance de l'ensemble des traces T2 (i.e. observées et non observées)
- L'étape de maximisation (M) de cette espérance pour obtenir θ^t

L'étape E se définit comme suit :

$Q(\theta \mid \theta^{t-1}) = E[\log Pr(T, X \mid \theta) \mid T, \theta^{t-1}]$ avec Q l'espérance de la vraisemblance sur T2

L'étape M :

$$\theta^t = \mathrm{argmax}_\theta\, Q(\theta|\theta^{t-1})$$

On note que l'algorithme vise en fait à maximiser le logarithme de la vraisemblance (log-vraisemblance), ce qui simplifie les expressions.

L'algorithme itère sur les étapes E et M jusqu'à convergence (i.e. jusqu'à ce qu'il ne soit plus possible de trouver une vraisemblance θ^t supérieure à θ^{t-1}).

L'application de l'algorithme EM dépend entièrement de chaque implémentation considérée, notamment car il faut redéfinir la log-vraisemblance et les paramètres θ, et donc les équations E et M. Les implémentations que nous utilisons étant classiques, ces réécritures ont déjà été faites dans la littérature : voir les démonstrations de Naïm et al. pour les réseaux bayésiens, de Baum et al. pour les modèles de Markov cachés, de Wilkinson et Rogers pour les régressions linéaires (Baum et al., 1970; Heckerman, 2008; Naïm, 2007; Wilkinson et Rogers, 1973). Nous réutilisons des librairies proposant pour chaque implémentation l'algorithme EM (cf. ci-dessous).

F. Librairies

Nous avons utilisé pour l'implémentation de l'algorithme entier (de l'étape 1 à la 3) plusieurs librairies nous permettant de réutiliser ce qui existe déjà.

Pour l'étape 1 de l'algorithme, nous avons utilisé et adapté pour l'heuristique de recherche locale et le calcul des scores statistiques *fS* des librairies R : bnlearn[2], igraph[3] et RJava[4]. On trouve dans les librairies bnlearn et igraph plusieurs heuristiques de recherche locale, dont Hill Climbing que nous avons utilisée (comme décrit plus haut) et de nombreux scores. RJava permet le dialogue entre du code R et du code JAVA, sachant que notre plateforme générale d'assistance à la construction et à la comparaison est en JAVA (cf. chapitre 6 pour les détails plus techniques).

Pour l'étape 2, nous avons utilisé pour l'implémentation des techniques les package R nlme[5], igraph et RJava pour les régressions linéaires, la librairie SMILE[6] de Microsoft pour les réseaux bayésiens et Drools[7] pour les règles. En effet, pour rappel, l'étape 2 dépend de l'implémentation de la technique en cours de construction. Ces librairies proposent toutes des fonctions pour construire une implémentation et des fonctions d'inférence qui permettront de diagnostiquer les connaissances des apprenants à partir de leurs traces.

Pour l'étape 3 (apprentissage des paramètres), nous avons utilisé l'algorithme EM développé dans SMILE pour les réseaux bayésiens et dans nlme pour les régressions linéaires, ainsi que

[2] http://www.bnlearn.com/
[3] http://cran.r-project.org/web/packages/igraph/index.html
[4] http://www.rforge.net/rJava/
[5] http://cran.r-project.org/web/packages/nlme/index.html
[6] http://genie.sis.pitt.edu/
[7] http://www.jboss.org/drools/

l'algorithme BNT-Brute-Force de Baker (Baker et al., 2010) pour les modèles de Markov cachés du Knowledge tracing.

IV. Exemple d'application de l'algorithme

Nous montrons un exemple d'exécution de l'algorithme sur le domaine de la chirurgie orthopédique via les traces de TELEOS (Vadcard et Luengo, 2004). Un exemple des traces est donné Table 4, où *Vertebra* représente l'exercice, *Action* une interaction avec l'EIAH, *Representation* désigne les types d'expression des problèmes et des actions, *Situation variables* est la règle ayant permis de calculer le diagnostic comportemental, *Behavioral diagnostic* est le diagnostic comportemental, et *Control* indique quelle connaissance permet de vérifier la validité du résultat de l'action. Ces traces, tout comme TELEOS en général, sont inspirées du modèle cKc dont dérive le modèle de diagnostic Control-based.

L'algorithme requiert une ontologie partielle des connaissances du domaine ainsi qu'une association entre cette ontologie et les traces de TELEOS, dont nous donnons un échantillon Table 4. Pour cet exemple, nous allons utiliser une ontologie simple donnée Figure 19 pour TELEOS : nous y représentons deux variables observables (nommées *Problem* et Ope*rator*) et une variable de connaissance (nommée *Skill*). L'association entre les traces et cette ontologie proposée pour TELEOS est donnée sur la Figure 20.

Student	Vert-ebra	Date	Action	Control	Represen-tation	Situation variables	Behavioral diagostic
Baptiste	T12	12/05 9 :12	Front X-ray	Vertebra centered	X-ray	Dist_RF_centerVert	Correct
Baptiste	T12	12/05 9 :12	Front X-ray	Pedicles symetric	X-ray	Dist_RF_symetryPedic	Incorrect
Baptiste	T12	12/05 9 :14	Profile X-ray	Interverte braldisc	X-ray	Dist_RP_Interdisc	Incorrect
Baptiste	L3	12/05 9 :14	Profile X-ray	Interverte braldisc	X-ray	Dist_RP_Interdisc	Correct

Table 4 : Traces utilisées dans les exemples de ce chapitre, dans le domaine de la chirurgie orthopédique.

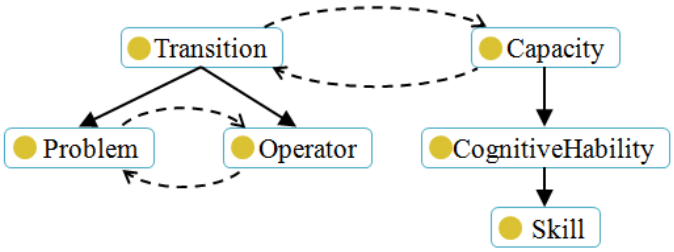
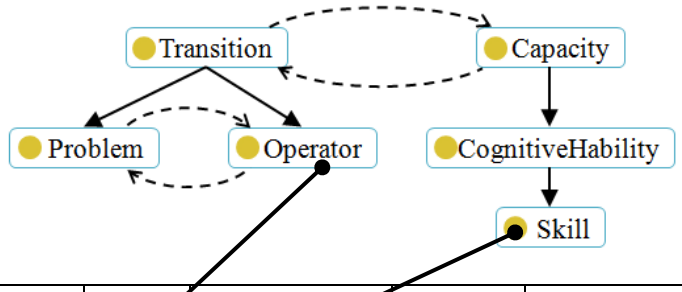


Figure 19 : Ontologie de haut niveau utilisé dans les exemples de ce chapitre.



Student	Vert-ebra	Date	Action	Control	Represen-tation	Situation variables	Behavioral diagostic
Baptiste	T12	12/05 9 :12	Front X-ray	Vertebra centered	X-ray	Dist_RF_centerVert	Correct
Baptiste	T12	12/05 9 :12	Front X-ray	Pedicles symetric	X-ray	Dist_RF_symetryPedic	InCorrect
Baptiste	T12	12/05 9 :14	Profile X-ray	Intervertebraldisc	X-ray	Dist_RP_Interdisc	InCorrect
Baptiste	L3	12/05 9 :14	Profile X-ray	Intervertebraldisc	X-ray	Dist_RP_Interdisc	Correct

Figure 20 : Association entre les traces en l'ontologie utilisées dans les exemples de ce chapitre.

Nous allons montrer l'exemple de l'application de l'algorithme pour trois techniques différentes : Control-based (implémenté par un réseau bayésien dynamique), Knowledge tracing (implémenté par des modèles de Markov caché), et Constraint-based (implémenté par des contraintes, i.e. des règles IF/THEN). Pour rappel, les modèles de diagnostic de ces techniques sont décrits dans le chapitre 3.

Afin de bien identifier les noms des variables, nous indiquerons dans les exemples en indices l'ensemble auquel elles appartiennent. Par exemple pour une variable X :

- X_{MD} est la variable X du modèle de diagnostic MD
- X_o est la variable X de l'ontologie conçue par le concepteur O
- X_T est la variable X des traces T

1. Control-based

La matrice *R* est initialisée comme montré Table 5, où les lignes représentent les variables du modèle de diagnostic *MD* et les colonnes les variables des traces *T*. $R(Control_T, Control_{MD})$ est initialisé à 1 car héritant de la classe Capacity dans l'ontologie. Or, le modèle de diagnostic du Control-based ne comporte qu'une seule variable de type connaissance, $Control_{MD}$. En revanche pour les transitions (observées), il n'est pas possible à ce stade d'identifier dans les traces les variables $Operator_{MD}$ et $Problem_{MD}$.

	$Vertebra_T$	**$Action_T$**	**$Representation_T$**	**$Control_T$**	**$Situation\ variables_T$**
$Problem_{MD}$	0.5	0.5	0.5	0	0.5
$Operator_{MD}$	0.5	0.5	0.5	0	0.5
$Register_{MD}$	0.5	0.5	0.5	0	0.5
$Control_{MD}$	0	0	0	1	0
$Situation\ variables_{MD}$	0.5	0.5	0.5	0	0.5

Table 5 : Initialisation de la matrice *R* pour le score. Les lignes sont les variables du modèle de diagnostic et les colonnes les variables des traces.

À noter que, pour simplifier l'exemple dans un souci de clarté, nous ne représentons pas dans la matrice toutes les variables des traces (notamment Student et Date).

L'algorithme génère les premiers candidats en partant d'un modèle à une variable ; prenons comme solution candidate initiale la variable $Vertebra_T$ assimilée à la première variable du modèle de diagnostic, $Problem_{MD}$,i.e. {$Vertebra_T$=$Problem_{MD}$}. Le voisinage de cette solution initiale est le suivant (notation : « variable de T=variable de MD ») :

{$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Operator_{MD}$ }

{$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Register_{MD}$ }

{$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Control_{MD}$ }

{$Vertebra_T$=$Problem_{MD}$; $Representation_T$=$Operator_{MD}$ }

{$Vertebra_T$=$Problem_{MD}$; Representationv=$Register_{MD}$ }

{$Vertebra_T$=$Problem_{MD}$; $Control_T$=$Control_{MD}$ }

…

{$Vertebra_T$=$Problem_{MD}$; Situation variables$_T$=Situation variables$_{MD}$ }

Sélectionnons les trois solutions suivantes :

1. {$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Control_{MD}$ }
2. {$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Operator_{MD}$ }
3. {$Vertebra_T$=$Problem_{MD}$; $Control_T$=$Control_{MD}$ }

L'application du calcul du score *S'* à partir de la matrice R indiquée ci-dessus donne respectivement :

1. 0.5×0 = 0
2. 0.5×0.5 = 0.25
3. 0.5×1 = 0.5

Les scores *S'* de chaque solution sont indiqués sur la gaussienne Figure 21 : l'axe des abscisses représente les probabilités issues de la matrice *R* et les ordonnées la valeur du score *S'*.

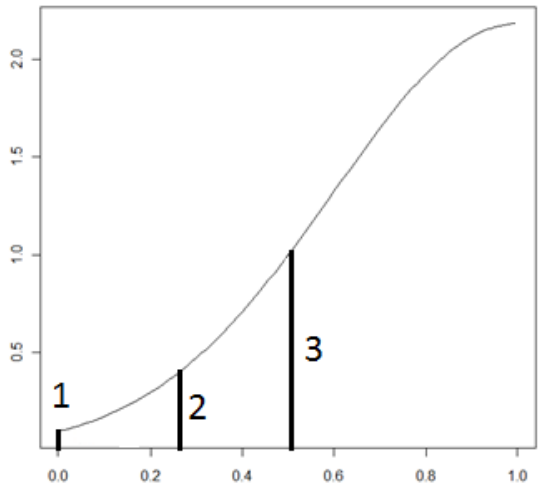


Figure 21 : Exemple de calcul du score fS' sur l'exemple. En abscisse, les valeurs de la matrice R. Les chiffres 1, 2 et 3 correspondent aux solutions candidates données dans l'exemple. L'ordonnée donne le résultat du score pour les solutions candidates.

Parmi ces trois solutions, la première ne satisfait pas le modèle de diagnostic MD car $Action_T$ a été déclarée comme observable dans l'ontologie (hérite de la classe Transition). $Action_T$ ne peut donc être associée qu'à une variable observable dans MD, ce que n'est pas $Control_{MD}$.

La matrice est ensuite mise à jour (exemple Table 6). La probabilité $R(Vertebra_T, Problem_{MD})$ augmente le plus car le score est le plus élevé pour cette solution tout en satisfaisant le modèle de diagnostic.

	$Vertebra_T$	**$Action_T$**	**$Representation_T$**	**$Control_T$**	**Situation $variables_T$**
$Problem_{MD}$	0.62	0.48	0.45	0.01	0.43
$Operator_{MD}$	0.5	0.5	0.5	0	0.5
$Register_{MD}$	0.5	0.5	0.5	0	0.5
$Control_{MD}$	0	0	0	1	0
Situation $variables_{MD}$	0.5	0.5	0.5	0	0.5

Table 6 : Mise à jour de la matrice *R* après la première itération de l'algorithme.

La matrice finale obtenue est donnée Table 7 (les probabilités les plus élevées sont en gras).

	$Vertebra_T$	**$Action_T$**	**$Representation_T$**	**$Control_T$**	**Situation $variables_T$**
$Problem_{MD}$	**0.88**	0.58	0.43	0.06	0.41
$Operator_{MD}$	0.49	**0.92**	0.38	0.22	0.39
$Register_{MD}$	0.52	0.54	**0.71**	0.12	0.56
$Control_{MD}$	0	0	0	**0.98**	0
Situation $variables_{MD}$	0.46	0.57	0.59	0.09	**0.76**

Table 7 : Matrice finale retournée par l'algorithme.

La solution qui en dérive est :

{$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Operator_{MD}$; $Representation_T$=$Register_{MD}$; $Control_T$=$Control_{MD}$; Situation $variables_T$=Situation $variables_{MD}$ }

La seconde étape de l'algorithme consiste à extraire les valeurs des variables du modèle de diagnostic de manière à l'implémenter. Dans l'extrait de traces donné ci-dessus, les valeurs de chaque variable dans les traces sont :

- $Vertebra_T$: T12, L3
- $Action_T$: Front X-ray, Profile X-ray
- $Representation_T$: X-ray
- $Control_T$: Vertebra centered, pedicles symetric, Intervetebraldisc
- Situation $variables_T$: Dist_RF_centerVert, Dist_RF_symetryPedic, Dist_RP_Interdisc

L'algorithme duplique le modèle de diagnostic et crée une classe pour chaque valeur héritant d'une des variables du modèle de diagnostic. Le résultat présenté sous forme de graphe (graphe d'instanciation) est le suivant :

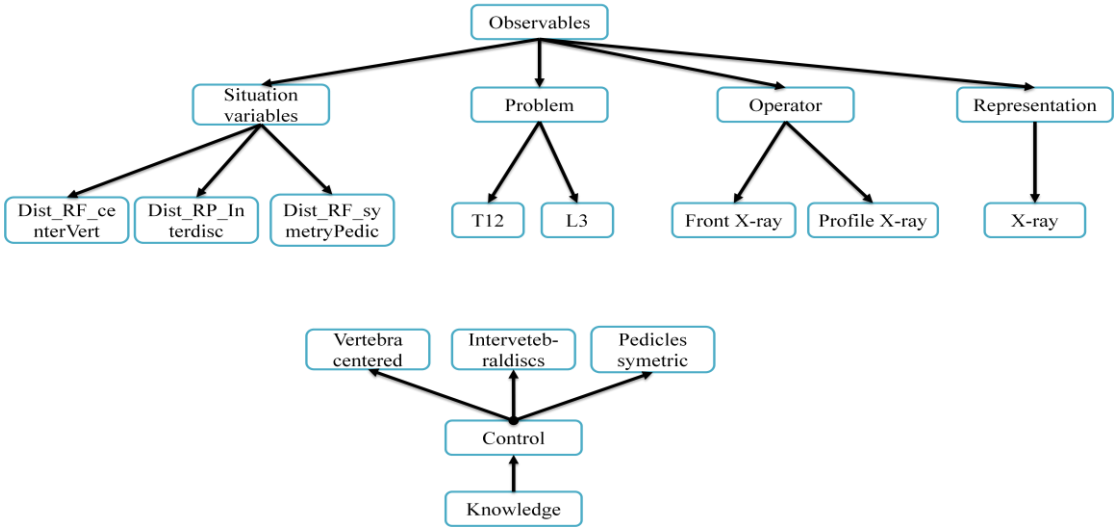


Figure 22 : Graphe d'instanciation de l'exemple, avec les valeurs des variables extraites des traces qui héritent des variables génériques du modèle de diagnostic.

Les relations sont extraites des traces. Prenons l'exemple des variables $Control_T$ et $Action_T$: une relation existe pour chaque cooccurrence d'une valeur de $Control_T$ et d'une valeur de $Action_T$ dans une trace. Dans notre extrait de traces, nous avons trois relations entre :

- Front X-ray et Vertebra centered
- Front X-ray et Pedicles symetric
- Profile X-ray et Intervetebraldisc

Soit trois relations de type Observables vers Connaissances (OtoK). Aucune de ces relations n'enfreint une contrainte du modèle de diagnostic, donc elles sont ajoutées dans le graphe d'instanciation ci-dessus (les relations sont représentées seulement pour les variables Control et Operator) :

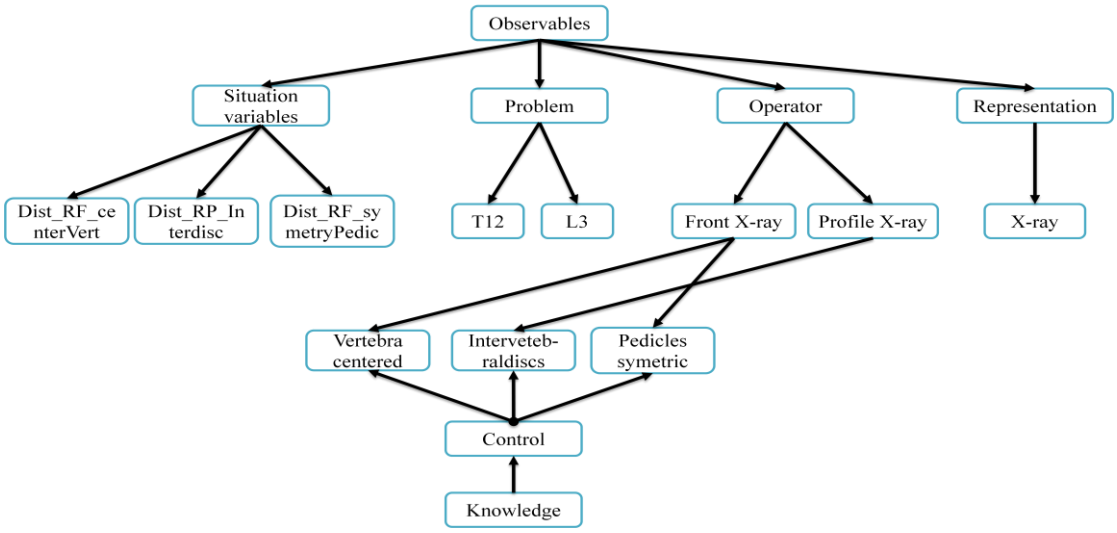


Figure 23 : Graphe d'instanciation de l'exemple, où sont montrées les relations entre les variables héritant de Operator et celles héritant de Control. Ces relations sont extraites des traces.

Le même procédé est répété pour toutes les relations du modèle de diagnostic Control-based, de manière à obtenir le graphe d'instanciation complet.

Pour l'implémentation finale via un réseau bayésien, il reste à implémenter le graphe d'instanciation ci-dessus et à utiliser l'algorithme EM pour apprendre les paramètres du réseau.

L'exemple d'instanciation du Control-based est intuitif car les traces ont été collectées en vue d'une implémentation de ce modèle de diagnostic. Nous allons maintenant donner un exemple d'application sur un autre modèle de diagnostic, le Knowledge tracing, implémenté par des modèles de Markov cachés.

2. Knowledge tracing

Comme pour le Control-based, le modèle de diagnostic du Knowledge tracing est donné dans le chapitre 3, et nous utilisons les mêmes traces avec la même ontologie.

La matrice *R* est initialisée comme montré Table 8 : seules les lignes changent avec les variables du Knowledge tracing. $R(KC_{MD}, Control_T)$ est initialisé à 1 car $Control_T$ est associée à la classe Capacity dans l'ontologie. Or, le modèle de diagnostic du Knowledge tracing ne comporte qu'une seule variable de type connaissance, KC_{MD}.

	Vertebra$_T$	**Action$_T$**	**Representation$_T$**	**Control$_T$**	**Situation variables$_T$**
Problem$_{MD}$	0.5	0.5	0.5	0	0.5
Step$_{MD}$	0.5	0.5	0.5	0	0.5
KC$_{MD}$	0	0	0	1	0

Table 8 : Initialisation de la matrice.

Le calcul du voisinage des solutions candidates et du score *S'* s'effectue de la même façon que pour le Control-based. La matrice finale obtenue donne la solution suivante :

{$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Step_{MD}$; $Control_T$=KC_{MD} }

La seconde étape de l'algorithme consiste à extraire les valeurs des variables du modèle de diagnostic de manière à l'implémenter. Dans l'extrait de traces donné ci-dessus, les valeurs de chaque variable dans les traces sont pour rappel :

- $Vertebra_T$: T12, L3
- $Action_T$: Front X-ray, Profile X-ray
- $Control_T$: Vertebra centered, pedicles symetric, Intervetebraldisc

Le graphe d'instanciation (simplifié sans les relations) du Knowledge tracing dans cet exemple est donc le suivant :

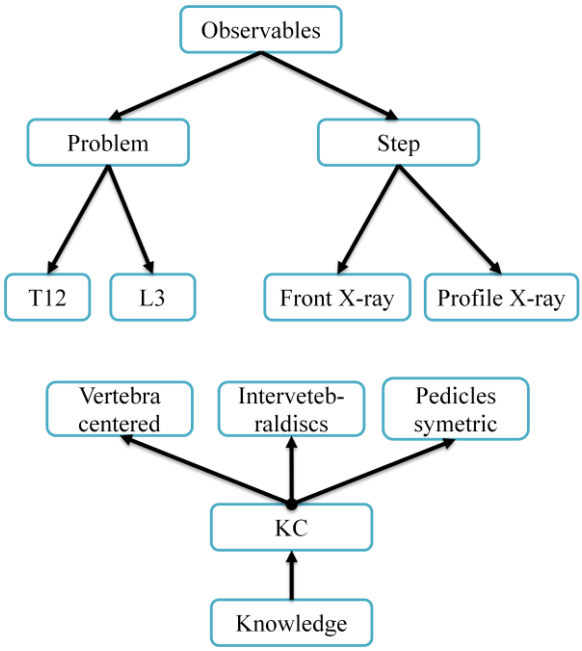


Figure 24 : Graphe d'instanciation sans les relations.

Vient enfin l'extraction des relations. Ici, le Knowledge tracing diffère du Control-based, car ce modèle impose une contrainte sur les relations entre les variables $Step_{MD}$ et KC_{MD} : une étape ($Step_{MD}$) ne peut être liée qu'à un et un seul KC. Or, dans l'exemple présent, les relations entre $Action_T$ et $Control_T$ enfreignent cette contrainte : Front X-ray est associée à Vertebra centered et à Pedicles symetric.

La résolution de ce problème peut être effectuée soit par agrégation des variables Vertebra centered et Pedicles symetric, soit par dissociation de la variable Front X-ray en deux nouvelles variables. Ces solutions sont illustrées Figure 25.

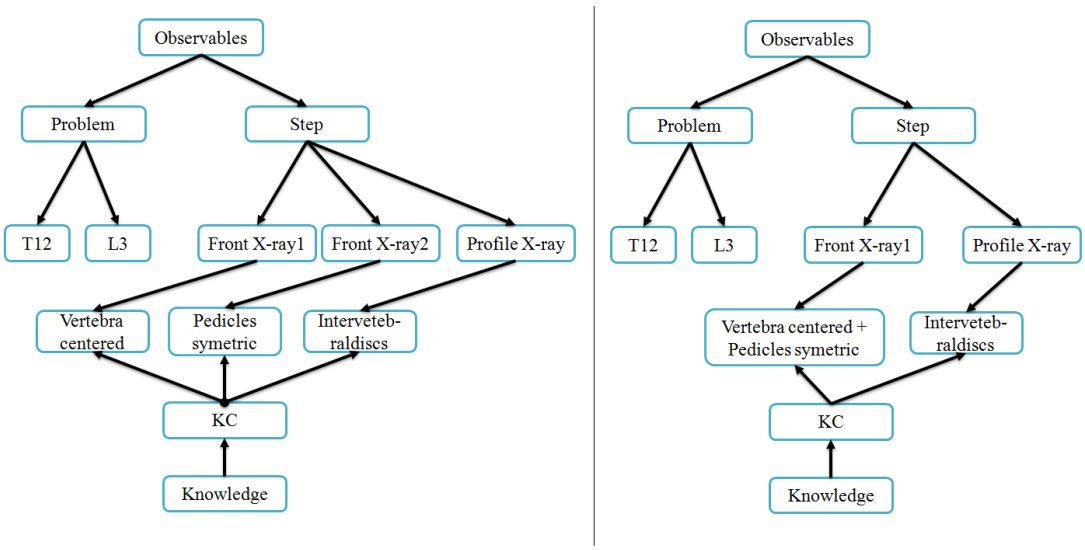


Figure 25 : Exemple de relations possibles entre les variables du domaine héritant de Step et celles héritant de KC. À gauche, la variable *Front X-ray* est dupliquée et liée à un seul KC. À droite, les variables *Vertebra centered* et *Pedicles symetric* sont fusionnées pour être liées à une seule variable héritant de *Step*.

Ces deux solutions respectent la contrainte. Le choix peut être guidé par le concepteur, via les liens entre variables du domaine dans l'ontologie des connaissances O. Dans cet exemple, ce n'est pas le cas, donc le choix se fait en calculant pour chacune leur score *fS*, comme expliqué dans l'algorithme. Pour information dans notre expérimentation sur les traces de TELEOS (chapitre 7), la solution de droite (fusion des deux connaissances) est celle qui obtient le meilleur score.

3. Constraint-based

Nous avons vu deux exemples d'application de l'algorithme pour des modèles de diagnostic ayant une structure proche d'un graphe. Nous allons maintenant étudier le cas du Constraint-based, implémenté par des règles IF/THEN. Il n'y a aucune contrainte sur les cardinalités ou les variables dans le Constraint-based, ce qui rend ce modèle de diagnostic très libre.

La matrice *R* est initialisée selon le même principe que les deux précédents exemples, comme montré Table 9.

	Vertebra$_T$	**Action$_T$**	**Representation$_T$**	**Control$_T$**	**Situation variables$_T$**
Cr$_{MD}$	0.5	0.5	0.5	0	0.5
Cs$_{MD}$	0	0	0	1	0

Table 9 : Initialisation de la matrice.

Le modèle de diagnostic ne comportant que deux variables, les voisinages sont facilement énumérables. Par exemple, le voisinage de la solution initiale associant la variable $Vertebra_T$ et Cr_{MD} est:

{$Vertebra_T$=Cr_{MD} ; $Action_T$=Cs_{MD} }
{ $Vertebra_T$=Cr_{MD} ; $Action_T$=Cs_{MD} }
{ $Vertebra_T$=Cr_{MD} ; $Representation_T$=Cs_{MD} }
{ $Vertebra_T$=Cr_{MD} ; $Control_T$=Cs_{MD} }
{$Vertebra_T$=$Problem_{MD}$; Situation $variables_T$=Cs_{MD} }

Toutes les solutions de ce voisinage respectent le modèle de diagnostic. La matrice finale obtenue est la suivante :

	$Vertebra_T$	**$Action_T$**	**$Representation_T$**	**$Control_T$**	**Situation $variables_T$**
Cr_{MD}	**0.76**	**0.71**	**0.74**	0.32	**0.73**
Cs_{MD}	0.16	0.34	0.17	**0.94**	0.19

Cette matrice donne la solution suivante : { {$Vertebra_T$, $Action_T$, $Representation_T$, Situation $variables_T$}=Cr_{MD} ; $Control_T$=Cs_{MD} }

Le graphe d'instanciation étant complexe à représenter, nous ne donnons sur la Figure 26 que la relation issue de la première trace de notre base de trace (que nous avons copiée-collée dans la Figure 26 en haut).

Baptiste	T12	9 :12	Front X-ray	Vertebra centered	X-ray	Dist_RF_centerVert	Correct

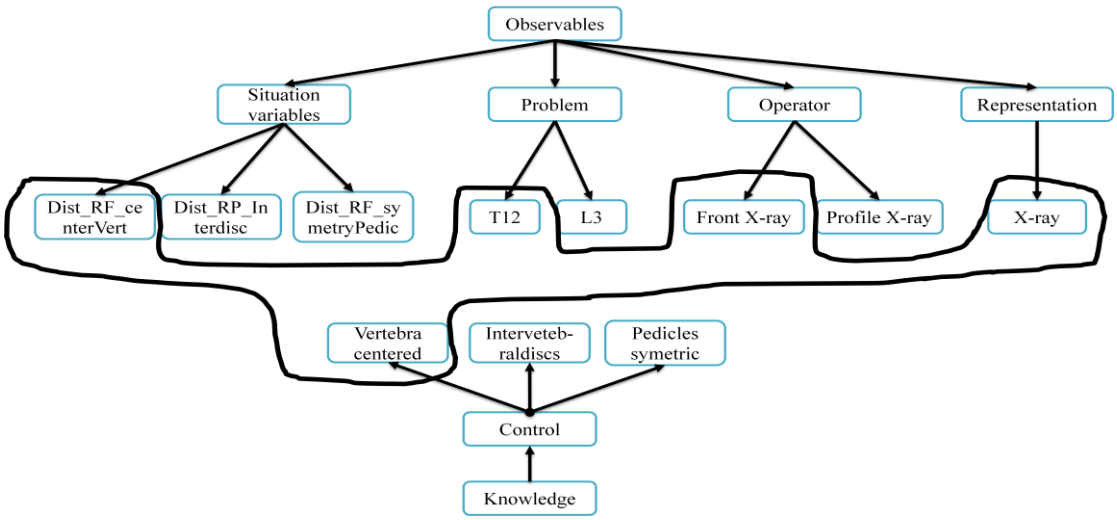
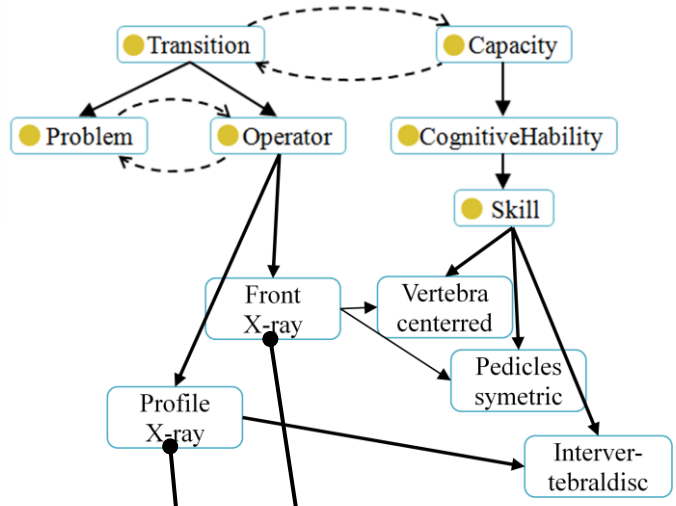


Figure 26 : En haut : rappel de la trace utilisée dans cet exemple. En bas, graphe d'instanciation avec l'arête issue de la trace, reliant *T12*, *Front X-ray*, *X-ray* et *Dist_RF_centervert* à *Vertebra centered*.

Sur la Figure 26, la trace contient les valeurs suivantes : Front X-ray pour le problème, Vertebra centered pour la connaissance, X-ray pour les registres de représentation et Dist_RF_centerVert pour la variable de situation. Chacune de ces valeurs est ajoutée sous la forme d'un sommet dans le graphe d'instanciation, et elles sont reliées par une arête (le trait noir continu sur la figure). Nous avons en effet défini dans le cadre général que le graphe d'instanciation est un hypergraphe où les arêtes peuvent relier plus de deux sommets (cf. section III. 2.D). L'arête représentée Figure 26 est en fait une contrainte spécifiant que : SI Problem=T12 et Operator=Front X-ray et Representation=X-ray et Situation variable=Dist_RF_centerVert ALORS Control=VertebraCentered. De façon informelle, cette contrainte spécifie que l'apprenant doit mobiliser la connaissance impliquant de vérifier le centrage de la vertèbre sur la radio de face, dans le cas où il prend une radio de face d'une vertèbre T12. À la fin, l'algorithme retourne l'ensemble des contraintes correspondant à l'ensemble des arêtes différentes.

4. Ontologie détaillée

Prenons pour terminer les mêmes traces, mais sans la variable $Control_T$. Ces traces sont donc incomplètes car elles ne contiennent aucune information sur les connaissances à diagnostiquer. Le concepteur peut dans ce cas spécifier dans une ontologie détaillée ces connaissances. Nous donnons l'ontologie détaillée : trois variables *Skill* (connaissances) sont spécifiées ainsi que les deux variables *Operator* (observable) dans l'ontologie. En associant les deux variables *Operator* Figure 27 de l'ontologie aux valeurs de la variable $Action_T$ des traces, le concepteur apporte la sémantique sur les traces nécessaire pour construire les techniques de diagnostic.



Student	Vert-ebra	Date	Action	Represen-tation	Situation variables	Behavioral diagostic
Baptiste	T12	12/05 9:12	Front X-ray	X-ray	Dist_RF_centerVert	Correct
Baptiste	T12	12/05 9:12	Front X-ray	X-ray	Dist_RF_symetryPedic	InCorrect
Baptiste	T12	12/05 9:14	Profile X-ray	X-ray	Dist_RP_Interdisc	InCorrect
Baptiste	L3	12/05 9:14	Profile X-ray	X-ray	Dist_RP_Interdisc	Correct

Figure 27 : Association entre l'ontologie détaillée et les traces.

Nous allons donc présenter ci-dessous les étapes ou phases de l'algorithme qui change avec l'usage d'une ontologie détaillée. Reprenons par exemple la construction de la technique de diagnostic Control-based+réseau bayésien. Dans l'étape 1 de l'algorithme, la matrice R est légèrement différente : la variable $Control_T$ des traces disparaît, et est remplacée par la variable $Skill_O$ de l'ontologie détaillée (Table 10). Comme nous l'avons défini dans l'algorithme, toutes les variables détaillées non associées aux traces sont ajoutées dans les colonnes de la matrice R, puisqu'il s'agit du moyen pour le concepteur d'apporter des informations sémantiques non observées dans les traces. Concernant l'initialisation de la matrice, R($Control_{MD}$, $Skill_O$) est initialisée à 1 car le concepteur a spécifié dans l'ontologie que $Skill_O$ est une variable de type connaissance.

	$Vertebra_T$	**$Action_T$**	**$Representation_T$**	**$Skill_O$**	**Situation $variables_T$**
$Problem_{MD}$	0.5	0.5	0.5	0	0.5
$Operator_{MD}$	0.5	0.5	0.5	0	0.5
$Register_{MD}$	0.5	0.5	0.5	0	0.5
$Control_{MD}$	0	0	0	1	0
Situation $variables_{MD}$	0.5	0.5	0.5	0	0.5

Table 10 : Initialisation de la matrice *R* pour le score. Les lignes sont les variables du modèle de diagnostic et les colonnes les variables des traces, plus $Skill_O$ qui est la variable de l'ontologie.

La solution qui en dérive est :

{$Vertebra_T$=$Problem_{MD}$; $Action_T$=$Operator_{MD}$; $Representation_T$=$Register_{MD}$; $Control_T$=$Skill_O$; Situation $variables_T$=Situation $variables_{MD}$ }

La seconde étape de l'algorithme consiste à extraire les valeurs des variables du modèle de diagnostic de manière à l'implémenter. Dans l'extrait de traces donné, les valeurs de chaque variable dans les traces sont :

- $Vertebra_T$: T12, L3
- $Action_T$: Front X-ray, Profile X-ray
- $Representation_T$: X-ray
- Situation $variables_T$: Dist_RF_centerVert, Dist_RF_symetryPedic, Dist_RP_Interdisc

Ici, il faut toutefois rajouter les valeurs de la variable $Skill_O$ qui ont été spécifiées dans l'ontologie par le concepteur :

- $Skill_O$: Vertebra centered, pedicles symetric, Intervetebraldisc

À partir de cette étape, la suite de l'algorithme est exactement la même. Dans cet exemple, nous voyons donc que l'ontologie ne sert pas qu'à décrire la sémantique des traces (en termes de variables observables et de variables de connaissances, ainsi que les relations entre les variables) pour guider la construction des techniques de diagnostic, mais également à compléter les traces en cas d'informations non observées. En particulier dans le domaine des EIAH, les traces n'incluent pas toujours les connaissances du domaine.

V. Caractéristiques et limites

1. Propriétés de l'algorithme d'apprentissage

La terminaison de notre algorithme est garantie par le fait que :

- L'algorithme de recherche locale accepte une solution candidate si et seulement si son score est meilleur que le score de la solution courante. Le score tend donc vers le score optimal (ou un optimum local).
- La matrice R converge soit dans un état stable, soit dans un état final, comme détaillé plus haut.

La complexité de l'algorithme est inconnue. En effet, il n'existe pas de preuve pour la complexité des algorithmes de recherche locale, qui sont par conséquent rangés dans la classe de complexité PLS (*Polynomial Local Search*)(Aarts et Lenstra, 2003). Il n'est ni prouvé que les problèmes LPS peuvent se résoudre en temps polynomial, ni prouvé qu'ils ne le peuvent pas. La raison de cette indécision est l'impossibilité de borner le nombre de solutions candidates explorées, qui peut être au pire la combinatoire du problème d'optimisation (si toutes les solutions sont candidates par ordre décroissant de score). En pratique, les temps d'exécution obtenus sont très courts du fait de la simplicité (en termes de nombre de variables) des modèles de diagnostic, de l'ordre de la seconde pour chaque technique de diagnostic.

L'algorithme fournit la preuve que la technique instanciée respecte toutes les contraintes du modèle de diagnostic: toute violation d'une contrainte du modèle de diagnostic est rejetée lors de la seconde phase d'instanciation.

2. Limites

Du point de vue algorithmique, un algorithme de recherche locale est une heuristique : il n'est pas garanti de trouver la solution optimale, mais un optimum local. Il en va de même pour l'algorithme EM d'apprentissage des paramètres.

Dans notre algorithme, il est possible d'associer plusieurs variables des traces à une variable du modèle de diagnostic. Ce point est nécessaire car nous n'imposons pas de contraintes fortes sur le format des traces : une variable d'un modèle de diagnostic peut donc être décrite par plusieurs variables dans les traces. Toutefois, nous ne prenons en compte ce point qu'à la fin de l'étape 1 de l'algorithme, et non lors de l'étape 1 dans l'heuristique de recherche locale. L'hypothèse est que l'étape 1 procède par renforcement et qu'un nombre suffisant de solutions candidates étudiées pour mettre à jour la matrice R permet d'associer les différentes variables des traces au modèle de diagnostic. Une amélioration possible serait toutefois de vérifier cette hypothèse en prenant en compte l'association de plusieurs variables des traces à une variable du modèle de diagnostic dans l'heuristique de recherche locale, lors de la génération du voisinage.

Du point de vue du diagnostic des connaissances, il est impossible d'évaluer la validité d'une technique de diagnostic instanciée. En effet, la notion de diagnostic optimal n'existe pas

formellement, comme l'a montré Self (Self, 1990). Similairement, l'algorithme ne garantit pas la plausibilité des techniques construites, qui peuvent donc inclure des erreurs du point de vue du domaine (exemple naïf, associer la connaissance « Calcul de l'aire d'un triangle » pour le problème du calcul de l'aire d'un rectangle). Toutefois, une erreur ne peut être qu'extraite des traces, dont la qualité a un impact déterminant. Les techniques de diagnostic sont donc évaluées empiriquement, comme vu dans l'état de l'art, par des expérimentations, par des experts ou par comparaison. Nous reviendrons sur ce sujet dans le chapitre Évaluation.

VI. Retour sur les scénarios

1. Premier cas d'usage

Dans le premier cas d'usage, Léa est une experte en EIAH, mais pas en diagnostic des connaissances, qui souhaite intégrer un diagnostic des connaissances dans son EIAH en chirurgie orthopédique. Elle souhaite utiliser le résultat du diagnostic pour obtenir des profils d'apprenant à la fin des séances de travail et donner des aides adaptées à chaque apprenant.

Léa consulte d'abord la spécification de l'ontologie des traces nécessaire pour la construction des diagnostics. Elle réalise ensuite une ontologie décrivant les traces. Afin d'obtenir des profils d'apprenant qui lui permettent de donner des aides adaptées, elle décide de lister explicitement dans l'ontologie les connaissances du domaine qui ont été identifiées auparavant. Il s'agit donc d'une ontologie détaillée. Elle associe l'ontologie à ses traces et lance l'algorithme d'apprentissage. Elle passe pour finir à l'étape de comparaison des techniques de diagnostic construites.

L'exemple donné dans la section IV de ce chapitre porte sur l'application de l'algorithme d'apprentissage sur des traces en chirurgie orthopédique. Il correspond à ce cas d'usage.

2. Second cas d'usage

Dans le second cas d'usage, Nadia est une chercheuse dans le domaine du diagnostic des connaissances qui souhaite comparer une nouvelle technique de diagnostic avec l'existant au moyen de traces enrichies dans deux domaines différents. Elle souhaite obtenir des premiers résultats d'évaluation publiables.

Nadia réalise une première ontologie pour le domaine de l'algèbre. Elle l'associe à ses traces et lance l'algorithme. Elle a choisi d'utiliser une ontologie de haut niveau, qui demande moins de travail et qu'elle adapte pour son second domaine, la géométrie. Elle associe cette seconde ontologie à ses traces et lance l'algorithme une seconde fois. Ensuite, elle pourra comparer les résultats des instanciations de sa technique de diagnostic avec les autres techniques de diagnostic construites (cette partie sera expliquée dans le chapitre suivant).

VII. Conclusion

Nous avons proposé une méthode d'assistance à la construction de techniques de diagnostic qui fonctionne selon un processus en trois étapes :

1. Instanciation du modèle de diagnostic en construisant à partir des traces et d'une ontologie des traces conçue par le concepteur une association entre chaque variable du modèle de diagnostic et une ou plusieurs variables dans les traces ou dans l'ontologie. Afin de résoudre le problème du grand nombre d'associations possibles, la prise en compte des informations expertes spécifiées dans l'ontologie et le respect des contraintes du modèle de diagnostic, nous proposons d'utiliser une heuristique locale par score, où le score est une combinaison entre un score statistique calculé à partir des traces et un score « biaisé » que nous proposons, et qui prend en compte l'ontologie du concepteur et les contraintes du modèle de diagnostic.
2. Instanciation et implémentation de la technique de diagnostic à partir du résultat de l'étape 1. Ici, le modèle de diagnostic est complété par toutes les valeurs du domaine qui sont contenues dans les traces et dans l'ontologie du concepteur. Puis le modèle de diagnostic ainsi instancié est transposé dans l'implémentation de la technique, comme expliqué au chapitre 3.
3. Apprentissage des paramètres de l'implémentation à partir des traces, dans le cas où l'implémentation possède des paramètres. Nous utilisons essentiellement l'algorithme EM pour nos différentes implémentations.

Ce processus possède les caractéristiques suivantes :

- Chaque technique n'est pas construite indépendamment par le concepteur, mais via un algorithme d'apprentissage semi-automatique.
- Les techniques construites respectent le modèle de diagnostic et ses contraintes. L'assistance est donc guidée par le modèle de diagnostic à construire.
- Il n'y a pas de contrainte forte sur le format des traces, qui peuvent de plus être incomplètes. Le concepteur apporte la sémantique des traces et les éléments non observés dans les traces grâce à une ontologie. L'assistance est donc également guidée par ces informations sémantiques expertes.

La méthode d'apprentissage permet donc de construire toute technique de diagnostic dont le modèle de diagnostic peut être décrit par notre formalisation (c'est le cas par exemple du Knowledge tracing, du Constraint-based et du Control-based, comme montré chapitre 3). La prise en compte du modèle de diagnostic et de l'ontologie doit en outre permettre de renforcer l'interprétabilité des techniques construites.

Le prix pour la construction de différentes techniques de diagnostic sans contrainte sur le format des traces est que l'algorithme d'apprentissage utilisé est semi-automatique : le concepteur a un travail d'apport de sémantique à faire. Nous faisons toutefois l'hypothèse que l'investissement requis de la part du concepteur est rentable de par la construction sans coût en programmation de plusieurs techniques de diagnostic, et la possibilité de réinvestir

ce travail pour la construction de techniques de diagnostic pour de futurs domaines et de futurs EIAH. Nous pouvons également caractériser plus précisément les concepteurs qui peuvent utiliser notre plateforme : le concepteur doit être en mesure de pouvoir concevoir une ontologie qui décrit des éléments clés du diagnostic des connaissances, i.e. les connaissances et les éléments observables permettant de les diagnostiquer. En revanche, le concepteur n'a pas besoin de savoir implémenter une technique de diagnostic, ni de connaître toutes les techniques construites par la plateforme.

