Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

وزارة التعليم العالي و البحث العلمي

UNIVERSITE BADJI MOKHTAR - ANNABA
BADJI MOKHTAR-ANNABA UNIVERSITY

جـامعـة باجـي مـختار – عنابة

Faculté des Sciences de l'Ingéniorat
Département d'Informatique

Année : 2012/2013

Thèse

Présentée en vue de l'obtention du diplôme de

Doctorat 3ième cycle

Un Système d'Évaluation du Savoir-faire
Application à l'algorithmique

Filière : Informatique

Spécialité : Ingénierie des Connaissances

par

Anis BEY

Devant le jury :

Laskri Med Tayeb	Professeur à l'Université Badji Mokhtar-Annaba	(Président)
Boufaida Mahmoud	Professeur à l'Université de Constantine	(Examinateur)
Kazar Okba	Professeur à l'Université de Biskra	(Examinateur)
Kimour Med Tahar	Professeur à l'Université Badji Mokhtar-Annaba	(Examinateur)
Bensebaa Tahar	Maitre de Conférence à l'Université Badji Mokhtar	(Directeur)
Denis Bouhineau	Maitre de Conférence à l'Université UJF-Grenoble	(Invité)

À mon adorable neveu Iyed

Remerciements

J'adresse mes premiers remerciements très particulièrement à Tahar Bensebaa, le directeur de ce travail, merci pour sa disponibilité indéfectible, ses conseils, ses remarques judicieuses qui ont enrichi mon travail et surtout d'avoir cru en moi et de m'avoir fait confiance.

Je tiens aussi à remercier Boufaida Mahmoud, professeur à l'Université de Constantin, Kazar Okba, professeur à l'Université de Biskra et Kimour Med Tahar, professeur à l'Université Badji Mokhtar-Annaba, de m'avoir fait l'honneur d'accepter d'examiner cette thèse.

Je remercie Laskri Med Tayeb, professeur à l'Université Badji Mokhtar-Annaba, de m'avoir fait l'honneur de présider le jury de ma soutenance.

Je remercie Denis Bouhineau, maitre de conférences à l'Université Joseph Fourier-Grenoble, pour avoir accepté de participer au jury.

Sans oublier de remercier les membres de l'équipe EIAH, Laboratoire de Recherche en Informatique de Annaba (LRI) , avec lesquelles j'ai pu avoir de nombreux échanges, et évoluer mon esprit de débat scientifique.

Mes remerciement s'adressent aussi aux membres de l'équipe Metah du LIG-France, en particulier, ceux que qui m'ont toujours accueilli et soutenu comme l'un des leurs : Denis Bouhineau, Cyrille Desmoulain, Pierre Tchounikine, Vanda Luengo, Claire Wajeman, Viviane Gueraud.

Je remercie aussi les gens qui ont participé à l'expérimentation : Hakim, Ahmed, Mohamed, Ismail, Anissa, Amina qui ont accepté de donner quelques heures de leur temps libre pour me permettre de réaliser les différentes expérimentations de cette thèse. Sans oublier de remercier le chef du département du MIAS pour sa disponibilité et sa gentillesse. Mes remerciements vont aussi à Meftouh Karima pour son aide pour réaliser cette expérimentation.

Je ne pourrais pas oublier mes amis, je les remercie tous, pour leur écoute, leur disponibilité et surtout leur gentillesse ainsi que les bons moments passés ensemble. Je leur souhaite une bonne continuation.

Évidemment, mes remerciements les plus chaleureux vont à ma source d'inspiration, ma très chère mère, mon très cher père ainsi que mon frère Raouf, qui ont toujours été là pour me soutenir et m'encourager.

Enfin, je remercie toutes les personnes (nombreuses) que je n'ai pas citées et qui, à un moment ou à un autre, m'ont donné l'envie et la force de continuer.

Anis Bey, le 13 Septembre 2013.

Table des matières

Table des matières **iv**

Liste des figures **vi**

Liste des tableaux **vii**

Résumé **2**

1 Introduction **3**
- 1.1 Contexte général du travail . . . 4
- 1.2 Motivation . . . 5
- 1.3 Problématique et approche . . . 7
- 1.4 Méthodologie de recherche . . . 9
- 1.5 Organisation du manuscrit . . . 10

2 L'évaluation dans la littérature **12**
- 2.1 La science de l'évaluation en quelques mots . . . 13
  - 2.1.1 Docimologie . . . 13
  - 2.1.2 Edumétrie . . . 14
- 2.2 Qu'est ce qu'évaluer dans le processus d'apprentissage . . . 15
  - 2.2.1 Définition de l'évaluation . . . 15
  - 2.2.2 La différence entre contrôle et évaluation . . . 15
- 2.3 A quoi sert l'évaluation et pour qui ? . . . 16
- 2.4 Les types d'évaluation . . . 16
  - 2.4.1 Par rapport à l'intention d'évaluation . . . 17
  - 2.4.2 Par rapport au moment d'évaluation . . . 20
  - 2.4.3 Par rapport à la forme d'évaluation . . . 20
- 2.5 Synthèse des types d'évaluation . . . 22
- 2.6 Outils pour l'évaluation . . . 22
- 2.7 Relation entre évaluation et processus d'apprentissage . . . 25
- 2.8 Objectivité et subjectivité dans l'évaluation . . . 26
  - 2.8.1 Les sources de subjectivité . . . 27
  - 2.8.2 La subjectivité : une nature de l'évaluation . . . 28

Conclusion . . . 29

3 L'évaluation dans les EIAH **30**
- 3.1 L'évaluation assistée par ordinateur . . . 31
- 3.2 Pourquoi évaluer dans les EIAH ? . . . 32
- 3.3 L'évaluation face à la standardisation . . . 34

3.4 Les qualités d'un outil d'évaluation 35
Conclusion 35

4 Le domaine d'application : L'algorithmique **37**
4.1 Introduction 38
4.2 Etude épistémologique de l'algorithmique 38
4.2.1 L'algorithmique et la conception d'algorithme 39
4.2.2 Les différents aspects de l'algorithmique 40
4.2.3 Réflexion algorithmique et compétences en résolution de problèmes 42
4.3 Les débutants en algorithmique 42
4.3.1 Difficultés de l'algorithmique 42
4.3.2 Comportement des débutants en algorithmique 44
4.4 Systèmes dédiés à l'algorithmique 45
4.4.1 Systèmes d'apprentissage de l'algorithmique 45
4.4.2 Systèmes d'évaluation intégrant une vision sommative . . . 50
4.5 Synthèse 52
4.5.1 Les techniques d'évaluation généralement utilisées en algorithmique/programmation 54
4.5.2 Synopsis des systèmes 56
Conclusion 58

5 Une approche d'évaluation des compétences algorithmique **60**
5.1 Modèle global de la méthode proposée 61
5.2 Stratégie de conception d'une solution 62
5.2.1 Définition d'un Plan de solution 65
5.2.2 La bibliothèque d'opérations de base 66
5.3 Une base de solutions correctes et de solutions erronées 67
5.4 L'évaluation à l'aide d'un processus de reconnaissance 69
5.5 Boucle d'alimentation de la base de solutions 69
Conclusion 70

6 Une méthode d'évaluation basée sur une mesure de similarité proposée **71**
6.1 Conception d'un Plan de Solution 72
6.1.1 Anatomie des Plans de Solution de la base 72
6.1.2 Feedback associé à chaque plan de solution 73
6.2 Scénario de l'évaluation 74
6.3 Proposition d'une mesure de similarité 75
6.3.1 Caractéristiques d'un plan de solution 75
6.3.2 Mesure de similarité proposée 76
6.4 Recherche d'un Plan de Solution référent 78
6.4.1 Cas d'un plan de solution référent trouvé 78
6.4.2 Cas d'un plan de solution référent non trouvé 79
6.5 Algorithme du processus d'évaluation 80

7 Algo+, Un système d'évaluation des solutions algorithmiques **81**

7.1 Démarche et objectifs pédagogiques d'Algo+ 82
7.2 Architecture d'Algo+ 82
7.2.1 Architecture logicielle 82
7.2.2 Architecture fonctionnelle 83
7.3 Expérimentation d'Algo+ 84
7.3.1 Résultats 85
7.3.2 Discussion et Analyse 90

Conclusion générale **92**
7.4 Conclusion 92
7.5 Perspectives 94
7.5.1 Perspectives expérimentales 95
7.5.2 Perspectives de recherche 95

Bibliographie **98**

A Annexes **106**
A.1 Quelques interfaces du système Algo+ 107
A.2 Les opérations de base 109
A.3 Résultats expérimentaux 110
A.4 Valeurs critiques du test des rangs pour échantillons appariés de Wilcoxon 110

Notations **112**

Travaux réalisés **113**

Liste des figures

1.1 A floating figure 5
1.2 Un élève débutant en pleine métacognition. 6
1.3 Démarche réelle et démarche poursuivie par les apprenants dans le développement d'un programme. 9
1.4 Organisation du manuscrit. 10

2.1 Le rôle de l'évaluation selon les différents acteurs. 17
2.2 Synthèse des types d'évaluation selon trois axex : Moment, Modalité, Intention. 23
2.3 Les outils appropriés à chaque niveau de Bloom. 24
2.4 Relation entre l'évaluation et un processus d'apprentissage. . 25

3.1 Critères d'efficacité des évaluations. 35

4.1 Interface de GPCeditor. 47
4.2 Interface de Solveit. 48
4.3 Interface de SICAS. 49

4.4 Interface principale de PROPL. 50
4.5 Interface de l'algorithme en pseudo code. 51
4.6 Interface principale d'AlgoBox. 52
4.7 Interface TRAKLA 2. 53
4.8 Interface BOSS. 53
4.9 Interface Web-CAT. 54
4.10 Exemple 1 d'utilisation des QCM en programmation. 56
4.11 Exemple 2 d'utilisation des QCM en programmation. 56

5.1 Architecture générale de la méthode. 61
5.2 Décomposition d'un problème algorithmique. 63
5.3 Décomposition d'un problème de l'abstraction au concret. . . 63
5.4 La décomposition d'un problème : vue par l'apprenant et par l'enseignant. 64
5.5 Rôle pédagogique de l'erreur. 67

6.1 Anatomie d'un plan de solution dans la base. 72
6.2 Carte descriptive d'un problème. 73
6.3 Scénario de l'évaluation. 74

7.1 Architecture logiciel d'Algo+. 83
7.2 Architecture fonctionnelle d'Algo+. 84
7.3 Éditeur de plan de solution. 85
7.4 Éditeur des paramètres d'une opération de base. 86
7.5 A floating figure 86
7.6 A floating figure 87
7.7 A floating figure 87
7.8 A floating figure 88
7.9 L'évaluation : le chef d'orchestre d'un processus d'apprentissage. 92
7.10 L'évolution de la méthode proposée. 96

A.1 Ajout d'un plan de solution 107
A.2 L'historique 108
A.3 Une solution reconnue 108
A.4 Comparaison des notes : Expert VS Algo+ VS Algo+ 'arrondies' 110

Liste des tableaux

1.1 Résultats du module algorithmique. 7
1.2 Taux de réussite en Algorithmique 2. 7
1.3 Taux de réussite d'Algorithmique 1 et des autres modules. . . 8

2.1 La différence entre Contrôle et Évaluation. 15
2.2 Le but de l'évaluation selon les acteurs. 16

2.3 La différence entre l'évaluation de l'apprentissage et l'évaluation pour l'apprentissage. 26

4.1 Les différentes compétences en résolution de problèmes et les activités d'apprentissage liées en algorithmique. 43
4.2 Synopsis des systèmes dédiés à l'algorithmique et à la programmation. 57

5.1 Ensemble d'opérations de base de la structure Pile. 67

6.1 Les différentes situations : Concordance, Discordance, Absence. 76
6.2 Exemple de mesure de similarité entre deux plans de solution : Référent et proposé. 77

7.1 Taux de reconnaissance des solution par Algo+. 85
7.2 Paramètres des distributions des notes attribuées par Algo+ et l'Expert. 87
7.3 Notation du système Algo+ par rapport à l'expert. 89

A.1 Table des opérations de base 109
A.2 Différences de notes entre l'Expert et Algo+ 110
A.3 Paramètres des distributions des notes : Expert VS Algo+ VS Algo+ 'arrondi' 110
A.4 Valeurs critiques du test des rangs pour échantillons appariés de Wilcoxon 111

"I practice until I have my life in my fingers"

Pianists' expression

Résumé

Le sujet abordé par cette thèse traite un des plus importants axes de recherche dans les environnements informatiques pour l'apprentissage Humain (EIAH) : l'évaluation des apprenants. Dans le processus d'apprentissage, l'évaluation joue un rôle fondamental car c'est le seul moyen d'assurer une bonne transmission des connaissances à l'apprenant. La plupart des plates-formes développées pour l'enseignement s'intéressent peu à l'évaluation, elles intègrent généralement des générateurs simples de questionnaire et évitent ainsi la difficulté de l'évaluation et la complexité de sa mise en œuvre dans un EIAH. De plus, la nature des connaissances à évaluer (connaissances déclaratives, connaissances procédurales) accroit exponentiellement cette difficulté.

Notre domaine d'application, l'algorithmique, a été depuis sa naissance une discipline difficile à enseigner pour l'enseignant et complexe à assimiler pour l'apprenant. Ceci est du aux caractéristiques intrinsèques de la matière et à la manière avec laquelle les enseignants (la majorité) appréhendent ses fondements.

Ce travail de recherche s'inscrit au confluent de ces deux axes et par conséquent hérite des difficultés des deux. Il vise plus précisément à élaborer une approche d'évaluation des compétences en algorithmique dans un EIAH. Outre le fait d'apporter une solution au délicat problème de l'évaluation automatique des productions en algorithmique, cette approche est, en plus, formative. Puisant son inspiration dans les principes de base de la matière, elle offre une efficacité certaine.

La démarche que nous avons adoptée consiste à modéliser une solution algorithmique en fonction d'opérations de bases et d'opérations élémentaires qui permettent à l'apprenant de dessiner sa propre démarche. Indépendante de tout langage de programmation, cette démarche favorise l'autonomie de l'apprenant dans la conception de sa solution. Elle repose sur l'utilisation d'une base évolutive de solutions types. Ainsi, toute production de l'apprenant, pour un problème donné, est soit automatiquement évaluée si elle est reconnue parmi les solutions de la base, soit évaluée par un expert humain dans le cas contraire. Dans ce dernier cas, elle viendra enrichir la base si elle est jugée pédagogiquement intéressante.

Mots clés : Évaluation dans un EIAH, compétences algorithmiques, évaluation du savoir-faire, évaluation sommative, formative, diagnostic

Introduction 1

Sommaire

1.1 Contexte général du travail 4
1.2 Motivation 5
1.3 Problématique et approche 7
1.4 Méthodologie de recherche 9
1.5 Organisation du manuscrit 10

1.1 Contexte général du travail

Ce travail de recherche concerne les EIAH (Environnements Informatiques pour l'Apprentissage Humain) d'une manière générale et en particulier, l'évaluation des compétences en résolution des problèmes en algorithmique chez les apprenants débutants.

Plusieurs systèmes d'évaluation ont été développés pour évaluer les connaissances ou les compétences des apprenants dans un EIAH. Dans les Systèmes Tuteurs Intelligents (STI), les résultats de l'évaluation devront être précis afin d'assurer un apprentissage adaptatif.

Parmi les outils les plus utilisés pour l'évaluation automatique, les QCM (Questions à Choix Multiples). D'après Labat (2002), les QCM sont des systèmes simples, qui ne nécessitent pas beaucoup de connaissances à la création, cependant ils demandent un investissement important lors de la création de chaque exercice. Les QCM restent l'outil le plus utilisable dans l'évaluation automatisée vu la facilité de leur mise en œuvre. Mais on ne peut pas tout évaluer en utilisant les QCM. Cela dépend du niveau cognitif des connaissances à évaluer.

En pédagogie, on distingue habituellement trois types de connaissances : déclaratives, procédurales et conditionnelles :

Des connaissances déclaratives : Les connaissances déclaratives correspondent essentiellement à des connaissances théoriques, aux connaissances qui, à une certaine période, furent reconnues comme des savoirs. Il s'agit, selon Gagné (Gagné, 1985), de la connaissance de faits, de règles, de lois, de principes. Par exemple, la connaissance de chacune des capitales des provinces canadiennes constitue une connaissance déclarative ...

Des connaissances procédurales : Les connaissances procédurales correspondent au comment de l'action, aux étapes pour réaliser une action et à la procédure permettant la réalisation d'une action.

Les connaissances conditionnelles : Les connaissances conditionnelles se réfèrent aux conditions de l'action. Elles concernent le quand et le pourquoi. A quel moment et dans quel contexte est-il approprié d'utiliser telle ou telle stratégie, telle ou telle démarche, d'engager telle ou telle action ? Pourquoi est-ce adéquat d'employer cette stratégie, cette démarche, de réaliser cette action ?

Face à l'évaluation des compétences, la réponse de divers spécialistes s'est souvent limitée à la mise en place d'un certain nombre de grilles d'analyse tant au niveau professionnel qu'éducatif. Il suffit dans ce cas de répondre via un logiciel à un certain nombre d'items en utilisant un questionnaire à choix multiple. Le résultat final nous permettant de déterminer si l'apprenant a réussi à acquérir une ou plusieurs compétences. Cette manière de pratiquer nous semble surprenante et surtout inadaptée car les compétences que nous désirons évaluer sont d'ordre procédural et conditionnel et elles devraient être jugées en laissant libre cours à la créativité des apprenants dans la construction de leurs solutions.