Chapitre 5 : Assistance à la comparaison de techniques de diagnostic

Sommaire

I. Aperçu 102

II. Critères de comparaisons 102

1. Définition 102
2. Caractéristiques des critères de comparaison 103
   - A. Nature du calcul 103
   - B. Type du résultat 104
   - C. But 105
   - D. Documentation d'un critère 105
3. Interprétation des critères de comparaison 105
4. Exemples de critères 106
   - A. Précision de la prédiction 106
   - B. Racine de l'erreur quadratique moyenne 106
   - C. Akaike Information Criterion 107
   - D. Bayesian Information Criterion 107
   - E. Area Under the Curve ROC 107
   - F. Temps d'exécution 108
   - G. Complexité de la technique 108
   - H. Entropie de Shannon 109
   - I. Corrélation avec un diagnostic externe 109
   - J. Sensibilité aux données manquantes 110
   - K. Inférence immédiate 110
   - L. Impact sur le choix d'un type de rétroaction 110
5. Description d'un critère : impact sur le choix d'un type d'aide 111
   - A. Principe 111
   - B. Prérequis 111
   - C. Algorithme 111

III. Exemples 113
1. Précision de la prédiction 114
2. Corrélation avec un diagnostic externe 115
3. Impact sur le choix d'un type d'aide 116
IV. Retour sur les cas d'usage 117
1. Premier cas d'usage 117
2. Second cas d'usage 117
V. Conclusion 118

I. Aperçu

Dans ce chapitre, nous allons présenter nos contributions pour la comparaison de techniques de diagnostic des connaissances. Dans la partie II, nous définissons la notion de critères de comparaison et proposons douze critères. Dans la partie III, nous montrons trois exemples d'application de critères. Dans la partie IV, nous revenons sur nos cas d'usage avant de conclure.

Le processus de comparaison s'appuie sur les propositions précédentes : il s'agit de comparer les résultats des techniques de diagnostic instanciées pour le domaine d'un EIAH. Les résultats des techniques sont calculés dans un EIAH à partir des traces d'apprenants. On souhaite donc obtenir, pour des traces d'apprenants donnés, les résultats de plusieurs techniques de diagnostic afin de comparer ces résultats. Cependant, les traces d'apprenants étant également utilisées pour la construction semi-automatique, Il convient, pour éviter tout biais, d'introduire une notion de validation croisée : une partie des traces est donc utilisée pour l'apprentissage automatique, le reste des traces pour tester et comparer les techniques (nous reviendrons dans le chapitre 6 sur la validation croisée).

Nous proposons de définir des critères de comparaison, c'est-à-dire des critères d'évaluation communs à un ensemble d'outils informatiques (ici, des techniques de diagnostic). Nous avons vu quelques exemples dans l'état de l'art. Leur nature peut être interne (portant sur les aspects du système liés à l'implémentation, comme la structure des techniques de diagnostic) ou externe (portant sur ce qui est mesurable à partir des sorties, comme l'efficacité, les erreurs détectées...).

II. Critères de comparaisons

1. Définition

Soit une technique de diagnostic TD et une base de traces d'apprenants enrichies T.

Définition

> Un **critère de comparaison de techniques de diagnostic** est un processus prenant en entrée TD et le résultat de la technique de diagnostic à un instant donné sur les traces T, et retournant en sortie un « résultat de comparaison ». Un critère est **calculable** par un algorithme.

La notion de « résultat de comparaison » est laissée volontairement très générale, car un critère de comparaison, ici entendu comme calculable, donc résultant d'un algorithme, n'a aucune limitation autre que technique.

Un critère de comparaison s'applique à une ou plusieurs techniques de diagnostic, c'est-à-dire qu'il peut dépendre du modèle de diagnostic et de l'implémentation. Par exemple, un critère peut ne s'intéresser qu'aux techniques implémentées par une méthode probabiliste. Le corollaire est donc qu'un critère ne doit pas nécessairement s'appliquer à toutes les techniques de diagnostics possibles ; il a un domaine de validité (cf. le chapitre 6 pour les questions d'architecture logicielle).

Définition

> Le **domaine de validité** d'un critère de comparaison est l'ensemble des techniques pour lesquelles le critère peut être calculé.

Par défaut, le domaine de validité n'est pas restreint, c'est-à-dire que le critère s'applique à toute technique de diagnostic qu'il est possible de modéliser selon la formalisation F_{MD} du modèle de diagnostic générique, décrite dans le chapitre 3.

Il est possible d'étendre le domaine de validité de façon ad hoc pour une technique de diagnostic donnée grâce à un **add-on** permettant le calcul du critère. Prenons l'exemple d'un critère qui s'applique aux méthodes probabilistes, i.e. le résultat du diagnostic des connaissances se présente sous la forme d'une probabilité. Une technique implémentée par des règles IF/THEN n'appartient pas au domaine de validité, car non probabiliste. Il est cependant possible de considérer des règles IF/THEN comme probabilistes en comptant par exemple pour une règle son taux de satisfaction (i.e. le nombre de traces pour lesquelles la règle est vraie sur le nombre total de traces). Un tel taux compris entre 0 et 1 peut être comparable à une probabilité, selon ce qu'évalue le critère. Ces add-on étant essentiellement un problème technique, nous reviendrons dessus dans le chapitre suivant présentant l'architecture logicielle de notre plateforme de comparaison.

2. Caractéristiques des critères de comparaison

Nous pouvons néanmoins identifier des caractéristiques communes à des ensembles de critères.

A. Nature du calcul

Comme première caractéristique, nous définissons la nature du processus de calcul du critère comme simple ou complexe. Un critère simple est un test ou une mesure calculée

directement sur les résultats de la technique de diagnostic. Tout autre calcul nécessitant d'autres informations ou plusieurs tests et mesures, est dite complexe. Les critères complexes peuvent nécessiter d'accéder à d'autres informations, via des add-on d'application des critères (cf. plus haut).

Définitions

Un critère de comparaison **simple** est une fonction *fc* prenant en entrée TD et le modèle de l'apprenant inféré par TD sur les traces T, et retournant en sortie un élément d'un ensemble de sortie ES. La définition de fc est fc : {TD, T} -> ES.

Un critère de comparaison **complexe** est une fonction *fc* muni d'un ensemble d'entrées EE et d'un ensemble de sortie ES. La définition de fc est fc : EE^n -> ES.

L'ensemble de sortie ES est défini ci-dessous.

La nature du calcul permet de générer partiellement le domaine de validité : tout critère simple a un domaine de validité non restreint, i.e. il peut être calculé pour toute technique de diagnostic. En revanche, pour les critères complexes, le domaine de validité, s'il est restreint, doit être déclaré par le concepteur, soit manuellement (en énumérant les techniques du domaine de validité), soit automatiquement dans le code du critère (par exemple en restreignant le domaine de validité en fonction des implémentations des techniques ou de leurs méta-données).

B. Type du résultat

Comme seconde caractéristique, le type de l'ensemble de sortie (donc de l'ensemble de résultats du critère) dépend de la nature du critère. Pour un critère simple, il peut être numérique ou symbolique. Par exemple, un ensemble de sortie numérique est l'ensemble des réels (donc le critère retourne un nombre), un ensemble de sortie symbolique est une valeur parmi la collection de valeurs possibles C={fortement plausible, faiblement plausible, non plausible} (donc le critère retourne une chaine textuelle).

Définitions

Soit un critère simple *fc*, son ensemble de sortie ES peut être un ensemble totalement ordonné, c'est-à-dire muni d'une relation d'ordre totale ≤. $\forall$ x, y, z $\in$ ES, la relation ≤ vérifie alors les propriétés suivantes :

- $x \leq x$
- $(x \leq y) \wedge (y \leq z) \Rightarrow (x \leq z)$
- $(x \leq y) \wedge (y \leq x) \Rightarrow (x = y)$
- $(x \leq y) \vee (y \leq x)$

Dans ce cas, la comparaison est mathématique et interprétable par un algorithme. L'ensemble de sortie est dit **numérique**. Ce peut notamment être le cas de mesures statistiques, de tests de génie logiciel ou de résultats d'algorithmes de data-mining.

L'ensemble de sortie ES n'est pas nécessairement totalement ordonné, il peut se présenter sous la forme d'un ensemble d'étiquettes, d'un ensemble de catégories… Dans ce cas, il est dit **symbolique**. Ce peut notamment être le cas de résultats d'algorithmes de data-mining.

Pour un critère complexe, le type de l'ensemble de sortie peut être numérique, symbolique, composé ou non typé. Les définitions des types numériques et symboliques sont les mêmes que ci-dessus.

Définitions

Soit le critère de comparaison complexe *fc* muni d'un ensemble d'entrée EE et d'un ensemble de sortie ES. ES est de type **composé** si et seulement si tout élément *es* ∈ ES est un ensemble de résultats numériques ou symboliques.

ES est **non typé** dans tous les autres cas. Le résultat est donc ad hoc au critère.

C. But

La troisième caractéristique est le but du critère du point de vue EIAH, c'est-à-dire du point de vue de la qualité des modèles d'apprenants inférés ou de l'impact possible du diagnostic sur d'autres composants de l'EIAH, tels que les processus de rétroaction, personnalisation… Il s'agit de documenter dans quel but un concepteur peut utiliser le critère et quels résultats il peut obtenir. Le but ainsi documenté est également une aide pour l'interprétation du résultat du critère. Le processus de calcul d'un critère de comparaison peut ne pas être lié au domaine des EIAH (par exemple un test statistique), aussi le but permet de justifier l'utilisation d'un critère pour une technique de diagnostic des connaissances.

D. Documentation d'un critère

Basé sur les trois caractéristiques définies ci-dessus, un critère est documenté par les éléments suivants :

- Nom
- Description
- But
- Nature du calcul
- Type de résultat
- Paramètre(s)
- Algorithme
- Référence(s)

Par la suite, nous présenterons les critères sous ce format.

3. Interprétation des critères de comparaison

Les critères de comparaison sont calculés en validation croisée sur les traces d'apprenants enrichies. Leurs résultats et l'interprétation de ces résultats ne sont donc valides que pour le domaine et les traces considérés. Un critère n'a pas pour but d'évaluer une technique de

diagnostic en général, mais de fournir des moyens pour évaluer, positionner et comparer plusieurs techniques pour le domaine d'application du concepteur de diagnostic.

En ce sens, ils apportent une réponse au problème actuellement pointé dans la littérature, qui est qu'il n'existe pas de moyen fiable de savoir si une technique de diagnostic sera « meilleure » qu'une autre sur un domaine et des traces donnés. Actuellement, il est nécessaire d'instancier les techniques au domaine, afin de les évaluer soit par des expérimentations, soit, comme nous le proposons, par des critères de comparaison calculables pour des techniques différentes (basées sur des modèles de diagnostic différents).

4. Exemples de critères

A. Précision de la prédiction

- Nom : Précision de la prédiction
- Description : taux de bonne prédiction au temps *t* de l'évaluation du comportement de l'apprenant au temps *t+1* (par exemple, prédiction de la réponse, correcte ou incorrecte, à la prochaine étape de résolution du problème). Le comportement est ici à entendre au sens diagnostic comportemental tel que défini chapitre 3.
- But : mesurer la capacité de la technique de diagnostic à prédire le comportement de l'apprenant à partir du modèle de l'apprenant inféré. L'hypothèse est que plus le modèle de l'apprenant inféré est fidèle, plus le taux de bonne prédiction sera élevé.
- Nature du diagnostic : simple
- Type de résultat : numérique
- Paramètre : -
- Algorithme :

  Pour chaque trace d'apprenant

  - Inférence du modèle de l'apprenant par la technique de diagnostic
  - Calcul de la probabilité que le résultat du diagnostic comportemental soit « correct » à partir du résultat du modèle d'apprenant
  - Si la probabilité est supérieure à 0,5 et que le diagnostic comportemental est correct, la prédiction est bonne

  Valeur retournée : taux prédictions correctes / (prédictions correctes + prédictions incorrectes)
- Référence : (Webb et al., 2001)

B. Racine de l'erreur quadratique moyenne

- Nom : Racine de l'erreur quadratique moyenne (RMSE, Root mean square error)
- Description : mesure de précision de la prédiction, mais attribuant un poids différent aux possibles erreurs de prédiction
- But : mesurer la capacité de la technique de diagnostic à prédire le comportement de l'apprenant à partir du modèle de l'apprenant inféré. L'hypothèse est que plus le modèle de l'apprenant inféré est fidèle, plus le taux de bonne prédiction sera élevé.
- Nature du diagnostic : simple

- Type de résultat : numérique
- Paramètre : -
- Algorithme : $\sqrt{E((\theta' - \theta)^2)}$ avec θ' la prédiction du comportement de l'apprenant et θ ce qui a réellement été observé dans les traces. Pour le calcul de la prédiction, cf. le critère A.
- Référence : (Armstrong et Collopy, 1992)

C. Akaike Information Criterion

- Nom : Akaike Information Criterion (AIC)
- Description : mesure de la vraisemblance d'un modèle prédictif, pénalisé par le nombre de paramètres du modèle
- But : mesurer la capacité de la technique de diagnostic à prédire le comportement de l'apprenant à partir du modèle de l'apprenant inféré, mais en prenant en compte le nombre de paramètres de la technique de manière à pénaliser sa complexité. L'hypothèse est que plus un modèle a de paramètres, plus le risque de sur-apprentissage augmente.
- Nature du diagnostic : simple
- Type de résultat : numérique
- Paramètre : -
- Algorithme : 2k-2ln(L) avec L la vraisemblance de la prédiction, k le nombre de paramètres
- Référence : (Akaike, 1974)

D. Bayesian Information Criterion

- Nom : Bayesian Information Criterion (BIC)
- Description : mesure de la vraisemblance d'un modèle prédictif, pénalisé par le nombre de paramètres du modèle et le nombre de traces
- But : le but est identique au critère AIC (décrit ci-dessus), mais prend également en compte le nombre de traces dans la base de traces d'apprenants. L'hypothèse est qu'un trop faible nombre de traces par rapport au nombre de paramètres augmente le risque de sur-apprentissage.
- Nature du diagnostic : simple
- Type de résultat : numérique
- Paramètre : -
- Algorithme : ln(N)k -2ln(L) avec L la vraisemblance de la prédiction, N la taille de la base de traces, k le nombre de paramètres
- Références : (Schwarz, 1978)

E. Area Under the Curve ROC

- Nom : Area Under the Curve (AUC)
- Description : étant donné deux traces quelconque, chacune évaluée par le diagnostic comportemental à deux entiers x et y (x inférieur à y) de façon binaire (x représentant par exemple « incorrect » et y « correct »), AUC est basiquement la probabilité Pr(x<y), c'est-à-dire la probabilité d'identifier correctement chaque trace

pour la technique de diagnostic. Dans le cas idéal, Pr(x<y)=1 signifie que la technique a une probabilité de 1 de prédire correctement quelle trace est « correcte » et quelle trace est « incorrecte ». Dans le cas au pire, Pr(x<y)=0,5 signifie que la technique fait des prédictions aléatoires.

- But : mesurer la capacité de la technique de diagnostic à prédire le comportement de l'apprenant à partir du modèle de l'apprenant inféré.
- Nature du diagnostic : simple
- Type de résultat : numérique
- Paramètre : -
- Algorithme : $\int_0^1 (f(x) - x)\, dx$ avec f(x) la courbe ROC de la prédiction.

  Pour le calcul de la courbe ROC, il faut calculer les valeurs suivantes (nous appelons ici x « incorrect » et y « correct », comme ci-dessus) : le nombre de traces correctes C, le nombre de traces incorrectes I, le nombre de traces correctes prédites comme correctes par la technique PC, le nombre de traces incorrectes prédites comme incorrectes par la technique PI. La sensibilité est la valeur PC/C et la spécificité est la valeur 1-PI/I.

  La courbe ROC vise à calculer le rapport entre sensibilité et spécificité sur l'intervalle [0;1] en faisant seulement varier le calcul de PC et PI avec un paramètre t sur [0;1] : chaque trace correcte est comptée dans PC si et seulement si le résultat de la prédiction de la technique est supérieur à t. Par exemple, si t=0,8, si la trace est correcte et si la technique prédit avec une probabilité de 0,95 que la trace est correcte, alors la prédiction est bonne car 0,95>0,8.
- Référence : (Draper et al., 1966; Sing et al., 2005)

Note : AUC n'est applicable que dans le cas où l'évaluation du comportement de l'apprenant est binaire (correct ou incorrect).

F. Temps d'exécution

- Nom : temps d'exécution
- Description : temps de calcul requis par la technique pour inférer le modèle de l'apprenant sur la base de traces.
- But : mesurer la rapidité de calcul de la technique, dans le cas où le modèle d'apprenant doit servir à d'autres composants de l'EIAH prenant des décisions en temps réel (par exemple donner une rétroaction immédiate après une erreur).
- Nature du diagnostic : simple
- Type de résultat : numérique
- Paramètre : -
- Algorithme : timestamp fin d'exécution – timestamp début d'exécution
- Référence : -

G. Complexité de la technique

- Nom : Complexité de la technique
- Description : mesure de la complexité de la technique, qui peut avoir un impact sur sa compréhension, son utilité, sa difficulté à être maintenue et mise à jour.

- But : mesurer le nombre de variables, de paramètres et de relations d'une technique de diagnostic.
- Nature du diagnostic : complexe (résultats composés, requiert des add-on tels que définis ci-dessus)
- Type de résultat : composé
- Paramètre : -
- Algorithme : exécution des add-on d'application de critères capables de compter les variables, paramètres et relations de chaque implémentation
- Référence : -

H. Entropie de Shannon

- Nom : entropie de Shannon
- Description : mesure de la quantité d'information délivrée par la technique
- But : mesurer si le résultat d'une technique est sensible aux informations marginales dans les traces. Plus l'entropie est grande, plus la technique renvoie des résultats différents (i.e. avec un grand nombre de valeurs différentes), tandis qu'une entropie plus faible signifie que la technique renvoie plus souvent les mêmes résultats.
- Nature du diagnostic : simple
- Type de résultat : numérique
- Paramètre : -
- Algorithme : $-\sum_i \Pr(X = xi)\, log2\, \Pr\,(X = xi)$ avec X la connaissance diagnostiquée pour chaque trace de la base de traces, xi un état (une valeur) de cette connaissance.
- Référence : (Shannon, 2001)

Note : l'entropie de Shannon n'est calculable que pour les techniques où les connaissances sont des variables aléatoires discrètes.

I. Corrélation avec un diagnostic externe

- Nom : corrélation avec un diagnostic externe
- Description : calcul de la corrélation entre les résultats de la technique de diagnostic et un diagnostic expert pour chaque apprenant. La corrélation se calcule sur le vecteur de probabilité qu'un apprenant ait appris chaque connaissance du domaine.
- But : mesurer la proximité avec un diagnostic externe, réalisé par exemple par un expert du domaine. Il s'agit de positionner la technique par rapport à une référence (le diagnostic externe).
- Nature du diagnostic : complexe (requiert le diagnostic externe)
- Type de résultat : numérique
- Paramètre : V2={p1'…p2'} (défini ci-dessous)
- Algorithme : soit C={c1…cn} le vecteur des connaissances du domaine, V1={p1…pn} le vecteur des probabilités qu'un apprenant ait appris les connaissances C, et V2={p1'…p2'} un vecteur de probabilité similaire mais fourni par le concepteur. Le résultat est la corrélation de Pearson entre V1 et V2.
- Référence : corrélation de (Pearson, 1896) :

J. Sensibilité aux données manquantes

- Nom : Sensibilité aux données manquantes
- Description : mesure de la sensibilité de la technique à des manques dans les traces, c'est-à-dire des variables pour lesquels il manque la valeur pour une trace t de la base de traces d'apprenants.
- But : évaluer la technique pour des domaines où les traces sont difficiles à collecter, notamment les domaines mal définis.
- Nature du diagnostic : complexe (résultat composé)
- Type de résultat : composé
- Paramètre : *x* le nombre de manques à simuler, *n* le nombre de simulations
- Algorithme :
    - Simuler aléatoirement *x* manques dans les traces
    - Calculer la précision de la prédiction sur les nouvelles traces
    - Calculer l'écart entre la précision courante et la précision initiale
    - Itérer *n* fois ce processus
    - Retourner la moyenne des écarts de précisions et le vecteur de ces écarts
- Référence : -

K. Inférence immédiate

- Nom : inférence immédiate
- Description : informe si une technique est capable d'inférer le modèle de l'apprenant trace par trace (immédiat), ou seulement sur un ensemble de traces (différé).
- But : évaluer les possibilités d'utilisation de la technique dans l'EIAH.
- Nature du diagnostic : simple
- Type de résultat : symbolique
- Paramètre : -
- Algorithme : accès aux méta-données de la technique (cf. chapitre 3)
- Référence : -

L. Impact sur le choix d'un type de rétroaction

- Nom : Impact sur le choix d'un type de rétroaction
- Description : mesure de l'impact du résultat du diagnostic sur un outil de feedback immédiat sélectionnant un type de rétroaction à fournir à l'apprenant (parmi une liste de types possibles) en cas d'erreur.
- But : évaluer l'impact de la technique sur un processus de prise de décision de l'EIAH (choix d'une rétroaction). Par extension, mesure de l'utilité de la technique de diagnostic
- Nature du diagnostic : complexe (requiert des informations sur les types de rétroaction et une mesure d'évaluation de l'impact d'un type de rétroaction sur l'apprentissage)
- Type de résultat : composé
- Paramètre : types de rétroaction
- Algorithme : cf. ci-après

- Référence : (Lallé et al., 2013)

5. Description d'un critère : impact sur le choix d'un type d'aide

A. Principe

Nous allons décrire en détail le critère L : Impact sur le choix d'un type de rétroaction, car il s'agit d'un critère complexe et spécifique au domaine des EIAH, le but étant d'évaluer l'impact des techniques de diagnostic des connaissances sur les prises de décision de l'EIAH, ici le choix d'un type d'aide (par exemple, choisir entre donner un indice ou la bonne réponse, entre donner une aide sous forme de schéma ou de formule…). À noter que nous donnons un exemple concret d'application du critère dans la partie III. 3 de ce chapitre.

Le principe du critère est le suivant : à partir de traces d'apprenants, il utilise une méthode de classification automatique pour prédire le résultat (succès ou échec) d'une aide donnée par l'EIAH, en prenant en considération des informations sur l'apprenant, la tâche et l'état des connaissances de l'apprenant donné par une technique de diagnostic. Il en résulte une stratégie d'aide apprise automatiquement et permettant de sélectionner dans une situation donnée l'aide ayant la probabilité la plus élevée de succès (exemple : « si pour un exercice de niveau facile, la technique de diagnostic indique que l'apprenant maîtrise la connaissance requise, alors donner le type d'aide indice »). Chaque technique de diagnostic est utilisée pour apprendre une stratégie d'aide. Le critère compare ces stratégies d'aide en fonction de la technique de diagnostic utilisée.

B. Prérequis

Ce critère demande les prérequis suivants :

- Chaque trace de la base de traces d'apprenants doit être annotée avec le type d'aide donné à l'apprenant.
- Par extension, l'EIAH ou le prototype utilisé pour la collecte des traces doit pouvoir donner des aides de plusieurs types à l'apprenant. Si ces types sont donnés aléatoirement, il n'y a pas de biais lors de l'apprentissage des stratégies d'aide que nous décrivons ci-dessous.

C. Algorithme

L'algorithme du calcul du critère fonctionne en trois étapes :

- application de la technique de diagnostic sur les traces afin d'inférer le modèle de chaque apprenant
- sélection des variables dans les traces ayant un impact significatif sur le succès d'un type d'aide via un modèle linéaire. Par exemple : la difficulté de l'exercice, le niveau de l'apprenant, le nombre de fois où l'apprenant a résolu cet exercice, le taux d'erreur moyen de l'apprenant, la présentation d'un exercice sous forme de texte ou de figure en géométrie…
- apprentissage d'une stratégie d'aide via un algorithme de classification automatique. Par exemple, une stratégie d'aide peut être un ensemble de règles recommandant un type d'aide ; un exemple de règle peut être : « Si

niveau_exercice=facile ET présentation_exercice=figure ALORS type_d'aide=dessin sur la figure ».

Une technique de diagnostic infère le modèle d'apprenant. La première étape de l'algorithme vise enrichir les traces grâce au modèle de l'apprenant inféré, en créant une nouvelle variable *prediction*. Cette variable représente pour chaque trace de la base de traces la probabilité pour l'apprenant que le résultat du diagnostic comportemental soit correct.

Pour trouver dans les traces les éléments qui prédisent le mieux le succès d'un type d'aide, la seconde étape de l'algorithme emploie une régression linéaire pas à pas (*stepwise linear regression*) avec le succès du type d'aide comme variable à expliquer et les variables des traces comme prédicteurs. La mesure d'optimisation utilisée pour la régression pas à pas est AIC (Akaike Information Criterion, cf. la description donnée dans le critère C). Un test ANOVA à un facteur mesure si une variable a un impact significatif sur le succès d'un type d'aide ($p<0.01$).

En utilisant le résultat des techniques de diagnostic, les variables sélectionnées lors de la seconde étape et le type d'aide, le problème est de prédire si une aide sera un succès ou non. Pour cela, la troisième étape apprend automatiquement une stratégie d'aide au moyen d'un algorithme de classification automatique. Nous utilisons l'algorithme PART (Cohen, 1995) implémenté dans Weka (Holmes et al., 1994).

Une stratégie d'aide basée sur un des classificateurs appris ci-dessus fonctionne comme suit : choisir le type d'aide selon la règle ayant la confiance la plus élevée dans la situation donnée. Si plusieurs types sont possibles, choisir aléatoirement.

Il reste enfin à calculer les résultats du critère, qui sont composés. Ces résultats sont calculés en validation croisée afin de limiter tout biais lié à l'apprentissage des stratégies d'aide.

Le premier résultat est la précision des stratégies d'aide, i.e. le taux de traces pour lesquelles la stratégie est capable de prédire si le type d'aide donné sera un succès ou non. Ce calcul de la prédiction est similaire à celui de la précision des diagnostics (critère A), mais appliqué aux stratégies d'aide pour la prédiction du succès des types d'aide. Prenons pour exemple une stratégie d'aide avec une seule règle (celle donnée ci-dessus) : « Si niveau_exercice=facile ET présentation_exercice=figure ALORS type_d'aide=dessin sur la figure ». Si, dans les traces de test, 80 % des cas où le type d'aide « dessin sur la figure » donné sur un exercice facile présenté sous forme de figure sont un succès, alors la précision de la stratégie d'aide est de 80 %.

Le second résultat est l'impact des techniques de diagnostic sur la précision des stratégies d'aide. Le critère emploie pour ce calcul de l'impact le test de McNemar (McNemar, 1947), qui évalue la signifiance de différences entre deux classificateurs C1 et C2. La formule est :

$$\chi^2 = (d1 - d2)^2 / (d1 + d2)$$

Ici, *d1* est le nombre d'instances classifiées comme positives par C1 et négatives par C2, et *d2* le nombre d'instances classifiées comme positives par C2 et négatives par C1. La somme *d1+d2* doit être supérieure à 80, la limite minimale définie par McNemar pour son test, qui peut alors être approximé par une distribution χ^2. Le test donne en résultat la valeur p permettant ou non de rejeter l'hypothèse nulle suivante : les techniques de diagnostic n'ont pas un impact significatif sur la précision des stratégies d'aide.

Le second résultat est la couverture des stratégies, i.e. le taux de traces pour lequel la stratégie est capable de recommander un type d'aide.

Le troisième résultat est l'espérance d'amélioration du taux de bonnes réponses faites par les apprenants. Le succès espéré E est calculé comme suit :

$$E(Success \mid h^*, TD, F)$$

Avec TD la technique de diagnostic, F les variables ayant un impact significatif sur le succès des types d'aide, et h* le type d'aide avec la probabilité de succès la plus élevée selon une stratégie d'aide pour une situation donnée :

$$h^* = \mathrm{argmax}_h \, E(Fluent \mid h, TD, F)$$

III. Exemples

Nous donnons dans cette section deux exemples d'application des critères. Nous considérons pour cela les mêmes traces que les exemples donnés au chapitre 3 sur la construction de techniques de diagnostic pour le domaine de la chirurgie orthopédique. L'extrait des traces utilisé est donné Table 11.

N°	Student	Vert-ebra	Date	Action	Control	Represen-tation	Situation variables	Behavioral diagostic
1	Baptiste	T12	12/05 9 :12	Front X-ray	Vertebra centered	X-ray	Dist_RF_centerVert	Correct
2	Baptiste	T12	12/05 9 :12	Front X-ray	Pedicles symetric	X-ray	Dist_RF_symetryPedic	Incorrect
3	Baptiste	T12	12/05 9 :14	Profile X-ray	Intervertebraldisc	X-ray	Dist_RP_Interdisc	Incorrect
4	Baptiste	L3	12/05 9 :14	Profile X-ray	Intervertebraldisc	X-ray	Dist_RP_Interdisc	Correct
5	Baptiste	L3	12/05 9 :14	Profile X-ray	Intervertebraldisc	X-ray	Dist_RP_Interdisc	Correct

Table 11 : Traces utilisées dans les exemples de ce chapitre, dans le domaine de la chirurgie orthopédique. *Vertebra* représente l'exercice, *Action* une interaction avec l'EIAH, *Representation* désigne les types d'expression des problèmes et des actions, *Situation variables* est la règle ayant permis de calculer le diagnostic comportemental, *Behavioral diagnostic* est le diagnostic comportemental, et *Control* indique quelle connaissance permet de vérifier la validité du résultat de l'action, ici correct ou incorrect.

Dans l'exemple, supposons que nous comparons trois techniques de diagnostic : Control-based implémenté par un réseau bayésien dynamique, Knowledge tracing implémenté par un modèle de Markov caché et Constraint-based implémenté par des contraintes (des règles IF/THEN). Ces techniques permettent d'inférer le modèle de l'apprenant trace par trace.

La comparaison requiert d'obtenir d'abord les résultats du diagnostic des connaissances. Sur notre exemple, nous avons un apprenant, Baptiste, cinq traces et trois connaissances (vertebra centered, pedicles symetric et Intervertebraldisc). Pour une trace, le résultat du diagnostic considéré ici est :

- Pour le Control-based + modèle de Markov caché, les probabilités que la connaissance soit mise en jeu de façon valide, invalide, ou la probabilité que la connaissance ne soit pas mise en jeu alors qu'il le fallait,
- Pour le Knowledge tracing + Modèle de Markov caché, la probabilité que la connaissance soit apprise,
- Pour le Constraint-based, le taux de satisfaction des contraintes associées à la connaissance dans l'historique de l'apprenant.

Dans notre exemple, nous avons cinq traces pour Baptiste. Supposons que les résultats simplifiés des techniques de diagnostic soient les suivants :

N° de la trace (connaissances dans la trace)	Control-based (probabilité de mise en jeu de manière valide)	Knowledge tracing (probabilité connaissance apprise)	Constraint-based (taux de satisfaction)
1 (Vertebra centred)	0,55	0,6	1/1
2 (Pedicle symetric)	0,40	0,3	0/1
3 (Intervetebraldisc)	0,40	0,3	0/1
4 (Intervetebraldisc)	0,52	0,45	1/2
5 (Intervetebraldisc)	0,65	0,7	2/3

Table 12 : résultat simplifié du diagnostic des connaissances pour les cinq traces et les trois techniques de diagnostic considérées dans cet exemple.

Nous allons calculer trois critères à partir de ces résultats.

1. Précision de la prédiction

La précision de la prédiction évalue si le résultat du diagnostic permet de prédire le comportement de l'apprenant. Nous pouvons calculer cette précision pour la connaissance *Intervetebraldisc*.

Pour le Control-based :

- Trace 4, la probabilité de mettre en jeu la connaissance de façon correcte est de 0,4 (obtenue à la trace 3), le diagnostic comportemental est correct, donc la prédiction est incorrecte car 0,4<0,5.
- Trace 5 : la probabilité de mettre en jeu la connaissance de façon correcte est de 0,52, le diagnostic comportemental est correct, donc la prédiction est correcte.

Pour le Knowledge tracing :

- Trace 4, la probabilité que la connaissance soit apprise est de 0,3, le diagnostic comportemental est correct, donc la prédiction est incorrecte.
- Trace 5 : la probabilité que la connaissance soit apprise est de 0,45, le diagnostic comportemental est correct, donc la prédiction est incorrecte.

Pour le Constraint-based :

- Trace 4, le taux de règles satisfaites est de 0/1=0 %, le diagnostic comportemental est correct, donc la prédiction est incorrecte car 0 %<50 %.
- Trace 5 : le taux de règles satisfaites est de 1/2=50 %, donc la prédiction est dite aléatoire.

En résultat, les taux de bonnes prédictions (i.e. la précision de la prédiction) pour cet exemple simplifié sont respectivement :

- Control-based : 1/2 (50 %)
- Knowledge tracing : 0/2 (0 %)
- Constraint-based : 0/1 (0 %)

2. Corrélation avec un diagnostic externe

La corrélation avec un diagnostic externe compare les techniques par rapport à un diagnostic externe, par exemple donné par un expert du domaine. Supposons que l'expert donne le diagnostic final suivant pour Baptiste :

- Vertebra centred : non apprise
- Pedicle symetric : non apprise
- Intervertebraldisc : apprise

Pour les techniques de diagnostic, le diagnostic final du diagnostic d'une connaissance est le résultat du diagnostic pour la dernière trace de l'apprenant mettant en jeu cette connaissance. Chaque connaissance est dite « apprise » si sa probabilité d'être apprise est supérieure à 0,5, ou si le taux de règles satisfaites est supérieur à 0,5 pour le Constraint-based. Cela donne donc :

Connaissance	Control-based	Knowledge tracing	Constraint-based
Vertebra centred	Apprise	Apprise	Aléatoire
Pedicle symetric	Non apprise	Non apprise	Non apprise
Intervetebraldisc	Apprise	Apprise	Apprise

Par rapport à l'expert, les trois techniques se trompent pour la connaissance *Vertebra centred*. Le critère représente le diagnostic final comme un vecteur de booléen (1 pour connaissance apprise, 0 sinon). Pour l'expert, son vecteur est donc {0, 0, 1} tandis que les techniques donnent toutes {1, 0, 1}, soit une corrélation de Pearson de 0,5. À noter que le résultat aléatoire du Constraint-based est fixé à faux (ici contraire à l'expert).

3. Impact sur le choix d'un type d'aide

Pour ce critère, il est nécessaire d'avoir pour chaque trace le dernier type d'aide donné avant sa collecte. Supposons que lors de la collecte des traces, deux types d'aide étaient possibles : type1 et type2 (par exemple, donner un indice ou donner la bonne réponse). Ces types ont été donnés de la façon suivante pour les cinq traces :

Traces	Type d'aide
1	Type1
2	Type2
3	Type2
4	Type1
4	Type1

Une aide est un succès si le diagnostic du comportement dans la trace est correct.

L'algorithme du critère fonctionne en trois étapes :

1. Inférence du modèle d'apprenant (Table 12)
2. Obtention des variables dans les traces ayant un impact significatif sur le succès de l'aide, via un modèle linéaire pas-à-pas avec le succès de l'aide comme variable dépendante. Supposons que les résultats suivants soient obtenus :

Variable dans les traces	**Student**	**Vertebra**	**Date**	**Action**	**Control**	**Represen-tation**	**Situation variables**	**Help type**
Valeur p		*		**			**	***

Légende : *** : p<0,01 ; ** : p<0,05 ; * : p<0,1

Les variables sélectionnées sont donc vertebra, Action, Situation variables et Help type

3. Apprentissage d'une stratégie d'aide via un algorithme de classification automatique. Supposons qu'un algorithme apprenant des règles de classification soit utilisé, un exemple de règle est le suivant :

```
IF Vertebra=T12

AND Action=Front X-ray

AND Help_Type = Type1

THEN Success (1)
```

Cette règle signifie que le type d'aide Type1 a une probabilité de 1 d'être un succès quand la vertèbre est T12 et l'action une radio de face (Front X-ray). Une stratégie d'aide serait ici un ensemble de règles de classification de ce type.

Le critère calcule une stratégie d'aide pour chaque technique de diagnostic. En validation croisée, il évalue ensuite la couverture de chaque stratégie, le gain espéré du nombre de réponses correctes, et l'impact des techniques sur les stratégies d'aide. Des résultats seront donnés dans les évaluations au chapitre 7.

IV. Retour sur les cas d'usage

1. Premier cas d'usage

Dans le premier cas d'usage, Léa est une experte en EIAH, mais pas en diagnostic des connaissances, qui souhaite intégrer un diagnostic des connaissances dans son EIAH. Elle souhaite utiliser le résultat du diagnostic pour obtenir des profils d'apprenants à la fin des séances de travail et donner des aides adaptées à chaque apprenant.

Selon notre méthode, elle compare les différentes techniques de diagnostic. Ces techniques sont construites au préalable, cf. chapitre 4. Pour la comparaison, elle détermine ses exigences : le résultat du diagnostic doit être informatif, c'est-à-dire donner un grand nombre d'informations sur chaque apprenant, et il doit être fortement spécifique à chaque apprenant afin de pouvoir donner des aides personnalisées. Elle sélectionne quelques critères qui lui semblent répondre à ses exigences : tout d'abord la précision de la prédiction, qui permet de mesurer la précision des résultats du diagnostic pour chaque apprenant, l'entropie de Shannon qui mesure la quantité d'information retournée par les techniques, et l'impact sur le choix d'un type de rétroaction.

2. Second cas d'usage

Dans le second cas d'usage, Nadia est une chercheuse dans le domaine du diagnostic des connaissances qui souhaite comparer une nouvelle technique de diagnostic avec l'existant au moyen de traces enrichies dans deux domaines différents. Elle souhaite obtenir des premiers résultats d'évaluation publiables.

Tout comme dans le premier cas d'usage, Nadia a au préalable construit plusieurs techniques de diagnostic pour ses deux domaines, en plus de la technique qu'elle propose. Elle cerne deux approches pour effectuer sa comparaison. Premièrement, elle veut évaluer si sa technique de diagnostic a une précision de prédiction au moins équivalente aux autres techniques établies. Ensuite, elle veut évaluer la complexité de sa technique par rapport aux autres, afin d'évaluer le rapport entre la complexité et la précision de sa technique. Enfin, elle veut obtenir ces résultats, précision et complexité, pour les traces d'apprenants qu'elle a identifiés au préalable comme ayant des misconceptions précises, afin d'évaluer sa technique dans des cas particuliers.

Elle sélectionne les critères qui correspondent à ses exigences : précision de la prédiction, RMSE, AUC, BIC, AUC et complexité. Elle lance sur ses deux jeux de traces (en géométrie et en algèbre) le calcul de tous ces critères. Puis elle sélectionne itérativement les sous-ensembles particuliers d'apprenants qu'elle a identifiés et lance pour chaque sous-ensemble le calcul de tous les critères. Elle recoupe tous ces résultats et les interprète afin de conclure

sur la qualité de sa technique par rapport aux autres, et les sous-ensembles d'apprenants ayant une misconception commune, en géométrie et en algèbre, afin d'identifier les cas pour lesquels sa technique est plus précise ou moins précise, plus complexe ou moins complexe.

V. Conclusion

Nous avons défini un critère de comparaison comme un algorithme prenant par défaut en entrée des traces d'apprenants et le résultat de techniques de diagnostic des connaissances pour ces traces, et retournant un résultat numérique, symbolique ou composé. Un critère de comparaison est applicable à toute technique de diagnostic, telle que définie chapitre 3, donc permet de comparer ou positionner différentes techniques de diagnostic, y compris si leurs modèles de diagnostic sont différents. Les critères de comparaison les plus utilisés sont ceux qui évaluent la précision de prédiction des techniques, c'est-à-dire leur capacité à prédire le futur comportement de l'apprenant à partir de son historique (par exemple, sa prochaine réponse sera-t-elle correcte ou incorrecte). Mais il est possible d'imaginer des critères plus complexes traitant par exemple de l'impact de techniques sur les prises de décision pédagogiques de l'EIAH (rétroaction, choix du prochain exercice…).

Par rapport à la littérature, nous proposons quelques critères de comparaison inédits, mais c'est surtout l'utilisation, la catégorisation et la généricité des critères, applicables à toute technique de diagnostic, qui permet de répondre à nos questions de recherche. Dans la littérature, aucun travail ne propose de comparer au moyen de traces d'apprenants des techniques de diagnostic basées sur des modèles de diagnostic différents. De plus, nos critères de comparaison peuvent être des algorithmes complexes, prenant en compte des éléments spécifiques du domaine des EIAH (nous avons donné l'exemple des types d'aide). Or, il n'existe pas de critères « spécifiques aux EIAH » dans la littérature, qui propose plutôt de fort nombreux critères statistiques, prédictifs ou issus de data-mining.

Notre méthode permet deux usages des critères pour un concepteur de diagnostic : soit calculer des critères sans avoir besoin de les développer ; soit développer et implémenter un nouveau critère qui sera par définition applicable à toute technique de diagnostic s'il respecte nos formalismes. Il s'agit donc bien d'une assistance au concepteur, soit en simplifiant l'utilisation des critères, soit en simplifiant leur développement (du point de vue de la généricité) et si possible leur partage au sein de la communauté et donc leur réutilisation.

La principale limite de ces critères vient des possibles biais qui peuvent survenir lors de leur application. En effet, les critères sont calculés à partir d'au moins deux entrées : les résultats des différentes techniques de diagnostic à comparer, et les traces. Une technique mal construite ou les traces peuvent donc conduire à privilégier une technique au profit d'une autre. Il convient donc de bien noter que notre méthode est une méthode d'assistance empirique : les résultats des critères ne sont valides que pour les traces du concepteur, et la question de la qualité des traces est laissée à charge du concepteur. Le concepteur doit donc

interpréter et croiser les résultats des critères. Nous reviendrons lors des évaluations et de la conclusion sur ce sujet.

Chapitre 6 : Plateforme PlaCID

Sommaire

I. Introduction 120
II. Aperçu de la plateforme 121
1. Schéma général 121
2. Bases de modèles de diagnostic, d'implémentations et de critères 124
3. Construction de techniques de diagnostic 124
4. Comparaison de techniques de diagnostic 124
III. Architecture logicielle 125
1. Technique de diagnostic 125
2. Traces 127
3. Critères de comparaison 128
4. Classe principale 130
5. Interfaces 130
IV. Utilisation de la plateforme 132
V. Limites de la plateforme 135
VI. Conclusion 135

I. Introduction

Nous proposons une plateforme nommée PlaCID (Platform for Comparing et Instanciating Diagnostics) permettant d'assister la construction et la comparaison de techniques de diagnostic des connaissances, telles que définies dans les sections précédentes. Comme détaillé dans les chapitres précédents, l'assistance requiert des traces d'apprenants enrichies par un diagnostic du comportement. Ces traces sont utilisées pour assister la construction de techniques de diagnostic via un algorithme d'apprentissage semi-automatique et d'une ontologie du domaine des traces, et pour comparer les diagnostics construits via un ensemble de critères de comparaison.

Notre plateforme calcule par défaut les critères de comparaison en validation croisée, c'est-à-dire que les traces sont séparées en deux sous-ensembles : une partie pour la construction (appelée base d'apprentissage), l'autre partie pour la comparaison (appelée base de test). En effet, si les mêmes traces étaient utilisées pour la construction (via l'algorithme d'apprentissage semi-automatique) et pour la comparaison (via les critères), nos résultats poseraient le problème du sur-apprentissage. Dans notre domaine, le sur-apprentissage

signifie qu'il est impossible de savoir si une technique de diagnostic donne de bons résultats seulement pour la base d'apprentissage, ou bien si ces bons résultats peuvent être généralisés. L'usage d'une base de test différente de la base d'apprentissage permet d'évaluer nos techniques de diagnostic sur de nouvelles traces indépendantes de la base d'apprentissage.

Nous allons présenter dans ce chapitre premièrement un aperçu des composants de la plateforme et de son fonctionnement. Nous détaillerons en suite précisément une architecture logicielle permettant d'implémenter nos deux méthodes d'assistance – construction et comparaison –, et illustrerons cette architecture logicielle avec nos choix d'implémentation dans notre plateforme en JAVA. Enfin, nous reviendrons sur l'utilisation de notre plateforme et ses limites.

II. Aperçu de la plateforme

1. Schéma général

Le schéma général de la plateforme est représenté Figure 28. Le cadre rectangulaire délimite les processus internes à la plateforme, qui se divisent en trois parties principales :

- une base de données stockant les techniques de diagnostic et les critères de comparaison,
- la construction des techniques pour le domaine du concepteur,
- la comparaison des techniques construites.

Les flèches indiquent les étapes successives d'utilisation de la plateforme, depuis les traces jusqu'à l'obtention des résultats. Nous ne présentons ici que les composants de la plateforme, pas les interfaces qui seront abordées partie III. 5.

Légende du schéma (Figure 28)

1. L'utilisateur de la plateforme, le concepteur du diagnostic tel que défini en introduction.
2. Les traces d'apprenants enrichies par un diagnostic comportemental. Comme défini dans le chapitre 3, ce processus doit être effectué en amont.
3. Séparation des traces en deux sous-ensembles pour la validation croisée.
4. Base de données contenant les modèles de diagnostic et les implémentations supportées par la plateforme.
5. Les techniques de diagnostic, composées d'un modèle de diagnostic et d'une implémentation, à instancier pour le domaine
6. Conception d'une ontologie des connaissances du domaine par le concepteur (apport d'information experte).
7. Processus de construction des techniques de diagnostic basé sur un algorithme d'apprentissage automatique semi-supervisé.
8. Techniques de diagnostics instanciées au domaine.

9. Application des techniques de diagnostic instanciées, c'est-à-dire inférence du modèle de l'apprenant, sur le second sous-ensemble de traces.
10. Base de données contenant les critères de comparaison calculables par la plateforme.
11. Base de données d'add-on permettant d'appliquer les critères pour chaque technique de diagnostic instanciée. L'application d'un critère dépend du modèle d'apprenant inféré par le diagnostic, tel que formalisé dans le modèle de diagnostic.
12. Application de chaque critère de comparaison pour toutes les techniques de diagnostic instanciées, en validation croisée.
13. Visualisation des résultats des critères et obtention des codes sources des techniques instanciées.