En effet l'évaluation de l'acquisition d'une connaissance ne se fonde plus

sur un constat que l'élève a pu reproduire la solution mais sur le fait qu'il peut produire la résolution, (...) car la réussite d'une tâche concrète n'assure pas pour autant l'acquisition d'un savoir ; il faut en plus pouvoir dégager la raison des choses, c'est-à-dire comprendre (Vinh-Bang 1989).

Et avec la démocratisation des nouvelles technologies de l'information et de la communication, l'évaluation assistée par ordinateur s'est ainsi très vite intégrée dans un certain nombre d'environnement pour l'apprentissage humain.

Notre domaine d'application, qui est l'algorithmique, est un domaine où le savoir-faire règne dans la pratique des apprenants. Son but est la résolution de problème en informatique, i.e. pour n'importe quel problème posé, on doit le résoudre d'une manière compréhensible par l'ordinateur.

Notre travail de recherche s'articule autour de l'évaluation du savoir-faire en algorithmique. On entend par le savoir-faire en algorithmique, les compétences de conception d'algorithme (figure 1.1).

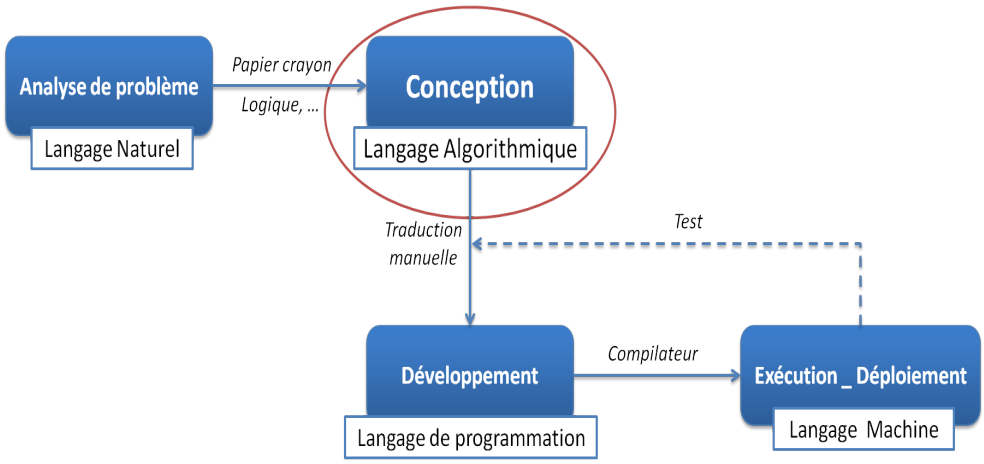

FIGURE 1.1 – *Objectif visée dans le cycle de développement d'un programme.*

1.2 MOTIVATION

Le projet dans lequel s'inscrit ce travail, ambitionne l'amélioration de l'apprentissage de l'algorithmique à l'université de Badji-Mokhtar. Basé sur l'apprentissage par résolution de problèmes, il vise à concevoir un EIAH dédié. Il a pour vocation de combattre la stéréotypie, la passivité, la logique de l'échec, ... dans lesquelles s'embourbent les étudiants en situation d'échec. Il a pour objectifs majeurs de responsabiliser l'apprenant et de l'amener à prendre conscience de ses insuffisances en matière de connaissances pour comprendre, s'exercer au raisonnement algorithmique afin de mieux résoudre un problème.

Dans ce travail, nous nous occupons de l'évaluation des compétences de conception d'algorithme (figure 1.2). Pour l'apprenant, ce besoin d'acquérir des compétences en résolution de problèmes est fondamental (Deek and McHugh 1999, Ragonis and Ben-Ari 2005).

Il est bien connu que plusieurs étudiants trouvent l'apprentissage de

FIGURE 1.2 – *Un élève débutant en pleine métacognition.*

l'algorithmique et de la programmation comme concepts difficiles à maitriser (Thomas et al. 2002). Une étude remarquable menée dans notre université par Bensalem and Bensebaa (2010) a consisté à comparer le taux de réussite des étudiants en algorithmique avec d'autres modules enseignés pendant la même année. Le résultat obtenu montre un taux d'échec de 75% sur les deux ans consécutifs concernés par l'étude.

L'étude a porté sur les étudiants de deuxième année LMD informatique de l'université Badji-Mokhtar de Annaba. Cette étude s'est étalée sur deux années universitaires en 2006/2007, ils ont considéré seulement les nouveaux étudiants, alors qu'en 2007/2008, ils se sont intéressés aux étudiants refaisant le module pour la première fois. L'encadrement a été assuré par un staff d'enseignants qualifiés et réputés pour leur maîtrise du domaine et leur expérience pédagogique. Cet encadrement est composé d'un chargé de cours et d'assistants de travaux dirigés. Ils ont tous contribué à homogénéiser l'évaluation des étudiants.

Pour cette catégorie d'étudiants, l'algorithmique est organisée sous forme de deux unités de valeurs : algorithmique 1 et algorithmique2. Considérée comme la plus difficile à assimiler, algorithmique1 est l'unité de valeur ciblée par cette analyse. Ce module se déroule au premier semestre et couvre des notions de bases sur : les sous-programmes, les structures de données complexes, la récursivité, les pointeurs, les structures arborescentes, etc.

Dans cette étude, ils ont suivi l'évolution du niveau des étudiants via leurs résultats en algorithmique1. Ensuite, ils ont validé les notes obtenues en les comparant avec celles obtenues dans les autres unités de valeur du même semestre. Enfin, ils ont démontré l'influence de l'échec en algorithmique1 sur les résultats d'algorithmique2 et leur forte corrélation. Cette étude s'est terminée par un recueil des constations faites par les enseignants.

a) Évolution des résultats des étudiants ayant poursuivi le module d'algorithmique :

a) Comparaison des résultats obtenus en module d'algorithmique 1 avec les modules enseignés en parallèles pour les nouveaux étudiants de 2006/2007 :

NOTE	Juin	Septembre	VARIATION
<5	84.55%	75.61%	-8.94%
>=5 et <10	13.41%	21.14%	+7.73%
>=10	2.04%	3.25%	+1.21%

Table 1.1 – *Résultats du module algorithmique.*

NOTE	Juin	Septembre	VARIATION
<5	85.31%	74.13%	-11.18%
>=5 et <10	12.59%	23.77%	+11.18%
>=10	2.1%	2.1%	2.1%

Table 1.2 – *Taux de réussite en Algorithmique 2.*

Ce problème d'échec en algorithmique ne concerne pas seulement notre institution. Plusieurs études sur l'apprentissage de l'algorithmique menées par différentes institutions dans d'autres pays (McCracken et al. 2001, Lister et al. 2004) ont convergé vers la même conclusion : l'apprentissage de l'algorithmique était toujours une source de difficulté non seulement pour les étudiants mais pour les enseignants aussi.

1.3 Problématique et approche

Face à ce problème d'apprentissage de l'algorithmique, plusieurs systèmes ont été développés pour aider les débutants en algorithmique et en programmation. Ces systèmes et outils négligent souvent la phase de conception d'algorithme pour se concentrer sur la programmation et le codage de la solution d'un problème donné. Alors que, la phase de conception d'algorithme qui représente la solution d'un problème donné est primordiale et importante dans la mesure où l'étudiant va se concentrer sur la stratégie de résolution seulement et non sur des questions secondaires comme le langage de programmation, la syntaxe, etc.

L'ignorance de la phase de conception avant le codage va engendrer plusieurs effets indésirables (figure 1.3) :

- Une phase de programmation assez compliquée,
- Des étudiants débordés et démotivés,
- Une surcharge cognitive.

Hélas, la conception des algorithmes et non des programmes directement conçus lors de la résolution des problèmes est nécessaire et ses compétences doivent être enseignées d'une manière explicite.

L'évaluation dans le domaine de l'algorithmique, objet de cette thèse, est très délicate. Cependant, il est utopique de croire qu'on peut évaluer les compétences algorithmiques avec des Questions à Choix Multiples (QCM) ou par complétion d'algorithmes. De plus, en algorithmique, un problème donné peut être résolu avec une multitude de solutions. Cette caractéristique

Note	Probabilité statistique	Système d'information	Logique mathématique	Analyse numérique	Architecture des ordinateurs	Cognition	Algorithmique1
<5	5.69%	6.10%	6.91%	8.94%	9.35%	12.60%	75.61%
>=5 et <10	18.29%	20.33%	55.28%	27.64%	33.33%	31.71%	21.14%
>=10	76.02%	73.57%	37.81%	63.42%	57.32%	55.69%	3.25%

TABLE 1.3 – *Taux de réussite d'Algorithmique 1 et des autres modules.*

accroit exponentiellement la difficulté du processus de l'évaluation automatique (Rivers and Koedinger 2012).