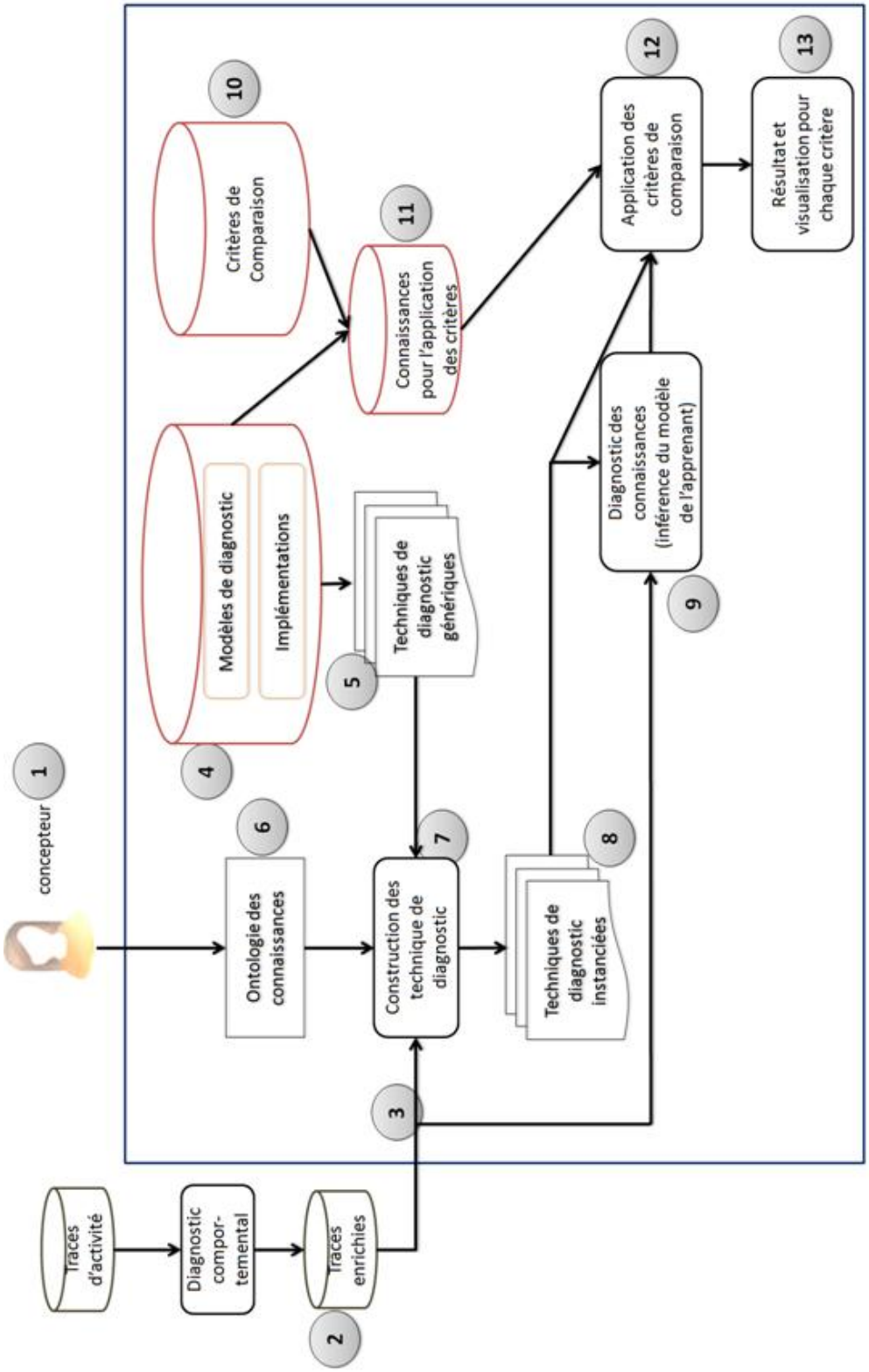


Figure 28 : Schéma des différents composants ou éléments de la plateforme. Les flèches indiquent qu'un composant ou un élément est un prérequis d'un autre.

2. Bases de modèles de diagnostic, d'implémentations et de critères

La plateforme intègre trois bases de données.

La première base (Figure 28, numéro 4) stocke tous les modèles de diagnostic génériques et les implémentations supportées par la plateforme. Un modèle de diagnostic est identifié par un nom unique et un fichier RDF contenant sa définition. Les modèles de diagnostic sont des instances du modèle de diagnostic générique défini dans le chapitre 3. Une implémentations est identifiée par un nom unique, la liste des modèles de diagnostic pour lesquels elle s'applique et un lien vers le fichier de code source.

La seconde base (Figure 28, numéro 10) stocke les critères de comparaison. Un critère est identifié par son nom unique et un fichier de code source, ainsi que la documentation du critère comme détaillé chapitre 5.

La troisième base (Figure 28, numéro 11) stocke les add-on permettant d'appliquer un critère à une technique de diagnostic implémentée de façon ad hoc si nécessaire (cf. chapitre 5).

3. Construction de techniques de diagnostic

La construction des techniques de diagnostic se fait en deux étapes.

Premièrement, le concepteur conçoit une ontologie des connaissances du domaine qu'il va lier aux traces enrichies (Figure 28, numéro 6). Cette ontologie a pour but d'apporter de la sémantique experte aux traces, de guider l'apprentissage automatique et de renforcer l'interprétabilité des techniques construites. La spécification de cette ontologie est décrite chapitre 4. En l'état, la plateforme n'intègre pas un éditeur d'ontologie ; la spécification de base est fournie via un fichier RDF créé avec Protégé[8].

Cette ontologie est associée aux traces d'apprenant par le concepteur. Pour cela, il lie manuellement des éléments de l'ontologie vers des éléments des traces d'apprenant en utilisant une interface (que nous présenterons plus bas partie III. .III. 5).

Deuxièmement, un algorithme d'apprentissage automatique instancie les techniques de diagnostic (Figure 28, numéro 7 et 8). Il prend en entrée les traces enrichies, l'ontologie des connaissances fournie par le concepteur et les techniques de diagnostics entreposées dans la plateforme.

4. Comparaison de techniques de diagnostic

Les techniques de diagnostic instanciées au domaine sont utilisées sur la seconde partie des traces (Figure 28, numéro 9) pour calculer le résultat de chaque technique (i.e. inférer le modèle de l'apprenant). La plupart des critères identifiés dans la littérature se calculent en effet sur les résultats des techniques de diagnostics.

L'application des critères de comparaison (cf. chapitre 5) entreposés dans la plateforme se fait pour toutes les techniques instanciées (Figure 28, numéro 12). Les résultats des critères

[8] http://protege.stanford.edu/

peuvent être visualisés dans l'interface (Figure 28, numéro 13). La plateforme affiche les résultats moyens pour toutes les traces de l'ensemble de test utilisé dans la validation croisée. Une fonctionnalité possible (non implémentée) est de pouvoir sélectionner un sous-ensemble de traces (par exemple les traces d'un apprenant, les traces d'un exercice, les traces mettant en jeu une connaissance donnée…) et de visualiser les résultats des critères pour ce sous-ensemble de traces uniquement.

Le code source des techniques implémentées est sauvegardé et peut être obtenu par le concepteur.

III. Architecture logicielle

Nous allons présenter plus en détail une architecture logicielle permettant d'implémenter nos deux méthodes d'assistance (pour la construction et la comparaison). Nous spécifions les différents composants requis, les variables requises par ces composants et les principales tâches que doivent remplir ces composants.

L'architecture se divise selon les composants suivants :

- La représentation des techniques de diagnostic, des modèles de diagnostic, et des implémentations
- La lecture des traces
- La représentation des critères de comparaison
- Les add-on d'application des critères
- L'interface principale et l'interface de visualisation des résultats de comparaison

Pour chaque composant, nous donnons en guise d'exemple l'implémentation que nous en avons faite dans notre plateforme, développée en JAVA, sauf l'apprentissage automatique qui a recourt à du code R et les différentes ontologies qui sont toutes au format RDF/OWL.

1. Technique de diagnostic

Pour rappel, une technique de diagnostic est le couple formé par un modèle de diagnostic et une implémentation. Nous avons déjà détaillé l'opérationnalisation des modèles de diagnostic via des fichiers RDF dans le chapitre 3. Quant à l'implémentation, elle doit respecter la spécification suivante :

- Elle est identifiée par un nom unique
- Elle peut lire les traces d'apprenants enrichies
- Elle peut réaliser trois tâches : apprendre automatiquement ses paramètres (cf. chapitre 4), inférer le modèle de l'apprenant à partir des traces enrichies, et transposer tout modèle de diagnostic spécifié selon la formalisation F_{MD} (cf. chapitres 3 et 4)

Dans notre plateforme, les implémentations sont définies par la classe abstraite JAVA suivante (pour la classe DataDiagnostic permettant l'accès aux traces d'apprenant, cf. plus bas) :

```
public abstract class Implementation {
	protected String name ;
	protected DataDiagnostic bufferdata; // traces d'apprenants enrichies

	abstract double learnParameter() ; //apprentissage des paramètres de
l'implémentation si besoin
	abstract void inference() ; //inférence d'un modèle de l'apprenant
	abstract void transpose() ; //transposition d'un modèle de diagnostic quelconque
}
```

Exemple pour l'implémentation AFM (décrite dans l'état de l'art) :

```
public class AFM extends Implementation {
	public AFM(DataDiagnostic bufferdata){
		this.name = "AFM" ;
		this.bufferdata = bufferdata;
	}
…
}
```

Une technique de diagnostic, association entre un modèle de diagnostic et une implémentation, est caractérisée par :

- Un nom unique de la technique
- Le nom du modèle de diagnostic
- Le lien vers le fichier décrivant le modèle de diagnostic (pour nous, une ontologie) tel que défini chapitre 3
- L'implémentation de la technique
- Un ensemble de métadonnées prédéfinies sur la technique

La technique de diagnostic peut réaliser les tâches suivantes :

- Initialisation (lecture du fichier RDF du modèle de diagnostic)
- Inférence du modèle de l'apprenant via son modèle et son implémentation
- Lecture et enregistrement du modèle de diagnostic

La classe JAVA abstraite donnant la spécification d'une technique de diagnostic est :

```
public abstract class DiagnosticTechnique {
	protected String name;
	protected String modelName;   //nom du modèle de diagnostic de la technique
	protected String modelRDFpath;  //chemin du fichier RDF du modèle de diagnostic
	protected Implementation imp;  //Classe JAVA de l'implémentation
	protected Metadata md;

	abstract void init(); //initialise la technique, i.e. son modèle et son implémentation
	abstract void inference(); //infère le modèle de l'apprenant (appel à la classe
Implementation)
	abstract void readModel() ; //ouverture d'un modèle de diagnostic
}
```

Exemple avec la technique de diagnostic composée du modèle de diagnostic Knowledge tracing et de l'implémentation AFM (régression) :

```
public class DiagnosticKT_AFM extends DiagnosticTechnique {
	public void init(){
		this.name = "Knowledge tracing – AFM";
		this.modelName = "Knowledge tracing";
		imp = new AFM();
		modelpath="DiagnosticModelKnowledgeTracing.owl";
		this.read_model();
	}
 }
```

Les métadonnées (classe Metadata) permettent de décrire les caractéristiques de la technique. En l'état, les caractéristiques suivantes peuvent être renseignées :

- La théorie d'apprentissage ou le modèle didactique sous-jacent
- Le temps de l'inférence, en temps réel (immédiat, i.e. trace par trace) ou différé (sur un ensemble de traces)
- La prise en compte de l'incertitude

2. Traces

La lecture et l'enregistrement des traces enrichies est effectuée de manière ad hoc. Comme expliqué plus haut, nous n'utilisons pas d'approche générique pour les traces, qui est une problématique de recherche en soit, du fait de la grande variété des formats de traces d'apprenants. Concrètement, chaque format de traces est lu par un code informatique spécifique, qui enregistre les traces de sorte que les implémentations puissent les lire pour inférer le modèle de l'apprenant.

Dans le code JAVA ci-dessus, DataDiagnostic est une classe abstraite ; à chaque implémentation doit correspondre un fichier ad hoc permettant la lecture des traces enrichies. La plateforme peut lire par défaut des traces d'apprenants au format CSV

(tableur), au format table SQL (base de données) et au format Datashop (via un Webservice et du XML). La spécification de la classe DataDiagnostic est :

```
public abstract class DataDiagnostic {
	abstract void entryData(); //fonction de lecture des traces enrichies
}
```

Exemple avec Datashop :

```
import java.util.Map.Entry;
public class DataDatashop extends DataDiagnostic {
	Entry<TransactionDataSubset, Integer> data;

	void entryData(){
		//accès au Webservice de Datashop
		//…
	}
}
```

La classe TransactionDataSubset décrit les variables d'une trace de Datashop :

```
public class TransactionDataSubset implements Comparable{
	private int ID;
	private String student;
	private String problem;
	private String step;
	private String outcome;
	private String input;
	private String Skill
}
```

3. Critères de comparaison

Un critère de comparaison est décrit par les éléments suivants :

- Un nom unique
- La documentation du critère (cf. chapitre 5)
- Le domaine de validité, c'est-à-dire l'ensemble des techniques de diagnostic pour lesquelles le critère peut être calculé (cf. chapitre 5)

Il doit de plus avoir accès aux traces enrichies et à la liste des techniques de diagnostic instanciées au domaine du concepteur pour lesquelles le critère doit être effectivement calculé.

Les critères sont implémentés dans notre plateforme dans une classe JAVA héritant de la classe abstraite ComparisonCriterion suivante (pour la classe DataDiagnostic, cf. les explications données plus haut) :

```
public abstract class ComparisonCriterion {
	protected String name;
	protected DataDiagnostic bufferdata; // traces d'apprenants enrichies
	protected Arraylist< DiagnosticTechnique > validDomain; // liste des techniques de
diagnostic pour lesquelles le critère peut être calculé. Vide par défaut, signifiant que le
critère s'applique à toutes les techniques.

	protected ArrayList< DiagnosticTechnique > listDiag ; // liste des techniques de
diagnostic à comparer

	abstract double compute() ; //calcule du critère
}
```

Le calcul des critères peut se faire sur les modèles de diagnostic, ou les modèles d'apprenants inférés par les techniques de diagnostic construites par la plateforme. Ces modèles étant formalisés selon F_{MD} (cf. chapitre 3)et inclus dans la plateforme, ils sont directement accessibles.

Le calcul des critères peut requérir d'autres éléments ad hoc, i.e. non prévus dans la plateforme. Considérons par exemple deux implémentations de deux techniques de diagnostic, l'une utilisant un réseau bayésien et l'autre un modèle de régression, ainsi qu'un critère qui requiert en entrée les paramètres de l'implémentation. Pour le réseau bayésien, cette information s'obtient via les tables de probabilité du réseau, tandis que pour la régression il s'agit des coefficients de régression. L'information dépend donc de l'implémentation. Il y a besoin d'un code intermédiaire de sorte que le critère de comparaison, qui est générique, puisse interroger chaque implémentation des techniques de diagnostic à comparer.

Pour pallier ce problème sans remettre en cause la généricité des critères, la plateforme intègre une base d'add-on d'application des critères (comme introduit dans le chapitre 5) : chaque add-on est lié à un critère et intègre le code ad hoc nécessaire au calcul du critère (sur notre exemple ci-dessus, comment accéder au paramètre d'un réseau bayésien, d'un modèle de régression, etc.). Chaque add-on est décrit :

- Par un nom unique
- Par une documentation attachée à celle du critère pour lequel l'add-on s'applique

Dans notre plateforme, un add-on est une classe JAVA ad hoc qui étend la classe abstraite AddOnCriteria. Nous n'avons pas encore implémenté la documentation :

```
public abstract class AddOnCriteria {
	protected  String name ;
}
```

Chaque critère peut accéder à sa liste d'add-on via un tableau associatif qui associe la classe principale de l'add-on (héritant de Script AddOnCriteria) à la technique (héritant de DiagnosticTechnique) pour laquelle il s'applique. En JAVA :

```
HashMap< DiagnosticTechnique, AddOnCriteria > scriptsList ;
```

Ce tableau associatif est ajouté à la classe abstraite ComparisonCriterion donnée ci-dessus :

```
public abstract class ComparisonCriterion {
	protected String name;
	protected DataDiagnostic bufferdata; // traces d'apprenants enrichies
	protected ArrayList< DiagnosticTechnique > listDiag ; // liste des techniques de
diagnostic à comparer
	HashMap< DiagnosticTechnique, AddOnCriteria > scriptsList ;

	abstract double init() ; //initialise les add-on, les traces et la liste des techniques
	abstract double compute() ; //calcule du critère
}
```

4. Classe principale

La classe principale coordonne tous les composants de la plateforme :

```
public class Platform {
	DataDiagnostic data ;
	ArrayList< DiagnosticTechnique > listDiag ;
	ArrayList< ComparisonCriterion > listCriteria ;
}
```

5. Interfaces

Concernant les interfaces, elles sont réalisées en JAVA avec la bibliothèque standard SWING. Du point de vue architecture logicielle, deux interfaces sont requises :

- L'interface pour la construction (travail sur les traces, l'ontologie des traces et des connaissances, l'association entre les traces et l'ontologie)
- L'interface pour la comparaison (visualisation des résultats des critères et navigation parmi les différents critères possibles)

Dans notre plateforme, pas souci de temps, tous ces éléments ne sont pas implémentés et nous avons recours à des logiciels externes, notamment Protégé[9] pour la conception des ontologies.

L'interface principale de travail pour le concepteur se compose de trois parties (Figure 29) :

1. Affichage synthétique des traces d'apprenants enrichies chargées par le concepteur
2. Affichage interactif de l'ontologie des connaissances du domaine
3. Affichage et édition de l'association entre les traces d'apprenants et l'ontologie.

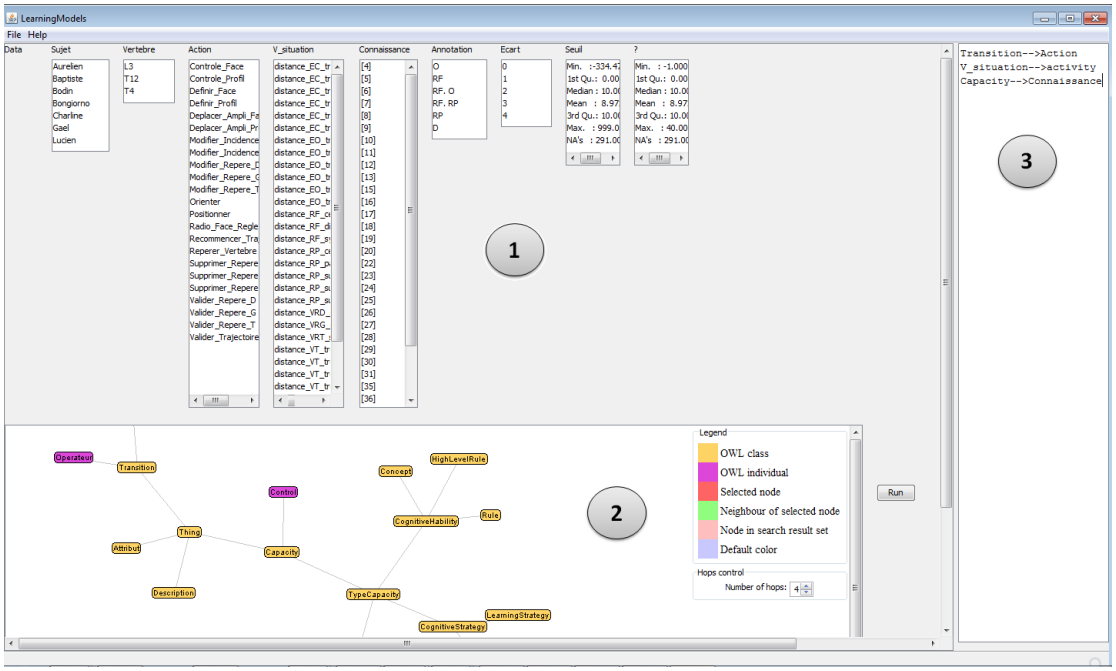


Figure 29 : Interface principale de la plateforme pour le chargement des traces d'apprenants enrichies et l'assistance à la construction de techniques de diagnostic.

Le bouton « Run » lance la construction et le calcul des critères de comparaison. La visualisation des résultats des critères de comparaison a recours à des graphiques. La Figure 30 donne un exemple de résultat d'un critère (nommé « Accuracy ») pour quatre techniques de diagnostic.

1. Liste des critères calculés par la plateforme sur les techniques de diagnostic construites pour les traces d'apprenants et le domaine du concepteur.
2. Graphe de résultat du critère courant.

En l'état, la plateforme vise à fournir une preuve de concept et à appliquer nos méthodes d'assistance sur des traces d'apprenants. Les interfaces n'ont donc pas fonction à être ergonomiques et facilement utilisables.

[9] http://protege.stanford.edu/

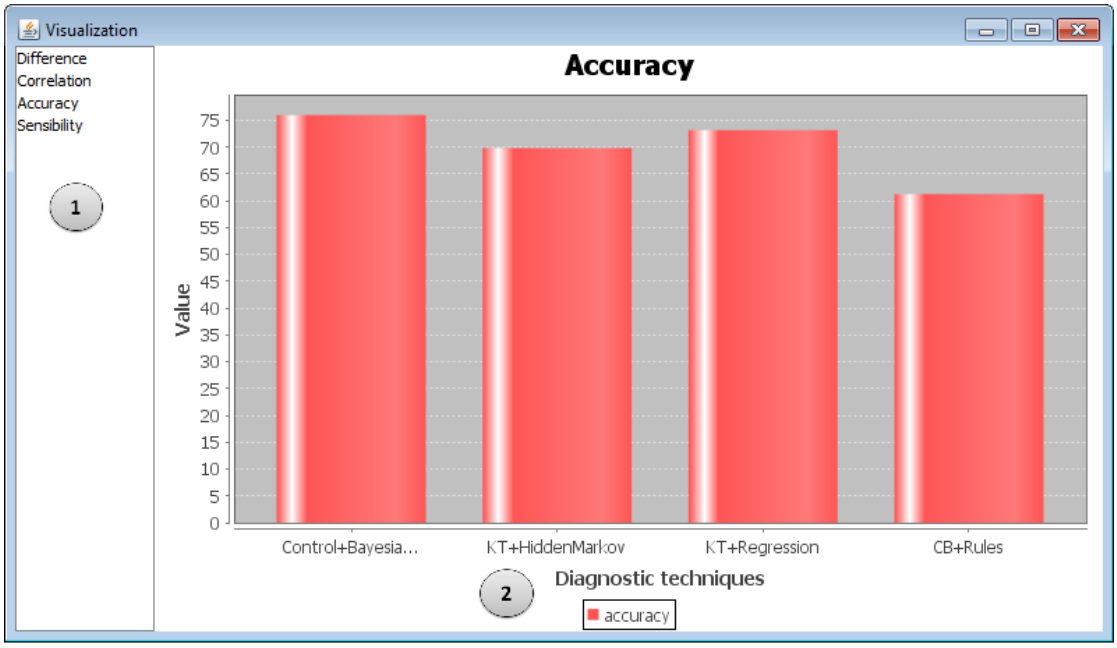


Figure 30 : Interface de visualisation des résultats des critères de comparaison.

IV. Utilisation de la plateforme

Le diagramme de séquence de la Figure 31 résume l'utilisation de PlaCID par le concepteur. L'interaction se fait entre trois entités :

- Le concepteur et utilisateur de notre plateforme (Designer)
- Un EIAH qui doit pouvoir collecter des traces d'apprenants enrichies, comme défini chapitre 3 (TEL system)
- Notre plateforme d'assistance (Assistance Platform)

Le concepteur n'est pas forcément la personne qui a collecté les traces enrichies, elle en est seulement l'utilisatrice. Comme défini en introduction chapitre 1, le but du concepteur peut être d'intégrer une technique de diagnostic dans son EIAH, ou bien de comparer des techniques de diagnostic en soi, sans vouloir les intégrer dans un EIAH. D'autre part, comme expliqué chapitre 1, « concepteur » est un terme général qui peut désigner en réalité plusieurs personnes concevant un diagnostic des connaissances.

L'élément principal de l'utilisation de notre plateforme est la possibilité de construire et comparer les techniques de diagnostic de façon itérative : après avoir chargé ses traces, le concepteur réalise une ontologie des traces et des connaissances qu'il associe aux traces, pour que la plateforme construise et compare les techniques. En fonction des résultats des critères qui intéressent le concepteur, ce dernier peut modifier et raffiner son ontologie de façon à améliorer les résultats des critères. Cette phase itérative est représentée sur la Figure 31.

Les différentes étapes du diagramme de séquence de la Figure 31 sont (les chiffres de la liste se réfèrent aux chiffres du diagramme de séquence) :

1. L'analyse du domaine d'apprentissage permet d'identifier les connaissances du domaine. Ce travail d'analyse des connaissances du domaine essentiel à la construction de tout diagnostic des connaissances ne change pas avec notre plateforme
2. Des traces d'apprenants enrichies par un diagnostic comportemental sont collectées
3. Le concepteur récupère les traces collectées
4. Le concepteur charge ses traces dans la plateforme
5. Le concepteur charge ou construit l'ontologie des traces et des connaissances du domaine requises par l'assistance à la construction, et associe cette ontologie aux traces
6. La plateforme construit les techniques de diagnostic, c'est-à-dire qu'elle instancie toutes les techniques de sa base de techniques au domaine de l'EIAH du concepteur
7. La plateforme diagnostique les connaissances, i.e. infère le modèle de l'apprenant
8. La plateforme calcule en validation croisée les critères de comparaison de sa base de critères
9. La plateforme renvoie à l'utilisateur les résultats des critères et les techniques implémentées

Début itération

10. En fonction des résultats des critères et des techniques construites, le concepteur peut affiner son ontologie des connaissances, ainsi que l'association avec les traces
11. Similaire à 6 avec la nouvelle ontologie et/ou la nouvelle association entre les traces et l'ontologie
12. Similaire à 7
13. Similaire à 8
14. Similaire à 9

Fin itération

15. Le concepteur peut intégrer l'implémentation d'une technique instanciée à son domaine dans son EIAH

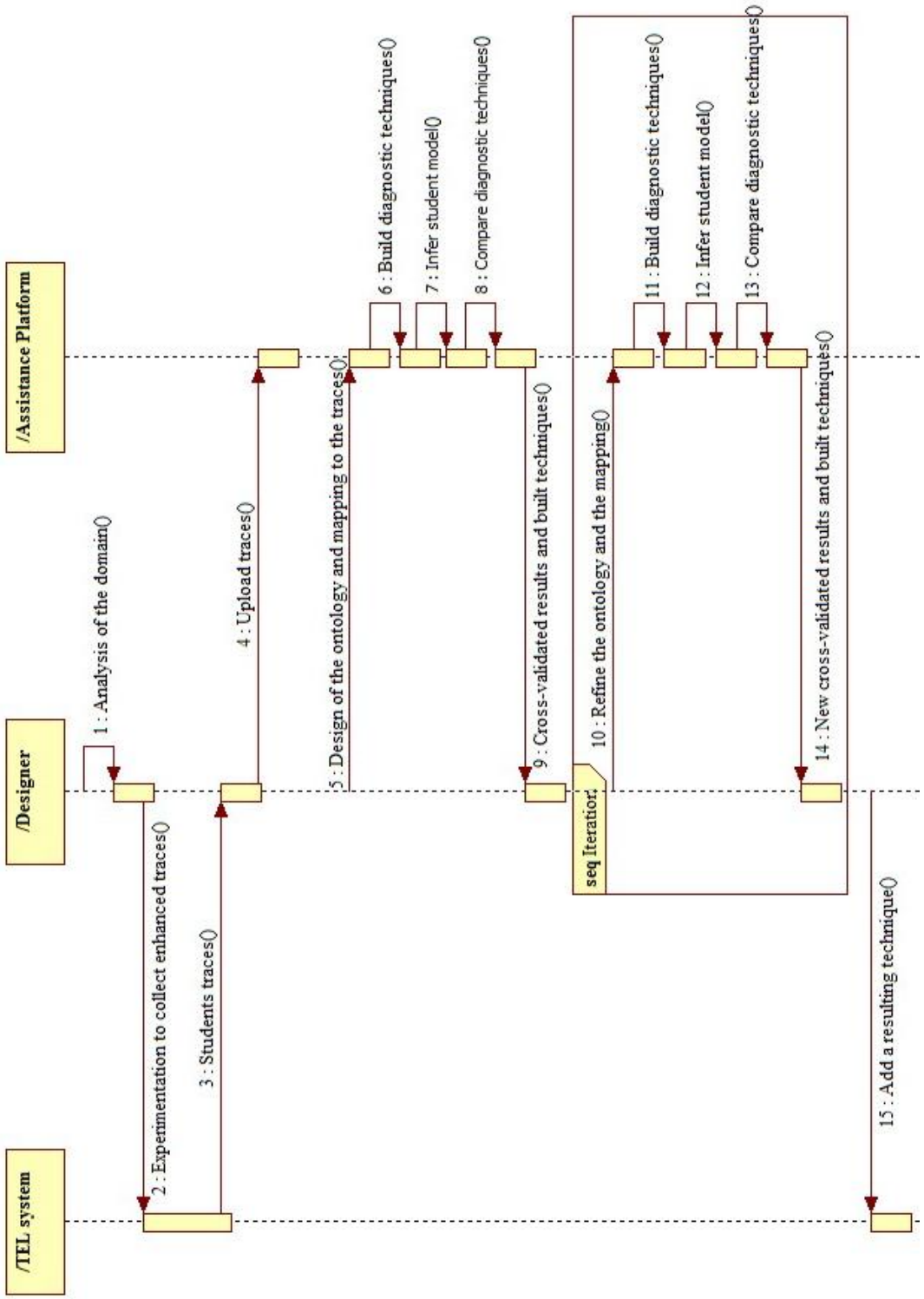


Figure 31 : Diagramme de séquence UML modélisant les interactions entre le concepteur, notre plateforme et l'EIAH du concepteur.

V. Limites de la plateforme

Notre plateforme étant une première preuve de concept, certains éléments des méthodes d'assistance n'y sont pas intégrés :

- Il n'y a pas d'éditeur d'ontologies, qui pourrait guider pas à pas le concepteur pour la définition de l'ontologie ou la formalisation d'un nouveau modèle de diagnostic.
- Il n'est pas possible de recalculer les critères dynamiquement sur de nouvelles traces (en l'état, il est soit nécessaire de refaire l'étape de construction, soit nécessaire de modifier le code source de la plateforme).
- Il n'est pas possible de visualiser ni d'agir sur le graphe d'instanciation (cf. chapitre 4) lors de la construction des techniques de diagnostic.
- Il n'y a pas d'interface pour définir un nouveau format de traces à prendre en compte, un nouveau critère de comparaison ou une nouvelle implémentation pour les techniques de diagnostic (il faut programmer).

VI. Conclusion

Dans ce chapitre, nous avons présenté les différents composants requis pour l'implémentation de nos méthodes d'assistance à la construction et à la comparaison de techniques de diagnostic. Puis nous avons donné une spécification d'une architecture logicielle permettant cette implémentation. Nous avons enfin montré une première plateforme en JAVA dérivée de cette architecture logicielle. Cette plateforme est fonctionnelle : elle peut lire certains formats de traces d'apprenants (formats CSV, SQL ou Datashop), construire et comparer des techniques de diagnostics. Cette preuve de concept va en outre nous permettre d'évaluer nos méthodes d'assistance dans le chapitre suivant. Pour finir, nous considérons cette plateforme comme une preuve de concept et non un logiciel abouti et utilisable, nous avons donc bien identifié les fonctionnalités d'utilisation que nous avons choisi de ne pas implémenter.

PlaCID propose une interface pour l'algorithme de construction semi-supervisé et une interface de visualisation des résultats des critères de comparaison. Parmi les travaux sur le diagnostic des connaissances, il existe deux outils auteurs offrant une interface à un algorithme semi-supervisé : ASPIRE (Mitrovic et al., 2006), qui intègre à la différence de notre plateforme un éditeur guidé d'ontologie, et Simstudent (Matsuda et al., 2005), qui fonctionne par exemples de résolution d'exercices. Ces deux outils auteurs requièrent de construire au préalable l'interface permettant de résoudre ces exercices (i.e. l'interface de l'EIAH). Au contraire, notre plateforme est centrée sur le diagnostic des connaissances et ne requiert aucun travail sur les autres composants de l'EIAH. Toujours dans le domaine du diagnostic des connaissances, deux plateformes permettent de visualiser une évaluation statistique d'une technique de diagnostic : Datashop (Koedinger et al., 2010) et Advisor (Beck et al., 2000). Ces plateformes n'offrent en revanche pas la possibilité d'intégrer plusieurs techniques de diagnostic, ni de définir de nouvelles mesures d'évaluation. L'architecture que nous proposons dans ce chapitre permet l'ajout et la documentation de différents critères et le calcul des résultats des critères si le concepteur fournit des traces

enrichies. Hors du domaine du diagnostic des connaissances, des environnements comme Matlab ou RStudio permettent également d'écrire du code source afin de comparer statistiquement plusieurs techniques de diagnostic, qu'il faut construire au préalable. Toutefois, ces environnements étant très génériques, il n'y a aucune assistance au concepteur et il n'est pas possible de créer une collection de critères de comparaison où les concepteurs peuvent ajouter et partager de nouveaux critères. PlaCID est donc la seule plateforme dans le domaine des EIAH qui permet à la fois de construire et comparer des techniques de diagnostic des connaissances différentes à partir de traces.

Chapitre 7 : Évaluation

Sommaire

I. Méthodologie d'évaluation 137
II. Bases de traces d'apprenants 139
1. Geometry Tutor 139
2. Reading Tutor 140
3. TELEOS 142
4. APLUSIX 143
5. Collecte et volumétrie 144
III. Première expérimentation : évaluation des méthodes d'assistance 145
1. Méthodologie 145
2. Évaluation sur l'application des critères de comparaison 147
3. Évaluation de la méthode de construction 153
4. Retour sur les questions de recherche 157
IV. Seconde expérimentation : développement d'un critère de comparaison spécifique aux EIAH 158
1. Méthodologie 158
2. Application du critère de comparaison 159
3. Résultats du critère de comparaison 160
4. Retour sur les questions de recherche 163
V. Troisième expérimentation : développement d'une nouvelle technique de diagnostic 163
1. Méthodologie 163
2. Modèle de diagnostic Praxéologie 164
3. Retour sur les questions de recherche 167
VI. Conclusion 167

I. Méthodologie d'évaluation

Nos travaux portent sur l'assistance à un concepteur de diagnostic des connaissances en EIAH, au moyen d'une méthode d'assistance à la construction de techniques de diagnostic via un algorithme d'apprentissage semi-automatique et une méthode d'assistance à leur

comparaison via un ensemble de critères de comparaison. Cette assistance permet d'obtenir deux types de résultats susceptibles d'aider le concepteur : des techniques de diagnostic instanciées et le résultat des critères de comparaison. Ce sont l'ensemble de ces points qui sont concernés par l'évaluation.

Nous évaluons notre méthode et notre plateforme au moyen de trois expérimentations, en utilisant des bases de traces d'apprenants enrichies de différents domaines et de différents formats. Afin de limiter les biais et le risque de sur-apprentissage, l'évaluation se fait en validation croisée, comme détaillé chapitre 6 : une partie des traces est utilisée pour assister la construction (jeu d'apprentissage), et le reste pour le traitement du diagnostic et le calcul des critères de comparaison (jeu de test). Ce découpage est fait aléatoirement sur les identifiants des apprenants (i.e. l'ensemble des traces d'un apprenant ne peut être coupé) de sorte que le jeu d'apprentissage représente environ 60 % de la base de traces.

Notre plateforme intègre plusieurs bases de données dont nous devons détailler le contenu lors de l'expérimentation. En premier lieu, la base des techniques de diagnostic (modèle+implémentation) est décrite Table 13.

Technique de diagnostic		**Traitement du diagnostic**		
Modèle de diagnostic	**Implémen-tation**	**Entrée**	**Traitement**	**Sortie**
Constraint-based	Contraintes	Valeurs de variables logiques	Moteur d'inférence et fonction puissance	Probabilité de violer une contrainte
Control-based	Réseau bayésien	Valeur des nœuds d'évidence dans le réseau	Inférence bayésienne	Tables de probabilités des nœuds « Contrôles »
Knowledge Tracing	Modèle de Markov caché	Valeur des nœuds non cachés	Inférence bayésienne	États des nœuds cachés
	Additive factor Model	Évaluation de la réponse de l'apprenant	Régression	Probabilités de maîtriser les connaissances
	Performance Factor Model	Historiques des évaluations de la réponse de l'apprenant	Régression	Probabilités de maîtriser les connaissances
	Logique floue	Valeur des variables logiques	Moteur d'inférence floue	État et valeurs des variables floues

Table 13 : Techniques de diagnostic utilisées dans nos expérimentations, identifiées par leur modèle de diagnostic et leur implémentation dans les deux colonnes de gauche. La partie droite du tableau décrit les entrées, sorties et types de traitement pour chaque technique.

La base des critères contient l'ensemble des critères décrits dans le chapitre 6. Un seul de ces critères requiert des add-on d'application : le critère G « Complexité de la technique ». Nous avons donc, pour chacune des six implémentations, un add-on capable d'extraire les informations requises par ce critère.

Il convient de noter que la technique « Knowledge tracing + logique floue », que nous avons proposée sur la base d'une collaboration avec un stagiaire (Goel et al., 2012), n'a pas été utilisée sur l'ensemble des évaluations en raison des limites techniques de l'implémentation.

Le code des implémentations est en JAVA pour le réseau bayésien, le modèle de Markov et les contraintes, en JAVA et C++ pour la logique floue, en JAVA et R pour les régressions. Les librairies partiellement ou totalement utilisées sont les suivantes :

- Contraintes : Drools[10]
- Réseau bayésien : Smile[11]
- Régression : package stats[12] et nlme[13]
- Logique floue : FisPro (Guillaume et al., 2002)

II. Bases de traces d'apprenants

Nos expérimentations ont porté sur quatre domaines d'apprentissage différents : la géométrie des aires (traces de l'EIAH Geometry Tutor), l'apprentissage de la lecture de l'anglais (traces de l'EIAH Reading Tutor), la chirurgie orthopédique (traces de l'EIAH TELEOS) et l'algèbre (traces simulées de l'EIAH APLUSIX).

Nous allons donner quelques détails sur chacun de ces domaines et les traces utilisées.

1. Geometry Tutor

Le Geometry Tutor (ou Cognitive Geometry Tutor) (Anderson et al., 1985) est destiné à l'apprentissage de la géométrie pour des apprenants du secondaire. Nous nous restreignons à la partie du tuteur dédiée à l'apprentissage de la géométrie des aires. La Figure 32 donne un exemple des exercices pratiqués par les apprenants.

Les traces du Geometry Tutor sont accessibles dans Datashop (Koedinger et al., 2010), une base de données pour la collecte de traces de tuteurs cognitifs. Une trace dans Datashop, nommée transaction, est un vecteur composé des variables suivantes (nous en avons omis certaines pour simplification) :

- Numéro anonymisé de l'apprenant*
- École de l'apprenant et sa classe
- Date et heure (timestamp)*
- Durée écoulée depuis la précédente transaction
- Catégorie du problème
- Problème en cours de résolution*
- Étape courante de résolution du problème effectuée par l'apprenant*
- Évaluation de la réponse courante, qui peut être « correcte » ou « incorrecte »*

[10] http://www.jboss.org/drools/
[11] http://genie.sis.pitt.edu/
[12] http://stat.ethz.ch/R-manual/R-patched/library/stats/html/00Index.html
[13] http://cran.r-project.org/web/packages/nlme/index.html

- Objet graphique sélectionné par l'apprenant dans l'interface du tuteur
- Réponse brute saisie par l'apprenant dans l'interface*
- Rétroaction éventuellement retournée par le tuteur
- Knowledge Component, i.e. connaissance mobilisée par l'apprenant pour résoudre une étape, tel que spécifié par le modèle des connaissances (KC Model, cf. l'état de l'art)*

Les champs marqués d'une étoile * sont obligatoires, les autres facultatifs.

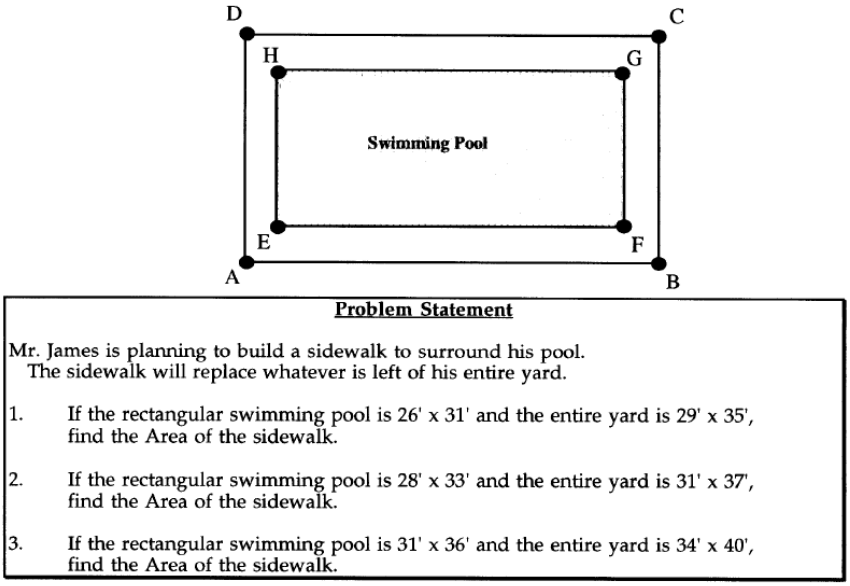


Figure 32 : Exemple d'énoncé d'un exercice de géométrie proposé par le Geometry Tutor.

Nous remarquons donc qu'une trace de Datashop présente les caractéristiques suivantes :

- Une trace est une réponse d'un apprenant à une étape de résolution d'un problème.
- Le résultat du diagnostic comportemental est inclus (évaluation de la réponse courante, correcte ou incorrecte), donc ce sont des traces enrichies selon la définition donnée en chapitre 3.
- Les traces sont temporellement situées
- Les traces incluent pour chaque étape de résolution d'un problème la connaissance qui doit être mobilisée, en se référant à un modèle des connaissances du domaine fourni par des experts.

Techniquement, Datashop est accessible via un Webservice.

2. Reading Tutor

Le Reading Tutor (Mostow et Aist, 2001) destiné à l'apprentissage de la lecture de l'anglais en milieu anglophone pour des enfants à l'école élémentaire (l'objectif n'est pas l'apprentissage de l'anglais mais l'apprentissage du mécanisme de la lecture). Le Reading Tutor propose des exercices de lecture de courtes histoires, lues à voix haute par l'apprenant. L'EIAH inclut un outil de reconnaissance vocale capable de reconnaître, modulo

une marge d'erreur, si les mots lus par l'apprenant sont ceux attendus. L'illustration de la Figure 33 montre un exercice de lecture en cours.

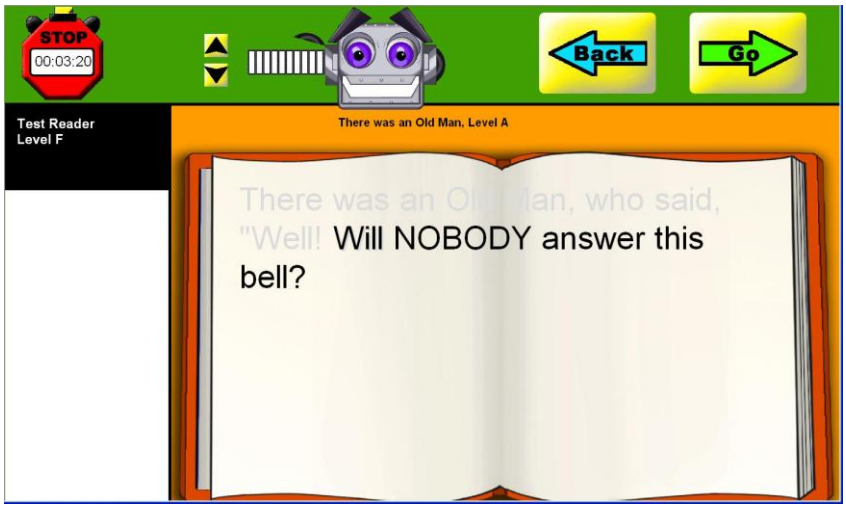


Figure 33 : Capture d'écran du Reading Tutor montrant un exercice de lecture en cours de résolution.

Les traces du Reading Tutor sont entreposées dans une base de données PostgreSQL. La table contenant les traces de l'apprenant possède le format suivant (nous avons omis certains éléments pour simplification) :

- ID (clé primaire)
- ID de l'utilisateur (clé étrangère)*
- ID de l'histoire lue par l'apprenant (clé étrangère)*
- Temps de début de lecture du mot*
- Temps de fin de lecture du mot*
- Mot courant que l'apprenant doit lire dans l'histoire*
- Mot reconnu par la reconnaissance vocale (vide si non reconnu)
- Place du mot dans la phrase*
- Temps d'hésitation avant la lecture du mot (0=pas d'hésitation)
- Un booléen indiquant si l'apprenant a reçu une aide de l'EIAH
- Un booléen indiquant si l'apprenant a demandé une aide
- Type d'aide donnée à l'apprenant (vide si pas d'aide)
- Les connaissances requises pour lire le mot courant, qui sont l'association entre les graphèmes du mot et les phonèmes correspondant à chaque graphème (l'association entre un graphème et un phonème est abrégée GtoP)*
- La fréquence du mot dans la langue anglaise

Les champs marqués d'une étoile * sont obligatoires, les autres facultatifs.

Nous remarquons donc qu'une trace du Reading Tutor présente les caractéristiques suivantes :

- Une trace est un mot d'une histoire lu par un apprenant.

- Le résultat du diagnostic comportemental est inclus (reconnaissance du mot lu à voix haute par l'apprenant, à comparer avec le mot attendu), donc ce sont des traces enrichies selon la définition donnée en chapitre 3.
- Les traces sont temporellement situées.
- Les traces incluent pour chaque mot les connaissances requises pour lire ce mot, en se référant à un modèle des connaissances du domaine fourni par des experts.

Techniquement, la base de données se trouve sur un serveur privé de l'université Carnegie Mellon, accessible par ODBC.

3. TELEOS

TELEOS (Vadcard et Luengo, 2004) est destiné à l'apprentissage de la conduite d'opérations de chirurgie orthopédique par les internes en médecine. L'apprenant doit réaliser des exercices de chirurgie via une interface 3D du corps humain. Nous nous concentrons sur une opération chirurgicale proposée par TELEOS, nommée vertébroplastie. La Figure 34 donne un aperçu de l'interface pour ce problème.