Nous supposons qu'il est possible d'évaluer n'importe quelle solution algorithmique d'un problème donné (une évaluation en terme de note et de diagnostic), en étudiant à cet effet les caractéristiques de l'algorithmique.

Nous nous sommes donc intéressés aux problèmes suivants :

1. Identifier un modèle qui nous permet de décrire une solution algorithmique,
2. Proposer une méthode d'évaluation qui se repose sur une méthode d'évaluation formative et sommative,
3. Développer un système informatique qui permet d'écrire une solution algorithmique puis l'évaluer,
4. Réaliser une expérimentation afin de valider les fondements de l'approche proposée en comparant l'évaluation automatisée par notre système et l'évaluation produite par l'enseignant humain.

C'est dans ce cadre que nous proposons un outil d'aide à la correction destiné à faciliter la tâche d'évaluation des apprenants à l'enseignant. L'outil pourra s"intégrer par la suite, comme une composante faisant partie d'un processus d'évaluation dans un EIAH (Environnement Informatique pour l'Apprentissage Humain) donné.

Notre objectif de recherche est divisé en deux :

Objectif à court terme : une évaluation sommative et diagnostic qui va être validée en la comparant avec l'évaluation humaine,

Objectif à long terme : évaluation formative selon une approche par essais et erreur.

L'évaluation des compétences en résolution de problèmes algorithmiques a un double rôle :

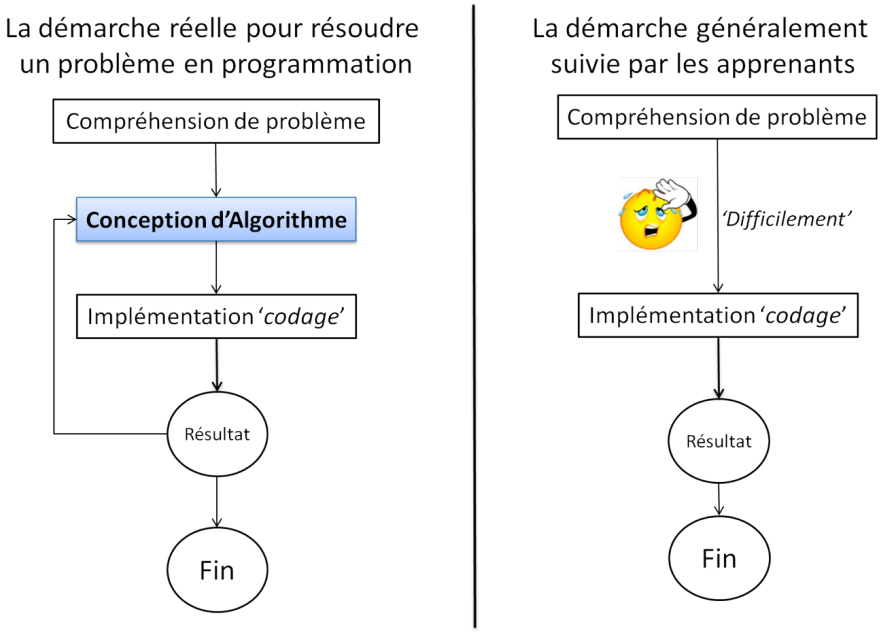

FIGURE 1.3 – *Démarche réelle et démarche poursuivie par les apprenants dans le développement d'un programme.*

1. Évaluer la production de l'apprenant et fournir un feedback pour l'apprenant et l'enseignant,
2. L'apprentissage par l'entrainement, car ce savoir-faire est acquis par l'approche apprentissage de résolution de problème ; *C'est en forgeant qu'on devient forgeron !*. Et selon Astolfi Astolfi (1997) : « *Apprendre c'est franchir progressivement une série d'obstacles* »,

1.4 Méthodologie de recherche

Après avoir bien cadré la problématique et objectif de recherche visés, nous avons essayé de dégager une approche d'évaluation du savoir-faire algorithmique chez les débutants en algorithmique. On entend par débutants en algorithmique, les étudiants ou les apprenants qui ont appris les notions de base en algorithmique (les structures de contrôles, lire une entrée, etc.). Un débutant en algorithmique peut distinguer une structure de test d'une boucle, contrairement au novice qui est en cours d'apprentissage de ces structures. Il peut identifier telle ou telle structure dans un programme mais les différents attributs et aspects identifiés de ces principes sont confondus.

La question de recherche a été abordée en commençant par une étude sur les outils et les modèles d'évaluation dans les EIAH et une étude épistémologique du domaine d'application, l'algorithmique.

A partir de là, une ossature d'idée de base a été conçue qui repose sur : une modélisation d'une solution algorithmique, méthode d'évaluation, une base de solutions et d'erreurs typiques et un mécanisme d'enrichissement de la base de solutions.

Le système d'évaluation Algo+ est l'implémentation de l'approche pro-

posée qui se base sur un : modèle de solution algorithmique et une méthode d'évaluation. Une mesure de similarité utilisée dans le processus de l'évaluation a été proposée. Pour valider cette approche, nous avons expérimenté le système Algo+ en confrontant les résultats d'évaluation donnés par Algo+ et les résultats d'évaluation donnés par un expert.

1.5 Organisation du manuscrit

Ce manuscrit est organisé selon le schéma de description suivant (figure 1.4).

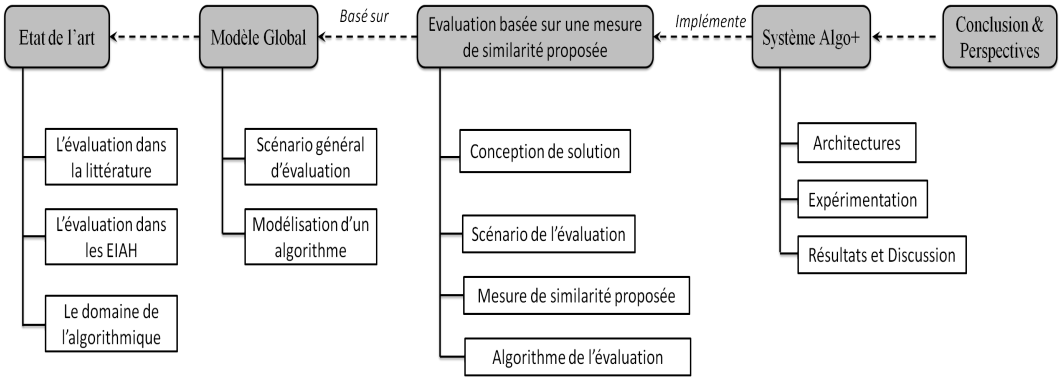

Figure 1.4 – *Organisation du manuscrit.*

Dans le chapitre 2, nous mettons en place une partie du cadre théorique qui concerne l'évaluation comme activité pédagogique à part entière dans un processus d'apprentissage ou de formation.

Dans le Chapitre 3, nous nous intéressons à l'évaluation dans les environnements informatiques pour l'apprentissage humain et l'évaluation des apprenants en utilisant les nouvelles technologies. Nous définissons quelques notions et concepts liées à l'évaluation dans les EIAH.

Le Chapitre 4 est consacré au domaine d'application, l'algorithmique. Une étude épistémologique est présentée afin de mieux cerner les différentes caractéristiques de l'algorithmique et de la programmation. Une panoplie de systèmes dédiés à l'apprentissage et à l'évaluation des productions en algorithmique sont étudiés pour terminer ce chapitre par une synthèse.

Dans le Chapitre 5, la philosophie et le modèle de notre méthode d'évaluation des compétences algorithmiques sont présentés. Les éléments nécessaires pour le fonctionnement de la méthode sont décrits.

Dans le Chapitre 6, nous présentons notre contribution basée sur une modélisation empirique d'une solution algorithmique et une mesure de similarité proposée pour répondre à nos besoins d'évaluation d'une production algorithmique.

Dans le Chapitre 7, l'implémentation de cette approche donne naissance au système Algo+ qui est un système d'évaluation paramétrable des compétences algorithmiques. La validation de ce système et la discussion des résultats obtenus sont aussi présentés.

Pour conclure, le Chapitre 8 nous permet de parachever cette thèse en dressant le bilan de nos travaux en ouvrant d'autres perspectives de recherche.

L'évaluation dans la littérature

2

Sommaire

2.1 La science de l'évaluation en quelques mots 13
2.1.1 Docimologie 13
2.1.2 Edumétrie 14
2.2 Qu'est ce qu'évaluer dans le processus d'apprentissage 15
2.2.1 Définition de l'évaluation 15
2.2.2 La différence entre contrôle et évaluation 15
2.3 A quoi sert l'évaluation et pour qui ? 16
2.4 Les types d'évaluation 16
2.4.1 Par rapport à l'intention d'évaluation 17
2.4.2 Par rapport au moment d'évaluation 20
2.4.3 Par rapport à la forme d'évaluation 20
2.5 Synthèse des types d'évaluation 22
2.6 Outils pour l'évaluation 22
2.7 Relation entre évaluation et processus d'apprentissage 25
2.8 Objectivité et subjectivité dans l'évaluation 26
2.8.1 Les sources de subjectivité 27
2.8.2 La subjectivité : une nature de l'évaluation 28
Conclusion 29

Avec l'apparition des Nouvelles Technologies de l'Information et de la Communication NTIC, de nouvelles méthodes et stratégies d'apprentissage ont vu le jour mais l'évaluation échoue généralement à soulever beaucoup d'enthousiasme dans de nombreux systèmes d'apprentissage (Shaw and Chen 2012).

Dans un processus d'apprentissage, la phase d'évaluation occupe une place prépondérante. Elle constitue l'élément clé dans un grand nombre d'activités pédagogiques. L'évaluation que se soit dans l'enseignement présentiel ou dans les Environnement Informatique pour l'Apprentissage Humains (EIAH) était toujours source d'ambigüités entre l'évalué et l'évaluateur d'une part, et entre les évaluateurs eux-mêmes d'une autre part. Ainsi, elle est le parent pauvre, car très souvent absente ou obsolète dans les différentes formations (Durand 2006).

Nous avons choisi de privilégier au cours de ce chapitre, la pertinence à l'exhaustivité, en commençant par la docimologie. On va discuter de l'état de l'art de l'évaluation dans la littérature (types, rôle et objectifs, etc.), ainsi que des questions à discuter :

Qu'est ce qu'évaluer ? Pourquoi évaluer ? Quels sont les types d'évaluation ? Quels sont les avantages et les inconvénients de l'évaluation ? Quels sont les méthodes pour évaluer ? Quels buts une évaluation doit avoir ? Comment évaluer l'évaluation ? Toutes ces questions vont êtres discutées dans cette partie.

2.1 La science de l'évaluation en quelques mots

L'évaluation peut être vue sous plusieurs angles : politique, social, pédagogique, etc. et avec plusieurs rôles : notation, sélection, diagnostic, maitrise, guidage, prédiction, etc Morris and Adamson (2010).

Depuis longtemps, l'évaluation est omniprésente surtout dans le domaine de l'éducation et elle constitue un concept plurivalent. Des scientifiques se sont intéressés à son étude afin d'éclaircir toutes les ambiguïtés liées à ce processus qui est synonyme d'anxiété, de pression, d'échec, de jugement, de feedback, de compétition, ... Cela dépend de la nature des personnes qui participent dans le processus d'évaluation.

Dans ce qui va suivre, on va aborder d'une manière succincte deux disciplines, qui sont liées, et qui traitent les aspects de l'activité d'évaluation à savoir la subjectivité et l'objectivité de cette activité.

2.1.1 Docimologie

Dans le vocabulaire de la psychologie de Piéron, la docimologie est définie comme suit :

Définition 2.1 *"l'étude systématique des examens (modes de notation, variabilité interindividuelle et intra-individuelle des examinateurs, facteurs subjectifs, etc.)".*

Ce terme proposé par Pieron lui-même est dérivé du grec *dokimé* qui veut dire épreuve. Les premiers travaux ont été réalisés par H. et M. Piéron, H. Laugier et D. Weinberg.

Les études docimologiques se sont appliquées à dégager des « effets » qui influencent les notations, les « jugements évaluatifs » des examinateurs/évaluateurs.

L'étude initiatrice de ce domaine était en 1930 où le professeur Laugier a réalisé une expérience de multi correction des copies d'agrégation d'histoire puisées dans les archives. L'étude a été menée sur 166 copies corrigées par deux professeurs expérimentés. Le résultat a été surprenant :

- La moyenne des notes du premier correcteur dépassait de près de deux points celle du second,
- Le candidat classé avant dernier par l'un était classé second par l'autre,
- Les écarts de notes allaient jusqu'à 9 points,
- Le premier correcteur donnait 5 à 21 copies cotées entre 2 et 14 par le second ; le second donnait 7 à 20 copies cotées entre 2 et 11,5 par le premier,
- La moitié des candidats reçus par un correcteur était refusée par l'autre.

Cette expérience caractéristique de docimologie, a amené de plus en plus des chercheurs de plus en plus nombreux à s'interroger sur les sources d'erreurs des procédures d'évaluation traditionnelles.

2.1.2 Edumétrie

L'édumétrie est une discipline relativement récente qui s'intéresse aux mesures des apprentissages sur les plans théorique, méthodologique et technique. On peut citer, à titre d'exemple d'application, la mesure et la quantification de la progression des apprentissages entre pré- et post-test. Elle est considérée habituellement comme une branche de la docimologie.

L'édumétrie est la mesure des résultats de l'apprentissage, au sens large que les sciences de l'éducation donnent à ce terme. Alors que la psychométrie classe les individus les uns par rapport aux autres, soit avant, soit après l'étude, l'édumétrie évalue le progrès réalisé par un individu entre les étapes successives de son apprentissage. Les méthodes utilisées classiquement pour construire des tests ne sont pas applicables à l'édumétrie, parce que cette dernière ne s'intéresse pas à la variance entre les performances individuelles, alors que les tests habituels définissent au contraire cette source de variance comme la variance '*vraie*'. Pour différencier et individualiser l'enseignement, ce sont manifestement des mesures édumétriques qui s'imposent.

2.2 Qu'est ce qu'évaluer dans le processus d'apprentissage

2.2.1 Définition de l'évaluation

Définition 2.2 *Le terme évaluer vient du latin « valere » signifiant valoir. Dans le dictionnaire Larousse, on trouve la définition de ce mot : Déterminer, fixer, apprécier la valeur, le prix de quelque chose, d'un bien, etc.*

Dans le cadre de l'apprentissage humain, pour Charles Hadji (Hadji 1977), Évaluer signifie : interpréter, vérifier ce qui a été appris, compris, retenu, vérifier les acquis dans le cadre d'une progression, juger un travail en fonction des critères donnés, estimer le niveau de compétence d'un apprenant, situer l'apprenant par rapport à ses compétences, déterminer le niveau d'une production donnée par l'apprenant.

A partir de ces définitions, on peut dire que l'évaluation est un acte de porter le jugement sur un travail, à partir d'objectifs d'apprentissages, dans le but de prendre une décision. Cette décision peut être selon Abdourahmane (2008) :

- un passage à la séquence suivante,
- une proposition de nouvelles activités,
- Revoir la stratégie de la séquence,
- Revoir les modalités d'évaluation,
- Modifier les objectifs ou les abandonner.

2.2.2 La différence entre contrôle et évaluation

A la suite des travaux de Ardoino and Berger (1989), on distingue les deux entités, contrôle et évaluation. La figure suivante illustre la différence.

Le contrôle	L'évaluation
Mesure des écarts entre des produits, des démarches et une norme extérieure, préétablie. Le contrôle est régi par des critères de conformité, de logique, de cohérence. Il vérifie pour valider ou rejeter, corriger ou sanctionner. Il vise à normaliser	Englobe et dépasse le contrôle. Elle privilégie le qualitatif sur le quantitatif Au-delà du contrôle analytique, elle est conçue comme un processus intervenant dans un système ouvert, en évolution, en vue d'en élucider le fonctionnement et l'évolution. En ce sens, elle ne peut être que partagée par les acteurs.

Table 2.1 – *La différence entre Contrôle et Évaluation.*

2.3 A quoi sert l'évaluation et pour qui ?

Le terme évaluation signifie un tas d'actions tenues pour collecter et utiliser des informations sur les connaissances, des attitudes ou des compétences d'un apprenant. Il y a plusieurs manières pour catégoriser les buts de l'évaluation selon le but final de l'évaluation. Dans ce cas, il y a deux raisons pour évaluer : (1) pour établir un jugement sur la performance des apprenants ou l'efficacité du système d'enseignement, (2) pour améliorer l'apprentissage Berry (2008). Ou selon les acteurs qui sont impliqués dans le processus de l'évaluation : Apprenants, enseignants ou ceux qui s'intéressent aux résultats de l'évaluation : institutions et parents.