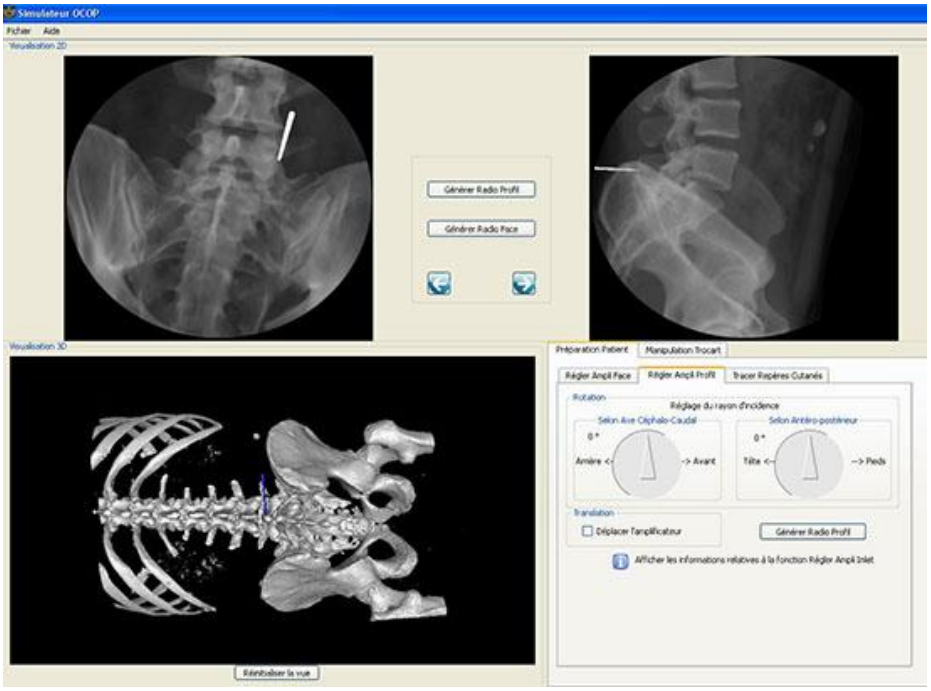


Figure 34 : Capture d'écran de TELEOS montrant l'interface de simulation 2D et 3D permettant de résoudre un exercice de chirurgie orthopédique.

Nous utilisons des traces de TELEOS collectées lors d'une expérimentation et stockées dans un fichier CSV. Le format des traces est le suivant (quelques éléments sont omis pour simplification) :

- Nom de l'apprenant*
- Vertèbre opérée*
- Date et heure*
- Action effectuée par l'apprenant*

- Connaissances liées à l'action (selon un modèle du domaine fourni par des experts)*
- Évaluation de l'état du problème consécutivement à l'action de l'apprenant (correct, incorrect, incorrect en cours de remédiation ou incorrect en aggravation)*
- Les registres de représentation dans lesquels sont exprimés le problème et l'action*

Les champs marqués d'une étoile * sont obligatoires, les autres facultatifs.

Nous remarquons donc qu'une trace de TELEOS présente les caractéristiques suivantes :

- Une trace est une action effectuée par un apprenant dans l'EIAH.
- Le résultat du diagnostic comportemental est inclus (évaluation de l'état du problème), donc ce sont des traces enrichies selon la définition donnée en chapitre 3.
- Les traces sont temporellement situées.
- Les traces incluent pour chaque action les connaissances requises pour contrôler la validité de cette action, en se référant à un modèle des connaissances du domaine fourni par des experts.

4. APLUSIX

APLUSIX (Chaachoua et al., 2004) est destiné à l'apprentissage de l'algèbre au secondaire. Il propose à l'apprenant des exercices algébriques à résoudre pas à pas. Nous avons considéré dans notre expérimentation des exercices de factorisation. Il est à noter que, pour des raisons techniques, nos traces ont été simulées par une didacticienne afin d'obtenir des traces enrichies. L'illustration de la Figure 35 montre un exercice d'algèbre en cours de résolution dans APLUSIX.

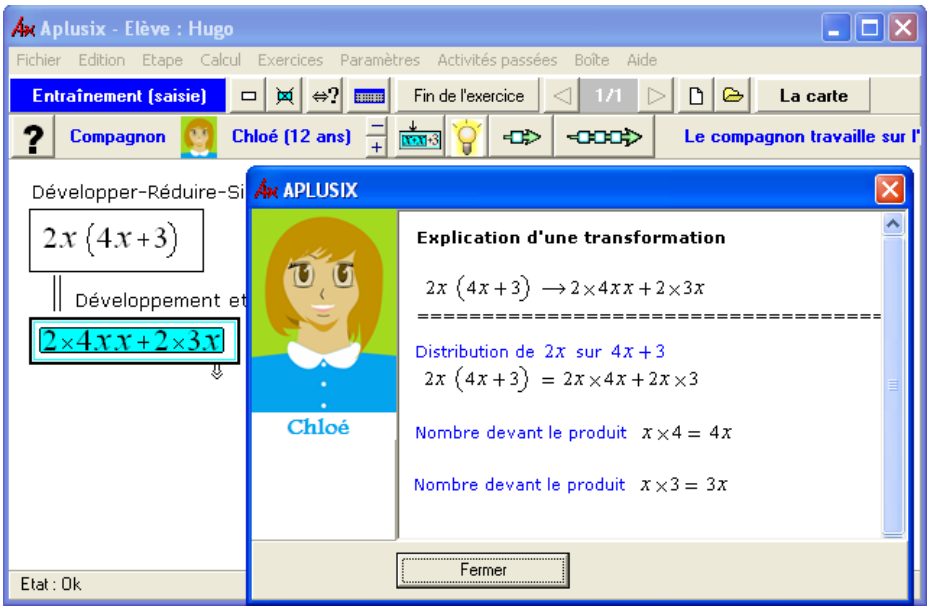


Figure 35 : Capture d'écran de APLUSIX lors de la résolution d'un exercice d'algèbre (développement et réduction d'une expression).

Les traces simulées sont stockées dans un fichier CSV contenant les informations principales suivantes (les détails secondaires sont omis pour simplification) :

- Numéro de l'exercice*
- Numéro de l'apprenant*
- Conformité du résultat final fourni par l'apprenant à l'exercice

Ensuite, pour chaque exercice, la didacticienne a fourni dans les fichiers CSV les informations suivantes :

- La technique mathématique courante mobilisée par l'apprenant*
- Le contexte courant représentant l'état de l'exercice*
- La règle permettant d'évaluer la validité mathématique du résultat courant*
- La valeur retournée par la règle (correct ou incorrect)*

Les champs marqués d'une étoile * sont obligatoires, les autres facultatifs.

Nous remarquons donc qu'une trace simulée d'APLUSIX présente les caractéristiques suivantes :

- Une trace est une technique mathématique mobilisée pour une étape de résolution d'un problème.
- Le résultat du diagnostic comportemental est inclus (règle permettant d'évaluer la validité mathématique du résultat courant), donc ce sont des traces enrichies selon la définition donnée en chapitre 3.
- Les traces sont temporellement situées.
- Les traces incluent pour chaque étape de résolution la technique mathématique mobilisée.

5. Collecte et volumétrie

Le volume de traces, telles que définies pour chaque EIAH ci-dessus, est résumé ci-dessous (Table 14). Nous constatons qu'une base de trace inclut un nombre élevé de traces (pour le Reading Tutor), deux bases ont un nombre de traces intermédiaire compris entre 2000 et 7000 (Geometry Tutor et TELEOS), et la base simulée d'APLUSIX est la plus réduite. Le nombre de traces collectées dépend principalement du coût et des moyens mobilisés pour la collecte :

- Pour le Geometry Tutor, les traces correspondent à une année d'utilisation dans une école étasunienne.
- Pour le Reading Tutor, des traces ont été collectées dans plusieurs écoles de la région de Pittsburgh aux États-Unis sur plus de dix ans (nous avons sélectionné pour l'étude les traces sur une année d'un échantillon d'apprenants). Le volume de traces est donc le plus fort.
- Pour TELEOS, les apprenants sont des internes en médecine qui ont des contraintes extrêmement fortes dans leur travail et qui sont peu nombreux dans la spécialité visée par TELEOS (chirurgie orthopédique).

- Enfin, APLUSIX est une base de traces simulées par une didacticienne des mathématiques, comme détaillé plus haut.

EIAH	Nombre de traces	Nombre d'apprenants	Nombre d'exercices différents
Geometry Tutor	6778	59	40
Reading Tutor	240 204	96	143
TELEOS	2695	7	4
APLUSIX	329	6	24

Table 14 : Volumes, nombre d'apprenants et nombre d'exercices différents pour les quatre bases de traces utilisées dans les expérimentations.

III. Première expérimentation : évaluation des méthodes d'assistance

1. Méthodologie

La première expérimentation visait à évaluer notre plateforme comme une preuve de concept de nos méthodes d'assistance. Pour cela, nous vérifions s'il est possible d'utiliser nos méthodes sur différents jeux de traces et d'obtenir des résultats de comparaison pour des techniques de diagnostic ayant un modèle de diagnostic différent, ce que nous avons identifié comme un verrou dans l'état de l'art. D'un point de vue méthodologique, nous appliquons nos méthodes d'assistance à la construction et à la comparaison sur trois bases de traces d'apprenant (Geometry Tutor, Reading Tutor et TELEOS, présentés plus haut) et appliquons divers critères de comparaison.

Les cinq techniques de diagnostic listées dans la Table 13 ont été construites manuellement par nos soins pour les traces du Geometry Tutor à partir des éléments fournis dans Datashop : liste des exercices avec leur solution et liste des connaissances identifiées dans le domaine.

Nous avons construit ces mêmes techniques, excepté le Knowledge tracing implémenté avec la logique floue, au moyen de notre plateforme, i.e. en utilisant les techniques décrites dans le chapitre 4, pour les traces du Reading Tutor et de TELEOS.

Dans notre plateforme, la première étape pour la construction assistée des techniques de diagnostic est donc la construction d'une ontologie du domaine. Pour cela, nous avons défini deux ontologies des traces et des connaissances pour chaque domaine : une de haut niveau et une détaillée. L'ontologie de haut niveau du Reading Tutor (Figure 36) décrit cinq observables : Story (l'exercice courant), Word (le mot lu), Graphem (les graphèmes du mot), Regular (un booléen indiquant si le mot est régulier ou non en anglais) et Common (la fréquence d'apparition du mot dans la langue anglaise) ; l'ontologie décrit une capacité : GtoP (l'association entre un graphème et un phonème). Ces éléments ont été décrits plus haut (paragraphe II.2) lorsque nous avons présenté les traces. L'ontologie de haut niveau de TELEOS (Figure 37) décrit deux observables (Problem et Operator) et une capacité (Control),

tous décrits au paragraphe II.3. Les ontologies doivent être associées aux traces par le concepteur. Nous avions présenté dans le chapitre 4 une association pour TELEOS, et nous montrons ici sur la Figure 38 l'association entre les traces du Reading Tutor et l'ontologie de haut niveau de la Figure 36.

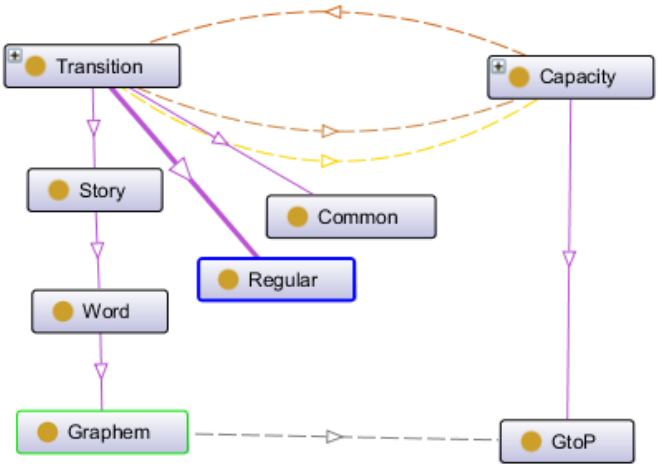


Figure 36 : Ontologie de haut niveau utilisée pour la construction des techniques de diagnostic pour les traces du Reading Tutor.

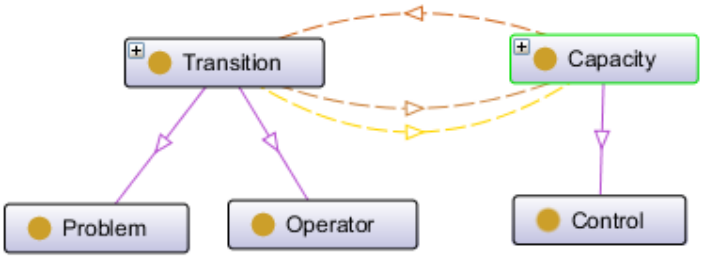
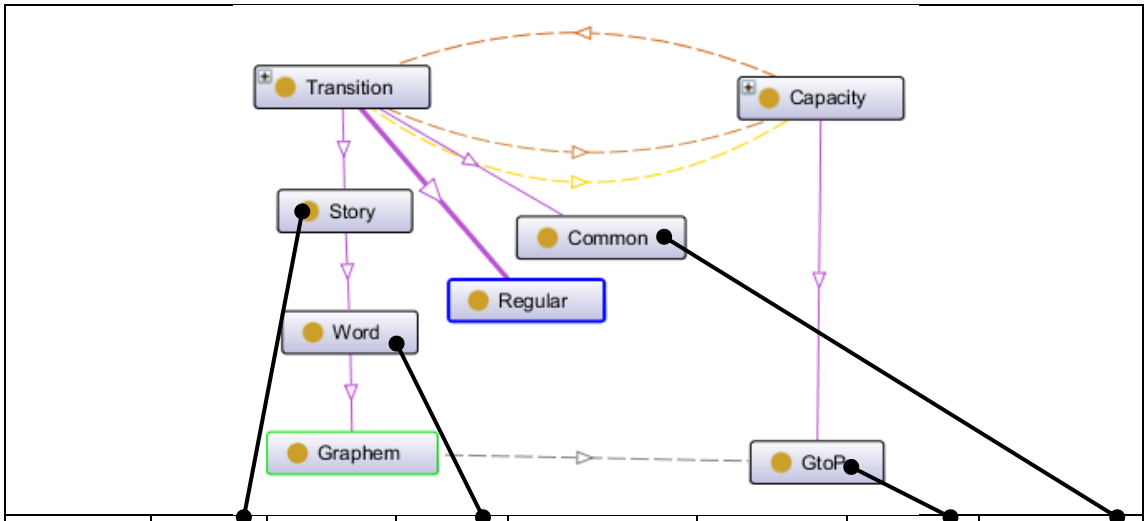


Figure 37 : Ontologie de haut niveau utilisée pour la construction des techniques de diagnostic pour les traces de TELEOS.

Comme détaillé plus haut, les connaissances du domaine de ces deux EIAH (Reading Tutor et TELEOS) sont incluses dans les traces. Nous avons construit les ontologies détaillées en sélectionnant aléatoirement 40 % des connaissances du domaine et en les représentant dans les ontologies, et nous avons ajouté respectivement les graphèmes pour le Reading Tutor et les opérateurs pour TELEOS, qui sont associés aux connaissances dans l'ontologie.

Dans le cadre de cette expérimentation, la logique floue n'a été utilisée que pour les traces du Geometry Tutor. En effet, elle a été implémentée par un stagiaire dans le cadre d'un travail annexe sur l'implémentation de la logique floue pour le Knowledge tracing et expérimentée sur les traces du Geometry Tutor. Seule la lecture des traces et le traitement du diagnostic ont été implémentés, les paramètres ayant été fixés manuellement. Toutefois, nous indiquons les résultats de cette technique en géométrie, car l'objet de cette expérimentation est de fournir une preuve de concept sur la possibilité de construire et comparer plusieurs techniques de diagnostic ayant un modèle de diagnostic différent.



Utilisateur	Histoire	Temps	Mot	Mot reconnu	Hésitation	GtoP	Fréquence
Fm1	S1.a	15:55	The	The	0	Th:TH	1.2
Fm1	S1.a	15:55	The	The	0	e:E	1.2
Fm1	S1.a	15:57	cat	-	2000	c:K	1.6
Fm1	S1.a	15:57	cat	-	2000	a:A	1.6
Fm1	S1.a	15:57	cat	-	2000	t:T	1.6

Figure 38 : Association entre les traces du Reading Tutor (partiellement représentées) et l'ontologie de la Figure 36.

Puisque nous avons construit toutes les techniques que nous allons comparer dans cette partie, cette expérimentation vise à présenter une preuve de concept pour la construction et la comparaison de techniques de diagnostic assistées par notre méthode. En revanche, elle ne vise pas à étudier l'utilisabilité, l'accessibilité ou l'utilité de nos travaux.

2. Évaluation sur l'application des critères de comparaison

La Table 15 donne les résultats des différents critères décrits dans le chapitre 5, à l'exception de la corrélation avec un diagnostic externe, ainsi que l'impact des techniques sur un type d'aide (qui sera lui abordé plus bas). Les critères de la Table 15 sont calculés après que la plateforme a appliqué chaque technique de diagnostic sur la base de test (cf. explication chapitre 6).

La Table 16 donne les résultats du critère « corrélation avec un diagnostic externe » qui requiert pour être calculé l'apport d'un diagnostic externe, comme expliqué chapitre 5. Ici, le critère est appliqué sur les techniques instanciées sur les traces du Geometry Tutor, et le diagnostic externe servant de référence est basé sur celui de Datashop.

La lecture des résultats domaine par domaine permet de positionner factuellement les techniques. On constate premièrement que les techniques ayant la meilleure précision et le meilleur AUC sont le Knowledge tracing + modèle de Markov caché pour le Reading Tutor et le Geometry Tutor, et le Control based + réseau bayésien dynamique pour TELEOS. Les intervalles de confiance à 95 % pour les précisions se recouvrent toujours pour au moins les trois techniques les plus précises. Deux intervalles de confiance qui se recouvrent signifient que la différence entre les deux techniques n'est statistiquement pas significative. Nous

notons ensuite que les AIC et les temps d'exécutions pénalisent le Control-based pour les trois domaines, en raison de la plus grande complexité en termes de nombre de paramètres de son réseau bayésien dynamique. Les entropies de Shannon sont les plus élevées pour le Control-based. Les sensibilités aux manques dans les données sont les plus faibles pour le Knowledge tracing + Modèle de Markov caché, et sont les plus faibles pour le Reading Tutor, probablement car il s'agit du domaine où le nombre de traces est significativement le plus élevé.

Technique de diagostic	**Modèle de diagnostic**	Knowledge Tracing			Constraint-based	Control-based
	Implémen-tation	Modèle de Markov caché	Additive Factor Model	Logique floue	Contraintes	Réseau bayésien dynamique
TELEOS	Précision (±95%)	71% (±2,7%)	70% (±2,8%)		73% (±3,6%)	75% (±3,3%)
	RMSE	0.32	0.32		0.31	0.30
	AIC	7829	7897		7305	15 194
	AUC	0.71	0.67		0.71	0.73
	Temps (s)	0.03	0.03		0.03	0.05
	Complexité	18 427	11 512		8804	26 651
	Entropie	0.3	0.28		0.31	0.33
	Sensibilité	2.6	2.8		3.4	3.2
Reading Tutor	Précision (±95%)	78% (±3,2%)	72% (±3,9%)		72% (±4,1%)	74% (±3,5%)
	RMSE	0.57	0.6		0.61	0.58
	AIC	226 126	224 447		220 003	244 655
	AUC	0.68	0.67		0.66	0.65
	Temps (s)	1.4	1.2		1.1	2.4
	Complexité	146 880	139 281		130 940	178 901
	Entropie	0.23	0.22		0.22	0.25
	Sensibilité	1.3	1.4		1.9	1.6
Geometry Tutor	Précision (±95%)	58,8% (±5,1%)	57.8% (±5,2%)	30,3% (±8.1%)	54,2% (±4,1%)	49,1% (±6.3%)
	RMSE	0.73	0.76	0.86	0.77	0.92
	AIC	14 729	14 815	20 018	14 901	19 753
	AUC	0.71	0.7	0.54	0.7	0.67
	Temps (s)	0.04	0.04		0.04	0.05
	Complexité	28 320	26 240		22 160	56 640
	Entropie	0.42	0.43	0.56	0.41	0.49
	Sensibilité	1.9	1.8		2.5	2.4

Table 15 : Résultats des critères de comparaison pour les cinq techniques de diagnostic et les trois bases de traces. Intervalles de confiance à 95 % entre parenthèse pour la précision. Les meilleurs résultats sont : la valeur la plus élevée pour la précision, AUC et l'entropie ; la valeur la plus faible pour RMSE, AIC, le temps d'exécution, la complexité, la sensibilité.

Concernant les résultats de la Table 16, il convient de noter que le diagnostic externe est basé sur le diagnostic de Datashop, qui est en réalité le Knowledge tracing implémenté avec AFM, faute d'avoir pu recourir à un expert durant cette expérimentation. Les résultats sont donc biaisés, mais ont pour but d'illustrer l'application de ce critère. Comme nous l'avons avancé au chapitre 5, l'interprétation de ces résultats dépend des objectifs du concepteur, car tous les critères de comparaison ne sont pas pertinents selon ses objectifs. Par exemple, la sensibilité aux manques dans les données n'est utile que lorsqu'il existe un risque de ne pas pouvoir observer toutes les traces, par exemple parce que le domaine est complexe ou a recourt à des dispositifs externes. De même, la précision, RMSE, AUC et AIC sont quatre mesures évaluant la précision de la prédiction des techniques, mais AIC est la seule à prendre en compte la complexité des techniques. En cas de contradictions significatives entre ces quatre mesures, il revient au concepteur d'analyser ces contradictions afin de sélectionner les mesures qui répondent le mieux à ses questions de recherche.

Modèle de diagnostic	Implémentation	Corrélation
Knowledge Tracing	Modèle de Markov caché	0.72
	Additive Factor Model	0.79
	Logique floue	0.63
Constraint-based	Contrainte	0.3
Control-based	Réseau bayésien dynamique	0.3

Table 16 : Résultats du critère de comparaison « Corrélation avec un diagnostic externe » pour les traces du Geometry Tutor.

L'interprétation des résultats peut également se faire en recalculant les critères sur un sous-ensemble de la base de test : par exemple pour une connaissance, un apprenant, un exercice, le groupe des apprenants faisant le plus d'erreur… Ainsi, par exemple, nous sélectionnons pour le Reading Tutor la connaissance « calcul de la hauteur d'un trapèze » du Geometry Tutor, qui est la connaissance pour laquelle le nombre de traces est le plus faible, et recalculons la précision de la prédiction (Table 17). Le résultat permet de positionner les techniques en fonction de leur précision pour une connaissance à partir du plus faible nombre de traces (sachant qu'un faible nombre de traces rend la prédiction plus difficile du point de vue statistique).

Modèle de diagnostic	Implémentation	Précision
Knowledge Tracing	Modèle de Markov caché	61%
	Additive Factor Model	61%
	Logique floue	32%
Constraint-based	Contrainte	52%
Control-based	Réseau bayésien dynamique	44%

Table 17 : Résultat du critère « précision de la prédiction » sur le sous-ensemble des traces du Geometry Tutor où la connaissance « calcul de la hauteur d'un trapèze » est présente.

Pour interpréter les résultats des critères sur des sous-ensembles de traces, il est également intéressant de recalculer les critères sur les traces de chaque apprenant de la base de test. En effet, du point de vue des EIAH, les techniques de diagnostic visent à inférer le modèle de chaque apprenant, comme défini en introduction. Du point de vue statistique, les résultats de la Table 18 sont donnés en moyenne sur l'ensemble des traces, ce qui tend à donner plus de poids aux apprenants ayant le plus de traces. À titre d'illustration, nous sélectionnons au hasard dix apprenants dans la base de test du Reading Tutor et recalculons quelques critères, dont les résultats sont donnés Table 18. Ces étudiants sont pour la plupart proches des valeurs moyennes (dans l'intervalle de confiance à 95% pour la précision de la prédiction). On remarque en revanche deux apprenants qui divergent des autres : le 3 et le 7. En étudiant les traces, nous constatons que ces apprenants ont un taux de mots lus correctement (61 % pour l'apprenant 3, 56 % pour l'apprenant 7) significativement inférieurs au taux moyen (74 %). De plus, la base de traces compte seulement 248 traces pour l'apprenant 7, bien inférieur au nombre moyen de traces par apprenant (2502). Ces résultats apprenant par apprenant permettent donc de positionner les techniques plus finement, i.e. sur des sous-ensembles de traces plus ciblés, ici par apprenant, plus haut par connaissance ou par diagnostic comportemental. Nous insistons sur le fait qu'il appartient toujours au concepteur d'interpréter ces résultats, et de décider si les sous-ensembles de traces sélectionnés sont pertinents ou non. Il convient également de noter que l'interface de notre plateforme n'offre pas encore la possibilité de sélectionner un sous-ensemble de traces.

Technique de diagostic	**Modèle de diagnostic**	Knowledge Tracing		Constraint-based	Control-based
	Implémen-tation	Modèle de Markov caché	Additive Factor Model	Contraintes	Réseau bayésien dynamique
Apprenant 1	Précision (±95%)	80% (±1%)	73% (±0.9%)	72% (±0.9%)	76% (±1.1%)
	RMSE	0.37	0.39	0.41	0.38
	AUC	0.72	0.70	0.69	0.74
	Entropie de Shannon	0.22	0.21	0.21	0.26
Apprenant 2	Précision (±95%)	83% (±1.1%)	71% (±0.9%)	74% (±0.8%)	79% (±1.2%)
	RMSE	0.39	0.4	0.41	0.39
	AUC	0.73	0.71	0.69	0.73
	Entropie de Shannon	0.26	0.22	0.23	0.27
Apprenant 3	Précision (±95%)	72% (±2,2%)	65% (±2,9%)	66% (±2,1%)	72% (±2,5%)
	RMSE	0.44	0.49	0.48	0.44
	AUC	0.67	0.64	0.64	0.65
	Entropie de Shannon	0.28	0.27	0.27	0.32

Apprenant 4	Précision (±95%)	78% (±0.9%)	70% (±1.2%)	72% (±1%)	74% (±1.2%)
	RMSE	0.36	0.44	0.42	0.39
	AUC	0.72	0.69	0.69	0.7
	Entropie de Shannon	0.23	0.22	0.2	0.25
Apprenant 5	Précision (±95%)	77% (±1.2%)	71% (±1%)	71% (±1.1%)	73% (±1.4%)
	RMSE	0.37	0.45	0.44	0.41
	AUC	0.73	0.69	0.68	0.71
	Entropie de Shannon	0.23	0.22	0.23	0.25
Apprenant 6	Précision (±95%)	79% (±1%)	72% (±1.1%)	72% (±0.9%)	74% (±1.4%)
	RMSE	0.4	0.44	0.45	0.42
	AUC	0.74	0.70	0.71	0.72
	Entropie de Shannon	0.24	0.22	0.23	0.27
Apprenant 7	Précision (±95%)	62% (±2,4%)	54% (±2,8%)	57% (±2,6%)	60% (±2,9%)
	RMSE	0.48	0.52	0.49	0.49
	AUC	0.66	0.63	0.66	0.65
	Entropie de Shannon	0.29	0.26	0.31	0.31
Apprenant 8	Précision (±95%)	78% (±0.9%)	70% (±1.2%)	70% (±1.1%)	75% (±1.3%)
	RMSE	0.39	0.45	0.45	0.41
	AUC	0.73	0.68	0.68	0.73
	Entropie de Shannon	0.24	0.23	0.23	0.27
Apprenant 9	Précision (±95%)	80% (±0.7%)	72% (±1%)	73% (±0.9%)	76% (±1.3%)
	RMSE	0.36	0.41	0.4	0.38
	AUC	0.75	0.7	0.71	0.74
	Entropie de Shannon	0.21	0.2	0.21	0.26
Apprenant 10	Précision (±95%)	78% (±0.8%)	70% (±0.8%)	71% (±0.7%)	74% (±1.2%)
	RMSE	0.38	0.43	0.42	0.4
	AUC	0.73	0.68	0.68	0.71
	Entropie de Shannon	0.2	0.18	0.19	0.24

Table 18 : Résultat de quatre critères (précision, RMSE, AUC, entropie) pour les traces de dix apprenants dans la base de test du Reading Tutor.

Une autre possibilité est de sélectionner les sous-ensembles des traces en fonction du diagnostic comportemental. Xu et Mostow (Xu et Mostow, 2012) ont par exemple montré

que sur les traces du Reading Tutor, les résultats des techniques de diagnostic sont plus précis lorsque l'apprenant lit le mot de façon correcte. Nous séparons donc la base de test du Reading Tutor en deux sous-ensembles : mot lu correctement et mot lu incorrectement, et recalculons quelques critères (Table 19). Les résultats obtenus sont similaires à ceux de Xu et Mostow : la précision de la prédiction, RMSE et AIC sont tous significativement plus élevés lorsque le mot est lu correctement dans les traces. La précision des techniques chute fortement pour les mots lus incorrectement. Par ailleurs, les intervalles de confiance ne se recouvrent pas dans le cas où les mots sont lus incorrectement entre d'une part Knowledge tracing + Modèle de Markov caché et Control-based, d'autre part Knowledge tracing + AFM et Constraint-based, indiquant qu'il y a cette fois une différence significative entre les performances du Knowledge tracing + Modèle de Markov caché et du Control-based par rapport aux deux autres.

Technique de diagostic	**Modèle de diagnostic**	Knowledge Tracing		Constraint-based	Control-based
	Implémen-tation	Modèle de Marlov caché	Additive Factor Model	Contraintes	Réseau bayésien dynamique
Mots lus de façon correcte	Précision (±95%)	97% (±0.7%)	95% (±0.89%)	94% (±1.02%)	96% (±1.13%)
	RMSE	0.39	0.42	0.42	0.41
	AIC	208 910	207 131	206 576	225 301
Mots lus de façon incorrecte	Précision (±95%)	37% (±3,7%)	27% (±4,3%)	26% (±4,2%)	35% (±3,3%)
	RMSE	0.52	0.61	0.63	0.54
	AIC	243 907	243 186	242 541	272 002

Table 19 : Résultats de trois critères (précision, RMSE, AIC) pour les traces du Reading Tutor séparées en deux sous-ensembles : mot lu correctement par l'apprenant et mot lu incorrectement, sachant qu'une trace est un mot lu par l'apprenant dans cet EIAH.

Nous pouvons nous intéresser dans ces résultats aux différences en fonction des domaines et des traces. On note premièrement que les traces du Geometry Tutor et du Reading Tutor ont été collectées dans l'optique d'utiliser le Knowledge tracing, et les traces de TELEOS pour utiliser le Control-based. Or, la précision, RMSE, AUC, AIC sont meilleurs pour le Knowledge tracing appliqué au Geometry turor et au Reading Tutor, et la précision, RMSE et AUC sont meilleurs pour le Control-based appliqué à TELEOS. Cependant, ces différences de précision ne sont pas significatives sur nos exemples, et on constate que le Knowledge tracing + Additive factor Model est moins performant que Knowledge tracing + modèle de Markov caché et Control-based pour le Reading Tutor. D'autre part, le domaine de TELEOS (chirurgie orthopédique) est un domaine mal défini (*ill defined* en anglais) au sens de Simmons (Simon, 1978, 1974), alors que le Reading Tutor et le Geometry Tutor n'en sont pas. En conclusion, la nature du domaine (*ill defined* ou non) et le format des traces peuvent avoir influé sur les résultats des critères, mais il est impossible de conclure dans quelle proportion et avec quelle signifiance. La volumétrie n'a en revanche pas importé sur nos résultats, puisque les

bases de TELEOS et du Geometry Tutor ont un volume similaire mais positionnent différemment les techniques.

Nous nous bornons ici à illustrer factuellement les biais que l'on peut identifier sur les performances des techniques et les résultats des critères (nature du domaine, format et quantité des traces), mais il est impossible et non pertinent de généraliser ces observations à partir de ces résultats.

3. Évaluation de la méthode de construction

Comme détaillé dans la présentation de la méthodologie, nous avons utilisé notre méthode d'assistance à la construction d'une technique de diagnostic pour les traces du Reading Tutor et de TELEOS, au moyen de deux ontologies pour chaque EIAH, une de haut niveau et une détaillée. Dans le cadre de cette expérimentation, nous évaluons donc la construction via trois aspects : l'impact du niveau de l'ontologie (haut niveau ou détaillé), la validité des techniques construites en rapport au domaine et les propriétés de l'algorithme d'apprentissage semi-automatique (convergence, temps d'exécution).

Nous évaluons d'abord l'impact des ontologies en calculant le résultat des critères de précision de prédiction et le RMSE (Table 20). Ces deux critères sont en effet aussi bien utilisés pour évaluer des techniques de diagnostic des connaissances que les résultats d'algorithmes d'apprentissage automatiques ou semi-automatiques. En effet, nous avons utilisé une méthode de recherche locale par score pour l'instanciation des techniques, et l'algorithme EM pour l'apprentissage des paramètres. Les scores statistiques et l'algorithme EM visent à maximiser la précision ou la vraisemblance de la technique construite, donc les mesures de précision de prédiction (précision, RMSE, BIC, AUC) sont utilisées pour évaluer les résultats des algorithmes d'apprentissage automatique. Du point de vue de l'interprétation, des précisions de prédictions significativement différentes entre les techniques construites pour l'ontologie détaillée et pour l'ontologie de haut niveau montreraient que le niveau d'ontologie a un impact significatif sur l'algorithme d'apprentissage, via le score biaisé *fS'* (cf. chapitre 4).

Pour les traces de TELEOS, les techniques construites avec l'ontologie détaillée donnent une meilleure précision, un meilleur RMSE et un meilleur AIC que la même technique construite avec l'ontologie de haut niveau. L'écart de précision est significatif pour le Knowledge tracing + modèle de Markov caché et pour le Control-based, les intervalles de confiance à 95 % ne se recouvrant pas. Cela signifie que les techniques construites avec l'ontologie détaillée sont significativement plus précises que les techniques avec l'ontologie de haut niveau pour le Knowledge tracing + modèle de Markov caché et pour le Control-based sur ces traces. Par extension, le score biaisé *fS'* que nous proposons (chapitre 4) peut avoir un impact significatif sur la précision de prédiction des techniques construites, selon le niveau de l'ontologie.Pour les traces du Reading Tutor, les précisions, RMSE et AIC des techniques construites avec les ontologies de haut niveau et détaillées sont quasiment similaires, et il n'y a aucune amélioration significative selon les intervalles de confiance. Ces résultats signifient que l'ontologie détaillée n'a pas permis d'obtenir de construire des techniques plus précises pour le Reading Tutor, contrairement à TELEOS. L'algorithme d'apprentissage

est guidé par l'ontologie et par les traces : dans le cas du Reading Tutor, il n'y avait pas besoin d'aller au niveau détaillé pour décrire les traces. Une hypothèse sur la différence entre les deux domaines est le volume de traces, bien plus élevé dans le Reading Tutor (cf. plus haut paragraphe II. 5), et le domaine, mal défini pour TELEOS et bien défini pour le Reading Tutor, comme montré plus haut.

Notons qu'en exécutant l'algorithme de construction sans usage des ontologies (donc de façon totalement automatique), nous avons mesuré une baisse de la précision de la prédiction de 16 % en moyenne et un RMSE plus élevé de 0,14 en moyenne sur toutes les techniques (par rapport aux résultats Table 20).

TELEOS							
Modèle de diagnostic	**Implémen-tation**	**Ontologie de haut niveau**			**Ontologie détaillée**		
		Précision	**RMSE**	**AIC**	**Précision**	**RMSE**	**AIC**
Knowledge Tracing	Modèle de Markov caché	66 % (± 2,8 %)	0.32	7914	71 % (± 2,7 %)	0.32	7 829
	Additive Factor Model	69 % (± 2,9 %)	0.34	7899	70 % (± 2,8 %)	0.32	7 897
Constraint-based	Contrainte	68 % (± 3,7 %)	0.33	7472	73 % (± 3,6 %)	0.31	7 305
Control-based	Réseau bayésien dynamique	67 % (± 3,5 %)	0.36	10 109	75 % (± 3,3 %)	0.30	10 194

Reading Tutor							
Modèle de diagnostic	**Implémen-tation**	**Ontologie de haut niveau**			**Ontologie détaillée**		
		Précision	**RMSE**	**AIC**	**Précision**	**RMSE**	**AIC**
Knowledge Tracing	Modèle de Markov caché	78 % (± 3,2 %)	0.577	226 723	78 % (± 3,2 %)	0.572	226 126
	Additive Factor Model	78 % (± 3,9 %)	0.586	215 894	78 % (± 3,9 %)	0.584	214 447
Constraint-based	Contrainte	72 % (± 4,1 %)	0.62	214 351	72 % (± 4,1 %)	0.61	215 003
Control-based	Réseau bayésien dynamique	74 % (± 3,5 %)	0.594	244 879	74 % (± 3,5 %)	0.59	244 655

Table 20 : Résultats de la précision et du RMSE pour les traces de TELEOS (tableau du haut) et du Reading Tutor (tableau du bas) en fonction de l'ontologie utilisée pour la construction des techniques de diagnostic. Entre parenthèse, les intervalles de confiance à 95%.

Dans la littérature, Xu et Mostow (Xu et Mostow, 2012) ont construit manuellement plusieurs techniques de diagnostic basées sur le modèle de diagnostic Knowledge Tracingpour le Reading Tutor. Leurs résultats donnaient des précisions entre 72 % et 87 % selon les techniques. Les résultats obtenus par notre plateforme sont compris dans cet intervalle, donc les techniques construites semi-automatiquement ont des performances comparables à des techniques construites par des experts. Nous n'atteignons pas les

meilleurs résultats experts (87 % de précision), mais qui ont été obtenus avec une implémentation poussée du Knowledge tracing que nous n'avons pas expérimentée. Toutefois, une telle comparaison est informative et non scientifique, car les données utilisées n'étaient pas les mêmes. Nous entendons juste montrer que les résultats de nos techniques sont comparables à des techniques d'experts, avec un coût de développement moindre et la possibilité de construire plusieurs techniques différentes (basées sur des modèles de diagnostic différents).

Concernant l'algorithme d'apprentissage, nous étudions la convergence de la matrice *R* lors de la première étape de l'algorithme, qui concerne l'association entre d'une part les variables du modèle de diagnostic, et d'autre part les variables des traces et de l'ontologie des traces et des connaissances. Pour rappel, l'étape 1 de l'algorithme redémarre l'heuristique de recherche locale jusqu'à convergence (cf. chapitre 4). Les graphes Figure 39 et Figure 40 montrent en abscisse le nombre de redémarrages de l'heuristique de recherche locale, et en ordonnée la variation entre deux redémarrages successifs définie comme suit : pour deux matrices R1 et R2, R2 étant obtenue à partir de la matrice R1 après un redémarrage de l'algorithme :

Variation = Moyenne (|r2-r1 |) pour tout r1 ∈ R1, r2 ∈ R2

L'heuristique converge entre 8 et 12 redémarrages pour TELEOS, et entre 15 et 18 redémarrages pour le Reading Tutor avec l'ontologie détaillée.

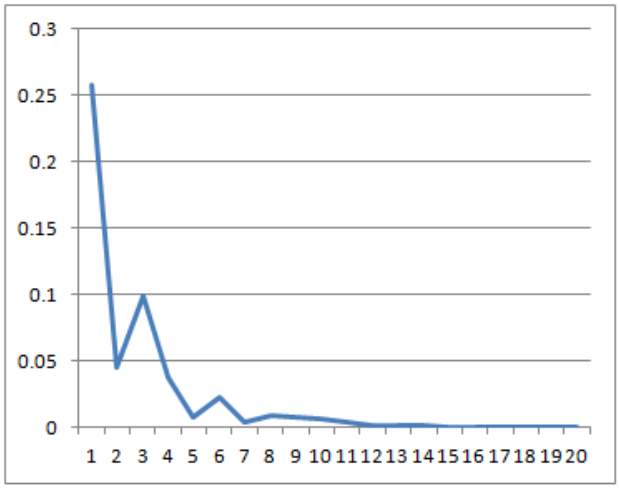


Figure 39 : Convergence de l'heuristique de recherche locale en fonction du nombre de redémarrages pour les traces de TELEOS.

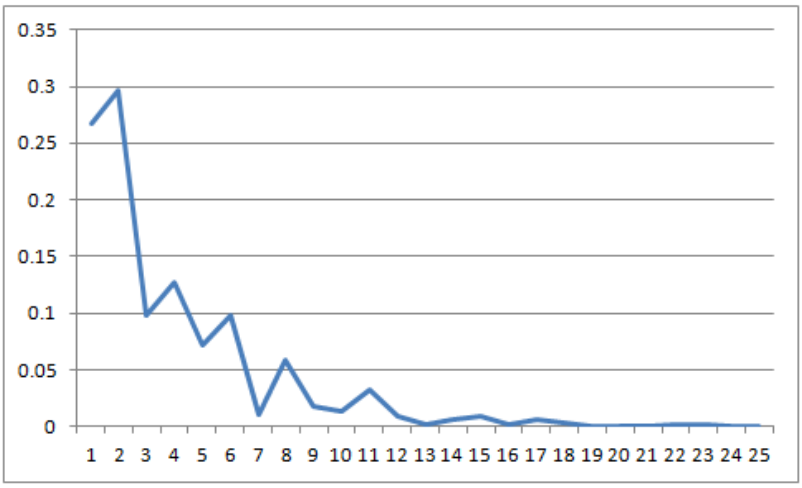


Figure 40 : Convergence de l'heuristique de recherche locale en fonction du nombre de redémarrages pour les traces du Reading Tutor.

Les courbes Figure 41 et Figure 42 montrent la convergence sur une exécution de l'algorithme, pour la construction des techniques utilisées jusqu'ici dans ce chapitre. Nous montrons maintenant la courbe de convergence extrapolée à partir de 1000 lancements de l'étape 1 de l'algorithme sur les traces de TELEOS et du Reading Tutor. Les abscisses montrent toujours le nombre de redémarrages de l'heuristique, et les ordonnées montrent la tendance de la moyenne des variations pour les 1000 lancements de l'étape. Sur ces deux graphes, la courbe de tendance montre que la convergence de l'heuristique de recherche locale est une fonction exponentielle f de la forme « f: x -> e^{-x} », avec R^2>=0.8. R^2 est le coefficient de détermination indiquant si une courbe extrapole bien les données (0 que l'extrapolation est nulle, 1 qu'elle est maximale). Ces deux figures montrent qu'après 1000 lancements, l'heuristique converge en moyenne après 13 redémarrages pour les traces de TELEOS, et 21 pour les traces du Reading Tutor (variation inférieure à 0.001). Pour ces deux domaines, on peut donc conclure que la convergence est rapidement atteinte (courbe de type exponentielle e^{-x}) et que le volume de traces semble avoir une influence sur le nombre de redémarrages (environ 3000 traces pour TELEOS contre 240 000 pour le Reading Tutor). Des expérimentations sur d'autres bases de traces permettraient de confirmer ou infirmer ces conclusions pour le cas général.

Concernant les temps d'exécution, la construction des quatre techniques avec TELEOS requiert environ 4 secondes et la construction avec le Reading Tutor 16 secondes. Ces temps d'exécution sont donc raisonnables, probablement en raison du nombre limité de variables dans les modèles de diagnostic de nos techniques (entre 4 et 5).

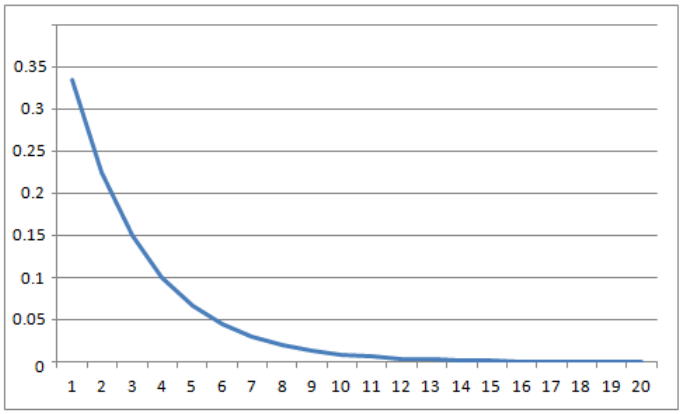


Figure 41 : Courbe de tendance de la convergence de l'heuristique de recherche locale à l'étape 1 de l'algorithme de construction de techniques de diagnostic après 1000 redémarrages sur les traces de TELEOS.

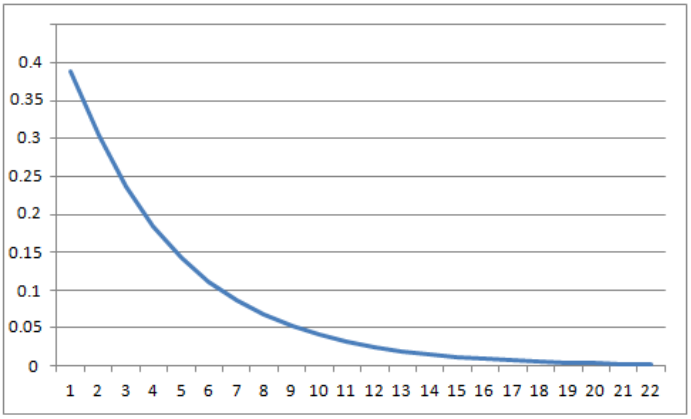


Figure 42 : Courbe de tendance de la convergence de l'heuristique de recherche locale à l'étape 1 de l'algorithme de construction de techniques de diagnostic après 1000 redémarrages sur les traces du Reading Tutor.

4. Retour sur les questions de recherche

Dans nos questions de recherche, nous posions le problème de pouvoir assister un concepteur à comparer et évaluer différentes techniques, ainsi qu'éventuellement en choisir une à intégrer dans un EIAH, pour les traces, le domaine et/ou l'EIAH du concepteur. Cette expérimentation montre la possibilité d'une telle comparaison sur des techniques basées sur des modèles de diagnostic différents (Knowledge tracing, Constraint-based et Control-based), ce qui est à notre connaissance nouveau dans la littérature. De plus, cette expérimentation montre que la comparaison, qui doit être générique dans nos questions de recherche, est applicable à des domaines et des traces différents.

Cette expérimentation montre également la possibilité de construire plusieurs techniques de diagnostics, là encore basées sur des modèles de diagnostic différents, via une même plateforme d'assistance.

IV. Seconde expérimentation : développement d'un critère de comparaison spécifique aux EIAH

1. Méthodologie

La seconde expérimentation visait à évaluer la possibilité d'appliquer des critères de comparaison spécifiques au domaine des EIAH sur des techniques de diagnostic basées sur différents modèles de diagnostic. Il s'agit premièrement de répondre à un second verrou dans la littérature : l'inexistence de comparaisons autres que statistiques ou prédictives, comme souligné dans l'état de l'art chapitre 2. Deuxièmement, il s'agit d'une preuve de concept sur la possibilité de développer des critères de comparaison complexes pouvant répondre à une question de recherche précise pour un concepteur de diagnostic.

Concernant la méthodologie, nous avons collaboré pendant six mois avec Jack Mostow et son équipe, qui sont les concepteurs du Reading Tutor, à l'université Carnegie Mellon de Pittsburgh (États-Unis). Nous avons identifié (Lallé et al., 2013) avec eux une question de recherche impliquant le développement d'un critère de comparaison spécifique à un sujet des EIAH : « une technique de diagnostic des connaissances a-t-elle un impact sur l'apprentissage automatique d'une stratégie d'aide basée sur un ensemble de types d'aide donnés par l'EIAH aux apprenants ? » Le critère de comparaison qui en a résulté est décrit dans le chapitre 5, section II.5, avec un exemple illustratif section II.6.C. Il s'agit ici de donner les résultats concrets obtenus par application de ce critère aux traces du Reading Tutor dans le cadre de cette expérimentation.

Le Reading Tutor utilise la reconnaissance automatique de la parole (RAP) pour décider si chaque mot est correctement lu ou non, et pour mesurer l'hésitation entre les mots. Nous disons qu'un mot est *lu couramment* si l'enfant le lit correctement selon la RAP et sans hésitation. Le Reading Tutor peut donner divers types d'aide sur un mot, comme le prononcer, l'épeler, prononcer un mot rimant avec, etc. (la liste exhaustive est donnée dans la Table 21). Certains types d'aide ne sont pas possibles pour tous les mots (comme épeler un mot d'une seule lettre). Le Reading Tutor choisit aléatoirement entre tous les types d'aide possibles pour un mot donné.

Type d'aide	**Action de l'EIAH pour le mot ou la phrase en cours**
SayWord	Joue un enregistrement vocal du mot (donne les différentes prononciations pour les homographes)
WordinContext	Joue un enregistrement vocal du mot extrait de la phrase en cours. Le mot doit avoir au moins trois caractères.
Autophonics	Prononce le graphème sélectionné dans un mot d'au moins trios caractères.
SoundOut	Joue un clips video d'une bouche d'enfant prononçant les phonèmes du mot. Le mot doit avoir deux caractères, ne pas être un homographe et avoir au plus quatre phonème.
Recue	Prononce tous les mots de la phrase précédant le mot en cours. Le mot en cours doit être au moins en troisième position dans la phrase.
OnsetRime	Prononce le premier phonème du mot, marque une pause, prononce le reste des phonèmes. Le mot doit avoir au moins trios caractères et ne pas être un homographe.

StartsLike	Dit "*start like*" (« commence par » en anglais) puis prononce un mot ayant le même début que le mot en cours. Le mot doit avoir au moins deux caractères.
RhymesWith	Dit "*Rhymes with*" (« rime avec » en anglais) puis prononce un mot rimant avec le mot en cours. Le mot doit avoir au moins deux caractères et les rimes doivent avoir la même orthographe.
Syllabify	Prononce les syllabes du mot séparées par de courtes pauses (en concaténant si besoin la prononciation des phonèmes de chaque syllabe). Le mot doit avoir au moins deux caractères et ne pas être un homographe.
ShowPicture	Montre une image du mot.
SoundEffect	Joue un son en rapport avec le mot.

Table 21 : Liste des types d'aide dans le Reading Tutor (Heiner et al., 2004).

Chaque type d'aide donné à un apprenant est collecté dans les traces. Nous disons qu'un type d'aide est un *succès* si l'apprenant lit le même mot couramment à la prochaine rencontre du mot (au moins un jour après pour éviter tout biais). Donc si un apprenant reçoit une aide *H* sur un mot *W* le jour *i*, nous considérons la prochaine lecture de *W* le jour *j* avec *j>i*, de façon à éviter les biais liés à la récence de l'aide (Figure 43). Pour simplifier l'analyse, nous ignorons les cas où un apprenant reçoit plusieurs types d'aide différents pour lire un même mot le jour *i*. Dans notre base de traces du Reading Tutor, nous avons 30 838 traces qui incluent un type d'aide. C'est ce sous-ensemble qui est utilisé dans cette expérimentation.

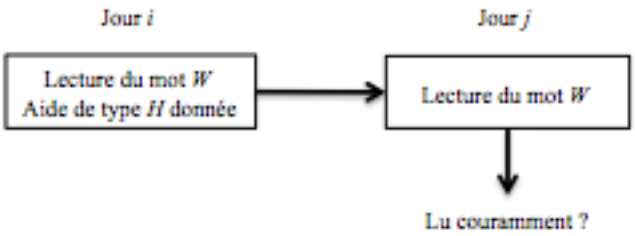


Figure 43 : L'aide *H* donnée sur le mot *W* le jour *i* est un succès si *W* est lu couramment le jour *j*.

2. Application du critère de comparaison

Nous avons défini les entrées à fournir pour le calcul du critère au chapitre 5 :

- Les types d'aide choisis aléatoirement sont inclus dans les traces du Reading Tutor, dans une variable nommée *Help_Type.*
- L'évaluation du succès d'un type d'aide, telle que définie ci-dessus (cf. l'exemple Figure 43), est également incluse dans les traces dans une variable nommée *Fluent.*

Nous rappelons succinctement le fonctionnement du critère :

1. application des techniques de diagnostic sur les traces afin d'inférer le modèle de chaque apprenant (étape effectuée par la plateforme, cf. chapitre 6),

2. sélection des variables dans les traces ayant un impact significatif sur le succès d'un type d'aide via un modèle linéaire,
3. apprentissage d'une stratégie d'aide via un algorithme de classification automatique.

À l'étape 2, les variables suivantes sont sélectionnées pour le Reading Tutor : le niveau de lecture de l'apprenant (évalué sur une échelle allant de 1 à F par le Reading Tutor, 1 étant le niveau de lecture le plus bas et F le meilleur niveau de lecture possible), son taux de mots lus couramment, le niveau de difficulté de l'exercice (évaluée de A à K dans le Reading Tutor), le nombre de lettres dans le mot, la fréquence du mot en anglais, la position du mot dans la phrase, le nombre de fois où l'apprenant a lu le mot auparavant, et la classe du mot (définie comme l'ensemble des aides possibles pour ce mot).

L'étape 3 donne une stratégie d'aide pour chacune des quatre techniques de diagnostic considérées :

- Knowledge tracing + modèle de Markov caché
- Knowledge tracing + Performance Factor Analysis (PFM)
- Constraint-based + contraintes
- Control-based + réseau bayésien

Une stratégie d'aide est un ensemble de règles de classification, dont voici un exemple :

1) Word = c145
2) AND Story_Level = B
3) AND Student_Model_Prediction > 0.6
4) AND Help_Type = "SayWord"
   ⇨ Fluent (22/22)

La clause 1 spécifie que la règle s'applique sur les mots de classe « c145 », c'est-à-dire pour lesquels les types d'aide 1, 4 et 5 sont possibles. La clause 2 spécifie la difficulté de l'exercice (niveau B). La clause 3 spécifie que la probabilité que l'apprenant réponde correctement selon la technique de diagnostic doit être supérieure à 0,6. La clause 4 spécifie le type d'aide. La prédiction est que le mot est lu couramment (*fluent*) avec une confiance de 22/22 (22 essais satisfont cette règle sur 22 dans les traces d'apprentissage).

3. Résultats du critère de comparaison

Nous commençons par évaluer les techniques de diagnostic construites pour la base de traces de cette expérimentation, au moyen des critères de comparaison suivants : précision, AUC et AIC (Table 22).

Modèle de diagnostic	Implémentation	Critères de comparaison		
		Taux	AUC	AIC
Knowledge Tracing	Modèle de Markov caché	84 % (± 2,6 %)	0.68	5.1 E+4
	Performance Factor Model	81 % (± 3 %)	0.65	5.5 E+4
Constraint-based	Contrainte	80 % (± 2,8 %)	0.65	5.6 E+4
Control-based	Réseau bayésien dynamique	83 % (± 2,9 %)	0.67	7.2 E+4

Table 22 : Résultats des critères de comparaison pour les techniques construites pour les traces du Reading Tutor incluant un type d'aide. Intervalle de confiance à 95 % entre parenthèses.

Chaque technique de diagnostic bat la classe majoritaire (76 % de mots lus couramment dans les traces). Ces résultats sont cohérents avec des évaluations passées du Knowledge Tracing, avec des précisions entre 72 % et 87 % (pour rappel, sur des traces différentes). Ces résultats montraient aussi des précisions en dessous de 35 % pour les mots lus non couramment, ce qui peut expliquer pourquoi les AUC sont de 0,68 au mieux dans nos résultats. AIC donne le Knowledge Tracing + modèle de Markov caché premier, et pénalise le Control-based, toujours en raison du grand nombre de paramètres du réseau bayésien dynamique.

La Table 23 montre l'évaluation de chaque stratégie d'aide par la précision de la prédiction du succès d'une aide (prochaine lecture sur un même mot lu couramment après avoir reçu une aide). Pour rappel, nous utilisons plusieurs algorithmes de classification automatique dans le critère pour apprendre une stratégie d'aide (comme décrit chapitre 5). Nous montrons ici les résultats seulement pour les classificateurs appris par JRip, qui donne la meilleure précision (l'écart étant de moins de 2 %). Le Knowledge Tracing + modèle de Markov caché donne la meilleure précision. Nous indiquons également le nombre total de règles apprises par JRip pour chaque technique.

Modèle de diagnostic	Implémentation	Précision de prédiction du succès de l'aide	Nombre de règles
Knowledge tracing	Modèle de Markov caché	75 % (± 4,1 %)	114
	Performance Factor Model	68 % (± 4,4 %)	95
Constraint-based	Contrainte	65 % (± 4,3 %)	89
Control-based	Réseau bayésien dynamique	73 % (± 4,4 %)	132

Table 23 : Précision de prédiction des stratégies d'aide construites pour chaque technique de diagnostic. Intervalle de confiance à 95 % entre parenthèses.

Les précisions de prédiction du succès d'une aide sont inférieures aux précisions des techniques de diagnostic utilisées. Prédire si un apprenant va lire un mot couramment est plus simple que de prédire si un type d'aide permettra de lire un mot couramment. Une raison est que les traces sont plus « éparses » : il y a parfois peu de traces pour certains types d'aide, et le nombre de types d'aide différents est de douze, ce qui crée un grand nombre de situations possibles.

Pour tester le degré de signifiance des différences entre les prédictions du succès des aides, nous utilisons le test de McNemar, qui évalue la signifiance de différences entre deux classificateurs C1 et C2. La formule est :

$$\chi^2 = (d1 - d2)^2 / (d1 + d2)$$

Ici, *d1* est le nombre d'instances classifiées comme positives par C1 et négatives par C2, et *d2* le nombre d'instances classifiées comme positives par C2 et négatives par C1. La somme *d1+d2* est supérieure à 80 dans nos données pour toutes les techniques, la limite minimale définie par McNemar pour son test, qui peut être approximé par une distribution χ^2. En résultat, le test de McNemmar appliqué à toutes les stratégies d'aide apprises deux à deux est significatif dans tous les cas ($p<0.025$)

Pour finir, le critère calcule l'espérance d'amélioration du taux de succès des stratégies d'aide, c'est-à-dire le taux de mots lus couramment selon les stratégies d'aide. La différence entre le taux espéré et le taux réel (dans les traces) est rapportée Table 24. La dernière ligne indique l'amélioration espérée en sélectionnant le type d'aide avec la plus grande probabilité dans les traces, sans utiliser de classificateur. Nous obtenons une espérance simulée, car calculée sur des traces précédemment collectées sans nouvelles expérimentations. Le succès espéré E est calculé comme suit :

$$E(Fluent \mid h^*, S, F)$$

Avec S la technique de diagnostic, F les éléments affectant le succès, et h* le type d'aide avec la probabilité de succès la plus élevée selon une stratégie d'aide pour une situation donnée :

$$h^* = \mathrm{argmax}_h \, E(Fluent \mid h, S, F)$$

La Table 24 indique également la couverture, qui est le pourcentage de traces pour lequel une stratégie d'aide est capable de recommander un type d'aide (donc pour lequel une règle de classification peut être appliquée. Les résultats montrent une amélioration espérée du succès des aides de 4,5 % à 5,2 % en utilisant les stratégies d'aide apprises par ce critère. En revanche, nous notons que les couvertures sont comprises en 25 % et 34 %, ce qui est peu car les stratégies sont incapables de recommander un type d'aide dans tous les autres cas. Cela est dû au fait que les traces sont éparses et incluent de nombreuses variables explicatives.

Modèle de diagnostic	Implémentation	Amélioration espérée du succès de l'aide	Couverture
Knowledge tracing	Modèle de Markov caché	5,2 %	32 %
	Performance Factor Model	4,7 %	26 %
Constraint-based	Contrainte	4,5 %	25 %
Control-based	Réseau bayésien dynamique	5,1 %	34 %
Aucun		2,4 %	

Table 24 : Amélioration espérée du taux de succès des aides.

4. Retour sur les questions de recherche

En ce qui concerne cette expérimentation avec l'équipe de Jack Mostow, elle a permis d'apprendre automatiquement une stratégie d'aide basée sur le résultat de techniques de diagnostic et les variables du domaine qui ont un impact sur le succès des aides (longueur des mots, place du mot dans la phrase, difficulté de l'exercice, etc.), afin de prédire les types d'aide ayant la plus forte probabilité de succès dans une situation donnée. Appliquées aux données du Reading Tutor, nous obtenons une amélioration estimée du succès des aides entre 4,5 % et 5,2 %. Toutefois, la faible couverture pénalise ce résultat : une amélioration espérée de 5,2 % avec une couverture de 32 % implique une amélioration espérée sur l'ensemble couvert de 16,3 %.

Les limites de ce critère sont la nécessité d'avoir des traces où les types d'aide sont sélectionnés aléatoirement (comme dans le Reading Tutor), ainsi que les couvertures trop faibles. Ces limites sont le prix à payer pour un apprentissage automatique de plusieurs stratégies d'aide et une évaluation automatique de ces stratégies.

En ce qui concerne nos questions de recherche, nous proposons un critère pour comparer des techniques de diagnostic et apprendre des stratégies d'aide. Les travaux précédents comparent les techniques de diagnostic basées sur un même modèle d'un point de vue prédictif, comme nous l'avons vu dans l'état de l'art. Nous comparons, au contraire, des techniques basées sur plusieurs modèles en fonction de leur impact sur l'apprentissage d'une stratégie d'aide. Une stratégie d'aide, entendue ici comme le choix d'un type d'aide à donner à l'apprenant, est une décision pédagogique prise par l'EIAH. Ce critère est donc spécifique au domaine des EIAH.

V. Troisième expérimentation : développement d'une nouvelle technique de diagnostic

1. Méthodologie

La troisième expérimentation visait à évaluer la possibilité pour un concepteur de diagnostic de spécifier et inclure dans notre plateforme des techniques de diagnostic basées sur un nouveau modèle de diagnostic. Il s'agissait d'évaluer l'application de notre formalisation

d'un modèle de diagnostic F_{MD} sur un cas concret, ainsi que le développement d'implémentations pour ce modèle de diagnostic.

Concernant la méthodologie, nous avons collaboré avec une didacticienne des mathématiques (Nathalie Brasset) qui a proposé un modèle de diagnostic que nous avons transcrit selon notre formalisation et nous avons proposé deux implémentations permettant de transposer ce modèle. La question de recherche en didactique était d'étudier la possibilité d'automatiser un diagnostic des connaissances, décrit plus bas, basé sur la praxéologie. Dans l'expérimentation, les traces sont similaires à celles de l'EIAH APLUSIX (décrites plus haut). Toutefois, cet EIAH ne pouvant pas au moment de l'expérimentation collecter dans ses traces le résultat du diagnostic comportemental, Nathalie a proposé de simuler des traces à partir d'exemples de résolutions des exercices d'APLUSIX réalisées sur papiers par des apprenants lors d'expérimentations passées. Ces traces, décrites plus haut, ont donc un faible volume (329 traces, 6 exercices, 24 apprenants).

Pour réaliser le diagnostic comportemental, APLUSIX évalue à chaque pas de résolution d'un problème d'algèbre (factorisation, développement…) un ensemble de règles mathématiques écrites par des experts qui permettent d'identifier si un apprenant a fait une erreur ou non, et si oui, quelle erreur. Un exemple de règle qui révèle une erreur est la suivante : « transformer ax-bx en a-b quand a≠b » (exemple, 9x-x=9-1=8). Chaque étape de résolution du problème pourra donc être correcte (si aucune règle erronée n'est détectée) ou incorrecte (si une règle erronée est détectée). Dans ce dernier cas, la règle erronée est indiquée dans les traces simulées par Nathalie.

2. Modèle de diagnostic Praxéologie

Nathalie propose dans son travail une simplification d'un modèle de diagnostic nommé Praxéologie (Chaachoua, 2011). Dans sa simplification, le modèle formalise les éléments suivants :

- Un ensemble d'exercices
- Un ensemble de « techniques » associées aux exercices. Par exemple, un exercice de factorisation met en jeu la technique de factorisation. Ces techniques sont donc des éléments de connaissance
    - Des sous-techniques décrivant à un niveau plus fin les connaissances que doit mobiliser l'apprenant. Par exemple, pour l'exercice de factorisation (x-1)(x+9)+(x-1), la technique de factorisation implique de mobiliser les sous-techniques factorisation simple (sur « x-1 »), puis développement et enfin réduction.
- Le diagnostic comportemental (présence de règles erronées ou non)
- Le contexte d'application (par exemple, un exercice peut porter sur un polynôme ou un monôme)
- La forme finale du résultat (qui peut avoir deux valeurs : conforme ou non conforme). Par exemple, le résultat d'une factorisation de degré deux ne peut être un polynôme de degré trois

Nous pouvons transcrire ces éléments selon notre formalisation F_{MD} comme suit (représentation graphique sur la Figure 44) :

Modèle simplifié de la praxéologie :

- O = { Exercise ; Rules ; Context ; Conform }
- K = { Technique, Subtechnique }
- OtoO = { RestoE : Result -> Exercise }
- OtoK = { EtoT : Exercise -> Technique ; RtoSub : Rules -> Subtechnique ; CxtoSub : Context -> Subtechnique ; CftoT : Conform -> Technique }
- KtoK = { TtoSub : Technique -> Subtechnique }
- C = { }
- BD = { Rules, Conform }

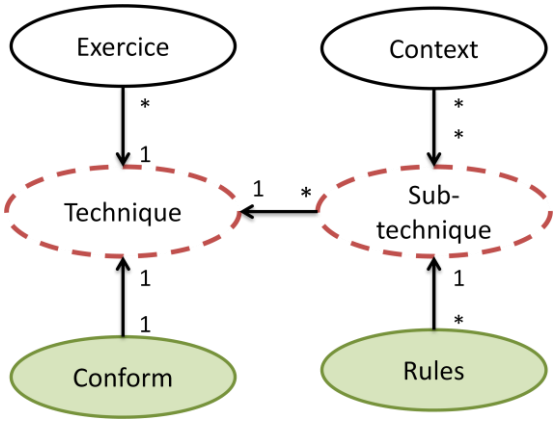


Figure 44 : Représentation graphique du modèle de diagnostic simplifié de la praxéologie.

L'objectif de ce diagnostic est d'inférer la « praxéologie » de l'apprenant, c'est-à-dire l'ensemble des techniques et sous-techniques qu'il maîtrise. Dans la praxéologie, la maîtrise d'une technique dépend de l'institution (généralement, les programmes scolaires de l'Éducation nationale en France). L'objectif est de déterminer si l'apprenant maîtrise la technique prônée par l'institution, ou bien s'il maîtrise une autre technique, qui peut être correcte en rapport au domaine mais non attendue (par exemple en géométrie, utiliser la figure plutôt que les théorèmes). Enfin, l'apprenant peut ne maîtriser aucune technique (ni attendue par l'institution, ni personnelle). Pour prendre en compte l'institution, le concepteur du diagnostic (ici Nathalie) doit apporter une « praxéologie institutionnelle » qui indique les techniques attendues par l'institution.

Dans notre expérimentation, nous proposons de définir la praxéologie institutionnelle dans un fichier XML qui décrit pour chaque technique les sous-techniques demandées par l'institution. Reprenons l'exemple de la factorisation d'une expression simple comme (x-1)(x+9)+(x-1) : la praxéologie institutionnelle indique qu'il est obligatoire de commencer par une factorisation simple, puis qu'un développement et une réduction sont optionnels. L'équivalent en XML est donné ci-dessous :

```
< !--Nom de la technique -->
<Technique name="Tfact.simple">

    < !--Première sous-technique « factorisation simple » obligatoire -->
    <SubTechnique name="TFact.simple_Formule"
    status="required"></SubTechnique>

    < !--Sous-technique de réduction facultative -->
    <SubTechnique name="Tcalc" status="optionnal">
    </SubTechnique>

    < !--Sous-technique de développement facultative -->
    <SubTechnique name="Tred" status="optionnal">
    </SubTechnique>
</Technique>
```

Pour ce modèle de diagnostic, nous proposons deux implémentations : règles expertes et réseau bayésien.

L'implémentation à base de règles expertes est basée sur un ensemble de règles de diagnostic des connaissances fournies par Nathalie. Ces règles sont les suivantes :

Si le résultat est conforme :

- 1er cas : la technique n'est pas celle attendue par l'institution, mais toutes les règles utilisées sont correctes. La technique est donc correcte mais pas institutionnelle
- 2e cas : la technique est celle attendue. La technique est institutionnelle et sue
- 3e cas : la technique n'est pas celle attendue et son utilisation est corrélée avec l'utilisation d'une règle erronée. La technique n'est pas correcte.

Si le résultat est non-conforme :

- 4e cas : la technique n'est pas celle attendue, mais toutes les règles utilisées sont correctes. La technique n'est donc pas correcte
- 5e cas : la technique est celle attendue par l'institution, l'apprenant a donc utilisé au moins une règle erronée. La technique est institutionnelle et une règle n'est pas maîtrisée.
- 6e cas : la technique n'est pas celle attendue et son utilisation est corrélée avec l'utilisation d'une règle erronée. La technique n'est pas correcte.

Ces règles expertes sont appliquées à la fin de chaque exercice. Dans ces règles, les cas 2 et 5 permettent d'inférer que l'apprenant maîtrise une technique institutionnelle. Les autres cas révèlent que la technique institutionnelle n'est pas maîtrisée, soit parce que l'apprenant utilise une autre technique personnelle, soit parce qu'il ne sait pas appliquer une quelconque technique pour résoudre le problème.

La seconde implémentation est basée sur un réseau bayésien, dont la structure est identique au modèle de diagnostic Figure 44. En raison du faible nombre de traces, il n'était pas

possible d'appliquer l'algorithme EM pour l'apprentissage des paramètres, ces derniers ont donc été fixés de façon manuelle par nous-mêmes. Cette méthode n'est donc pas rigoureuse, mais nous l'avons appliquée à titre d'exemple.

En guise de résultat, nous avons confronté le résultat des techniques de diagnostic Praxéologie + règles expertes et Praxéologie + réseau bayésien avec le diagnostic réalisé par Nathalie sur les traces simulées. Pour chaque exercice, elle a fourni les éléments suivants : la technique utilisée par l'apprenant est correcte ou non, est institutionnelle ou non. Les résultats de nos techniques de diagnostic appliquées aux traces simulées donnent les mêmes résultats que le diagnostic expert dans 100 % des cas.

3. Retour sur les questions de recherche

Par rapport aux questions de recherche de Nathalie Brasset (la didacticienne des mathématiques), cette expérimentation montre qu'il est possible d'évaluer automatiquement les techniques de l'apprenant, leur validité et leur adéquation avec l'institution.

Concernant nos travaux, l'objectif de cette expérimentation était d'étudier l'apport d'un nouveau modèle de diagnostic dans notre plateforme selon notre formalisation, avec la possibilité de comparer plusieurs implémentations pour ce modèle. Cependant, en raison du faible nombre de traces simulées, nous n'avons pu utiliser l'algorithme de construction semi-automatique et le calcul des critères de comparaison.

VI. Conclusion

Nous avons évalué nos propositions de recherche à travers trois expérimentations. La première fournit une preuve de concept sur la possibilité de construire et de comparer au moyen de notre plateforme plusieurs techniques de diagnostic basées sur des modèles de diagnostic différents. Nous avons pour cela utilisé les traces d'apprenants de trois domaines différents (chirurgie orthopédique, géométrie des aires, lecture de l'anglais). Pour chacun de ces domaines, notre algorithme de construction a pu construire les quatre techniques de diagnostic considérées, et tous les critères de comparaison ont pu être calculés. La seconde expérimentation portait sur le développement d'un critère de comparaison plus complexe qui portait sur une question de recherche spécifique au domaine des EIAH : l'impact des techniques de diagnostic sur le choix d'un type d'aide à donner à un apprenant. Les résultats ont montré d'une part la possibilité de développer un tel critère dans notre plateforme, et d'autre part que l'impact des techniques de diagnostic s'est révélé significatif sur l'apprentissage de stratégies d'aide dans le domaine de la lecture de l'anglais. Enfin, la troisième expérimentation portait sur l'apport d'un nouveau modèle de diagnostic dans la plateforme, associé à deux implémentations. Toutefois, le faible volume de traces simulées par une didacticienne n'a pas permis d'exploiter les méthodes d'assistance de la plateforme.

Ces expérimentations ont permis de montrer que nos propositions permettent de répondre à nos questions de recherche et à plusieurs verrous de la littérature :

- La possibilité de comparer à partir d'un même jeu de traces différentes techniques de diagnostic basées sur des modèles de diagnostic différents.
- La possibilité d'assister la construction de ces techniques par un algorithme semi-automatique, en respectant toutes les contraintes de chaque modèle de diagnostic et la sémantique apporté par le concepteur du diagnostic.
- La possibilité d'appliquer un grand nombre de critères de comparaison simples ou complexes, qui portent aussi bien sur la statistique (précision de prédiction), le génie logiciel (complexité des techniques), ou le domaine des EIAH (impact sur le choix d'un type d'aide, corrélation avec un diagnostic externe).

Parmi les limites de ces évaluations figurent essentiellement le fait que nous n'avons pas pu proposer à des concepteurs d'utiliser par eux-mêmes notre plateforme, faute d'une interface utilisable. En effet elle requiert encore un peu de programmation manuelle, notamment pour analyser les résultats des critères sur des sous-ensembles de traces, ce qui est essentiel à la phase d'interprétation. Nous reviendrons sur ces limites dans la conclusion et les perspectives.

Chapitre 8 : Conclusion

Sommaire

I. Bilan des contributions 169
II. Retour sur les questions de recherches 171
1. Rappel des questions de recherche et des principaux verrous 171
2. Réponses aux questions de recherche 172
3. Positionnement par rapport à la littérature 175
III. Prises de position par rapport aux contributions et limites 176
1. Rôle du concepteur et coût d'utilisation 176
A. Compétences du concepteur 176
B. Coût d'utilisation de PlaCID 179
2. Confiance dans les résultats et possibles biais 180
C. Confiance dans les techniques construites 180
D. Validité des résultats des critères de comparaison 181
3. Sur le diagnostic des connaissances 181
A. Prise de position induite par notre formalisation 181
B. Processus de construction et d'évaluation des techniques de diagnostic des connaissances 182
C. Positionnement par rapport à la communauté EIAH 183
IV. Perspectives 183
4. Prolongement des contributions 184
A. Utilisabilité de PlaCID 184
B. Expérimentations sur l'utilité 185
C. Interaction et paramétrisation de l'algorithme de construction et du score biaisé 186
D. Collecte et partage de critères et critères EIAH 187
5. Transposition de nos questions de recherche dans d'autres domaines des EIAH 189
E. Collaboration, émotions, jeux sérieux et modèles ouverts 190
F. Conception d'EIAH 191

Dans ce chapitre, nous allons faire dans la partie I un bilan de nos contributions. Dans la partie II, nous reviendrons sur nos questions de recherche et les réponses que nous avons pu

y apporter. Dans la partie III, nous aborderons les limites de nos contributions, les choix et prises de positions que nous avons faits et les implications de ces choix. Dans la partie IV, nous présenterons nos perspectives à court et long terme.

I. Bilan des contributions

Cette thèse propose deux méthodes d'assistance respectivement pour la construction et la comparaison de techniques génériques de diagnostic des connaissances dans le domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH).

Nous proposons en premier lieu une définition formelle de notre objet de recherche – les techniques de diagnostic générique – afin de pouvoir développer des méthodes et des outils qui fonctionnent pour un ensemble de techniques différentes. Une technique de diagnostic générique est l'association entre un *modèle de diagnostic* et une *implémentation*. Le modèle de diagnostic, qui est proche d'un réseau sémantique dans notre formalisation, vise à spécifier les éléments requis et les contraintes pour diagnostiquer les connaissances des apprenants. Un modèle de diagnostic est souvent basé sur une théorie de l'apprentissage et représente donc un point de vue ou une approche particulière. Une implémentation permet de mettre en œuvre le modèle de diagnostic, c'est-à-dire de pouvoir lire les traces des apprenants et inférer l'état de leur connaissance. Une telle formalisation nous permet de délimiter le cadre de notre recherche : l'assistance à la construction et à la comparaison de techniques de diagnostic. Cette formalisation porte uniquement sur le diagnostic des connaissances, et non sur le suivi et l'évaluation du comportement de l'apprenant : ce processus, appelé *diagnostic comportemental*, donne en sortie des *traces enrichies* qui sont les entrées d'une technique de diagnostic générique.

L'utilisateur cible de nos méthodes d'assistance est appelé *concepteur* de diagnostic des connaissances. Un concepteur peut être en premier lieu un expert du diagnostic des connaissances : ses besoins peuvent être d'étudier les propriétés d'une technique, d'évaluer une nouvelle technique par rapport à l'existant, d'améliorer une technique existante… Un concepteur peut être en second lieu un expert d'un autre domaine lié aux EIAH : ses besoins sont de construire et utiliser une technique de diagnostic qui puisse répondre à ses problématiques, comme fournir des aides personnalisées aux apprenants.

Nous proposons, pour assister la construction de techniques de diagnostic, un algorithme d'apprentissage semi-automatique, guidé par une ontologie qui permet au concepteur d'apporter de la sémantique aux traces. L'ontologie est construite par le concepteur à partir d'une ontologie par défaut, qui permet de spécifier deux types de variables : *observables* et *capacités* (=connaissances), en s'inspirant de la formalisation d'une technique de diagnostic. Les variables spécifiées sont ensuite liées aux traces par le concepteur, en associant une variable de l'ontologie à une ou plusieurs variables des traces. L'algorithme semi-automatique est ensuite exécuté afin de construire un ensemble de techniques de diagnostic à partir d'une base de traces d'apprenants et de l'ontologie du concepteur. Il en résulte des *techniques instanciées*, dont le code source peut par exemple être intégré à un EIAH pour diagnostiquer les connaissances des apprenants.

Nous proposons pour assister la comparaison de techniques de diagnostic d'appliquer un ensemble de critères de comparaison sur un ensemble de techniques de diagnostic instanciées, là encore au moyen de traces d'apprenants. Ces critères sont des algorithmes dont les résultats aident à positionner les techniques instanciées entre elles, par exemple via des tests statistiques, des algorithmes de data-mining, des critères de génie logiciel, etc. Ces critères étant calculés pour comparer des techniques instanciées, leurs résultats ne sont valides que pour les traces et le domaine du concepteur. Ce dernier doit analyser et interpréter ces résultats.

II. Retour sur les questions de recherches

1. Rappel des questions de recherche et des principaux verrous

Nous avons formulé dans l'introduction la question de recherche suivante : comment assister des concepteurs de diagnostic des connaissances pour la création et la comparaison de différents diagnostics des connaissances de façon indépendante du domaine au moyen de traces d'apprenant ?

Pour répondre à cette question, nous avons identifié dans l'état de l'art plusieurs verrous à résoudre. En premier lieu, concernant la construction de techniques de diagnostic :

- Le manque d'outils auteurs permettant de construire une technique de diagnostic de façon générique, sans imposer la construction d'un EIAH complet
- La difficulté d'interpréter et de réutiliser les techniques construites à partir d'algorithmes d'apprentissage automatique
- Le coût de développement d'une technique de diagnostic, qui est une tâche pluridisciplinaire nécessitant du temps de programmation et une bonne connaissance des modèles de diagnostic

En second lieu, concernant la comparaison de techniques de diagnostic :

- La prédominance des tests de comparaison statistiques et l'inexistence de mesures de comparaison spécifiques au domaine des EIAH
- L'inexistence de plateforme ou bibliothèque permettant de proposer, partager et calculer des mesures de comparaison pour des techniques de diagnostic
- La dépendance aux traces d'apprenants (le format et le volume) pour le calcul de mesures de comparaison et le biais que peuvent apporter les traces sur les résultats des mesures

Enfin, de façon plus générale, nous avons souligné l'inexistence de modèles ou de frameworks permettant de concevoir et développer des méthodes ou outils qui peuvent s'appliquer à des techniques issues de modèles de diagnostic différents. Ce verrou entraîne un cloisonnement des travaux en fonction du modèle de diagnostic pour lequel ils s'appliquent :

- Pour la construction, les outils auteurs et algorithmes automatiques imposent un modèle de diagnostic, donc ne permettent de construire que des techniques basées sur un même modèle de diagnostic en faisant éventuellement, mais pas toujours, varier les implémentations.
- Pour la comparaison, les mesures de comparaison utilisées ne sont calculées que pour des techniques basées sur un même modèle de diagnostic.

2. Réponses aux questions de recherche

Nous allons à présent reprendre les verrous listés ci-dessus et positionner nos contributions par rapport à ces verrous. Nous positionnerons nos contributions par rapport à la littérature dans la partie suivante.

Sur la construction :

- Le manque d'outils auteurs permettant de construire une technique de diagnostic de façon générique, sans imposer la construction d'un EIAH complet

Notre méthode et notre plateforme sont faiblement couplées aux autres composants d`un EIAH : il n'est pas imposé de construire un EIAH complet ou plusieurs composants d'EIAH. Nous nous sommes centrés entièrement sur la construction de techniques de diagnostic des connaissances, qui sont capables de prendre en entrée des traces d'apprenants enrichies fournies par le concepteur. En résultat, nous fournissons le code source de la technique, qui peut être intégré dans un EIAH par un informaticien.

- La difficulté d'interpréter et de réutiliser les techniques construites à partir d'algorithmes d'apprentissage automatique

Notre méthode de construction est basée sur un algorithme d'apprentissage semi-automatique. Nous avons proposé deux moyens de guider l'apprentissage. Le premier est la conception d'une ontologie par le concepteur qui apporte de la sémantique sur les traces, des liens entre les variables des traces et qui permet de spécifier des éléments absents dans les traces. Le second est que l'apprentissage vise à instancier un modèle de diagnostic connu et formalisé. Le résultat du diagnostic n'est donc pas libre, mais contraint par un modèle de diagnostic dont les variables et les relations sont spécifiées par un concepteur. L'apprentissage est donc guidé par une ontologie et par le modèle de diagnostic. Dans la littérature, seul SimStudent permet de guider l'apprentissage d'une technique de diagnostic par un apport de sémantique experte. Quelques travaux permettent de guider l'apprentissage par la spécification du modèle de diagnostic à apprendre (SimStudent et le Dynamic Cognitive Tracing), mais de façon ad hoc : le modèle de diagnostic est imposé et ces travaux dépendent de ce modèle de diagnostic imposé.

- Le coût de développement d'une technique de diagnostic, qui est une tâche pluridisciplinaire nécessitant du temps de programmation et une bonne connaissance des modèles de diagnostic

Dans notre approche, nous ne remettons pas en cause l'aspect pluridisciplinaire : les connaissances du domaine doivent bien être identifiées par un ou des experts (didacticiens, psychologues, enseignants…), afin qu'elles puissent être soit collectées dans les traces, soit spécifiées dans l'ontologie réalisée par le concepteur pour guider l'algorithme d'apprentissage semi-automatique. En revanche, notre méthode ne requiert pas de programmer partiellement ou en totalité les techniques de diagnostic : la construction est seulement guidée par l'ontologie du concepteur et les modèles de diagnostic. L'instanciation des techniques de diagnostic et l'apprentissage des paramètres sont réalisés par notre plateforme qui donne en résultat l'implémentation (le code source) de chaque technique de diagnostic construite. Nous avons montré dans les expérimentations que notre méthode a permis de construire plusieurs techniques de diagnostic dans deux domaines (chirurgie orthopédique, lecture de l'anglais) avec des précisions supérieures à 70 %, et des précisions proches de techniques construites par des concepteurs experts pour la lecture de l'anglais.

Sur la comparaison :

- La prédominance des tests de comparaison statistiques et l'inexistence de mesures de comparaison spécifiques au domaine des EIAH

Nous avons proposé dans nos travaux au moins un critère de comparaison qui est spécifique au domaine des EIAH : la comparaison de l'impact de techniques de diagnostic sur l'apprentissage d'une stratégie d'aide. Nous avons montré dans les expérimentations que ce critère a pu être calculé pour plusieurs techniques de diagnostic et a permis de positionner ces techniques.

- L'inexistence de plateforme ou bibliothèque permettant de proposer, partager et calculer des mesures de comparaison pour des techniques de diagnostic

Notre méthode repose sur le calcul d'un ensemble de critères de comparaison, qui sont stockés dans une base de données dans notre plateforme. Nous avons montré lors des expérimentations que ces critères – issus de la statistique, du data-mining, du génie logiciel ou spécifiques aux EIAH – on pu être calculés pour différentes techniques de diagnostic et différents domaines, et que leurs résultats permettent de positionner les techniques entre elles. Les résultats ne montrent pas toujours de différences significatives entre les techniques, et doivent être interprétés par le concepteur. Par rapport à la littérature, seule la plateforme Datashop permet de calculer trois critères de comparaison différents (AIC, BIC, RMSE), mais pour une seule technique de diagnostic (Knowledge tracing + AFM) dont seuls les paramètres et la forme du résultat changent.

- La dépendance aux traces d'apprenants (le format et le volume) pour le calcul de mesures de comparaison et le biais que peuvent apporter les traces sur les résultats des mesures

Nous n'avons premièrement traité le problème du format des traces que de façon ad hoc, en programmant directement la lecture de quelques formats (CSV, table SQL, Datashop). La lecture d'un nouveau format de trace nécessite un travail de programmation. La seule

contrainte forte que nous imposons dans notre méthode d'assistance est que les traces soient enrichies par un diagnostic comportemental. Dans la littérature, cette exigence est déjà présente dans plusieurs modèles de diagnostic, par exemple le Knowledge tracing. Cette contrainte permet de s'affranchir de la dépendance au domaine.

Nos expérimentations suggèrent que la nature des traces a un impact sur les performances des techniques de diagnostic : par exemple, les traces du Geometry Tutor issues de Datashop ont entrainé de meilleures précisions pour les techniques basées sur le Knowledge tracing, les traces de Datashop étant prévues pour fonctionner avec le Knowledge tracing. Toutefois, les traces du Reading Tutor sont également proches du Knowledge tracing mais les différences de précisions entre les techniques n'étaient cette fois pas significatives. De façon plus générale, nos travaux proposent de comparer des techniques de diagnostic instanciées pour les traces et le domaine du concepteur, et l'interprétation des résultats de comparaison ne sont valides que pour ces traces et ce domaine. La qualité et la complétude des traces sont laissées à la charge du concepteur.

Sur la prise en compte de modèles de diagnostic différents

- Pour la construction, les outils auteurs et algorithmes automatiques imposent un modèle de diagnostic, donc ne permettent de construire que des techniques basées sur un même modèle de diagnostic, en faisant éventuellement, mais pas toujours, varier les implémentations
- Pour la comparaison, les mesures de comparaison utilisées ne sont calculées que pour des techniques basées sur un même modèle de diagnostic

La principale contribution dans nos travaux est la proposition d'une méthode d'assistance à la construction et à la comparaison qui puissent être appliquées pour des techniques de diagnostic différentes basées sur des modèles de diagnostic différents. Pour cela, nous avons dû formaliser l'ensemble des techniques de diagnostic pour lesquelles nos contributions s'appliquent (la formalisation F_{MD} proposée chapitre 3). Cette formalisation nous a permis ensuite de proposer une méthode d'assistance à la construction et à la comparaison de différentes techniques de diagnostic, instanciées au domaine du concepteur et basées sur des modèles de diagnostic différents. Nos expérimentations ont montré la possibilité de construire et comparer différentes techniques de diagnostic sur différents domaines. Comme illustré dans l'état de l'art, il s'agit de la première méthode d'assistance qui soit à la fois :

- générique par rapport au domaine d'apprentissage (les techniques sont génériques et peuvent être construites et comparées pour plusieurs domaines)
- générique par rapport aux techniques de diagnostic (plusieurs techniques peuvent être construites, tant que le modèle de diagnostic peut être représenté dans notre formalisation F_{MD})

Pour conclure, revenons sur la question de recherche identifiée pour l'ensemble de notre travail :

- comment assister des concepteurs de diagnostic des connaissances pour la création et la comparaison de différents diagnostics des connaissances de façon indépendante du domaine au moyen de traces d'apprenants ?

Comme nous l'avons détaillé dans cette section, notre méthode d'assistance permet de construire et de comparer des techniques de diagnostic basées sur des modèles de diagnostic différents, de façon aussi bien indépendante du domaine que des modèles de diagnostic.

3. Positionnement par rapport à la littérature

Nous pouvons maintenant positionner nos travaux par rapport à l'état de l'art.

Pour les outils auteurs, nous avions identifié deux propriétés dans l'état de l'art : le couplage du diagnostic des connaissances avec les autres composants d'un EIAH (les outils auteurs fortement couplés imposent de construire tout ou partie d'un EIAH), et la nature de la construction du diagnostic (manuel, semi-automatique ou automatique). La Figure 45 résume les travaux de la littérature selon ces deux aspects. Nous constatons que notre travail est faiblement couplé et semi-automatique, comme SimStudent, et qu'il n'existe actuellement pas d'outil auteur à la fois automatique et faiblement couplé.

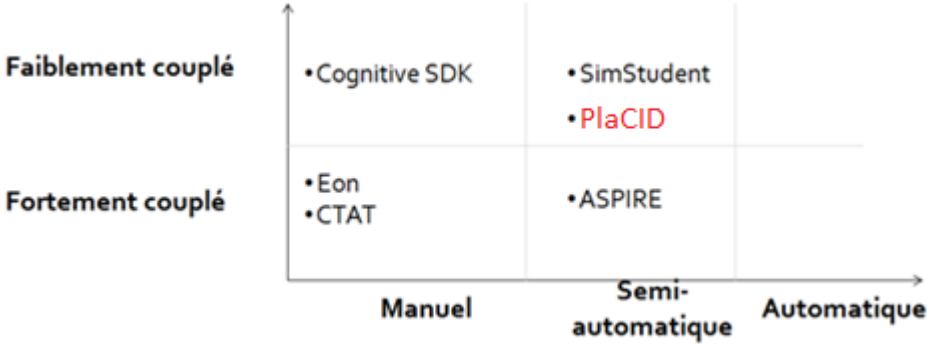


Figure 45 : Positionnement des outils auteurs et de nos travaux en fonction de leur couplage sur le diagnostic des connaissances et de leur approche pour la construction du diagnostic, de manuel à automatique.

Par rapport aux algorithmes d'apprentissage automatique, nous avions identifié deux propriétés dans l'état de l'art : l'interprétabilité (la facilité de compréhension du diagnostic construit) et la réutilisabilité (les contraintes imposées). La Figure 46 résume ces deux aspects. Nos travaux prennent en compte l'interprétabilité, tout comme SimStudent, car ce sont des approches semi-automatiques prenant en entrée des informations sémantiques expertes. Il est donc un peu biaisé de comparer notre travail par rapport aux algorithmes d'apprentissage automatique. Concernant la réutilisabilité, nous n'imposons aucune contrainte sur les traces si ce n'est l'enrichissement par un diagnostic comportemental. En revanche, aucune autre variable n'est requise dans les traces, notamment les connaissances qui peuvent être spécifiées dans l'ontologie par le concepteur, et il n'est pas requis comme SimStudent de fournir une base d'exemples de résolution d'exercices du domaine par un expert. En revanche, notre approche est plus coûteuse à utiliser, puisque semi-automatique,

et plus complexe que SimStudent, qui peut être utilisé par exemple par un enseignant après avoir été formé.

Figure 46 : Positionnement des différents algorithmes d'apprentissage automatique de diagnostic des connaissances et de notre travail en fonction de l'interprétabilité du diagnostic construit et de la réutilisabilité de l'algorithme.

Pour la comparaison, il n'existe pas de travaux proposant une collection de critères de comparaison génériques et partageables, et aucun travail ne porte sur la comparaison de l'impact de techniques de diagnostic sur les prises de décision de l'EIAH.

Enfin, comme indiqué ci-dessus, la principale différence par rapport à la littérature est la prise en compte de différentes techniques de diagnostic basées sur des modèles de diagnostic différents, ce qui est nouveau tant pour la construction que pour la comparaison de techniques de diagnostic des connaissances.

III. Prises de position par rapport aux contributions et limites

Notre travail implique des choix de formalisation et de méthodes qui présentent des limites, et qui peuvent relever de prises de position de notre part. Dans ce paragraphe, nous allons revenir sur les principales limites de nos travaux, sur leurs implications et sur nos prises de position sur le domaine de l'étude du diagnostic des connaissances.

1. Rôle du concepteur et coût d'utilisation

A. Compétences du concepteur

Nous avons identifié le concepteur soit comme un expert du diagnostic des connaissances, soit comme un expert des EIAH utilisateur de diagnostics des connaissances. Ce concepteur peut travailler seul ou au sein d'une équipe. Nous allons ici définir plus précisément le concepteur par les compétences qui sont requises ou facultatives pour l'utilisation de notre méthode d'assistance.