Apprenants	Enseignants	Institutions
• Favoriser la motivation et l'autonomie ; • Inciter l'élève à identifier ses erreurs par des questions précises ou des références à des outils précis ; • Sélectionner les erreurs prioritaires.	• Donner du sens à partir des analyses des réponses des élèves ; • Faire en sorte qu'elles soient utiles dans la classe, réfléchir sur les besoins spécifiques des élèves ; • Créer des groupes de besoin en ciblant les difficultés ; • Agir après l'évaluation ; • Évaluer pour prendre des décisions ; • Comptabiliser ce qui vient d'être compris que remédier aux dysfonctionnements de la leçon ; • Contrôler en permanence la progression d'un élève ;	• Agir après l'évaluation ; • Évaluer pour prendre des décisions ; • Diagnostiquer, prévoir, positionner les formés ; • Faire le point, conduire l'action, régulée pour les formateurs ; • Juger de l'efficacité : pédagogique, dans la profession, sur le terrain.

Table 2.2 – *Le but de l'évaluation selon les acteurs.*

2.4 Les types d'évaluation

L'évaluation, en effet, (du produit ou du processus d'apprentissage) revêt des aspects différents selon le but recherché, le moment où elle intervient

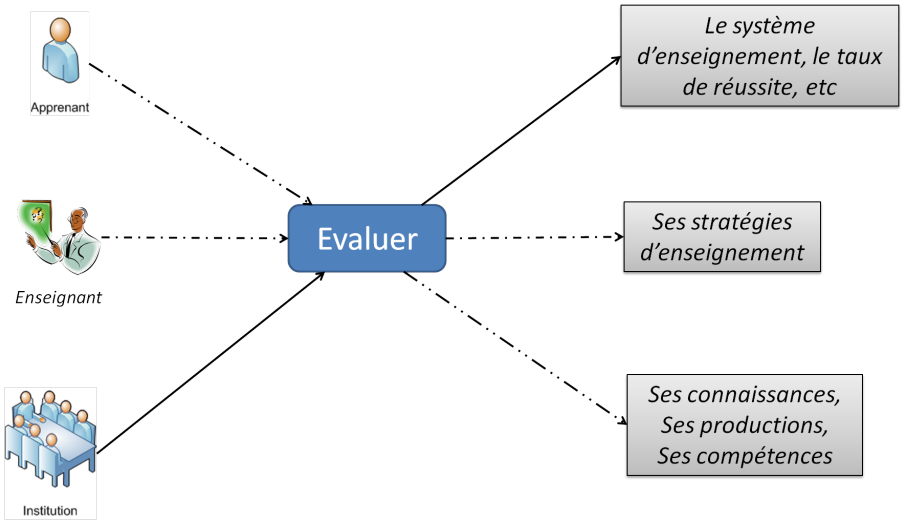

FIGURE 2.1 – *Le rôle de l'évaluation selon les différents acteurs.*

dans le processus pédagogique ou dans l'élaboration du curriculum ou la forme sous laquelle elle est présentée.

A partir delà, nous allons voir les différents types d'évaluation par rapport à son intention, à son moment et à sa forme.

2.4.1 Par rapport à l'intention d'évaluation

Évaluation formative

L'évaluation formative est une évaluation pour le processus d'apprentissage car elle est toujours présente pendant les cours. En quelque sorte, l'évaluation formative est une évaluation diagnostic et elle fournit un feedback non seulement pour l'apprenant mais aussi pour l'enseignant par rapport à son processus d'enseignement. Aussi, elle permet aux apprenants et aux enseignants de modifier leurs comportements dans le but d'améliorer et de faciliter l'apprentissage.

L'évaluation formative est une évaluation des apprenants favorisant l'apprentissage. Ce mode d'évaluation peut reposer sur divers modèles pédagogiques :

- Évaluation des progrès réalisés au regard des objectifs attendus, donc c'est une source de motivation ;
- Évaluation par essais/erreurs : l'erreur est formatrice pour peu qu'elle ne soit pas perçue comme un échec ;
- Évaluation croisée : des apprenants évaluent d'autres apprenants, ce qui leur permet de prendre du recul par rapport à la formation, et participer à la mémorisation ;
- Auto-évaluation.

Selon Hadji (Hadji 1990), l'évaluation formative est une "évaluation dont l'ambition est de contribuer à la formation". Dans ce contexte, elle a pour

but de réguler l'enseignement. Ce type d'évaluation fournit des informations permettant à l'enseignant d'adapter son enseignement aux particularités de l'apprenant.

Contrairement aux autres évaluations, l'évaluation formative ne se contente pas d'évaluer les productions des élèves mais aussi les situations actives permettant de comprendre la démarche des apprenants, leurs rapports aux savoirs, leurs capacités de métacognition, etc (Perrenoud 2001).

Voici quelques raisons pour utiliser une évaluation formative :

- Stimuler la réflexion et engager les apprenants dans le processus d'apprentissage,
- Expliciter les idées des apprenants pour eux mêmes et pour l'enseignant,
- Confronter les idées des apprenants et les encourager à la curiosité intellectuelle ,
- Aider les apprenants à penser à d'autres solutions,
- Fournir le stimulus pour la discussion et l'argumentation/débat,
- Aider les apprenants à reconnaitre ce qui est appris et non appris,
- Développer le transfert formel des concepts,
- Déterminer si les apprenants ont capable d'appliquer des idées à des nouvelles situations,
- Aider les apprenants à développer l'auto évaluation et l'évaluation par pair,
- Donner et utiliser un feedback (apprenant à apprenant, enseignant à apprenant et apprenant à enseignant),
- Encourager la construction sociale des idées,
- Réaliser un ajustement immédiat ou en différé de la démarche d'apprentissage,
- Encourager la participation de tous les apprenants,
- La qualité d'apprentissage de l'apprenant,
- La qualité de l'apprentissage,
- L'identification par les enseignants et apprenants des objectifs pédagogiques, des intentions ou des conclusions et des critères pour les achever,
- Prévoir un feedback pour permettre aux apprenants de pousser leur apprentissage,
- La participation des apprenants dans leur apprentissage,
- Les enseignants à identifier les manques du processus d'apprentissage et modifier leurs approches d'enseignements.

Par analogie, imaginons que les apprenants sont des plantes, l'évaluation formative est dans ce cas équivalente à l'arrosage de ces plantes qui va les aider à pousser, s'il s'est présenté dans le moment approprié.

Évaluation formatrice

C'est une forme particulière d'évaluation formative. Selon Bonniol (1986), l'évaluation formative s'inscrit dans une visée de régulation de l'apprentissage par l'enseignant tandis que dans l'évaluation formatrice, la régu-

lation est assurée par l'apprenant. En ce sens, l'activité d'auto-évaluation, qu'elle soit individuelle, mutuelle ou collective, est une évaluation formatrice.

Évaluation réflexive

Elle permet à l'apprenant de prendre du recul par rapport à ses apprentissages. Elle lui permet de prendre conscience : des processus d'apprentissage qu'elle met en œuvre, de sa façon d'organiser son travail, de gérer son effort, etc.

Évaluation régulatrice

Elle permet au formateur de savoir où en sont les apprenants afin d'ajuster ses interventions, est-ce trop rapide ? trop lent ? trop théorique ? trop pratico-pratique ? les apprentissages répondent-ils aux attentes des apprenants ?, etc. C'est une sorte d'état des lieux des interventions qui permet au formateur de réguler son intervention, de la réajuster.

Évaluation sommative

Les méthodes d'évaluation sommative sont les plus anciennes méthodes d'évaluation des travaux d'apprenants. L'évaluation sommative est une évaluation du processus d'apprentissage. Elle intervient après une séance d'apprentissage ou à la fin d'une session, d'un semestre, d'une année, pour juger et quantifier ce qui a été appris pendant le processus d'apprentissage. Elle constitue un bilan des apprentissages d'un sujet ou d'un groupe de sujets : dans quelle mesure ont-ils atteints les objectifs prévus ? La décision consécutive à ce bilan se traduit par une note, une mention, une promotion ou une certification.

L'évaluation sommative peut prendre la forme d'une interrogation, mais pas nécessairement. Une bonne évaluation sommative, tests et autres doivent être bien évidemment consistantes, valides et non biaisées (Angelo and Cross 1993).

C'est l'évaluation par laquelle on fait un inventaire des compétences acquises, ou un bilan, après une séquence de formation d'une durée plus ou moins longue Hadji (1990). Elle est en opposition avec l'évaluation formative, elle ne régule pas l'apprentissage, mais elle le contrôle.

L'enseignant peut varier ses procédures et proposer par exemple la résolution d'un cas ou d'un problème, la réalisation d'un projet, une production personnelle pour autant, toutefois, que ces procédures restent cohérentes par rapport aux objectifs dûment définis.

L'évaluation sommative peut prendre la forme d'examens périodiques qui valident un apprentissage. Elle conduit à l'obtention d'une note qui sanctionne une activité d'apprentissage afin d'établir un classement, sélectionner les apprenants ou certifier leur niveau. En mettant l'accent sur les perfor-

mances, l'évaluation sommative s'intéresse essentiellement aux productions réalisées par les apprenants.