Compétences requises pour l'utilisation de notre méthode de construction et de comparaison

Par défaut, l'utilisation de notre méthode et de notre plateforme passe par trois tâches : construction de techniques de diagnostic, comparaison des techniques construites, utilisation du code source d'une technique construite.

La construction de techniques requiert la conception d'une ontologie des traces et des connaissances. Cette ontologie permet aussi bien d'apporter de la sémantique sur les traces que de spécifier des éléments non observés, comme des connaissances. Le concepteur doit être familier avec la conception d'une ontologie, notamment les principes d'héritage et de relations entre classes, ainsi qu'avec la représentation de connaissances sous forme de graphe. Pour rappel, notre ontologie dérive d'une ontologie existante proposant un vocabulaire de base sur la représentation de connaissances (i.e. sur la nature des connaissances : loi, concept, règle, stratégie…). La compréhension de ce vocabulaire est donc nécessaire. Les compétences requises sont donc la familiarité avec la représentation des connaissances sous forme de graphe ou de réseau, et la maîtrise technique des ontologies. Il n'est en revanche pas nécessaire d'avoir des compétences sur les mécanismes plus complexes des ontologies (comme l'inférence, le raisonnement, les requêtes…).

Les connaissances du domaine doivent être tracées dans les traces d'apprenants ou spécifiées dans l'ontologie par le concepteur. L'identification de ces connaissances du domaine n'est pas une compétence spécifiquement requise dans nos travaux, mais un prérequis à la construction de toute technique de diagnostic, qui relève avant tout de la didactique.

La conception de l'ontologie, tout comme la formalisation des modèles de diagnostic, reposent dans notre travail sur le postulat suivant : en EIAH, les connaissances peuvent être diagnostiquées à partir de l'observation des interactions de l'apprenant avec l'EIAH (donc à partir de variables observables dans les traces enrichies). Pour concevoir l'ontologie des traces et des connaissances, la maîtrise de toutes les techniques de diagnostic construites par la plateforme n'est pas requise, grâce à l'algorithme d'apprentissage semi-automatique, mais la maîtrise de ce postulat – une connaissance est diagnostiquée à partir d'observables – semble nécessaire pour pouvoir concevoir et raffiner l'ontologie. Les compétences requises sont donc la compréhension théorique du processus de diagnostic des connaissances.

La construction, tout comme la comparaison, nécessite des traces enrichies par un diagnostic comportemental (pour rappel, dans un objectif de généricité). Le concepteur doit donc être capable soit de mettre en place un diagnostic comportemental sur ses traces, soit utiliser des traces déjà enrichies. La mise en place du diagnostic comportemental est à la fois un problème d'algorithmique et un problème d'acquisition et de représentation des connaissances du domaine nécessaires pour réaliser ce diagnostic comportemental, qui dépend le plus souvent du domaine d'apprentissage. La mise en œuvre du diagnostic comportemental peut avoir été réalisée précédemment, comme ce fut le cas dans nos expérimentations où nous avons directement utilisé des traces enrichies.Dans le cas contraire, les compétences requises pour mettre en œuvre ce diagnostic comportemental sont donc le traitement des traces collectées par un EIAH.

Les résultats des critères de comparaison doivent être interprétés par le concepteur. L'interprétation nécessite la compréhension du critère, ce qui peut nécessiter des compétences différentes selon le critère : en statistique, en data-mining, en qualité du génie logiciel, etc. Comme nous l'avons montré dans nos expérimentations, cette interprétation

demande de recouper les résultats des critères, de sélectionner des sous-ensembles de traces, d'interpréter les intervalles de confiance… Plus généralement, les compétences requises relèvent de l'analyse de données.

En résultat, notre méthode permet d'obtenir le code source d'une ou plusieurs techniques de diagnostic instanciées pour un domaine et un jeu de traces enrichies. L'utilisation du code source d'une technique ou son intégration dans un EIAH nécessite des compétences de programmation et d'architecture logicielle. Il est également nécessaire à ce stade de maîtriser la technique dont on souhaite utiliser et intégrer le code source, ce qui nécessite un travail de documentation.

En résumé, nous pouvons identifier quatre pôles de compétences : représentation de connaissances, analyse de données, programmation et maîtrise de la notion de diagnostic des connaissances. Les trois premiers pôles sont des sous-domaines de l'informatique et/ou de la statistique (pour l'analyse de données). Le dernier pôle est lui spécifique au domaine des EIAH. Le concepteur doit donc avoir des connaissances pluridisciplinaires, en informatique avec des connaissances spécifiques aux EIAH. Or, le domaine des EIAH étant lui-même au carrefour entre informatique, didactique et analyse de données, nous pensons que notre méthode peut être utilisée par les experts des EIAH maîtrisant les compétences que nous avons listées plus haut et la notion de diagnostic des connaissances, au moins théoriquement. Nous notons également que le domaine des EIAH étant pluridisciplinaire, il se trouve en son sein des experts en analyse de données, en représentation de connaissance… Le processus d'utilisation de notre méthode permet la collaboration entre plusieurs concepteurs, en séparant la construction (qui relève essentiellement de la représentation de connaissances), la comparaison (qui relève essentiellement de l'analyse de données) et l'intégration dans un EIAH existant (qui relève du programmeur).

Compétences pour l'ajout de modèles de diagnostic, d'implémentations et de critères

Il est possible de réaliser un certain nombre de tâches facultatives avancées dans notre méthode : la formalisation d'un nouveau modèle de diagnostic, l'ajout d'une implémentation, l'ajout d'un critère de comparaison, le raffinement du code source d'une technique construite.

Dans nos travaux, un modèle de diagnostic est une instance de notre formalisation F_{MD} proposée au chapitre 3. La spécification d'un nouveau modèle de diagnostic requiert donc des compétences en représentation de données pour la compréhension de ce modèle et la construction d'une instance, ainsi que la maîtrise de l'outil des ontologies. Des compétences théoriques sur la définition du diagnostic des connaissances sont également requises, notamment car notre formalisation repose sur des prises de position (les connaissances sont diagnostiquées à partir d'observables et les connaissances sont représentables sous forme de graphe sémantique) et impose des contraintes (par exemple, absence de cycle entre les ensembles de connaissances et d'observables). Il faut donc tenir compte de ces aspects.

L'ajout d'une implémentation dans la plateforme nécessite de proposer un code source capable de lire des traces, d'inférer le modèle de l'apprenant, et de transposer un modèle de

diagnostic instancié à un domaine (cf. chapitre 3 pour plus de précisions). Pour ce faire, des compétences solides en représentation de connaissances et raisonnement (inférence en informatique) sont nécessaires. Il est également requis les mêmes compétences en représentation de données que pour la spécification d'un nouveau modèle de diagnostic, puisqu'une technique de diagnostic est le couple formé par un modèle de diagnostic et une implémentation.

Un critère de comparaison est défini de façon générale comme un algorithme évaluant le résultat d'une technique de diagnostic sur une base de traces d'apprenants. Le développement d'un nouveau critère de comparaison requiert donc par défaut des compétences en programmation et en analyse de données (statistique, data-mining…). D'autres compétences facultatives peuvent être requises selon l'objectif du critère. Par exemple, un critère portant sur l'évaluation d'un point de vue génie logiciel des techniques de diagnostic requiert des compétences en génie logiciel.

La plateforme donne en résultat le code source des techniques instanciées aux traces du concepteur. Il est donc possible de raffiner, modifier ou enrichir ce code source, par exemple pour ajouter des variables, raffiner les paramètres, etc. Ces tâches requièrent des compétences spécifiques en conception de diagnostic des connaissances.

Les compétences requises pour l'utilisation avancée de notre méthode demandent donc une bonne connaissance du domaine du diagnostic des connaissances ainsi que des compétences en représentation des connaissances et en raisonnement. Le concepteur ayant ces connaissances est probablement un expert du diagnostic des connaissances ayant lu et maîtrisant notre formalisation F_{MD}.

B. Coût d'utilisation de PlaCID

L'utilisation par défaut de notre méthode (construction de techniques et comparaison) présente les coûts suivants : collecte de traces enrichies et spécification d'une ontologie des traces et des connaissances. Depuis les années 2000, la collecte de traces enrichies est de plus en plus requise pour la construction d'une technique de diagnostic, pour l'apprentissage automatique des paramètres de ces techniques. Ces traces sont aussi utilisées depuis les années 1980 pour l'évaluation des techniques de diagnostic via des courbes d'apprentissage, bien qu'il n'y ait alors ni comparaison, ni évaluation *a priori*. En ce sens, notre méthode ne présente pas nécessairement un surcoût par rapport à la littérature. Similairement, l'exigence de traces qui soient enrichies par un diagnostic comportemental est également fortement présente dans la littérature ; la séparation entre le processus d'enrichissement des traces et le processus de diagnostic des connaissances est en revanche souvent floue. Notre méthode exige donc un effort pour dissocier les deux diagnostics (comportemental et épistémique).

Concernant la conception de l'ontologie des traces, il y a un surcoût lié à la compréhension du vocabulaire et des relations de l'ontologie attendue dans notre plateforme.

L'utilisation avancée de notre méthode (la spécification de nouveaux modèles de diagnostic, implémentations ou critères de comparaison) présente un coût plus important car il est requis de maîtriser notre formalisation d'un modèle de diagnostic. Il y a donc un temps de lecture des spécifications, puis un temps de mise en application.

2. Confiance dans les résultats et possibles biais

A. Confiance dans les techniques construites

La construction des techniques de diagnostic se fait via un algorithme d'apprentissage semi-automatique. Du point de vue du concepteur, l'apprentissage semi-automatique peut poser deux problèmes de confiance dans les résultats. Premièrement, il n'y a pas de preuve que le résultat de l'apprentissage automatique est valide du point de vue du domaine d'apprentissage. Par exemple, il n'y a pas de preuve garantissant que l'algorithme ne fera pas de relations erronées entre une connaissance et une variable observable. Deuxièmement, les résultats de l'apprentissage automatique sont dépendants de la qualité et du volume de traces. Notre méthode ne garantit que deux propriétés : les techniques construites respecteront toutes les contraintes des modèles de diagnostic, et donneront un poids fort aux informations sémantiques apportées par le concepteur via l'ontologie. Ces propriétés visent à renforcer l'interprétabilité des techniques et leur validité par rapport au domaine, mais le concepteur n'a pas de garantie sur la validité par rapport au domaine. Il y a donc un triple travail qui est requis pour le concepteur : évaluer la qualité de ses traces afin de pouvoir les enrichir ou les compléter dans l'ontologie, analyser les résultats de l'algorithme (i.e. les techniques construites), notamment au moyen des critères de comparaison, et enfin analyser le code source d'une technique instanciée avant intégration dans un EIAH.

Il n'existe pas dans la littérature de méthode automatique ou semi-automatique de construction d'une technique de diagnostic qui garantit la validité de la technique par rapport au domaine. Le temps gagné par l'apprentissage (semi-)automatique est donc minoré par le temps nécessaire à l'analyse de la validité du résultat de l'apprentissage. Dans nos travaux, nous prenons le parti de faciliter l'analyse de la validité des résultats en prenant en compte des éléments à même de renforcer l'interprétabilité des techniques construites. De plus, nous lions construction et évaluation (au moyen des critères de comparaison) des techniques dans une même méthode.

Nous ne garantissons donc pas la validité des techniques construites, mais nous proposons d'assister, donc de faciliter, la construction des techniques et leur évaluation. Nos expérimentations ont montré que la construction de différentes techniques à partir de traces et guidée par le modèle de diagnostic de la technique et par de la sémantique experte est possible. Du point de vue de la précision de prédiction, les techniques construites battaient la classe majoritaire et étaient comparables à des techniques construites par des experts.

En permettant au concepteur d'influer sur l'apprentissage automatique au moyen de l'ontologie des traces, nous incluons de fait un biais lors de la phase de construction. Chaque

concepteur peut avoir une conception différente du diagnostic des connaissances et du domaine d'apprentissage, et les informations apportées dans l'ontologie ont un impact sur les techniques. Toutefois, notre travail a pour optique d'être suffisamment générique pour pouvoir assister chaque concepteur pour la construction et la comparaison de techniques pour un EIAH et des traces données. De plus, notre méthode est conçue pour qu'il soit possible de tester de manière simple plusieurs ontologies des traces différentes, voire même plusieurs bases de traces, permettant alors au concepteur de recouper les résultats. Ainsi, les techniques construites ne sont valides que pour le domaine, l'EIAH et les traces fournies par le concepteur.

B. Validité des résultats des critères de comparaison

Les critères de comparaison visent à positionner les techniques de diagnostic en fonction de leurs performances et de leur utilité. Ils sont essentiellement calculés à partir des résultats des techniques de diagnostic appliquées sur une base de traces donnée. Les critères de comparaison dépendent donc directement de l'application des techniques de diagnostic, c'est-à-dire : de la construction des techniques, de la qualité des paramètres des techniques (s'il y a des paramètres) et des traces d'apprenants.

Premièrement, la construction des techniques de diagnostic et la qualité des paramètres des techniques ont un impact sur les résultats des critères. Nous avons par exemple montré dans nos expérimentations que l'implémentation du Knowledge tracing avec la logique floue donnait de mauvais résultats de prédiction en raison d'un mauvais paramétrage.

Deuxièmement, les traces (leur qualité, leur format et leur volume) ont une influence sur les résultats des critères. Nous avons montré dans nos expérimentations que des traces adaptées au Knowledge tracing ont tendance à privilégier les techniques basées sur le Knowledge tracing du point de vue de la précision de la prédiction, mais pas nécessairement de façon significative. D'une façon générale, l'objectif de nos travaux est là encore d'assister le travail de concepteurs de diagnostic des connaissances pour leurs domaines et leurs traces.

Nous insistons sur le fait que notre méthode porte sur la comparaison de techniques de diagnostic instanciées pour un domaine, et non sur l'évaluation des techniques de façon générique. Par exemple, nous ne visons pas à montrer que le Knowledge tracing sera toujours supérieur au Constraint-based en termes de précision, mais seulement à faciliter cette comparaison pour un domaine donné. Les résultats des critères de comparaison nécessitent d'être analysés et interprétés par le concepteur en fonction de ses besoins.

3. Sur le diagnostic des connaissances

A. Prise de position induite par notre formalisation

Dans notre formalisation F_{MD} d'une technique de diagnostic, nous posons pour postulat qu'une connaissance est diagnostiquée à partir d'observables et que les connaissances peuvent être représentées sous forme de graphe ou de réseau qui peut être simple ou complexe. Une telle formalisation de techniques de diagnostic génériques apporte un cadre

restrictif à nos travaux. Nous ne proposons pas d'assister la construction et la comparaison de toute technique de diagnostic existante, mais uniquement du sous-ensemble des techniques qui sont des instances de notre formalisation. Nous avons montré au chapitre 3 et dans les expérimentations que les techniques de diagnostic les plus connues dans la littérature peuvent être décrites avec notre formalisation.

Si une telle formalisation des techniques existantes n'existe pas dans la littérature, c'est avant tout car les techniques sont très différentes (les variables, les relations, l'inférence changent). C'est également la raison pour laquelle notre formalisation reste très générique, en représentant les techniques de diagnostic comme des ensembles de connaissances et d'observables.

Notre approche est centrée sur l'utilisation de techniques de diagnostic génériques. Nous postulons que la généricité des techniques est un élément important qui permet le développement d'outils et de méthodes génériques. En contrepartie, il n'est pas garanti que ces techniques génériques soient adaptées ou pertinentes pour tous les domaines, notamment les domaines mal définis (*ill defined*). Dans nos expérimentations, nous avons pu néanmoins construire avec PlaCID quatre techniques différentes pour un domaine mal défini (chirurgie orthopédique).

Enfin, notre approche s'applique exclusivement au diagnostic des connaissances en situation d'apprentissage individuel.

B. Processus de construction et d'évaluation des techniques de diagnostic des connaissances

Dans la littérature, la construction et l'évaluation de techniques de diagnostic est un processus qui repose sur un choix *a priori* de la ou des techniques considérées. Les techniques peuvent ne pas être génériques (i.e. une technique développée par un concepteur pour son domaine, sans possibilité d'être réappliquée sur d'autres domaines), ne pas utiliser de traces (i.e. pas d'apprentissage automatique des techniques ou des paramètres), etc. De même, l'évaluation des techniques de diagnostic peut se faire avec ou sans comparaison, avec des traces réelles ou simulées, lors de la construction d'un EIAH ou après, etc.

Dans notre travail, nous proposons une nouvelle approche du processus de construction et de comparaison qui est indépendante de l'EIAH. Nous renversons pour cela le processus en postulant que l'évaluation (par comparaison) de techniques de diagnostic doit en précéder le choix. Plus le nombre de techniques comparées est grand, plus le choix est large. L'utilisation des traces d'apprenants aussi bien pour la construction que l'évaluation est un second postulat. Les traces sont de plus en plus collectées en grand volume et stockées dans des plateformes comme Datashop, qui permettent l'extraction d'informations dans les traces de façon automatique et une évaluation empirique basée sur de réelles traces d'interaction d'apprenants. Dans la littérature, une telle approche n'existe pas en raison du coût de construction d'une technique de diagnostic et de l'absence de méthodes et outils permettant de construire et comparer des techniques différentes basées sur des modèles de

diagnostic différents. Notre travail propose donc de résoudre ces verrous en aidant le concepteur à construire, puis à comparer et enfin à choisir des techniques de diagnostic.

C. Positionnement par rapport à la communauté EIAH

Dans les premiers travaux dédiés au diagnostic des connaissances, jusqu'aux années 1980, les concepteurs visaient à représenter tant les connaissances des experts sur le domaine que les erreurs de la façon la plus exhaustive et précise possible, ce qui était coûteux en temps de développement, pour des résultats sur l'apprentissage qui n'étaient pas toujours clairs (Wenger, 1987). Dans les années 1990, des personnes comme Self ou Ohlsson (Ohlsson, 1994; Self, 1994) ont proposé de faire évoluer le principe du diagnostic des connaissances afin de répondre à ces critiques. Self postule notamment qu'un diagnostic n'a pas à être exhaustif et complet en toutes situations ; il suffit que le diagnostic des connaissances soit fiable et utile lorsqu'il est appliqué, mais qu'il n'est pas requis d'appliquer le diagnostic dans toutes les situations. Ohlsson postule également qu'il est possible d'ignorer toutes les situations où l'apprenant n'est pas en mesure d'apprendre par l'erreur. Les objectifs du diagnostic des connaissances sont donc réévalués plus modestement, mais afin de faciliter leur développement et leur utilisation. Dans les années 2000, les traces d'apprenants sont de plus en plus utilisées pour l'apprentissage automatique, aussi bien l'apprentissage de paramètres de techniques de diagnostic, que l'apprentissage de la structure complète de techniques de diagnostic (Desmarais et Baker, 2012). On observe donc une simplification progressive de la construction de techniques de diagnostic. Toutefois, c'est la phase d'évaluation qui n'évolue que très peu : elle est toujours effectuée *a posteriori*, par des traces ou des expérimentations. Datashop est par exemple toujours basé sur le calcul de courbes d'apprentissage et de précision de prédiction d'une technique, tout comme dans les travaux d'Anderson sur le LISP Tutor dans les années 1980. Similairement, les méthodes et outils d'évaluation sont encore peu standardisés et partagés, et la plupart des analyses des résultats de techniques de diagnostic se fait via des environnements comme R ou Matlab. Nous constatons aussi actuellement un cloisonnement plus fort entre les communautés en fonction des modèles de diagnostic considérés – par exemple, la recherche sur le Knowledge tracing et la recherche sur le Constraint-based –, avec peu d'interaction entre ces communautés. Une raison pour cela est la spécificité plus grande des travaux, tangible notamment dans les études très détaillées liées au Knowledge tracing.

Dans notre travail, nous nous positionnons pour une évaluation comparative *a priori* systématique, et nous proposons pour cela une méthode d'assistance permettant de construire et comparer des techniques différentes empiriquement. Nous pensons que ces études comparatives permettent de renforcer la valeur scientifique des travaux basés sur des techniques de diagnostic des connaissances.

IV. Perspectives

Nous allons ci-dessous dans la partie 1 commencer par exposer différentes perspectives dans le prolongement de nos contributions, c'est-à-dire des perspectives pour améliorer nos contributions, notamment en rapport aux limites que nous avons identifiées, ou des

perspectives pour approfondir nos contributions. Dans la partie 2, nous nous intéresserons à des perspectives de mise en application de nos questions de recherche et du principe de nos contributions dans d'autres domaines des EIAH.

1. Prolongement des contributions

A. Utilisabilité de PlaCID

Comme nous l'avons noté dans le mémoire et dans la conclusion, PlaCID n'est actuellement pas utilisable par un concepteur du fait que des composants importants pour la construction et la comparaison ne sont pas utilisables par des interfaces (il faut modifier le code de PlaCID). Une première perspective est donc l'amélioration de l'utilisabilité de PlaCID. Nous proposons plusieurs pistes d'amélioration dans cette partie.

Premièrement, nous pouvons ajouter une interface dédiée pour guider la conception de l'ontologie des traces requises pour la construction des techniques. Nous utilisons pour le moment Protégé qui est une interface de conception d'ontologie complexe et générique, ne pouvant fournir aucune aide ni guider le concepteur pour la réalisation de notre ontologie. On peut pour cela soit proposer une interface dédiée, soit mieux intégrer l'utilisation de Protégé à l'utilisation de notre plateforme, à la manière du logiciel ABSTRACT (Georgeon, 2008) dont le but est de visualiser des traces associées à une ontologie. Pour aller plus loin, il serait utile de pouvoir guider le concepteur en lui recommandant le niveau le plus adapté de l'ontologie en fonction des traces : haut niveau ou détaillée (cf. chapitre 4). Nous avons par exemple constaté dans nos expérimentations qu'une ontologie de haut niveau est suffisante pour une base de traces importante (>200 000 traces) sur un domaine bien défini, et insuffisante pour une base de traces moyenne (entre 3000 et 4000 traces) sur un domaine mal défini.

Deuxièmement, nous pouvons ajouter une interface permettant de calculer les critères sur des sous-ensembles de traces, par exemple en sélectionnant un ensemble d'apprenants, de connaissances, d'exercices… Le but est de faciliter l'interprétation et l'analyse des résultats des critères. Nous pouvons nous inspirer ici de Datashop qui permet de calculer des courbes d'apprentissage en sélectionnant dans un menu des apprenants, des connaissances ou des problèmes.

À plus long terme, une interface de visualisation des techniques construites (actuellement il faut consulter le code source) peut permettre aux concepteurs non experts de mieux appréhender la validité des techniques construites et leur utilité. Nous pouvons *a priori* proposer deux grandes catégories de visualisation : sous forme de graphe (avec possibilité de visualiser les paramètres associés à chaque nœud du graphe) ou sous forme de règles. Il existe par ailleurs des travaux portant sur la visualisation des paramètres d'une technique de diagnostic dont nous pourrions nous inspirer.

Plus haut, nous avons vu que l'utilisateur de notre plateforme est nécessairement un expert du domaine des EIAH. Nous pourrions nous intéresser en perspective à rendre PlaCID utilisable par un enseignant. Pour ce faire, une perspective est d'adopter l'approche de

SimStudent qui permet à un enseignant de construire une technique de diagnostic par l'exemple : l'enseignant choisi un ensemble d'exercices représentatifs du domaine et les résout. SimStudent construit à partir des résolutions des exercices par l'enseignant la technique de diagnostic.

B. Expérimentations sur l'utilité

Notre plateforme fournit une preuve de concept sur la possibilité d'appliquer notre méthode d'assistance, et nos expérimentations montrent que cette méthode est applicable. Nos expérimentations ne permettent en revanche pas d'évaluer l'utilité de notre méthode et l'utilisabilité de notre plateforme. Il y a pour cela des contraintes techniques – notre plateforme n'est pas utilisable car plusieurs fonctionnalités ou interfaces n'y ont pas été implémentées – et des contraintes de temps.

Une perspective est donc la tenue de nouvelles expérimentations pour évaluer l'utilité et la validité de notre méthode d'assistance à la construction et à la comparaison. Ces expérimentations doivent permettre de collecter les retours des concepteurs sur l'utilité de nos travaux et sur la validité des techniques construites et des résultats des critères, ainsi que d'évaluer les technique construites par les concepteurs. Les expérimentations doivent aussi permettre d'identifier si nos travaux permettent de répondre aux besoins et aux questions de recherche des concepteurs qui nécessitent l'utilisation ou l'étude de techniques de diagnostic des connaissances. Enfin, un élément important de nos travaux est la proposition d'une seule méthode d'assistance, qui pourra donc être utilisée par les concepteurs pour plusieurs travaux, sur plusieurs traces/domaines. Les expérimentations peuvent permettre d'évaluer si le coût investi pour comprendre l'utilisation de nos travaux est amorti d'une part par la facilité de construire et comparer sans programmation un ensemble de techniques de diagnostic, d'autre part par la possibilité de réinvestir l'expérience acquise pour construire et comparer des techniques sur d'autres domaines ou d'autres traces. La Table 25 synthétise les objectifs de ces expérimentations prévues en perspectives.

De telles expérimentations peuvent fournir *a minima* les résultats ou données suivantes : les productions des concepteurs (i.e. les ontologies et les techniques construites), les traces d'interaction entre les concepteurs et notre plateforme (notamment lors de l'analyse des résultats des critères), des questionnaires posés aux concepteurs. Le protocole n'étant pas encore défini, d'autres résultats sont envisageables. L'analyse des résultats de ces expérimentations peut être aussi bien quantitative (analyse des traces d'interaction, des productions, synthèse numérique des questionnaires) que qualitative (analyse des réponses aux questionnaires, entretiens).

QUESTIONS	REPONSE ET SYNTHESE
Objectifs des expérimentations	
Que veut-on faire ou démontrer ?	Notre méthode et la plateforme correspondante est-elle utile pour un expert en EIAH pour la construction et la comparaison de techniques de diagnostic des connaissances ? Les résultats des critères de comparaison sont-ils interprétables et utiles ? Le coût investi pour la compréhension de la méthode est-il amorti en étant réinvesti pour la construction de techniques sur plusieurs domaines/traces ? Notre méthode est-elle utile pour un expert en diagnostic des connaissances, soit pour la formalisation de nouveaux modèles de diagnostic, soit pour l'apport de nouveaux critères de comparaison ?
Quel est l'objet à étudier ?	Méthode d'assistance à la construction ou à la comparaison de techniques de diagnostic des connaissances et plateforme correspondante.
La valeur ajoutée ou éléments novateurs ?	La possibilité de construire et comparer plusieurs techniques de diagnostic basées sur des modèles de diagnostic différents. La possibilité de partager et réutiliser un grand nombre de critères de comparaison de techniques de diagnostic, notamment des critères spécifiques au domaine des EIAH.
Que souhaite-t-on connaître ?	On souhaite connaître les besoins des utilisateurs qui justifieraient pour eux l'usage de nos travaux. On souhaite collecter des techniques de diagnostic construites par les concepteurs pour les évaluer, et connaître l'opinion des concepteurs sur la validité de ces techniques. On souhaite connaître les retours et les usages des concepteurs pour évaluer l'utilité de nos travaux.
Quels sont les acteurs ?	Experts en EIAH maîtrisant la notion de diagnostic des connaissances. Experts en conception et étude du diagnostic des connaissances.

Table 25 : Fiche de présentation des objectifs principaux des expérimentations prévues sur l'utilité et l'utilisabilité de nos travaux.

C. Introduction d'interactions et de paramètres dans l'algorithme de construction et dans le calcul du score biaisé

Notre algorithme semi-automatique pour la construction de techniques de diagnostic est en l'état guidé par le concepteur, mais il n'est pas possible d'intervenir au cours de l'exécution de l'algorithme ou de modifier ses paramètres.

Une première perspective intéressante serait de proposer une phase d'interaction entre la plateforme et le constructeur entre les étapes 2 et 3 de l'algorithme (pour rappel, l'étape 2 est l'instanciation du diagnostic au domaine et l'étape 3 est l'apprentissage des paramètres si besoin). Une telle phase d'interaction permettrait à un expert de modifier directement la structure de la technique de diagnostic ; actuellement, le concepteur ne peut qu'obtenir le code source final qu'il peut modifier, mais il est alors nécessaire de recalculer les paramètres. Permettre de modifier la structure des techniques via une interface, et non

dans le code source, ainsi que permettre de recalculer les paramètres après modification serait une amélioration de l'utilisabilité de notre plateforme. Cette perspective simplifierait également le travail itératif que prône notre méthode (construction d'une ontologie des traces pour la construction, comparaison, modification de l'ontologie, comparaison, etc.).

Une seconde perspective est de rendre certains composants de l'algorithme paramétrables pour un expert. Nous pensons en premier lieu aux méthodes d'apprentissage des paramètres des techniques de diagnostic. Baker et al. (Baker et al., 2010) ont par exemple mené une expérimentation sur différents algorithmes d'apprentissage de paramètres pour le Knowledge tracing, et ont montré que ces algorithmes peuvent avoir un impact significatif sur les performances de la technique de diagnostic en terme de précision de la prédiction.

Une troisième perspective est d'expérimenter de nouveaux scores pour la construction des techniques de diagnostic. En l'état, nous utilisons un score combinant un score statistique et un score biaisé que nous proposons. Lors de futures expérimentations, il est possible d'étudier nos scores en faisant varier soit le score statistique, soit le score biaisé. Dans nos travaux, notre apport est le score biaisé : nous proposons en guise de perspective de faire varier les éléments suivants du score biaisé : l'initialisation de la matrice *R* (matrice représentant les probabilités d'association entre les variables de traces et les variables du modèle de diagnostic), la loi de probabilité utilisée pour le calcul du score, et le poids accordé aux informations apportées par le concepteur via à l'ontologie.

Une quatrième perspective est d'expérimenter notre algorithme d'apprentissage sur un grand nombre de bases de traces d'apprenants. Nous pourrions par exemple sélectionner un ensemble de bases de traces dans des plateformes telles que Datashop ou Undertracks (Bouhineau et al., 2013; Koedinger et Stamper, 2010). L'objectif de ces expérimentations serait d'évaluer plus finalement :

- L'impact du volume de traces sur l'algorithme de construction semi-automatique et sur le calcul des critères de comparaison
- L'impact de la nature du domaine en fonction de caractéristiques du domaine comme sa définition (bien ou mal défini)
- L'impact de l'ontologie conçue par le concepteur, notamment en fonction du niveau de détails (haut niveau ou détaillé)

D. Collecte et partage de critères et critères EIAH

Dans nos travaux, nous proposons une collection de critères de comparaison qui peuvent être calculés automatiquement par notre plateforme pour les techniques construites.

Une première perspective est donc de permettre le partage de nouveaux critères de comparaison pour la plateforme. Cette perspective pose en premier lieu des questions techniques : il faut un dépôt pour le code et la documentation des critères, avec une interface en ligne pour la navigation dans le dépôt. Cette notion de dépôt pose le problème de pouvoir naviguer parmi ces critères et de choisir ceux qui intéressent le concepteur. Pour ce faire, nous pensons qu'une catégorisation plus fine des critères permettrait de faire des

requêtes dans le dépôt. Dans nos travaux, nous proposons trois formes de catégorisation (chapitre 5) : critères simples ou complexes, types de résultats, buts du critère. En l'état, le but est informel (textuel) et ne permet pas facilement de faire des recherches parmi les critères. Une perspective est donc d'affiner la catégorisation par les buts. Une catégorisation plus fine peut permettre également de faire de la recommandation si le concepteur fournit une liste de mots clés ou d'objectifs qui l'intéressent dans son travail. Du point de vue de l'utilisabilité, une perspective à terme est de développer notre plateforme sur le Web, et de permettre le calcul des critères sur des techniques déjà construites, donc sans utiliser notre méthode d'assistance à la construction. Dans nos travaux, nous avons bien séparé les deux méthodes d'assistance (construction et comparaison), mais notre plateforme ne permet pas d'utiliser séparément chaque méthode facilement.

Cette perspective sur la collecte et le partage de critères peut inclure la question de critères de comparaison qui seraient calculés à partir des résultats d'autres critères de comparaison. Nous avons un exemple simple dans nos travaux : les critères AIC et BIC sont calculés à partir de la complexité, en termes de nombre de paramètres, des techniques. Or, la complexité en termes de nombre de paramètres est elle-même un critère de comparaison. Nous pouvons faire l'hypothèse que plus les critères seront nombreux et variés, plus il sera intéressant de proposer des critères de haut niveau permettant de synthétiser les résultats d'un ensemble de critères de comparaison. Nous avons par exemple constaté que, lors du calcul des divers critères que nous avons proposés pour la précision de la prédiction (avec notamment RMSE, BIC, AIC, AUC, précision de prédiction simple), les résultats de ces critères pouvaient être contradictoires ou non, significatifs ou non. Il serait, à notre sens, possible de proposer un critère de synthèse sur la précision de prédiction qui attribuerait un poids plus ou moins important à chaque critère de comparaison soit en fonction de la signifiance statistique de leurs résultats, soit en fonction d'objectifs du concepteur. Une perspective est donc d'étudier si de tels critères de synthèse seraient utiles au concepteur pour l'interprétation des résultats.

Une seconde perspective est de poursuivre le travail sur la conception de critères spécifiques aux EIAH, comme le critère que nous avons proposé mesurant l'impact des techniques de diagnostic sur l'apprentissage d'une stratégie d'aide pour un EIAH, et qui a montré dans nos expérimentations une différence statistiquement significative entre les techniques.

Par exemple, un des buts d'une technique de diagnostic est d'inférer un modèle des connaissances des apprenants qui puissent être utilisés pour adapter les prises de décision de l'EIAH. Parmi ces processus de prises de décisions figurent donc le choix d'un type d'aide, d'un indice, d'une rétroaction épistémique, du prochain exercice à proposer à l'apprenant, d'un ensemble d'exercices (parcours) à proposer à l'apprenant, d'un affichage personnalisé de son profil à l'apprenant, d'une présentation personnalisée d'un exercice… Il existe dans la littérature, pour tous ces aspects, des travaux (Barnes et Stamper, 2008; Chi et al., 2010; Gertner et al., 1998) qui se sont d'une part intéressés à l'apprentissage (semi-)automatique

de stratégies de prise de décision dans l'EIAH, d'autre part à l'apprentissage de stratégies utilisant le résultat d'une technique de diagnostic.

Schute et Zapata-Rivera (Shute et Zapata-Rivera, 2007) définissent dans leurs travaux un framework (Figure 47) décrivant l'usage de modèles d'apprenants pour l'adaptation de l'EIAH à l'apprenant. Elle décrit un cycle temporel à quatre états (pouvant être facultatifs) :

- Capture : collecte d'informations sur l'apprenant, y compris des informations sur ses connaissances via le modèle de l'apprenant
- Analyze : inférence et mise à jour du modèle de l'apprenant (le diagnostic des connaissances est fait à cette étape, bien que Shute ne fasse pas de distinction entre diagnostic des connaissances et diagnostic comportemental)
- Select : sélection de ressources ou de contenus en fonction du modèle de l'apprenant et des objectifs de l'EIAH
- Present : présentation des ressources ou contenus sélectionnés dans l'EIAH via des techniques adaptatives

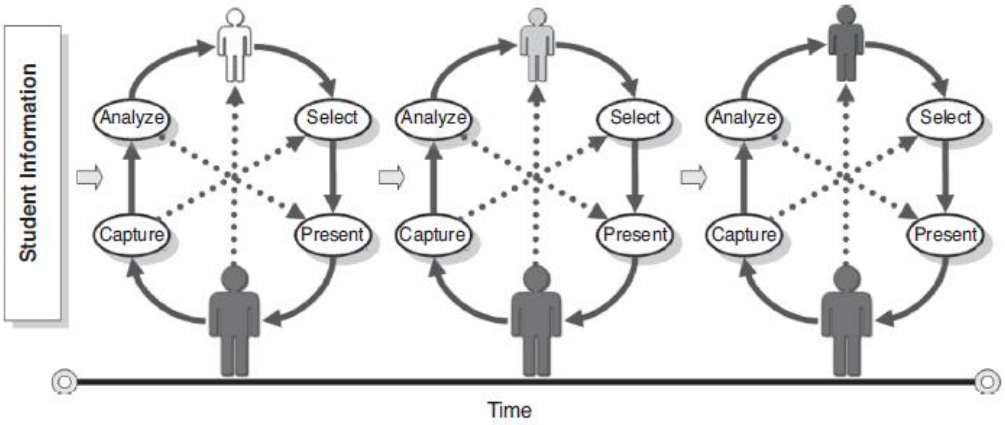


Figure 47 : Cycle à quatre états proposé par Shute pour décrire les processus d'adaptation dans les EIAH à partir du modèle de l'apprenant. La silhouette en bas représente l'apprenant, et celle du haut représente le modèle de l'apprenant qui est censé, selon Shute et Zapata-Rivera, devenir de plus en plus riche et précis en fonction du temps, donc au cours de l'interaction de l'apprenant avec l'EIAH.

Le critère que nous avons proposé sur l'impact des techniques de diagnostic sur l'apprentissage d'une stratégie d'aide se positionne dans le cycle entre le modèle de l'apprenant et la sélection de ressources ou contenus (ici, sélection d'un type d'aide). Nous pouvons donc préciser notre perspective comme la proposition de critères de comparaison permettant l'étude de l'impact d'une technique de diagnostic sur la sélection de contenus ou ressources dans l'EIAH.

2. Transposition de nos questions de recherche dans d'autres domaines des EIAH

Nous avons détaillé jusqu'ici nos perspectives dans le prolongement de nos contributions. Dans cette section, nous allons nous intéresser à des perspectives de mise en application de

nos questions de recherche et du principe de nos contributions dans d'autres domaines des EIAH.

A. Collaboration, émotions, jeux sérieux et modèles ouverts

Nous nous sommes intéressés dans nos contributions au diagnostic des connaissances dans le contexte de l'apprentissage individuel. Or, les EIAH proposent de plus en plus des situations d'apprentissage plus complexes, notamment en s'intéressant :

- à la collaboration entre les apprenants, qui influe sur l'acquisition des connaissances
- à la motivation des apprenants, qui impacte l'apprentissage
- aux autres émotions des apprenants, qui peuvent aussi avoir un impact sur l'apprentissage

L'apprentissage collaboratif vise à favoriser l'acquisition de connaissances en poussant les apprenants à poser des questions, dialoguer, se justifier ou raisonner en groupe. Dans ce contexte, deux types de modèles d'apprenants sont possibles : 1/ les modèles d'apprenants qui visent à évaluer les capacités de collaboration des apprenants (i.e. les apprenants maîtrisent-ils les connaissances pour expliquer leurs opinions, pour justifier leur choix, pour travailler en groupe…) (Lee et al., 2006; McCalla et al., 2000; Murray et al., 2012; Soller et al., 2002) ; 2/ les modèles d'apprenants classiques, qui visent à évaluer les connaissances maîtrisées par l'apprenant sur un domaine, mais prenant en compte le groupe (état des connaissances du groupe, collaboration au sein du groupe…)(Gijlers et De Jong, 2005; Roth et Roychoudhury, 1993).

Concernant la motivation, le but est de proposer des situations d'apprentissage motivantes et engageantes pour les apprenants, par exemple via le jeu. Dans ce contexte, divers travaux sur les jeux sérieux (Garris et al., 2002; Lane et al., 2010; Shute et al., 2012) se sont intéressés à diagnostiquer les connaissances acquises par les apprenants ou à utiliser le diagnostic des connaissances pour proposer des rétroactions à l'apprenant lors de son interaction avec le jeu sérieux. Ces travaux prennent en compte pour le diagnostic des connaissances la motivation ou les ressorts de jeu utilisés. D'autres travaux (Baker et al., 2006, 2004) s'intéressent à l'opposé à détecter la démotivation d'un apprenant en utilisant une technique de diagnostic des connaissances.

Enfin, plusieurs travaux montrent que la prise en compte des émotions lors des interactions (par exemple, ennui, satisfaction…) entre l'apprenant et l'EIAH peut avoir un impact sur l'apprentissage, notamment en adaptant le comportement de l'EIAH en fonction de ces émotions afin de renforcer la motivation des apprenants. Parmi ces travaux, plusieurs (Conati, 2002; D'Mello et al., 2006; Litman et Forbes-Riley, 2004) proposent la conception d'un modèle prédictif des émotions des apprenants (et donc d'un « diagnostic des émotions »), ou, plus proche de notre thématique, d'autres travaux (Arroyo et Woolf, 2001; Craig et al., 2004; D'Mello et al., 2005; Kort et Reilly, 2002; Stein et Levine, 1990) proposent de diagnostiquer l'acquisition des connaissances des apprenants en prenant en compte ces émotions.

Nous pouvons identifier dans ces trois domaines (collaboration, émotion, jeux sérieux) des problématiques proches des nôtres : usage de traces pour construire (semi-automatiquement) des diagnostics ou des modèles d'apprenants, absence de formalisation claire des modèles d'apprenants, et peu d'analyses comparatives facilement reproductibles. Une question de recherche à long terme serait de s'intéresser à la possibilité de transposer nos contributions dans ces domaines des EIAH.

Prenons l'exemple des émotions. D'Mello et al. sujet (D'Mello et al., 2005) synthétisent dans un article trois modèles théoriques permettant d'exprimer le lien entre émotions et apprentissage. Le premier modèle stipule que l'apprentissage ne se fait que si l'état émotionnel de l'apprenant est favorable, et cet état émotionnel dépend des objectifs de l'apprenant. Le second modèle stipule que les émotions peuvent être classées comme positives ou négatives ; les émotions négatives favorisent l'apparition de misconceptions alors que les émotions positives favorisent l'apprentissage. Le troisième modèle stipule que l'apprentissage est favorisé par les émotions de types « confusion », « hésitation », « perturbation ». Les questions qui se posent pour la possibilité d'appliquer nos contributions sont donc les suivantes : est-il possible de proposer une formalisation de ces modèles théoriques ? Est-il possible de les implémenter ? Si oui, il est envisageable de pouvoir assister la construction et la comparaison de ces modèles en reprenant notre méthode d'assistance et en proposant des critères de comparaison adaptés à ce domaine.

Une autre approche dans l'étude du diagnostic des connaissances sont les modèles d'apprenants ouverts (*open student models*). Dans les travaux sur le sujet (Bull et Kay, 2007; Bull et al., 1995; Dimitrova et al., 2001; Hartley et Mitrovic, 2002), le résultat du diagnostic des connaissances n'est pas seulement utilisé pour les prises de décision de l'EIAH, mais est affiché à l'apprenant dans un but d'auto-évaluation et de motivation. Une question de recherche qui se pose dans ce contexte est le processus d'extraction dans le résultat du diagnostic des informations à afficher aux apprenants susceptibles de favoriser l'apprentissage. Par extension, il serait possible d'étudier l'impact des différentes techniques de diagnostics existantes sur ce processus d'extraction.

B. Conception d'EIAH

Dans nos travaux, nous proposons une méthode pour la conception d'une technique de diagnostic de façon indépendante des autres composants de l'EIAH (approche modulaire) et en prenant en compte lors de la phase de conception des résultats de comparaison entre plusieurs techniques au moyen de traces d'apprenants. Une perspective est l'étude de la possibilité d'appliquer une telle approche modulaire pour les autres composants des EIAH, *a minima* pour les composants réalisant de l'inférence ou des prises de décision, par exemple les composants pédagogiques (rétroaction, choix des exercices, personnalisation).

Prenons l'exemple du choix des rétroactions : il existe plusieurs similitudes avec le diagnostic des connaissances. Concernant leur construction, on observe les deux mêmes méthodes génériques que pour le diagnostic des connaissances : les outils auteurs (Murray, 1999) qui permettent aux concepteurs de concevoir des rétroactions avec un coût réduit en programmation, ou l'apprentissage automatique qui permet d'apprendre une stratégie de

choix de rétroactions à partir de traces d'apprenants (Barnes et Stamper, 2008; Chi et al., 2010; Gertner et al., 1998). Il existe de même plusieurs approches pour choisir un feedback et plusieurs variables pouvant être prises en compte (Luengo, 2009; Shute et Zapata-Rivera, 2007). Du point de vue implémentation, le choix du feedback peut se faire via des réseaux bayésiens, des règles… Enfin, il existe des études (Barnes et al., 2008; Mostow et al., 2003; Razzaq et Heffernan, 2006) comparant deux stratégies pour le choix d'une rétroaction appliquées sur un domaine donné, mais basée sur des expérimentations en classe.

Une perspective à long terme est donc d'étudier la possibilité de proposer une méthode similaire à la nôtre pour le feedback, permettant la construction de différentes stratégies de feedback et leur comparaison, de façon modulaire. Cette perspective pourrait donc s'articuler avec notre travail pour fournir plusieurs méthodes permettant la construction de modules indépendants pour un EIAH, avec l'obtention de résultats d'évaluation pour chacun de ces modules tout au long du développement de l'EIAH.

Chapitre 9 : Bibliographie

Aarts, E.H.L., Lenstra, J.K., 2003. *Local Search in Combinatorial Optimization*. Princeton University Press.

Akaike, H., 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19, 716–723.

Aleven, V., McLaren, B., Sewall, J., Koedinger, K., 2006. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 61–70.

Amershi, S., Conati, C., 2007. Unsupervised and supervised machine learning in user modeling for intelligent learning environments. Actes de l'International Conference on Intelligent User Interfaces, pp. 72–81.

Anderson, J.R., 1983. *The architecture of cognition*. Harvard University Press, Cambridge, Cognitive science series.

Anderson, J.R., 1990. Analysis of student performance with the LISP tutor, in: *Diagnostic Monitoring of Skill and Knowledge Acquisition*. N. Fredericksen, R. Glaser, A. Lesgold, & M. Shaffo, pp. 27–50.

Anderson, J.R., 1996. ACT: A simple theory of complex cognition. *American Psychologist* 51, 355–365.

Anderson, J.R., Boyle, C.F., Yost, G., 1985. The Geometry Tutor. Actes de l'International Joint Conferences on Artificial Intelligence, pp. 1–7.

Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R., 1995. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences* 4, 167–207.

Armstrong, J.S., Collopy, F., 1992. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting* 8, 69–80.

Arroyo, I., Murray, T., Woolf, B.P., Beal, C., 2004. Inferring unobservable learning variables from students' help seeking behavior. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 244–270.

Arroyo, I., Woolf, B.P., 2001. Improving student models by reasoning about cognitive ability, emotions and gender. Actes de l'International Conference on User Modeling. Springer, pp. 265–267.

Baader, F., 2003. *The description logic handbook: theory, implementation, and applications*. Cambridge university press.

Baffes, P., Mooney, R., 1996. Refinement-based student modeling and automated bug library construction. *Journal of Artificial Intelligence in Education* 7, 75–116.

Baker, R.S.J.d., Corbett, A.T., Gowda, S., M., Wagner, A.Z., MacLaren, B.M., Kauffman, L.R., Mitchell, A.P., Giguere, S, 2010. Contextual Slip and Prediction of Student Performance after Use of an Intelligent Tutor. Actes de l'International Conference on User Modeling, Adaptation, and Personalization, Springer Berlin Heidelberg, pp. 52–63.

Baker, R., Corbett, A., Aleven, V., 2008. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. Actes de l'International Conference Intelligent Tutoring Systems, pp. 406–415.

Baker, R.S., Corbett, A.T., Koedinger, K.R., 2004. Detecting student misuse of intelligent tutoring systems. Actes de l'International Conference on Intelligent tutoring systems, pp. 531–540.

Baker, R.S., Corbett, A.T., Koedinger, K.R., Evenson, S., Roll, I., Wagner, A.Z., Naim, M., Raspat, J., Baker, D.J., Beck, J.E., 2006. Adapting to when students game an

intelligent tutoring system. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 392–401.
Balacheff, N., 1994. Didactique et intelligence artificielle. *Recherches en didactique des mathématiques* 14, 9–42.
Balacheff, N., 1995. Conception, propriété du système sujet/milieu. Actes de l'IREM de Clermont-Ferrand.
Balacheff, N., Gaudin, N., 2002. Students conceptions: an introduction to a formal characterization. *Cahier Leibniz* 65, 1–21.
Barnes, T., Stamper, J., 2008. Toward automatic hint generation for logic proof tutoring using historical student data. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 373–382.
Barnes, T., Stamper, J., Lehman, L., Croy, M., 2008. A pilot study on logic proof tutoring using hints generated from historical student data. Actes de l'International Conference on Educational Data Mining, p. 197.
Baum, L.E., Petrie, T., 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics* 37, 1554–1563.
Baum, L.E., Petrie, T., Soules, G., Weiss, N., 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics* 41, 164–171.
Beck, J., Woolf, B., 1998. Using a learning agent with a student model. Actes de l'International Conference Intelligent Tutoring Systems, pp. 6–15.
Beck, J.E., 2007. Difficulties in inferring student knowledge from observations (and why you should care), in: Educational Data Mining: Supplementary Proceedings of the 13 Th International Conference of Artificial Intelligence in Education. pp. 21–30.
Beck, J.E., Sison, J., 2006. Using knowledge tracing in a noisy environment to measure student reading proficiencies. *International Journal of Artificial Intelligence in Education* 16, 129–143.
Beck, J.E., Woolf, B.P., Beal, C.R., 2000. ADVISOR: a machine-learning architecture for intelligent tutor construction. Actes de l'AAAI Conference on Artificial Intelligence, University of Massachusetts at Amherst, pp. 552–557.
Beck, J.E., Xiong, X., 2013. Limits to Accuracy : How Well Can We Do at Student Modeling? Actes de l'International Conference on Educational Data Mining, pp. 4–11.
Berge, C., 1987. Hypergraphes: combinatoire des ensembles finis. Gauthier-Villars.
Blessing, S., Gilbert, S., 2008. Evaluating an authoring tool for model-tracing intelligent tutoring systems. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 204–215.
Bouhineau, D., Luengo, V., Mandran, N., Ortega, M., Wajeman, C., 2013. Conception et mise en place d'un entrepôt de traces et processus de traitement EIAH: UnderTracks. Actes de la Conférence sur les Environnements Informatiques pour l'Apprentissage Humain, p. 41.
Bull, S., Brna, P., Pain, H., 1995. Extending the scope of the student model. *User Modeling and User-Adapted Interaction* 5, 45–65.
Bull, S., Kay, J., 2007. Student Models that Invite the Learner In: The SMILI:() Open Learner Modelling Framework. *International Journal of Artificial Intelligence in Education* 17, 89–120.
Burton, R.R., 1981. *Diagnosing bugs in a simple procedural skill*. Xerox Corporation, Palo Alto Research Center.
Cen, H., Koedinger, K., Junker, B., 2006. Learning factors analysis–a general method for cognitive model evaluation and improvement. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 164–175.

Cen, H., Koedinger, K., Junker, B., 2008. Comparing two IRT models for conjunctive skills. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 796–798.

Chaachoua, H., 2011. La praxéologie comme modèle didactique pour la problématique EIAH. Etude de cas: la modélisation des connaissances des élèves. Actes du Séminaire National de Didactique des mathématiques 2011., pp. 81–101.

Chaachoua, H., Nicaud, J.-F., Bronner, A., Bouhineau, D., 2004. Aplusix, a learning environment for algebra, actual use and benefits. International Congress on Mathematical Education 8.

Chi, M., Koedinger, K., Gordon, G., Jordan, P., VanLehn, K., 2011. Instructional factors analysis: A cognitive model for multiple instructional interventions, in: International Conference on Educational Data Mining. Eindhoven, the Netherlands.

Chi, M., VanLehn, K., Litman, D., Jordan, P., 2010. Inducing effective pedagogical strategies using learning context features. Actes de l'International Conference on User Modeling, Adaptation, and Personalization, pp. 147–158.

Cohen, W.W., 1995. Fast Effective Rule Induction. Actes de l'International Conference on Machine Learning, Morgan Kaufmann, pp. 115–123.

Conati, C., 2002. Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence* 16, 555–575.

Corbett, A.T., Anderson, J.R., 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modelling and User-Adapted Interaction* 4, 253–278.

Cornuéjols, A., Miclet, L., 2011. *Apprentissage artificiel: concepts et algorithmes*. Eyrolles.

Craig, S., Graesser, A., Sullins, J., Gholson, B., 2004. Affect and learning: an exploratory look into the role of affect in learning with AutoTutor. *Journal of Educational Media* 29, 241–250.

D'Mello, S., Craig, S.D., Gholson, B., Franklin, S., Picard, R., Graesser, A.C., 2005. Integrating affect sensors in an intelligent tutoring system. Actes de Affective Interactions: The Computer in the Affective Loop Workshop at Intelligent User Interface, pp. 7–13.

D'Mello, S.K., Craig, S.D., Sullins, J., Graesser, A.C., 2006. Predicting affective states expressed through an emote-aloud procedure from AutoTutor's mixed-initiative dialogue. *International Journal of Artificial Intelligence in Education* 16, 3–28.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*. Series B (Methodological) 39, 1–38.

Desmarais, M., 2011. Performance comparison of item-to-item skills models with the IRT single latent trait model. Actes de l'International Conference on User Modeling, Adaption and Personalization, pp. 75–86.

Desmarais, M.C., Baker, R.S.J. d, 2012. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction* 22, 9–38.

Desmarais, M.C., Meshkinfam, P., Gagnon, M., 2006. Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction* 16, 403–434.

Dimitrova, V., Self, J., Brna, P., 2001. Applying interactive open learner models to learning technical terminology. Actes de l'International Conference on User Modeling, Springer, pp. 148–157.

Draper, N.R., Smith, H., Pownell, E., 1966. *Applied regression analysis*. Wiley New York.

Fournier, J.-C., 2011. *Théorie des graphes et applications: Avec exercices et problèmes*. Lavoisier.

Fournier-Viger, P., Nkambou, R., Mayers, A., Mephu-Nguifo, E., Faghihi, U., 2012. Multi-paradigm generation of tutoring feedback in robotic arm manipulation training, in:

ITS'12. Actes de l'International Conference on Intelligent Tutoring Systems, Springer-Verlag, Berlin, Heidelberg, pp. 233–242.

Garris, R., Ahlers, R., Driskell, J.E., 2002. Games, motivation, and learning: A research and practice model. *Simulation & gaming* 33, 441–467.

Georgeon, O., 2008. Analyzing traces of activity for modeling cognitive schemes of operators. *Artificial Intelligence and Simulation of Behaviour Quarterly Bulletin* 127, 1–3.

Gertner, A.S., Conati, C., VanLehn, K., 1998. Procedural help in Andes: Generating hints using a Bayesian network student model. Actes de la National Conference on Artificial Intelligence, pp. 106–111.

Gijlers, H., De Jong, T., 2005. The relation between prior knowledge and students' collaborative discovery learning processes. *Journal of Research in Science Teaching* 42, 264–282.

Goel, G., Lallé, S., Luengo, V., 2012. Fuzzy Logic Representation for Student Modelling. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 428–433.

Gong, Y., Beck, J., Heffernan, N., 2010. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 35–44.

Gong, Y., Beck, J.E., Heffernan, N.T., 2011. How to Construct More Accurate Student Models: Comparing and Optimizing Knowledge Tracing and Performance Factor Analysis. *International Journal of Artificial Intelligence in Education* 21, 27–46.

Gonzales-Brenes, J., Mostow, J., 2012. Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models. Actes de l'International Conference on Educational Data Mining, pp. 49–56.

Gonzalez-Brenes, J., Mostow, J., 2013. What and When do Students Learn? Fully Data-Driven Joint Estimation of Cognitive and Student Models. Actes de l'International Conference on Educational Data Mining, pp. 236–239.

Guillaume, S., Charnomordic, B., Lablée, J.-L., 2002. Fispro: An open source portable software for fuzzy inference systems. INRA-Cemagref.

Hanley, J.A., McNeil, B.J., 1983. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology* 148, 839–843.

Hartley, D., Mitrovic, A., 2002. Supporting learning by opening the student model. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 453–462.

Heckerman, D., 2008. A tutorial on learning with Bayesian networks. *Innovations in Bayesian Networks* 33–82.

Heiner, C., Beck, J., Mostow, J., 2004. Improving the help selection policy in a Reading Tutor that listens. Actes de l'InSTIL/ICALL Symposium on NLP and Speech Technologies in Advanced Language Learning Systems, pp. 195–198.

Holmes, G., Donkin, A., Witten, I.H., 1994. Weka: A machine learning workbench, in: Second Australian and New Zealand Conference on Intelligent Information Systems. pp. 357–361.

Jean, S., 2000. *PÉPITE: un système d'assistance au diagnostic de compétences* (thèse). Université du Maine.

Johns, J., Mahadevan, S., Woolf, B., 2006. Estimating student proficiency using an item response theory model. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 473–480.

Jonsson, A., Johns, J., Mehranian, H., Arroyo, I., Woolf, B., Barto, A., Fisher, D., Mahadevan, S., 2005. Evaluating the feasibility of learning student models from data. Actes du AAAI Workshop on Educational Data Mining, pp. 1–6.

Kodaganallur, V., Weitz, R.R., Rosenthal, D., 2005. A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education* 15, 117–144.

Kodaganallur, V., Weitz, R.R., Rosenthal, D., 2006. An Assessment of Constraint-Based Tutors: A Response to Mitrovic and Ohlsson's Critique of" A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms". *International Journal of Artificial Intelligence in Education* 16, 291–321.

Koedinger, K.R., Baker, R., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J., 2010. A data repository for the EDM community: The PSLC DataShop, in: *Handbook of Educational Data Mining*. Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d., pp. 43–55.

Koedinger, K.R., McLaughlin, E.A., Stamper, J.C., 2012. Automated Student Model Improvement. Actes de l'International Conference on Educational Data Mining. pp. 17–24.

Koedinger, K.R., Pavlik Jr, P.I., Stamper, J.C., Nixon, T., Ritter, S., 2011. Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing. Actes de l'International Conference on Educational Data Mining. pp. 91–100.

Koedinger, K.R., Stamper, J.C., 2010. A Data Driven Approach to the Discovery of Better Cognitive Models. Actes de l'International Conference on Educational Data Mining. pp. 325–326.

Koedinger, K.R., Stamper, J.C., McLaughlin, E.A., Nixon, T., 2013. Using data-driven discovery of better student models to improve student learning. Actes de l'International Conference on Artificial Intelligence in Education, pp. 421–430.

Kort, B., Reilly, R., 2002. Analytical models of emotions, learning and relationships: towards an affect-sensitive cognitive machine. Actes de la Conference on virtual worlds and simulation.

Lallé, S., Mostow, J., Luengo, V., Guin, N., 2013. Comparing Student Models in Different Formalisms by Predicting their Impact on Help Success. Actes de l'International Conference on Artificial Intelligence in Education, Memphis, TN.

Lane, H.C., Hays, M.J., Auerbach, D., Core, M.G., 2010. Investigating the relationship between presence and learning in a serious game. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 274–284.

Langley, P., Gennari, J.H., Iba, W., 1987. Hill-Climbing Theories of Learning. DTIC Document.

Langley, P., Ohlsson, S., 1984. Automated cognitive modeling. Actes de la Second National Conference on Artificial Intelligence.

Le, N.-T., Pinkwart, N., 2012. Can Soft Computing Techniques Enhance the Error Diagnosis Accuracy for Intelligent Tutors?, in: Lecture Notes in Computer Science. Actes de l'International Conference on Intelligent Tutoring Systems, Springer Berlin / Heidelberg, pp. 320–329.

Lee, E.Y., Chan, C.K., van Aalst, J., 2006. Students assessing their own collaborative knowledge building. *International Journal of Computer-Supported Collaborative Learning* 1, 57–87.

Lee, Y.J., Palazzo, D.J., Warnakulasooriya, R., Pritchard, D.E., 2008. Measuring student learning with item response theory. *Physical Review Special Topics-Physics Education Research* 4, 010102.

Litman, D.J., Forbes-Riley, K., 2004. Predicting student emotions in computer-human tutoring dialogues. Actes du Annual Meeting on Association for Computational Linguistics, p. 351.

Luengo, V., 2009. *Les rétroactions épistémiques dans les Environnements Informatiques pour l'Apprentissage Humain* (Habilitation à diriger la recherche). Université de Grenoble.

Luengo, V., Vadcard, L., Tonetti, J., Dubois, M., 2011. Diagnostic des connaissances et rétroaction épistémique adaptative en chirurgie. *Revue d'intelligence artificielle* 25, 499–524.

Martin, B., Koedinger, K.R., Mitrovic, A., Mathan, S., 2005. On using learning curves to evaluate ITS. Actes de l'International Conference on Artificial Intelligence in Education, pp. 419–426.

Matsuda, N., Cohen, W.W., Koedinger, K.R., 2005. Building cognitive tutors with programming by demonstration. Actes de l'International Conference on Inductive Logic Programming, pp. 41–46.

Matsuda, N., Cohen, W.W., Sewall, J., Lacerda, G., Koedinger, K.R., 2007. Predicting students' performance with simstudent: Learning cognitive skills from observation. *Frontiers in Artificial Inteligence and Applications* 158, 467.

Mayo, M., Mitrovic, A., 2001. Optimising ITS behaviour with Bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education* 12, 124–153.

McCalla, G., Vassileva, J., Greer, J., Bull, S., 2000. Active learner modelling. Actes de l'International Conference on Intelligent Tutoring Systems. pp. 53–62.

McNemar, Q., 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12, 153–157.

Minh Chieu, V., Luengo, V., Vadcard, L., Tonetti, J., 2010. Student modeling in complex domains: Exploiting symbiosis between temporal Bayesian networks and fine-grained didactical analysis. *International Journal of Artificial Intelligence in Education* 20, 269–301.

Mitrović, A., 1998. Experiences in implementing constraint-based modeling in SQL-Tutor, in: Lecture Notes in Computer Science. Actes de l'International Conference on Intelligent Tutoring Systems, Springer Berlin / Heidelberg, pp. 414–423.

Mitrovic, A., Koedinger, K., Martin, B., 2003. A comparative analysis of cognitive tutoring and constraint-based modeling. Actes de l'International Conference on User Modeling, pp. 313–322.

Mitrovic, A., Martin, B., Suraweera, P., 2007. Intelligent Tutors for All: The Constraint-Based Approach. *IEEE Intelligent Systems* 22, 38–45.

Mitrovic, A., McGuigan, N., Martin, B., Suraweera, P., Milik, N., Holland, J., 2008. Authoring constraint-based tutors in ASPIRE: A case study of a capital investment tutor. Actes de la World Conference on Educational Multimedia, Hypermedia & Telecommunications, pp. 4607–4616.

Mitrovic, A., Ohlsson, S., 2006. A Critique of Kodaganallur, Weitz and Rosenthal," A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms". *International Journal of Artificial Intelligence in Education* 16, 277–289.

Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., Holland, J., 2006. Authoring constraint-based tutors in ASPIRE. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 41–50.

Mostow, J., Aist, G., 2001. Evaluating tutors that listen: An overview of Project LISTEN., in: *Smart Machines in Education: The Coming Revolution in Educational Technology*. MIT/AAAI Press, pp. 169 – 234.

Mostow, J., Aist, G., Burkhead, P., Corbett, A., Cuneo, A., Eitelman, S., Huang, C., Junker, B., Sklar, M.B., Tobin, B., 2003. Evaluation of an automated reading tutor that listens: comparison to human tutoring and classroom instruction. *Journal of Educational Computing Research* 29, 61–117.

Murray, T., 1999. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education* 10, 98–129.

Murray, T., 2003. Eon: Authoring tools for content, instructional strategy, student model, and interface design, in: *Authoring Tools for Advanced Technology Learning Environments, Toward Cost-effective Adaptive, Interactive, and Intelligent Educational Software*. Murray, T., Blessing, S., Ainsworth, S., pp. 309–340.

Murray, T., Blessing, S., Ainsworth, S., 2003. *Authoring tools for advanced technology learning environments: toward cost-effective adaptive, interactive, and intelligent educational software*. Springer.

Murray, T., Woolf, B.P., Xu, X., Shipe, S., Howard, S., Wing, L., 2012. Towards supporting social deliberative skills in online classroom dialogues and beyond. Actes de l'International Conference on Intelligent Tutoring Systems.

Naïm, P., 2007. *Réseaux bayésiens*. Eyrolles.

Neapolitan, R.E., 2004. *Learning Bayesian networks*. Pearson Prentice Hall.

Nicaud, J.-F., Chaachoua, H., Bittar, M., 2006. Automatic Calculation of Students' Conceptions in Elementary Algebra from Aplusix Log Files. Actes de l'International Conference on Intelligent Tutoring Systems, Springer Berlin Heidelberg, pp. 433–442.

Nkambou, R., Frasson, C., Gauthier, G., 2003. CREAM-Tools: An authoring environment for knowledge engineering in intelligent tutoring systems, in: *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-effective Adaptive, Interactive, and Intelligent Educational Software*. Murray, T., Blessing, S., Ainsworth, S., pp. 93–138.

Nkambou, R., Gauthier, G., Frasson, C., 1997. Un modèle de représentation des connaissances relatives au contenu dans un système tutoriel intelligent. *Sciences et techniques éducatives* 4, 299–330.

Ohlsson, S., 1994. Constraint-based student modeling. *NATO ASI Series F Computer and Systems Sciences* 125, 167–189.

Ohlsson, S., 1996. Learning from performance errors. *Psychological review* 103, 241.

Pardos, Z., Heffernan, N., 2010. Modeling individualization in a bayesian networks implementation of knowledge tracing. Actes de l'International Conference on User Modeling, Adaptation, and Personalization, pp. 255–266.

Pavlik, P.I., Cen, H., Koedinger, K.R., 2009. Performance Factors Analysis–A New Alternative to Knowledge Tracing. Actes de l'International Conference on Artificial Intelligence in Education, pp. 531–538.

Pearl, J., 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Pearson, K., 1896. Mathematical Contributions to the Theory of Evolution. On a Form of Spurious Correlation Which May Arise When Indices Are Used in the Measurement of Organs. *Proceedings of the Royal Society of London* 60, 489–498.

Py, D., 1998. Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève. *Sciences et techniques éducatives* 5.

Razzaq, L., Heffernan, N.T., 2006. Scaffolding vs. Hints in the Assistment System. Actes de l'International Conference Intelligent Tutoring Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 635–644.

Roth, W.-M., Roychoudhury, A., 1993. The concept map as a tool for the collaborative construction of knowledge: A microanalysis of high school physics students. *Journal of Research in Science Teaching* 30, 503–534.

Russell, S.J., Norvig, P., Canny, J.F., 2003. *Artificial intelligence : a modern approach*. Prentice Hall, Englewood Cliffs, N.J.

Schwarz, G., 1978. Estimating the dimension of a model. *The annals of statistics* 6, 461–464.

Self, J.A., 1990. Bypassing the intractable problem of student modelling, in: *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*. Claude Frasson, Gilles Gauthier, pp. 107–123.

Self, J.A., 1994. Formal approaches to student modelling. *NATO ASI Series F Computer and Systems Sciences* 125, 295–295.

Shannon, C.E., 2001. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* 5, 3–55.

Shute, V.J., Rieber, L., Van Eck, R., 2012. Games... and... learning, in: *Trends and Issues in Instructional Design and Technology*. Reiser, R. A., Dempsey, J. V., Boston, pp. 321–332.

Shute, V.J., Zapata-Rivera, D., 2007. Adaptive technologies, in: *Handbook of Research on Educational Communications and Technology*. pp. 277–294.

Simon, H.A., 1974. The structure of ill structured problems. *Artificial intelligence* 4, 181–201.

Simon, H.A., 1978. Information-processing theory of human problem solving, in: *Handbook of Learning and Cognitive Processes*. pp. 271–295.

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T., 2005. ROCR: visualizing classifier performance in R. *Bioinformatics* 21, 3940–3941.

Sison, R., Shimura, M., 1998. Student modeling and machine learning. *International Journal of Artificial Intelligence in Education* 9, 128–158.

Soller, A., Wiebe, J., Lesgold, A., 2002. A machine learning approach to assessing knowledge sharing during collaborative learning activities. Actes de la Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community. pp. 128–137.

Stamper, J.C., Koedinger, K.R., 2011. Human-machine student model discovery and improvement using DataShop. Actes de l'International Conference on Artificial Intelligence in Education, pp. 353–360.

Stein, N.L., Levine, L.J., 1990. Making sense out of emotion: The representation and use of goal-structured knowledge. *Psychological and biological approaches to emotion* 45–73.

Vadcard, L., Luengo, V., 2004. Embedding knowledge in the design of an orthopaedic surgery learning environment. Actes de l'International Conference on Computer Aided Learning in Engineering Education, Grenoble.

VanLehn, K., 1988. Student modeling, in: *Foundations of Intelligent Tutoring Systems*. pp. 55–78.

Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M., 2005. The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education* 15, 147–204.

Vomlel, J., 2004. Bayesian networks in educational testing. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12, 83–100.

Wang, Y., Heffernan, N.T., 2012. The student skill model. Actes de l'International Conference on Intelligent Tutoring Systems, pp. 399–404.

Webb, G.I., Pazzani, M.J., Billsus, D., 2001. Machine learning for user modeling. *User modeling and user-adapted interaction* 11, 19–29.

Wenger, E., 1987. *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann Publishers.

Wilkinson, G.N., Rogers, C.E., 1973. Symbolic description of factorial models for analysis of variance. *Applied Statistics* 392–399.

Woolf, B.P., 2008. *Building intelligent interactive tutors: student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.

Xu, Y., Mostow, J., 2011. Using Logistic Regression to Trace Multiple Subskills in a Dynamic Bayes Net. Actes de l'International Conference on Educational Data Mining, pp. 6–8.
Xu, Y., Mostow, J., 2012. Comparison of methods to trace multiple subskills: Is LR-DBN best? Actes de l'International Conference on Educational Data Mining, pp. 41–48.

Annexes

Sommaire

I. Catégorisation d'EIAH en fonction de leur diagnostic des connaissances 203
II. Traces d'EIAH 204
III. Modèles de diagnostic 208
IV. Ontologies des traces 226
V. Stratégie d'aide pour le Reading tutor 257
VI. Ressources pour la praxéologie 270

I. Catégorisation d'EIAH en fonction de leur diagnostic des connaissances

EIAH	Domaine	Diagnostic des connaissances	Technique d'inférence	Objectifs
PAT	Algèbre	Model tracing (ACT-R)	Règles logiques sur les connaissances du domaine	Rétroactions immédiates
Animal Watch	Maths (arithmétique)	Graphe du domaine (node=connaissance, parent=prérequis) et hiérarchisation des problèmes	Mise à jour des connaissances (des nœuds du graphe)	Pour le choix des problèmes proposés à l'apprenant (en fonction de son profil)
Cardiac tutor	Médecine	Modèle expert et overlay	Reconnaissance de plan. Réseau probabiliste (pour la résolution des problèmes)	Choix des problèmes en fonction du profil de l'apprenant
Wayang outpost	Géométrie	Overlay et modèle affectif de l'apprenant.	Réseau bayésien	Profil de l'apprenant
Andes	Pysique	Model tracing (ACT-R)	Reconnaissance de plan (pour les feedcack) et réseau bayésien	feedback pendant et après l'exercice
Geometry tutor	Géométrie	Model tracing Knowledge tracing	Règles logiques sur les connaissances du domaine Régression linéaire	Rétroactions immédiates Obtenir le profil des apprenants

Teleos	chirurgie (othopédique)	CKC (constructivisme)	réseau bayésien	Rétroactions (en fin d'exercice)
Ambre-add	mathématiques	overlay / erreur	règles logiques, statistique.	Rétroactions (notamment conseils et indices pendant l'exercice), profil
LISP Tutor	informatique (programmation)	Model tracing (ACT-R)	règles logiques	
SQL Tutor	informatique (Base de données)	Constraint-Based	reconnaissance de plan (pattern marching)	
German tutor	langue (apprentissage de l'allemand)	* Natural language processing (NLP) * Score (pour le modèle de l'apprenant)	NLP=grammaire + parseur (règles logiques définies avec Attributed Logic Engine)	Définir le feedback (sous forme de conseils et exercices)
Reading tutor	lecture de l'anglais	* Knowledge tracing	Modèle de Markov caché	Obtenir le profil des apprenants

II. Traces d'EIAH

Datashop

Les traces de la plateforme en ligne Datashop proviennent de divers tuteurs cognitifs développés au PSLC de Pittsburgh aux USA. Elles sont disponibles sur *https://pslcdatashop.web.cmu.edu* (inscription gratuite).

Format des traces :

- **row** clé
- **anon_student_id** numéro de l'étudiant anonymisé
- **session_id** ID de la session
- **time**
- **time_zone**
- **duration** durée pris par l'apprenant pour répondre à une étape
- **student_response_type** type de réponse (par exemple « essai », « demande d'indice »…)
- **student_response_subtype**
- **tutor_response_type** réponse du tuteur (par exemple « évaluation », « aide »…)
- **tutor_response_subtype**
- **problem_hierarchy**
- **problem_name** Problème propose à l'apprenant
- **step_name** Etape corrante de resolution du problem effectée par l'apprenant
- **attempt_at_step** Indique le nombre de réponse successive proposée par l'apprenant pour une étape
- **outcome** Indique si la réponse courante est "correcte" ou "incorrecte"
- **selection** Objet sélectionné par l'apprenant dans le tuteur

- **action** (facultatif)
- **input** Réponse brute saisie par l'apprenant dans l'interface
- **feedback** Rétroaction éventuellement retourné par le tuteur
- **help_level** Niveau de l'aide (générallement de 1 à 3)
- **total_num_hints** Nombre d'indices fournis par le tuteur pour une étape
- **kcs** Knowledge Component, connaissance mobilisée par l'apprenant pour résoudre une étape. Se réfère à un modèle des connaissances.
- **school**
- **class**

Teleos

Tuteur pour l'apprentissage de gestes de chirurgie orthopédique, basé sur un modèle didactique nommé cKc.

Deux fichiers de traces :

1. fichier de traces enregistrées par le tuteur (avec la date, l'action réalisée par l'apprenant, les paramètres du modèle 3D (extrait simplifié dans le tableau ci-dessous)
2. fichier de traces « haptiques » enregistrées directement par un petit robot (donc des données numériques indiquant la position dans l'espace et les forces appliquées par l'élève sur le robot)

Time code	Action	cam3 Dx	cam3 Dy	cam 3Dz	focale 3Dx	focale 3Dy	focale 3Dz	camF acex	camF acey	camF acez
1.29E +12	Impacter Trocart	190.9 277	- 699. 893	114. 75	178.7 11	0	114.7 5	178.7 11	- 245.1 17	114.7 5
1.29E +12	Recommencer Trajectoire	190.9 277	- 699. 893	114. 75	178.7 11	0	114.7 5	178.7 11	- 245.1 17	114.7 5
1.29E +12	Impacter Trocart	190.9 277	- 699. 893	114. 75	178.7 11	0	114.7 5	178.7 11	- 245.1 17	114.7 5
1.29E +12	Recommencer Trajectoire	190.9 277	- 699. 893	114. 75	178.7 11	0	114.7 5	178.7 11	- 245.1 17	114.7 5

Aplusix

Tuteur pour l'algèbre au collège. Deux parties dans les traces :

1. L'en-tête (en rouge) avec des informations sur l'élève, les paramètres et les options utilisés dans l'outil
2. L'ensemble des actions enregistrées (en bleu) pour un apprenant. Le format est le suivant : No (clé unique); duree; action; erreur (vide si aucune); etape; expression; etat (état de l'exercice obtenu par l'élève); curseur (ce que l'élève pointe à l'écran); selection (ce que l'élève à sélectionné)

```
%;NOUVELLE SESSION
#VersionProtocoles=1
#Appli=Aplusix 1.34 - 4/7/2003 #Date=26/09/2003
#Heure=12:32:51
%;ELEVE
#NumeroAnonyme=9027
#laclasse=2°6
... (partie supprimée)
%;PARAMETRES
#PermetMicromonde=non
#PermetExercices=oui
... (partie supprimée)
%;OPTIONS
#petiteFleche=nouvelle
#verification=permanente
... (partie supprimée)
%;CHAMPS No;duree;action;erreur;etape;expression;etat;curseur;selection
%;ACTIONS;#Heure=12:32:51;#TypeProbleme=TpbReduire
1;0.0;enonce;();0;3x+1-2x+4;BienForme;(0 0 0 devant);rien;
2;10.5;placerCurseur;();0;3x+1-2x+4;BienForme;(0 0 0 devant);rien;
3;5.1;placerCurseur;();0;3x+1-2x+4;BienForme;(0 0 0 devant);rien;
... (partie supprimée)
```

EDBA

EIAH pour l'apprentissage de l'algorithmique. Deux types de traces :

1. centrée sur l'activité de l'utilisateur
2. centrée sur un exercice

Dans le cas 1 (activité), les traces contiennent :

- une date
- un type d'information (soit « code » si la trace contient un algorithme écrit par l'apprenant, soit « action » si la trace contient une action effectuée par l'apprenant à l'écran, soit « sav » pour les sauvegardes, soit « exec » pour les résultats de l'exécution de l'algorithme écrit par l'apprenant, soit « tests » pour les tests effectués par l'apprenant sur son algorithme),
- l'identifiant utilisateur
- l'identifiant de l'algorithme (car un apprenant écrit généralement plusieurs algos s'il se trompe)
- le code de l'algorithme, l'action, ou le résultat de l'exécution
- d'autres informations selon si la trace et de type « sav », « exec » ou « tests »

Dans le cas 2 (exercice), les traces contiennent : les algorithmes successifs écrits pas les apprenants pour cet exercice, les résultats des tests effectués sur ces algorithmes, et un tableau récapitulatif (mis à jour directement).

Andes

Andes, tuteur pour l'apprentissage de la physique. A noter que les traces sont aussi enregistrées dans Datashop.

Format des traces pour une session avec un utilisateur : timestamp, élément tracé, valeurs éléments tracés. Les éléments tracés peuvent être de différents types : action de l'élève, paramètre de l'exercice, calcul du tuteur (identifiés par le préfixe « DDE »…)

SQL-Tutor

Tuteur pour l'apprentissage du langage SQL. Les traces se composent de trois parties :

1. partie initiale (pre-process) avec : le niveau d'aide, les options du feedback, la base de données de travail en cours et le numéro de l'exercice proposé à l'apprenant
2. partie réponse (attempt) avec les réponses successives de l'apprenant (c'est-à-dire les requête SQL qu'il tape dans l'interface)
3. partie finale (post-process) avec la liste des contraintes satisfaites et non satisfaites (contraintes = misconceptions ou erreurs)

TPelec

TPelec est un peu différent puisqu'il s'agit d'un micro-monde pour l'électricité (travail sur des circuits électriques). Les traces enregistrent donc les états successifs du micro-monde, pour une session de travail d'un utilisateur. L'état d'un circuit est décrit en trois parties :

1. la liste des composants présents (lampe, générateur, résistance, condensateur…)
2. leur position à l'écran
3. les actions de l'apprenant sur le circuit (généralement une modification comme allumer et éteindre un interrupteur)

Reading tutor

Les données sont organisées par années dans une base de données Postgres-SQL (donc une année d'utilisation du tuteur dans plusieurs écoles = une base de données). Dans chaque base, on trouve trois types de tables :

1. les données sur les expérimentations (tables user, school, exercices, audio-recording, feedback given…)
2. des données bas niveau enregistrées par le tuteur (sur les sessions, les clics souris…)
3. les données des apprenants proprement dites, c'est-à-dire l'enregistrement des traces d'activité pour les différentes catégories d'exercices pratiqués.

Nous ne donnons ici la description (en anglais) que d'une seule table de la base, celle qui enregistre les mots lus par les apprenants :

Champ	Description
Machine_Name	= Reading_tutor.machine_name
Utterance_Start_Time	= Utterance.start_Time
Utterance_sms	= Utterance.sms
Target_Word_Number	The position of the target word in the sentence
Start_Time	The start time when the reads the word
Sms	The millisecond of the start time
End_Time	The end time when the student reads the word
Ems	The millisecond of the end time
User_ID	= Student.User_ID
Utterance_Count	= Utterance.utterance_count

Sentence_Encounter_Start_Time	= Sentence_Encounter.start_time
Sentence_Encounter_sms	= Sentence_Encounter.sms
Target_Word	The word the student should read
Aligned_Word	The word the speech recognizer thinks what the student says
Silence	The silent time periods from the end of the previous word of the target word to the beginning of the target word.
Best_Case_Latency	The silent time periods from the end of the previous word of the target word to the beginning of the target word, if the previous word is read. Otherwise “und” (undefined).
Worst_Case_Latency	The silent time periods from the end of the previous word of the target word to the beginning of the target word, if the previous word is read correctly. Otherwise “und” (undefined).
Before_Context	The number of aligned words before the target word.
Center_Context	“1” – if the aligned word matches the target word “.5” – if the aligned word truncates the target word “0” – else.
After_Context	The number of aligned words after the target word.
BestYet_Before_Context	Max(Before_Context), if multiple utterance heard for one sentence.
BestYet_Center_Context	Max(Center_Context), if multiple utterance heard for one sentence.
BestYet_After_Context	Max(Center_Context), if multiple utterance heard for one sentence.
BestYet_Silence	Min(Silence), if multiple utterance heard for one sentence.
BestYet_Latency	Min(Best_Case_Latency), if multiple utterance heard for one sentence.
WorstYet_Latency	Min(Worst_Case_Latency), if multiple utterance heard for one sentence.
Total_Before_Context	
Total_After_Context	
Total_Match	The total number aligned words matched with their target words in one utterance.
Total_Substituations	The total number aligned words that are substitutions of their target words in one utterance.

III. Modèles de diagnostic

Nous donnons dans cette annexe les ontologies des divers modèles de diagnostic considérées dans le mémoire, ainsi que celle de la formalisation F_{MD} permettant de décrire ces modèles de diagnostics.

Formalisation F_{MD}


```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2002/07/owl#"

xml:base="http://www.semanticweb.org/ontologies/2013/8/Ontology1378170810155.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"

ontologyIRI="http://www.semanticweb.org/ontologies/2013/8/Ontology1378170810155.o
wl">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Declaration>
      <Class IRI="#Comportement"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Knowledge"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Observable"/>
   </Declaration>
   <Declaration>
      <ObjectProperty IRI="#KtoK"/>
   </Declaration>
   <Declaration>
      <ObjectProperty IRI="#OtoK"/>
   </Declaration>
   <Declaration>
      <ObjectProperty IRI="#OtoO"/>
   </Declaration>
   <SubClassOf>
      <Class IRI="#Comportement"/>
      <Class IRI="#Observable"/>
   </SubClassOf>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#KtoK"/>
      <ObjectSomeValuesFrom>
```

```
        <ObjectProperty IRI="#KtoK"/>
        <Class IRI="#Knowledge"/>
      </ObjectSomeValuesFrom>
    </ObjectPropertyDomain>
    <ObjectPropertyDomain>
      <ObjectProperty IRI="#OtoK"/>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#OtoK"/>
        <Class IRI="#Observable"/>
      </ObjectSomeValuesFrom>
    </ObjectPropertyDomain>
    <ObjectPropertyDomain>
      <ObjectProperty IRI="#OtoO"/>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#OtoO"/>
        <Class IRI="#Observable"/>
      </ObjectSomeValuesFrom>
    </ObjectPropertyDomain>
    <ObjectPropertyRange>
      <ObjectProperty IRI="#KtoK"/>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#KtoK"/>
        <Class IRI="#Knowledge"/>
      </ObjectSomeValuesFrom>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
      <ObjectProperty IRI="#OtoK"/>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#OtoK"/>
        <Class IRI="#Knowledge"/>
      </ObjectSomeValuesFrom>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
      <ObjectProperty IRI="#OtoO"/>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#OtoO"/>
        <Class IRI="#Observable"/>
      </ObjectSomeValuesFrom>
    </ObjectPropertyRange>
</Ontology>

<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```


Knowledge tracing

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1372328202073
"http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#"

xml:base="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:Ontology1372328202073="http://www.semanticweb.org/ontologies/2013/5/Ontolog
y1372328202073.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl"
/>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#KtoK
-->

   <owl:ObjectProperty rdf:about="&Ontology1372328202073;KtoK">
      <rdfs:range rdf:resource="&Ontology1372328202073;Knowledge"/>
      <rdfs:domain rdf:resource="&Ontology1372328202073;Knowledge"/>
   </owl:ObjectProperty>
```

```
   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#OtoK
-->

   <owl:ObjectProperty rdf:about="&Ontology1372328202073;OtoK">
      <rdfs:range rdf:resource="&Ontology1372328202073;Knowledge"/>
      <rdfs:domain rdf:resource="&Ontology1372328202073;Observable"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#OtoO -->

   <owl:ObjectProperty rdf:about="&Ontology1372328202073;OtoO">
      <rdfs:range rdf:resource="&Ontology1372328202073;Observable"/>
      <rdfs:domain rdf:resource="&Ontology1372328202073;Observable"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#ProblemtoStep -->

   <owl:ObjectProperty rdf:about="&Ontology1372328202073;ProblemtoStep">
      <rdfs:subPropertyOf rdf:resource="&Ontology1372328202073;OtoO"/>
      <rdfs:domain rdf:resource="&Ontology1372328202073;Problem"/>
      <rdfs:range rdf:resource="&Ontology1372328202073;Step"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#SteptoKC -->

   <owl:ObjectProperty rdf:about="&Ontology1372328202073;SteptoKC">
      <rdfs:range rdf:resource="&Ontology1372328202073;KnowledgeComponent"/>
      <rdfs:subPropertyOf rdf:resource="&Ontology1372328202073;OtoK"/>
      <rdfs:domain rdf:resource="&Ontology1372328202073;Step"/>
      <rdfs:domain>
         <owl:Restriction>
            <owl:onProperty rdf:resource="&Ontology1372328202073;SteptoKC"/>
            <owl:onClass rdf:resource="&Ontology1372328202073;Step"/>
            <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
         </owl:Restriction>
      </rdfs:domain>
      <rdfs:range>
         <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="&Ontology1372328202073;SteptoKC"/>
            <owl:onClass rdf:resource="&Ontology1372328202073;KnowledgeComponent"/>
            <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
         </owl:Restriction>
      </rdfs:range>
   </owl:ObjectProperty>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Classes
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Behavioral
Diagnostic -->

   <owl:Class rdf:about="&Ontology1372328202073;BehavioralDiagnostic">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202073;Observable"/>
   </owl:Class>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Knowledge
-->

   <owl:Class rdf:about="&Ontology1372328202073;Knowledge">
      <rdfs:subClassOf rdf:resource="&owl;Thing"/>
   </owl:Class>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Knowledge
Component -->

   <owl:Class rdf:about="&Ontology1372328202073;KnowledgeComponent">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202073;Knowledge"/>
   </owl:Class>
```

```
   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Observable
-->

   <owl:Class rdf:about="&Ontology1372328202073;Observable"/>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Outcome --
>

   <owl:Class rdf:about="&Ontology1372328202073;Outcome">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202073;BehavioralDiagnostic"/>
   </owl:Class>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Problem --
>

   <owl:Class rdf:about="&Ontology1372328202073;Problem">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202073;Observable"/>
   </owl:Class>



   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202073.owl#Step
-->

   <owl:Class rdf:about="&Ontology1372328202073;Step">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202073;Observable"/>
   </owl:Class>
</rdf:RDF>


<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```


Constraint-based


```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
```

```
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1372328202075
"http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#"

xml:base="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:Ontology1372328202075="http://www.semanticweb.org/ontologies/2013/5/Ontolog
y1372328202075.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl"
/>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#KtoK
-->

   <owl:ObjectProperty rdf:about="&Ontology1372328202075;KtoK">
      <rdfs:range rdf:resource="&Ontology1372328202075;Knowledge"/>
      <rdfs:domain rdf:resource="&Ontology1372328202075;Knowledge"/>
   </owl:ObjectProperty>



   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#OtoK
-->

   <owl:ObjectProperty rdf:about="&Ontology1372328202075;OtoK">
```

```
    <rdfs:range rdf:resource="&Ontology1372328202075;Knowledge"/>
    <rdfs:domain rdf:resource="&Ontology1372328202075;Observable"/>
  </owl:ObjectProperty>



  <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#OtoO -->

  <owl:ObjectProperty rdf:about="&Ontology1372328202075;OtoO">
    <rdfs:range rdf:resource="&Ontology1372328202075;Observable"/>
    <rdfs:domain rdf:resource="&Ontology1372328202075;Observable"/>
  </owl:ObjectProperty>



  <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#CrtoCs -->

  <owl:ObjectProperty rdf:about="&Ontology1372328202075;CrtoCs">
    <rdfs:range rdf:resource="&Ontology1372328202075;Cs"/>
    <rdfs:subPropertyOf rdf:resource="&Ontology1372328202075;OtoK"/>
    <rdfs:domain rdf:resource="&Ontology1372328202075;Cr"/>
    <rdfs:domain>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&Ontology1372328202075;CrtoCs"/>
        <owl:onClass rdf:resource="&Ontology1372328202075;Cr"/>
        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
      </owl:Restriction>
    </rdfs:domain>
                <rdfs:range>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&Ontology1372328202075;CrtoCs"/>
        <owl:onClass rdf:resource="&Ontology1372328202075;Cs"/>
        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
      </owl:Restriction>
    </rdfs:range>
  </owl:ObjectProperty>



  <!--
  ///////////////////////////////////////////////////////////////////////////////////////
  //
  // Classes
  //
  ///////////////////////////////////////////////////////////////////////////////////////
   -->
```

```
    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#Behavioral
Diagnostic -->

    <owl:Class rdf:about="&Ontology1372328202075;BehavioralDiagnostic">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202075;Observable"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#Knowledge
-->

    <owl:Class rdf:about="&Ontology1372328202075;Knowledge">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#Cs --
>

    <owl:Class rdf:about="&Ontology1372328202075;Cs">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202075;Knowledge"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#Observable
-->

    <owl:Class rdf:about="&Ontology1372328202075;Observable"/>



    <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202075.owl#Cr --
>

    <owl:Class rdf:about="&Ontology1372328202075;Cr">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202075;Observable"/>
    </owl:Class>


</rdf:RDF>
```

```
<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```

Control-based

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1372328202074
"http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#"

xml:base="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:Ontology1372328202074="http://www.semanticweb.org/ontologies/2013/5/Ontolog
y1372328202074.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl"
/>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#KtoK
-->
```

```
   <owl:ObjectProperty rdf:about="&Ontology1372328202074;KtoK">
      <rdfs:range rdf:resource="&Ontology1372328202074;Knowledge"/>
      <rdfs:domain rdf:resource="&Ontology1372328202074;Knowledge"/>
   </owl:ObjectProperty>



   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#OtoK
-->

   <owl:ObjectProperty rdf:about="&Ontology1372328202074;OtoK">
      <rdfs:range rdf:resource="&Ontology1372328202074;Knowledge"/>
      <rdfs:domain rdf:resource="&Ontology1372328202074;Observable"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#OtoO -->

   <owl:ObjectProperty rdf:about="&Ontology1372328202074;OtoO">
      <rdfs:range rdf:resource="&Ontology1372328202074;Observable"/>
      <rdfs:domain rdf:resource="&Ontology1372328202074;Observable"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Problemto
Operator -->

   <owl:ObjectProperty rdf:about="&Ontology1372328202074;ProblemtoOperator">
      <rdfs:subPropertyOf rdf:resource="&Ontology1372328202074;OtoO"/>
      <rdfs:domain rdf:resource="&Ontology1372328202074;Problem"/>
      <rdfs:range rdf:resource="&Ontology1372328202074;Operator"/>
   </owl:ObjectProperty>


         <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Problemto
Control -->

   <owl:ObjectProperty rdf:about="&Ontology1372328202074;ProblemtoControl">
      <rdfs:subPropertyOf rdf:resource="&Ontology1372328202074;OtoK"/>
      <rdfs:domain rdf:resource="&Ontology1372328202074;Problem"/>
      <rdfs:range rdf:resource="&Ontology1372328202074;Control"/>
   </owl:ObjectProperty>
```

```
        <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Registerto
Operator -->

  <owl:ObjectProperty rdf:about="&Ontology1372328202074;RegistertoOperator">
    <rdfs:subPropertyOf rdf:resource="&Ontology1372328202074;OtoK"/>
    <rdfs:domain rdf:resource="&Ontology1372328202074;Register"/>
    <rdfs:range rdf:resource="&Ontology1372328202074;Operator"/>
  </owl:ObjectProperty>