Si on reprend la métaphore des plantes, l'évaluation sommative des plantes est simplement de les mesurer. Il se peut être intéressant de comparer et analyser ces mesures mais, cela ne va pas influencer la croissance de ces arbustes.

2.4.2 Par rapport au moment d'évaluation

Pronostique

Ce type d'évaluation se situe en amont d'un cursus. Elle permet d'évaluer la capacité d'un apprenant à commencer un apprentissage, un cycle d'étude ou à exercer une profession. C'est une évaluation en amont d'une réalisation ou d'un apprentissage. Elle permet de prévoir les chances de succès d'un sujet dans un apprentissage donné. Avant un cours, l'enseignant peut, par exemple, contrôler les pré-requis, les points forts et les points faibles de ses étudiants en vue d'optimiser sa démarche didactique. Bien entendu, une telle évaluation n'est pas nécessairement cotée et elle s'effectue surtout à l'occasion d'une nouvelle formation.

Diagnostic/Formative

L'évaluation diagnostique permet d'évaluer un niveau de compétence bien souvent juste avant une nouvelle phase d'apprentissage. Dans le cadre d'une évaluation formative, ce diagnostic permet la remédiation et la mise en œuvre d'une pédagogie différenciée.

2.4.3 Par rapport à la forme d'évaluation

Une évaluation peut être conçue selon deux modalités : une évaluation qui compare les élèves les uns aux autres, dans le groupe classe ou entre des groupes de même niveau, et une évaluation qui permet de certifier ce que les élèves peuvent faire et ne peuvent pas faire, indépendamment les uns des autres, en rapportant cette certification à un ensemble de critères prédéfinis.

On parlera, dans le premier cas, d'évaluation normative, et, dans le second, d'évaluation critériée.

Normative

Une évaluation qui situe les individus les uns par rapport aux autres, en fonction des scores obtenus par les membres d'un groupe de référence. Dans l'évaluation scolaire courante, la norme de référence est bien souvent constituée par les performances moyennes du groupe classe (Résultats comparés aux résultats du groupe).

Elle traduit donc les résultats d'un apprentissage, le produit d'une tâche, en termes de comparaison avec les résultats des autres élèves de la classe voire avec des élèves appartenant à d'autres groupes de même niveau. Elle permet ainsi de situer chaque élève par rapport à la moyenne de son groupe ; c'est cette moyenne qui représente la norme du groupe en question.

Il est capital de bien comprendre que ce n'est pas par rapport à une moyenne absolue qu'on classe les élèves. En effet, la moyenne normative est toujours fonction du niveau d'un groupe de référence donné parce que, lors de l'élaboration de l'épreuve d'évaluation, il est nécessaire d'insérer des questions faciles, de difficulté moyenne et des questions difficiles dans le but de discriminer les élèves les uns par rapport aux autres.

A l'occasion d'une évaluation normative, les élèves sont classés du plus fort au plus faible parce que l'épreuve a été construite dans ce sens ; on prend essentiellement en compte l'écart de la moyenne de chacun par rapport à la moyenne du groupe dont il fait partie.

L'évaluation normée comporte des avantages. Elle fournit des éléments de décision aux responsables du système éducatif chargés de prendre des décisions sur l'organisation des études, le contenu des programmes et des méthodes d'enseignement. Elle permet de remédier à des carences localisées ou de réfléchir sur des résultats exceptionnellement bons. Elle présente, par contre, les inconvénients critiqués traditionnellement. Elle ramène tout à une norme et ne pose pas le problème des objectifs, de méthodes didactiques et des moyens d'évaluation. C'est un instrument de sélection et d'orientation. Elle ne donne prise ni à une pédagogie corrective, ni à une pédagogie de maîtrise.

Critériée

Selon Philippe Meirieu [1] : « *Critérier l'évaluation, c'est permettre à l'élève de savoir précisément ce qu'il sait faire et ce qu'il ne sait pas faire ... précisément et en termes opérationnels : il ne s'agira pas d'expliquer à un élève qu'il ne sait pas faire une dissertation française, mais de lui dire où il en est dans le domaine du plan, des exemples, de l'introduction, des transitions, du style' Une fois ce repérage effectué, l'on peut mettre en place les remédiations nécessaires et regrouper alors les élèves en fonction des reprises ou des entraînements qui leur seront nécessaires.*»

L'évaluation critériée vérifie les performances d'un élève en fonction d'un ensemble de compétences (savoirs, savoir-faire, savoir-être) constituées en critères par rapport à un modèle défini par avance. Dans l'enseignement technique par exemple, les référentiels de diplômes sont ces modèles, à un niveau donné et pour une formation donnée.

C'est une évaluation qui apprécie un comportement en le situant par rapport à une cible (le critère qui correspond à l'objectif à atteindre). Autrement dit, le résultat obtenu de l'évaluation est comparé à un critère/performance.

R. Glaser, cité par V. de Landsheere, précise « Comme les tests critériés

1. http ://meirieu.com/

sont spécialement conçus pour fournir une information (directement interprétable) sur les niveaux de performance, ces niveaux doivent être définis avant de construire le test. Le but du testing est d'évaluer la position de chaque individu par rapport à ces niveaux. »

Contrairement à l'évaluation normative, la démarche critériée ne situe pas les élèves les uns par rapport aux autres ; elle prend plutôt compte des performances (que ce soit en plus ou en moins) par rapport à une sorte de « capital-cible ». Le but poursuivi est, bien entendu, 100% de réussite. Il n'y a pas de questions trop faciles ou de questions plus difficiles puisqu'il n'est pas question d'aboutir à un classement des élèves sur une échelle ordinale. Les tâches proposées permettent d'évaluer à quel point « d'un continuum dans l'acquisition de la connaissance, allant de l'absence totale de compétence à la performance parfaite » (De Landsheere 1979).

2.5 Synthèse des types d'évaluation

Comme nous pouvons le remarquer dans la section précédente, certaines évaluations peuvent avoir un rôle différent. La même évaluation peut être utilisée avant ou pendant le temps de formation, une autre pendant ou après la formation. Nous voyons bien, après l'énumération de ses différentes formes, que l'évaluation, selon le moment où elle intervient, joue un rôle très différent dans une formation. Elle permet de faire un bilan de compétence, une sélection avant, une réorientation ou un rééquilibrage pendant et une certification ou une orientation après.

La figure 2.2 résume les différentes évaluations selon trois dimensions : le moment de l'apprentissage, l'intention de l'évaluateur et sa modalité d'évaluation.

2.6 Outils pour l'évaluation

Bien entendu, les outils d'évaluation seront très largement déterminés par les objectifs, mais il est possible, de mettre en relation ces outils et la taxonomie du domaine cognitif proposée par Bloom et al. (1956). Les outils d'évaluation les plus fermés sont parfaitement adaptés à des objectifs dont le niveau d'exigence les situe sur les premiers degrés de cette taxonomie, alors qu'aux niveaux les plus élevés correspondent des outils plus ouverts.

Le schéma décrit dans la figure 2.3 permet de donner une idée précise de ces correspondances ; il faut toutefois ajouter qu'il n'a qu'une valeur indicative : en effet, une catégorie d'objectifs, à un niveau d'exigence déterminé, peut être évaluée à l'aide de plusieurs types d'outils voisins.

L'outil de suivi pédagogique ou d'évaluation est d'abord qualifié par le type de réponse qu'il implique. Ainsi, le questionnaire à choix multiple (QCM) ne demande que l'apposition de croix dans des cases, ce qui est la forme de réponse dite la plus fermée. Inversement, la production d'un travail personnel est un outil très ouvert.