  <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Operatorto
Control -->

  <owl:ObjectProperty rdf:about="&Ontology1372328202074;OperatortoControl">
    <rdfs:range rdf:resource="&Ontology1372328202074;Control"/>
    <rdfs:subPropertyOf rdf:resource="&Ontology1372328202074;OtoK"/>
    <rdfs:domain rdf:resource="&Ontology1372328202074;Operator"/>
    <rdfs:domain>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&Ontology1372328202074;OperatortoControl"/>
        <owl:onClass rdf:resource="&Ontology1372328202074;Operator"/>
        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
      </owl:Restriction>
    </rdfs:domain>
  </owl:ObjectProperty>



  <!--
  ///////////////////////////////////////////////////////////////////////////////////////
  //
  // Classes
  //
  ///////////////////////////////////////////////////////////////////////////////////////
   -->




  <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Behavioral
Diagnostic -->

  <owl:Class rdf:about="&Ontology1372328202074;BehavioralDiagnostic">
    <rdfs:subClassOf rdf:resource="&Ontology1372328202074;Observable"/>
  </owl:Class>
```

```
    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Knowledge
-->

    <owl:Class rdf:about="&Ontology1372328202074;Knowledge">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Control -->

    <owl:Class rdf:about="&Ontology1372328202074;Control">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202074;Knowledge"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Observable
-->

    <owl:Class rdf:about="&Ontology1372328202074;Observable"/>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#SituationVa
riable -->

    <owl:Class rdf:about="&Ontology1372328202074;SituationVariable">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202074;BehavioralDiagnostic"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Problem --
>

    <owl:Class rdf:about="&Ontology1372328202074;Problem">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202074;Observable"/>
    </owl:Class>
```

```
   <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Step
-->

   <owl:Class rdf:about="&Ontology1372328202074;Operator">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202074;Observable"/>
   </owl:Class>


           <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202074.owl#Register -->

   <owl:Class rdf:about="&Ontology1372328202074;Register">
      <rdfs:subClassOf rdf:resource="&Ontology1372328202074;Observable"/>
   </owl:Class>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```

Simple praxeology

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1372328202076
"http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#"

xml:base="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:Ontology1372328202076="http://www.semanticweb.org/ontologies/2013/5/Ontolog
y1372328202076.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl"
/>
```

```
<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Object Properties
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#KtoK
-->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;KtoK">
    <rdfs:range rdf:resource="&Ontology1372328202076;Knowledge"/>
    <rdfs:domain rdf:resource="&Ontology1372328202076;Knowledge"/>
</owl:ObjectProperty>



<!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#OtoK
-->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;OtoK">
    <rdfs:range rdf:resource="&Ontology1372328202076;Knowledge"/>
    <rdfs:domain rdf:resource="&Ontology1372328202076;Observable"/>
</owl:ObjectProperty>



<!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#OtoO -->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;OtoO">
    <rdfs:range rdf:resource="&Ontology1372328202076;Observable"/>
    <rdfs:domain rdf:resource="&Ontology1372328202076;Observable"/>
</owl:ObjectProperty>



<!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#TasktoTech
-->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;TasktoTech">
    <rdfs:subPropertyOf rdf:resource="&Ontology1372328202076;OtoK"/>
    <rdfs:domain rdf:resource="&Ontology1372328202076;Task"/>
    <rdfs:range rdf:resource="&Ontology1372328202076;Technique"/>
</owl:ObjectProperty>
```

```
<!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#SubTasktoS
ubTech -->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;SubTasktoSubTech">
  <rdfs:subPropertyOf rdf:resource="&Ontology1372328202076;OtoK"/>
  <rdfs:domain rdf:resource="&Ontology1372328202076;SubTask"/>
  <rdfs:range rdf:resource="&Ontology1372328202076;SubTechnique"/>
</owl:ObjectProperty>




<!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#TasktoSubT
ask -->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;TasktoSubTask">
  <rdfs:range rdf:resource="&Ontology1372328202076;SubTask"/>
  <rdfs:subPropertyOf rdf:resource="&Ontology1372328202076;OtoO"/>
  <rdfs:domain rdf:resource="&Ontology1372328202076;Task"/>
  <rdfs:domain>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&Ontology1372328202076;TasktoSubTask"/>
      <owl:onClass rdf:resource="&Ontology1372328202076;Task"/>
      <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:domain>
</owl:ObjectProperty>


<!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#TechtoSub
Tech -->

<owl:ObjectProperty rdf:about="&Ontology1372328202076;TechtoSubTech">
  <rdfs:range rdf:resource="&Ontology1372328202076;SubTechnique"/>
  <rdfs:subPropertyOf rdf:resource="&Ontology1372328202076;KtoK"/>
  <rdfs:domain rdf:resource="&Ontology1372328202076;Technique"/>
  <rdfs:domain>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&Ontology1372328202076;TechtoSubTech"/>
      <owl:onClass rdf:resource="&Ontology1372328202076;Technique"/>
      <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:domain>
```

```
    </owl:ObjectProperty>


    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Behavioral
Diagnostic -->

    <owl:Class rdf:about="&Ontology1372328202076;BehavioralDiagnostic">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202076;Observable"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Knowledge
-->

    <owl:Class rdf:about="&Ontology1372328202076;Knowledge">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Technique
-->

    <owl:Class rdf:about="&Ontology1372328202076;Technique">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202076;Knowledge"/>
    </owl:Class>


   <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#SubTechniq
ue -->

    <owl:Class rdf:about="&Ontology1372328202076;SubTechnique">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202076;Knowledge"/>
    </owl:Class>
```

```
    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Observable
-->

    <owl:Class rdf:about="&Ontology1372328202076;Observable"/>



    <!--
http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Outcome --
>

    <owl:Class rdf:about="&Ontology1372328202076;Outcome">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202076;BehavioralDiagnostic"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Task
-->

    <owl:Class rdf:about="&Ontology1372328202076;Task">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202076;Observable"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2013/5/Ontology1372328202076.owl#Step
-->

    <owl:Class rdf:about="&Ontology1372328202076;SubTask">
        <rdfs:subClassOf rdf:resource="&Ontology1372328202076;SubTask"/>
    </owl:Class>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```

IV. Ontologies des traces

Définition de l'ontologie dérivée de CREAM-C

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
```

```
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1333695474043
"http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#"

xml:base="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:Ontology1333695474043="http://www.semanticweb.org/ontologies/2012/3/Ontolog
y1333695474043.owl#">
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl"
/>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Annotation properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Datatypes
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
```

```
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Describe --
>

   <owl:ObjectProperty rdf:about="&Ontology1333695474043;Describe">
      <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
      <rdfs:domain rdf:resource="&Ontology1333695474043;Description"/>
      <owl:inverseOf rdf:resource="&Ontology1333695474043;IsDescribedBy"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#IsDescribed
By -->

   <owl:ObjectProperty rdf:about="&Ontology1333695474043;IsDescribedBy">
      <rdfs:domain rdf:resource="&Ontology1333695474043;Capacity"/>
      <rdfs:range rdf:resource="&Ontology1333695474043;Description"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#PrealableO
f -->

   <owl:ObjectProperty rdf:about="&Ontology1333695474043;PrealableOf">
      <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
   </owl:ObjectProperty>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TransitionF
rom -->

   <owl:ObjectProperty rdf:about="&Ontology1333695474043;TransitionFrom">
      <rdfs:domain rdf:resource="&Ontology1333695474043;Capacity"/>
      <rdfs:range rdf:resource="&Ontology1333695474043;Transition"/>
   </owl:ObjectProperty>
```

```
<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TransitionTo -->

<owl:ObjectProperty rdf:about="&Ontology1333695474043;TransitionTo">
    <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
    <rdfs:domain rdf:resource="&Ontology1333695474043;Transition"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#hasPrealable -->

<owl:ObjectProperty rdf:about="&Ontology1333695474043;hasPrealable">
    <owl:inverseOf rdf:resource="&Ontology1333695474043;PrealableOf"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>



<!-- http://www.w3.org/2002/07/owl#topObjectProperty -->

<rdf:Description rdf:about="&owl;topObjectProperty">
    <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
    <rdfs:domain rdf:resource="&Ontology1333695474043;Transition"/>
</rdf:Description>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Attribut -->

<owl:Class rdf:about="&Ontology1333695474043;Attribut">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

```
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Capacity --
>

    <owl:Class rdf:about="&Ontology1333695474043;Capacity">
        <rdfs:comment>A capacity is a knowledge or skill. </rdfs:comment>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#CognitiveH
ability -->

    <owl:Class rdf:about="&Ontology1333695474043;CognitiveHability">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#CognitiveSt
rategy -->

    <owl:Class rdf:about="&Ontology1333695474043;CognitiveStrategy">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Concept -->

    <owl:Class rdf:about="&Ontology1333695474043;Concept">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#ConcreteC
oncept -->

    <owl:Class rdf:about="&Ontology1333695474043;ConcreteConcept">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Concept"/>
    </owl:Class>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#DefinedCo
ncept -->

    <owl:Class rdf:about="&Ontology1333695474043;DefinedConcept">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Concept"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Description
-->

    <owl:Class rdf:about="&Ontology1333695474043;Description">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#DomainStr
ategy -->

    <owl:Class rdf:about="&Ontology1333695474043;DomainStrategy">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveStrategy"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#HighLevelR
ule -->

    <owl:Class rdf:about="&Ontology1333695474043;HighLevelRule">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Informatio
n -->

    <owl:Class rdf:about="&Ontology1333695474043;Information">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Law -
->

    <owl:Class rdf:about="&Ontology1333695474043;Law">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#LearningStr
ategy -->

    <owl:Class rdf:about="&Ontology1333695474043;LearningStrategy">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveStrategy"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Proposition
-->

    <owl:Class rdf:about="&Ontology1333695474043;Proposition">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Proposition
Composite -->

    <owl:Class rdf:about="&Ontology1333695474043;PropositionComposite">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Rule
-->

    <owl:Class rdf:about="&Ontology1333695474043;Rule">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>
```

```
   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Transition -
->

   <owl:Class rdf:about="&Ontology1333695474043;Transition"/>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TypeCapaci
ty -->

   <owl:Class rdf:about="&Ontology1333695474043;TypeCapacity">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Capacity"/>
   </owl:Class>
</rdf:RDF>
```


Ontologie détaillée pour TELEOS


```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1333695474043
"http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#" >
   <!ENTITY Pointd
"http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Pointd&#
39;" >
]>


<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#"

xml:base="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:Pointd="&Ontology1333695474043;Pointd&#39;"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:Ontology1333695474043="http://www.semanticweb.org/ontologies/2012/3/Ontolog
y1333695474043.owl#">
```

```
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl"
/>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Annotation properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Datatypes
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Describe --
>

   <owl:ObjectProperty rdf:about="&Ontology1333695474043;Describe">
      <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
      <rdfs:domain rdf:resource="&Ontology1333695474043;Description"/>
      <owl:inverseOf rdf:resource="&Ontology1333695474043;IsDescribedBy"/>
   </owl:ObjectProperty>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#IsDescribed
By -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;IsDescribedBy">
        <rdfs:domain rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:range rdf:resource="&Ontology1333695474043;Description"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#PrealableO
f -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;PrealableOf">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TransitionF
rom -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;TransitionFrom">
        <rdfs:domain rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:range rdf:resource="&Ontology1333695474043;Transition"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TransitionT
o -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;TransitionTo">
        <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:domain rdf:resource="&Ontology1333695474043;Transition"/>
        <owl:inverseOf rdf:resource="&Ontology1333695474043;TransitionFrom"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#hasPrealabl
e -->
```

```
    <owl:ObjectProperty rdf:about="&Ontology1333695474043;hasPrealable">
        <owl:inverseOf rdf:resource="&Ontology1333695474043;PrealableOf"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://www.w3.org/2002/07/owl#topObjectProperty -->

    <rdf:Description rdf:about="&owl;topObjectProperty">
        <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:domain rdf:resource="&Ontology1333695474043;Transition"/>
    </rdf:Description>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Attribut -->

    <owl:Class rdf:about="&Ontology1333695474043;Attribut">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Capacity --
>

    <owl:Class rdf:about="&Ontology1333695474043;Capacity">
        <rdfs:comment>A capacity is a knowledge or skill. </rdfs:comment>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#CognitiveH
ability -->
```

```
    <owl:Class rdf:about="&Ontology1333695474043;CognitiveHability">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#CognitiveSt
rategy -->

    <owl:Class rdf:about="&Ontology1333695474043;CognitiveStrategy">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Concept -->

    <owl:Class rdf:about="&Ontology1333695474043;Concept">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#ConcreteC
oncept -->

    <owl:Class rdf:about="&Ontology1333695474043;ConcreteConcept">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Concept"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#DefinedCo
ncept -->

    <owl:Class rdf:about="&Ontology1333695474043;DefinedConcept">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Concept"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Description
-->

    <owl:Class rdf:about="&Ontology1333695474043;Description">
```

```
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
  </owl:Class>



  <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#DomainStr
ategy -->

  <owl:Class rdf:about="&Ontology1333695474043;DomainStrategy">
    <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveStrategy"/>
  </owl:Class>



  <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#HighLevelR
ule -->

  <owl:Class rdf:about="&Ontology1333695474043;HighLevelRule">
    <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
  </owl:Class>



  <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Informatio
n -->

  <owl:Class rdf:about="&Ontology1333695474043;Information">
    <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
  </owl:Class>



  <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Law -
->

  <owl:Class rdf:about="&Ontology1333695474043;Law">
    <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
  </owl:Class>



  <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#LearningStr
ategy -->

  <owl:Class rdf:about="&Ontology1333695474043;LearningStrategy">
    <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveStrategy"/>
```

```
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Proposition
-->

    <owl:Class rdf:about="&Ontology1333695474043;Proposition">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Proposition
Composite -->

    <owl:Class rdf:about="&Ontology1333695474043;PropositionComposite">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Rule
-->

    <owl:Class rdf:about="&Ontology1333695474043;Rule">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Transition -
->

    <owl:Class rdf:about="&Ontology1333695474043;Transition"/>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TypeCapaci
ty -->

    <owl:Class rdf:about="&Ontology1333695474043;TypeCapacity">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Capacity"/>
    </owl:Class>
```

```
<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#ControleImpact -->

<Capacity rdf:about="&Ontology1333695474043;ControleImpact">

</Capacity>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Controle_Face -->

<Transition rdf:about="&Ontology1333695474043;Controle_Face">

</Transition>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Controle_Profil -->

<Transition rdf:about="&Ontology1333695474043;Controle_Profil">

</Transition>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#ControlerRadio -->

<Capacity rdf:about="&Ontology1333695474043;ControlerRadio">

    <hasPrealable rdf:resource="&Ontology1333695474043;Definir_Face"/>
    <hasPrealable rdf:resource="&Ontology1333695474043;Definir_Profil"/>
```

```
      <hasPrealable rdf:resource="&Ontology1333695474043;PositionnerFace"/>
      <hasPrealable rdf:resource="&Ontology1333695474043;PositionnerProfil"/>
   </Capacity>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Definir_Fac
e -->

   <Transition rdf:about="&Ontology1333695474043;Definir_Face">

   </Transition>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Definir_Pro
fil -->

   <Transition rdf:about="&Ontology1333695474043;Definir_Profil">

   </Transition>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#EntreeCorp
s -->

   <Capacity rdf:about="&Ontology1333695474043;EntreeCorps">

      <hasPrealable rdf:resource="&Ontology1333695474043;Controle_Face"/>
      <hasPrealable rdf:resource="&Ontology1333695474043;Controle_Profil"/>
      <hasPrealable rdf:resource="&Ontology1333695474043;EntreeOsseuse"/>
   </Capacity>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#EntreeOsse
use -->

   <Capacity rdf:about="&Ontology1333695474043;EntreeOsseuse">

      <hasPrealable rdf:resource="&Ontology1333695474043;Controle_Face"/>
      <hasPrealable rdf:resource="&Ontology1333695474043;Controle_Profil"/>
      <hasPrealable rdf:resource="&Ontology1333695474043;Impact"/>
   </Capacity>
```

```
<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#IdentifierPl
an -->

<Capacity rdf:about="&Ontology1333695474043;IdentifierPlan">

    <TransitionFrom rdf:resource="&Ontology1333695474043;Controle_Face"/>
    <TransitionFrom rdf:resource="&Ontology1333695474043;Controle_Profil"/>
</Capacity>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Impact -->

<Capacity rdf:about="&Ontology1333695474043;Impact">

    <hasPrealable rdf:resource="&Ontology1333695474043;Controle_Face"/>
    <hasPrealable rdf:resource="&Ontology1333695474043;Controle_Profil"/>
    <hasPrealable rdf:resource="&Ontology1333695474043;Orienter"/>
</Capacity>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Orientation
entree -->

<Capacity rdf:about="&Ontology1333695474043;Orientationentree">

    <hasPrealable rdf:resource="&Ontology1333695474043;Orienter"/>
    <hasPrealable rdf:resource="&Ontology1333695474043;Positionner"/>
</Capacity>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Orienter --
>

<Capacity rdf:about="&Ontology1333695474043;Orienter">

</Capacity>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Pointd&#3
9;entree -->

    <Capacity rdf:about="&Ontology1333695474043;Pointd&#39;entree">

    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Positionner
-->

    <Capacity rdf:about="&Ontology1333695474043;Positionner">

    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Positionner
Face -->

    <Capacity rdf:about="&Ontology1333695474043;PositionnerFace">

        <hasPrealable rdf:resource="&Ontology1333695474043;IdentifierPlan"/>
        <TransitionFrom rdf:resource="&Ontology1333695474043;PositionnerFace"/>
        <TransitionFrom rdf:resource="&Ontology1333695474043;PositionnerProfil"/>
    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Positionner
Profil -->

    <Capacity rdf:about="&Ontology1333695474043;PositionnerProfil">

        <hasPrealable rdf:resource="&Ontology1333695474043;IdentifierPlan"/>
        <TransitionFrom rdf:resource="&Ontology1333695474043;PositionnerFace"/>
        <TransitionFrom rdf:resource="&Ontology1333695474043;PositionnerProfil"/>
    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#RepereDroi
t -->
```

```
    <Capacity rdf:about="&Ontology1333695474043;RepereDroit">

        <TransitionFrom rdf:resource="&Ontology1333695474043;Valider_Repere_D"/>
    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Reperegau
che -->

    <Capacity rdf:about="&Ontology1333695474043;Reperegauche">

        <TransitionFrom rdf:resource="&Ontology1333695474043;Valider_Repere_G"/>
    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#RepererVer
tebre -->

    <Capacity rdf:about="&Ontology1333695474043;RepererVertebre">

        <hasPrealable rdf:resource="&Ontology1333695474043;Reperer_Vertebre"/>
    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Reperer_V
ertebre -->

    <Capacity rdf:about="&Ontology1333695474043;Reperer_Vertebre">

    </Capacity>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Valider_Re
pere_D -->

    <Capacity rdf:about="&Ontology1333695474043;Valider_Repere_D">

    </Capacity>
```

```
   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Valider_Re
pere_G -->

   <Capacity rdf:about="&Ontology1333695474043;Valider_Repere_G">

   </Capacity>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Valider_Re
pere_T -->

   <Capacity rdf:about="&Ontology1333695474043;Valider_Repere_T">

   </Capacity>



   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Valider_Tra
jectoire -->

   <Capacity rdf:about="&Ontology1333695474043;Valider_Trajectoire">

   </Capacity>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```


Ontologie détaillée pour le Reading tutor


```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Ontology1333695474043
"http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#"

xml:base="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl"
```

```
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:Ontology1333695474043="http://www.semanticweb.org/ontologies/2012/3/Ontolog
y1333695474043.owl#">
   <owl:Ontology
rdf:about="http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl"
/>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Annotation properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Datatypes
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->




   <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Describe --
>
```

```
    <owl:ObjectProperty rdf:about="&Ontology1333695474043;Describe">
        <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:domain rdf:resource="&Ontology1333695474043;Description"/>
        <owl:inverseOf rdf:resource="&Ontology1333695474043;IsDescribedBy"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#IsDescribed
By -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;IsDescribedBy">
        <rdfs:domain rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:range rdf:resource="&Ontology1333695474043;Description"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#PrealableO
f -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;PrealableOf">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TransitionF
rom -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;TransitionFrom">
        <rdfs:domain rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:range rdf:resource="&Ontology1333695474043;Transition"/>
    </owl:ObjectProperty>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#TransitionT
o -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;TransitionTo">
        <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:domain rdf:resource="&Ontology1333695474043;Transition"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#hasPrealabl
e -->

    <owl:ObjectProperty rdf:about="&Ontology1333695474043;hasPrealable">
        <owl:inverseOf rdf:resource="&Ontology1333695474043;PrealableOf"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://www.w3.org/2002/07/owl#topObjectProperty -->

    <rdf:Description rdf:about="&owl;topObjectProperty">
        <rdfs:range rdf:resource="&Ontology1333695474043;Capacity"/>
        <rdfs:domain rdf:resource="&Ontology1333695474043;Transition"/>
    </rdf:Description>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Attribut -->

    <owl:Class rdf:about="&Ontology1333695474043;Attribut">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Capacity --
>

    <owl:Class rdf:about="&Ontology1333695474043;Capacity">
        <rdfs:comment>A capacity is a knowledge or skill. </rdfs:comment>
    </owl:Class>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#CognitiveH
ability -->

    <owl:Class rdf:about="&Ontology1333695474043;CognitiveHability">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#CognitiveSt
rategy -->

    <owl:Class rdf:about="&Ontology1333695474043;CognitiveStrategy">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Concept -->

    <owl:Class rdf:about="&Ontology1333695474043;Concept">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#ConcreteC
oncept -->

    <owl:Class rdf:about="&Ontology1333695474043;ConcreteConcept">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Concept"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#DefinedCo
ncept -->

    <owl:Class rdf:about="&Ontology1333695474043;DefinedConcept">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Concept"/>
    </owl:Class>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Description
-->

    <owl:Class rdf:about="&Ontology1333695474043;Description">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#DomainStr
ategy -->

    <owl:Class rdf:about="&Ontology1333695474043;DomainStrategy">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveStrategy"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#HighLevelR
ule -->

    <owl:Class rdf:about="&Ontology1333695474043;HighLevelRule">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Informatio
n -->

    <owl:Class rdf:about="&Ontology1333695474043;Information">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;TypeCapacity"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Law -
->

    <owl:Class rdf:about="&Ontology1333695474043;Law">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>
```

```
    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#LearningStr
ategy -->

    <owl:Class rdf:about="&Ontology1333695474043;LearningStrategy">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveStrategy"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Proposition
-->

    <owl:Class rdf:about="&Ontology1333695474043;Proposition">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Proposition
Composite -->

    <owl:Class rdf:about="&Ontology1333695474043;PropositionComposite">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Information"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Rule
-->

    <owl:Class rdf:about="&Ontology1333695474043;Rule">
        <rdfs:subClassOf rdf:resource="&Ontology1333695474043;CognitiveHability"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Transition -
->

    <owl:Class rdf:about="&Ontology1333695474043;Transition"/>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#GtoP
-->
```

```
   <owl:Class rdf:about="&Ontology1333695474043;GtoP">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Capacity"/>
   </owl:Class>

         <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Word -->

   <owl:Class rdf:about="&Ontology1333695474043;Word">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Story"/>
   </owl:Class>

         <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Story -->

   <owl:Class rdf:about="&Ontology1333695474043;Story">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Transition"/>
   </owl:Class>

         <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Regular -->

   <owl:Class rdf:about="&Ontology1333695474043;Regular">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Transition"/>
   </owl:Class>

         <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Common --
>

   <owl:Class rdf:about="&Ontology1333695474043;Common">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Transition"/>
   </owl:Class>

         <!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1333695474043.owl#Graphem --
>

   <owl:Class rdf:about="&Ontology1333695474043;Graphem">
      <rdfs:subClassOf rdf:resource="&Ontology1333695474043;Word"/>
   </owl:Class>



   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Individuals
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->
```

```
<GtoP rdf:about="&Ontology1333695474043;a:AE">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;a:EY-->SilentEe">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;a:EY-->SilentEvowel">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ai:EY">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ar:AA-R">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;au:AO">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;aw:AO">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ay:EY">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;b:B">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;c:K">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ch:CH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;d:D">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;e:EH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ea:EH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ea:IY">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ed:D">
</GtoP>
```

```
<GtoP rdf:about="&Ontology1333695474043;ee:IY">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;en:AH-N">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;er:ER">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;est:EH-S-T">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;f:F">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;g:G">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;h:HH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;i:AY-->SilentEe">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;i:AY-->SilentEvowel">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;i:IH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ing:IH-NG">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ir:ER">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;j:JH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;k:K">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;kn:N">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;l:L">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;m:M">
</GtoP>
```

```
<GtoP rdf:about="&Ontology1333695474043;n:N">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;o:AA">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;o:OW-->SilentEe">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;o:OW-->SilentEvowel">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;oa:OW">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;oi:OY">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;oo:UH">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;oo:UW">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;or:AO-R">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;ou:AW">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;oy:OY">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;p:P">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;qu:K-W">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;r:R">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;s:S">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;s:Z">
</GtoP>

<GtoP rdf:about="&Ontology1333695474043;sh:SH">
```

```
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;t:T">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;th:DH">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;th:TH">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;u:AH">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;u:UW-->SilentEe">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;u:UW-->SilentEvowel">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;ur:ER">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;v:V">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;w:W">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;wh:W">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;wr:R">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;x:K-S">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;y:Y">
        </GtoP>

        <GtoP rdf:about="&Ontology1333695474043;z:Z">
        </GtoP>


</rdf:RDF>

<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```

V. Stratégie d'aide pour le Reading tutor

Stratégie d'aide apprise pour le Reading tutor durant l'expéprimentation 2 (stratégie obtenue avec la technique de diagnostic la plus précise).

WordClass = c12456CD AND
Story_Level = A AND
HelpType = OnsetRime: 1 (0.83)

WordClass = c1256BD AND
HelpType = OnsetRime AND
Prediction > 0.5046: 1 (0.87)

Reading_Level = G AND
HelpType = Recue AND
Prediction > 0.514: 1 (0.86)

WordClass = c12456CD AND
Story_Level = G AND
HelpType = SayWord: 1 (0.95)

WordClass = c1245679BD AND
HelpType = SayWord AND
Prediction > 0.6375: 1 (0.82)

WordClass = c1245679BD AND
HelpType = StartsLike: 1 (0.83)

WordClass = c16C AND
HelpType = Syllabify: 1 (0.8)

WordClass = c146BD AND
HelpType = SayWord: 1 (0.8)

WordClass = c1246D AND
HelpType = OnsetRime AND
Prediction > 0.5274: 1 (0.81)

WordClass = c4 AND
HelpType = Recue: 1 (0.8)

WordClass = c6CD AND
HelpType = SayWord: 1 (0.84)

WordClass = c6B AND
HelpType = SayWord: 1 (0.84)

WordClass = c26D AND
WordPosition <= 1 AND
HelpType = WordInContext: 1 (16.0)

WordClass = c1469D AND
Reading_Level = A AND
HelpType = SoundOut AND
Prediction > 0.5051: 1 (0.79)

WordClass = c1469D AND
Reading_Level = B AND
HelpType = SoundOut: 1 (0.82)

WordClass = c1469 AND
WordFrequency > 1 AND
Story_Level = B AND
HelpType = Autophonics: 1 (0.85)

WordClass = c1469 AND
WordFrequency > 1 AND
Story_Level = B AND
HelpType = SayWord: 1 (0.92)

WordClass = c4569B AND
HelpType = SoundOut AND
Prediction > 0.6451: 1 (0.8)

WordClass = c1469D AND
Reading_Level = C AND
HelpType = Autophonics: 1 (0.81)

WordClass = c1469D AND
Reading_Level = C AND
HelpType = SayWord: 1 (0.81)

WordClass = c1469D AND
Reading_Level = D AND
HelpType = SoundOut AND
Prediction > 0.6005: 1 (0.86)

WordClass = c1469 AND
WordFrequency > 1 AND
Story_Level = C AND
HelpType = SayWord: 1 (0.86)

WordClass = c1246BCD AND
HelpType = SayWord: 1 (0.78)

WordClass = c1246BCD AND
HelpType = StartsLike: 1 (0.79)

WordClass = c124569 AND
Reading_Level = B AND

HelpType = SayWord: 1 (0.88)

WordClass = c146D AND
HelpType = WordInContext AND
Prediction > 0.5650: 1 (0.78)

WordClass = c12569 AND
HelpType = SayWord: 1 (0.78)

WordClass = c16D AND
HelpType = WordInContext: 1 (0.78)

WordClass = c246D AND
HelpType = SayWord: 1 (0.79)

WordClass = c12456CD AND
Story_Level = C AND
WordPosition <= 3 AND
HelpType = OnsetRime: 1 (0.83)

WordClass = c12456CD AND
Story_Level = C AND
WordPosition <= 3 AND
HelpType = Syllabify AND
Prediction > 0.5028: 1 (0.84)

WordClass = c124569D AND
Story_Level = C AND
HelpType = OnsetRime AND
Prediction > 0.5651: 1 (0.82)

WordClass = c1469CD AND
Reading_Level = A AND
HelpType = SoundOut AND
Prediction > 0.5651: 1 (0.79)

WordClass = c1469 AND
HelpType = SayWord: 1 (0.77)

WordClass = c12469BD AND
HelpType = SayWord: 1 (0.78)

WordClass = c1469CD AND
Reading_Level = C AND
HelpType = Autophonics: 1 (0.9)

WordClass = c2456BD AND
HelpType = OnsetRime: 1 (0.77)

WordClass = c256D AND

HelpType = SayWord AND
Prediction > 0.5276: 1 (0.78)

WordClass = c169 AND
HelpType = SayWord: 1 (0.77)

WordClass = c146CD AND
WordPosition <= 2 AND
Reading_Level = B AND
HelpType = Autophonics: 1 (0.89)

WordClass = c4569 AND
WordPrev > 2 AND
Story_Level = A AND
HelpType = RhymesWith: 1 (0.76)

WordClass = c4569 AND
WordPrev > 2 AND
Story_Level = C AND
HelpType = SoundOut AND
Prediction > 0.5413: 1 (0.79)

WordClass = c124569 AND
Reading_Level = C AND
HelpType = SoundOut AND
Prediction > 0.5713: 1 (0.8)

WordClass = c1469D AND
Story_Level = A AND
HelpType = SoundOut: 1 (0.83)

WordClass = c12469B AND
HelpType = SayWord: 1 (0.77)

WordClass = c1469BD AND
Story_Level = C AND
HelpType = StartsLike: 1 (0.81)

WordClass = c124569BD AND
Reading_Level = D AND
HelpType = SayWord AND
Prediction > 0.5901: 1 (0.84)

WordClass = c12456D AND
HelpType = OnsetRime AND
Prediction > 0.5114: 1 (0.77)

WordClass = c12456CD AND
HelpType = OnsetRime: 1 (0.77)

WordClass = c12456CD AND
HelpType = Syllabify: 1 (0.78)

WordClass = c4569 AND
WordPrev <= 2 AND
HelpType = RhymesWith : 1 (0.92)

WordClass = c46D AND
HelpType = SayWord: 1 (0.77)

WordClass = c124569 AND
Reading_Level = A AND
Story_Level = C AND
HelpType = SoundOut: 1 (0.79)

WordClass = c124569D AND
Story_Level = B AND
HelpType = SayWord AND
Prediction > 0.5231: 1 (0.76)

WordClass = c124569D AND
Story_Level = D AND
HelpType = RhymesWith: 1 (0.84)

Reading_Level = E AND
HelpType = RhymesWith AND
Prediction > 0.547: 1 (0.82)

Reading_Level = F AND
HelpType = WordInContext AND
Prediction > 0.5712: 1 (0.85)

Reading_Level = F AND
HelpType = Recue AND
Prediction > 0.5446: 1 (0.8)

WordClass = c1246BD AND
HelpType = SayWord: 1 (0.77)

WordClass = c1256C AND
HelpType = RhymesWith: 1 (0.77)

WordClass = c56 AND
HelpType = SayWord AND
Prediction > 0.5295: 1 (0.79)

WordClass = c69D AND
HelpType = SoundOut: 1 (0.77)

WordClass = c1269 AND

HelpType = SoundOut: 1 (0.89)

WordClass = c6C AND
HelpType = Syllabify: 1 (0.79)

WordClass = c126BD AND
HelpType = WordInContext: 1 (0.79)

WordClass = c24569D AND
HelpType = WordInContext: 1 (0.79)

WordClass = c126D AND
HelpType = Autophonics AND
Prediction > 0.5691: 1 (0.83)

WordClass = c16BCD AND
HelpType = SayWord: 1 (0.79)

WordClass = c1245679B AND
Reading_Level = C AND
HelpType = OnsetRime: 1 (0.93)

WordClass = c16CD AND
HelpType = Syllabify: 1 (0.76)

WordClass = c1469BD AND
HelpType = StartsLike AND
Prediction > 0.551: 1 (0.8)

WordClass = c1469D AND
HelpType = Autophonics: 1 (0.85)

WordClass = c4569 AND
Story_Level = B AND
HelpType = SoundOut: 1 (0.78)

WordClass = c124569BD AND
Story_Level = C AND
Reading_Level = C AND
HelpType = SoundOut: 1 (0.84)

WordClass = c124569D AND
WordLength <= 1 AND
HelpType = OnsetRime: 1 (0.77)

WordClass = c124569BD AND
Story_Level = C AND
Reading_Level = C AND
HelpType = SayWord AND
Prediction > 0.5514: 1 (0.8)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = SoundOut: 1 (0.79)

WordClass = c124569BD AND
Story_Level = C AND
Reading_Level = C AND
HelpType = Autophonics: 1 (0.74)

WordClass = c124569BD AND
Story_Level = C AND
HelpType = StartsLike: 1 (0.83)

WordClass = c1469BD AND
HelpType = Recue AND
Prediction > 0.5981: 1 (0.95)

WordClass = c124569D AND
HelpType = Recue: 1 (0.8)

WordClass = c124569BD AND
Reading_Level = C AND
HelpType = RhymesWith AND
Prediction > 0.641: 1 (0.75)

WordClass = c124569BD AND
Reading_Level = C AND
HelpType = OnsetRime AND
Prediction <= 0.5647: 1 (0.84)

WordClass = c146CD AND
HelpType = Autophonics: 1 (0.85)

WordClass = c1469B AND
Story_Level = B AND
HelpType = SayWord: 1 (0.71)

WordClass = c1469BD AND
HelpType = SayWord AND
Reading_Level = A AND
WordPosition <= 3: 1 (0.87)

WordClass = c124569D AND
HelpType = Autophonics AND
Story_Level = A AND
WordFrequency <= 3: 0 (0.7)

WordClass = c124569D AND
HelpType = SoundOut: 1 (0.74)

WordClass = c124569D AND
HelpType = RhymesWith: 1 (0.74)

WordClass = c124569D AND
HelpType = Autophonics: 1 (0.84)

WordClass = c1469B AND
HelpType = Recue: 1 (0.82)

WordClass = c46 AND
Story_Level = K AND
HelpType = SayWord AND
Prediction > 0.6824: 1 (0.75)

WordClass = c124569BD AND
Story_Level = K AND
HelpType = SayWord AND
Prediction <= 0.585308: 1 (0.79)

WordClass = c469B AND
Reading_Level = A AND
Story_Level = K AND
HelpType = SoundOut: 1 (0.76)

WordClass = cD AND
HelpType = WordInContext : 1 (0.73)

WordClass = c4569 AND
Story_Level = B AND
HelpType = RhymesWith: 1 (0.81)

WordClass = c124569BD AND
Story_Level = B AND
HelpType = RhymesWith AND
Prediction > 0.5109: 1 (0.74)

WordClass = c124569BD AND
Story_Level = B AND
HelpType = SayWord: 1 (0.7)

WordClass = c124569BD AND
Story_Level = B AND
HelpType = SoundOut: 1 (0.73)

WordClass = c124569BD AND
Story_Level = B AND
HelpType = StartsLike: 1 (0.76)

WordClass = c124569BD AND

Story_Level = B AND
HelpType = WordInContext: 1 (0.76)

WordClass = c12456BD AND
HelpType = SayWord: 1 (0.68)

WordClass = c12456BD AND
HelpType = WordInContext AND
Prediction > 0.5487: 0 (0.69)

WordClass = c12456BD AND
HelpType = RhymesWith: 1 (16.0)

WordClass = c469B AND
Reading_Level = A AND
HelpType = StartsLike: 1 (0.76)

WordClass = c1469B AND
Reading_Level = B AND
Story_Level = C AND
HelpType = SoundOut: 1 (0.92)

WordClass = c124569BD AND
Story_Level = B AND
HelpType = Autophonics: 1 (0.8)

WordClass = c124569B AND
Reading_Level = A AND
HelpType = OnsetRime AND
Prediction > 0.5156: 1 (0.81)

WordClass = c6BD AND
HelpType = StartsLike AND
Prediction > 0.594: 1 (0.72)

WordClass = c26D AND
HelpType = SayWord: 1 (0.8)

WordClass = c469B AND
Reading_Level = A AND
Story_Level = B AND
WordFrequency > 1 AND
WordPosition <= 1 AND
HelpType = SayWord: 1 (0.85)

WordClass = c16 AND
HelpType = SayWord: 1 (0.77)

WordClass = c124569BD AND
Story_Level = B AND

HelpType = Recue AND
Prediction > 0.5574: 1 (0.84)

WordClass = c256 AND
HelpType = OnsetRime: 1 (0.71)

WordClass = c124569B AND
Reading_Level = A AND
HelpType = SoundOut AND
Prediction <= 0.561385: 1 (0.9)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = SayWord AND
Prediction > 0.6178: 1 (0.73)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = WordInContext: 1 (0.74)

WordClass = c1469B AND
Story_Level = B AND
HelpType = StartsLike: 1 (0.84)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = OnsetRime: 1 (0.78)

WordClass = c124569D AND
Reading_Level = A AND AND
HelpType = WordInContext
Prediction <= 0.563949: 0 (0.71)

WordClass = c124569D AND
HelpType = WordInContext: 1 (0.81)

WordClass = c46 AND
Reading_Level = C AND
HelpType = SayWord AND
Prediction > 0.6073: 1 (0.8)

WordClass = c1469B AND
Story_Level = A AND
WordPrev > 2 AND
HelpType = SayWord: 1 (0.8)

WordClass = c6D AND
WordFrequency > 3 AND
HelpType = WordInContext AND
Prediction > 0.5713: 1 (0.74)

WordClass = c124569B AND
Reading_Level = A AND
HelpType = Autophonics: 1 (0.77)

WordClass = cB AND
HelpType = StartsLike: 1 (0.9)

WordClass = c69B AND
HelpType = SoundOut: 1 (0.71)

WordClass = c56B AND
HelpType = SayWord AND
Prediction > 0.5493: 0 (0.79)

WordClass = c29 AND
HelpType = SoundOut: 1 (0.92)

WordClass = c2569 AND
HelpType = SoundOut AND
Prediction > 0.5688: 1 (0.77)

WordClass = c169B AND
WordLength <= 2 AND
HelpType = Autophonics: 0 (0.83)

WordClass = c124569B AND
Reading_Level = K AND
HelpType = Autophonics: 1 (0.7)

WordClass = c124569B AND
Story_Level = K AND
HelpType = SayWord AND
WordPosition > 1: 1 (0.7)

WordClass = c124569BD AND
Story_Level = E AND
HelpType = SayWord: 1 (0.74)

WordClass = c124569B AND
Story_Level = K AND
WordPosition > 1 AND
HelpType = OnsetRime AND
Prediction > 0.5561: 1 (0.71)

WordClass = c469B AND
HelpType = SayWord AND
Prediction <= 0.514095: 1 (0.83)

WordClass = c124569B AND

HelpType = StartsLike: 1 (0.74)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = StartsLike AND
WordFrequency <= 1: 1 (0.76)

WordClass = c1469B AND
Reading_Level = K AND
HelpType = Autophonics AND
Prediction <= 0.546062: 1 (0.73)

WordClass = c169B AND
HelpType = Autophonics: 1 (0.83)

WordClass = c146CD AND
Reading_Level = A AND
Story_Level = B AND
HelpType = SayWord: 0 (0.85)

WordClass = c1469B AND
Story_Level = C AND
HelpType = SayWord AND
Prediction > 0.5836: 0 (0.72)

WordClass = c46 AND
Story_Level = A AND
WordPrev > 2 AND
Reading_Level = B AND
HelpType = SayWord AND
Prediction > 0.562711: 1 (0.81)

WordClass = c124569B AND
HelpType = RhymesWith: 1 (0.71)

WordClass = c1469B AND
Reading_Level = B AND
HelpType = SoundOut: 0 (0.76)

WordClass = c1469B AND
Story_Level = A AND
WordFrequency <= 3 AND
Reading_Level = A AND
HelpType = SoundOut: 1 (0.7)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = Recue: 1 (0.71)

WordClass = c1469B AND

Story_Level = A AND
HelpType = StartsLike: 1 (0.82)

WordClass = c124569B AND
Story_Level = K AND
HelpType = SoundOut AND
Prediction > 0.5643: 1 (0.72)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = StartsLike: 0 (0.71)

WordClass = c124569BD AND
Story_Level = A AND
HelpType = RhymesWith: 1 (0.69)

WordClass = c124569BD AND
HelpType = Autophonics AND
Prediction <= 0.5453: 1 (0.71)

WordClass = c46 AND
Reading_Level = A AND
HelpType = SayWord AND
Story_Level = D AND
WordFrequency > 2: 0 (0.71)

WordClass = c46 AND
WordPrev > 1 AND
Story_Level = A AND
HelpType = SayWord: 1 (0.73)

WordClass = c46 AND
Reading_Level = A AND
HelpType = SayWord AND
Prediction > 0.5837: 1 (0.67)

WordClass = c469B AND
Story_Level = C AND
Reading_Level = A AND
HelpType = SayWord: 1 (0.74)

WordClass = c469B AND
Story_Level = C AND
HelpType = SoundOut: 1 (0.72)

WordClass = c1469B AND
Story_Level = K AND
HelpType = StartsLike AND
WordFrequency <= 1: 0 (0.92)

WordClass = c1469B AND
Story_Level = K AND
HelpType = SayWord AND
Prediction > 0.5393: 1 (0.73)

WordClass = c1469B AND
Story_Level = K AND
HelpType = SoundOut AND
Reading_Level = A: 0 (0.7)

WordClass = c1469B AND
Story_Level = K AND
HelpType = SoundOut: 1 (0.71)

WordClass = c5 AND
WordPosition > 2 AND
HelpType = RhymesWith: 0 (0.75)

WordClass = c1 AND
WordLength <= 1 AND
HelpType = Autophonics : 1 (0.8)

WordClass = c1 AND
WordLength <= 1 AND
HelpType = PlaybackWord : 1 (0.85)

VI. Ressources pour la praxéologie

Ontologie institutionnelle

```
<praxeoInst>
	<Technique name="Tfact.simple">
		<SubTechnique name="TFact.simple_Formule" status="required">
			<v1>Simple</v1>
		</SubTechnique>
		<SubTechnique name="Tcalc" status="optionnal">
		</SubTechnique>
		<SubTechnique name="Tred" status="optionnal">
		</SubTechnique>
	</Technique>

	<Technique name="Tfact.comb">
		<SubTechnique name="TFact.simple" status="required">
			<v1>Simple</v1>
			<v3>Partiel</v3>
		</SubTechnique>
		<SubTechnique name="TFact.simple" status="required">
			<v1>Simple</v1>
```

```
                    <v3>Complet</v3>
                </SubTechnique>
                <SubTechnique name="Tred" status="optionnal">
                </SubTechnique>
        </Technique>
</praxeoInst>
```

Contexte des exercices (fourni par l'expert)

Exercise	Technique	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10	Prax
1	Tfact	simple	type 1	complet	+/-	polynôme	2/2	1/2	polynôme	polynôme	PI	Tfact.simple
2	Tfact	comb	type 1	complet	+/+	polynôme	2/2	PartielVisible	numérique	numérique	PI	Tfact.comb
3	Tfact	simple	type 2	complet	+/-	polynôme	2/2	2/2	monôme	polynôme	PI	Tfact.simple
4	Tfact	simple	type 2	complet	+/+	monôme	NonVisible	NonVisible	monôme	numérique	PI	Tfact.simple
5	Tfact	comb	type 3	complet	+/+	polynôme	1/2	PartielVisible	polynôme	numérique	PI	Tfact.comb
6	Tfact	simple	type 3	complet	+/-	polynôme	1/2	1/1	polynôme	numérique	PI	Tfact.simple

Légende :

- V1 : simple ou comb, correspond en fait au nombre de termes dans la somme de départ 2 ou 3
- V2 : type 1 qui correspond à (AB+AC) ; type 2 qui correspond à (A^2B+AC) ; type 3 qui correspond à (AB+A)
- V3 : partiel ou complet, quand il s'agit d'un énoncé on considère toujours complet
- V4 : +/+ ; –/+ ; +/- et -/-, correspond aux signes des termes
- V5 : numérique ou monôme ou polynôme, qui correspond à la forme du facteur commun
- V6 : 1/1 ou 1/2 ou 2/2 ou 1/3 ou 2/3 ou 3/3 ou Non Visible ou Partiel Visible, correspond à la place dans le 1er terme. Partiel Visible n'est possible que si V1 = comb
- V7 : idem V6, correspond à la place dans le 2ième terme. Si [V2 = type 3] alors ([V6 = 1/1] ou [V7 = 1/1])
- V8 : numérique ou monôme ou polynôme, correspond à la forme du cofacteur dans le 1er terme
- V9 : numérique ou monôme ou polynôme, correspond à la forme du cofacteur dans le 2ième terme

- V10 : PI et OE, question Posée par l'Institution PI ou Opportunité Crée par l'élève OE

Exercices de factorisation

1. $(5x - 2)(3x + 4) - (3x + 4)(-x + 3)$
2. $2(x - 1) + 3x - 3$
3. $2x(3x + 7) - (x + 1)(3x + 7)^2$
4. $9x^2 + 6x$
5. $(2x - 7)(x + 4) + 2x - 7$
6. $(x + 1)(x + 9) - (x + 1)$