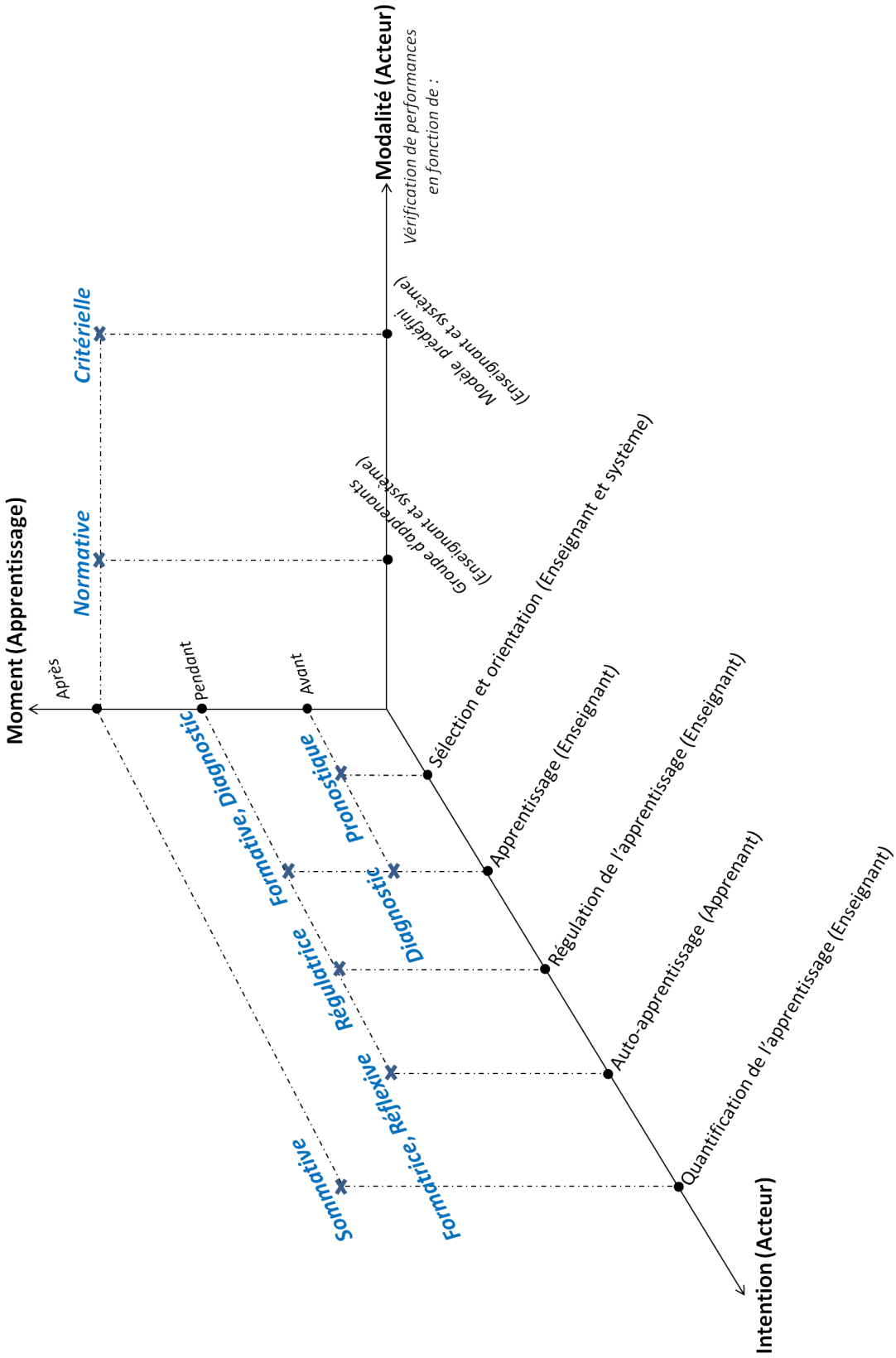

FIGURE 2.2 – *Synthèse des types d'évaluation selon trois axes : Moment, Modalité, Intention.*

Les outils d'évaluation ouverts requièrent de l'élève une plus grande maîtrise ; ils mettent en œuvre des compétences de plus haut niveau et supposent la maîtrise d'un faisceau de plus en plus large de compétences.

Entre les formes extrêmes de fermeture et d'ouverture, existe toute une variété d'outils où le degré de liberté laissé à l'apprenant est variable.

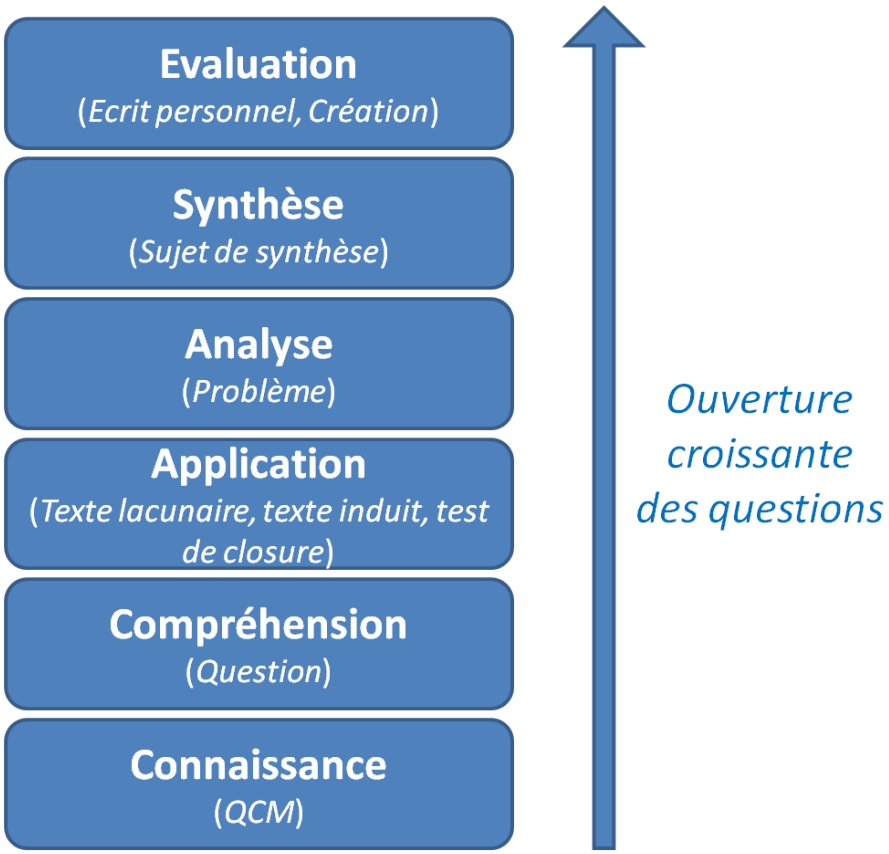

FIGURE 2.3 – *Les outils appropriés à chaque niveau de Bloom.*

De l'apposition d'une croix dans une case à la production d'un document personnel, une infinité d'outils s'offrent au pédagogue, lui permettant d'adapter le degré d'ouverture de ses modes de suivi et d'évaluation à la complexité de ses objectifs. La liste suivante, n'est pas exhaustive, mais elle donne un aperçu de la variété des possibilités.

- *Questionnaire à choix multiple* : A l'intérieur même de cette forme d'outil, de nombreuses possibilités sont offertes de faire varier la difficulté et la compétence impliquée. Les items sont dits de sélection, car il s'agit de choisir la ou les bonnes réponses parmi plusieurs distracteurs (destinés à induire en erreur). QCM intelligent (Mickael 2002) qui posent de nouvelles questions suivant les réponses de l'apprenant. Inconvénients : la conception d'un bon questionnaire demande du temps, une réponse exacte peut être due au hasard, un questionnaire ne permet pas de tester la capacité à rédiger, à créer, ou à produire des raisonnements complexes.
- *Question classique (et exercice)* : Elle appelle théoriquement une réponse attendue. Par exemple : "Quel est l'auteur de tel ouvrage, ou quel est le résultat de telle opération ?". Mais, elle peut aussi consister

en une consigne telle que : "Recopier telle figure, tourner telle phrase à la forme passive", etc. L'ouverture diffère selon qu'il s'agit de reproduire (savoirs) ou d'appliquer une règle (savoir-faire).

- *Texte lacunaire* : Il est très employé en langue vivante pour vérifier le vocabulaire ou l'application de règles. Dans ce dernier cas, les phrases sont composées de telle manière que le mot manquant s'accorde par exemple, d'une manière enseignée dans le cours qui a précédé. La réponse est normalement univoque.
- *Test de closure* : il s'agit d'un texte lacunaire particulier destiné à mesurer la lisibilité d'un texte ou la capacité de compréhension des apprenants. Il s'agit de trouer un texte existant en laissant en blanc un mot sur cinq (en règle générale). L'apprenant doit compléter ce texte en proposant les mots manquants.
- *Questionnaire à réponses ouvertes courtes* : A la différence de la question classique, il appelle une réponse dont la forme et parfois le contenu est libre à l'intérieur de limites. Donner une définition de telle notion, citer cinq des auteurs d'un traité, etc.
- *Texte induit* : Après avoir enseigné un certain nombre de notions, il est possible d'en vérifier la compréhension en proposant aux apprenants de composer un texte à partir d'une liste de notions imposées. À l'aide de consignes, le degré d'ouverture est adaptable avec précision. Il faudra, par exemple, employer les termes dans l'ordre ou non, exclusivement dans le sens attendu ou non, donner des définitions ou non, le texte sera limité ou non, etc.
- *Problème* : Faisant appel à l'esprit d'analyse, le problème propose une situation nouvelle demandant l'application de règles connues, mais dont le choix est à faire par l'apprenant, contrairement à l'exercice où il s'agit d'appliquer une règle indiquée. Un certain recul par rapport à la situation proposée est requis, ainsi qu'une capacité de juger de la vraisemblance des résultats.
- *Production personnelle* : L'imposition du thème et celle du temps restent en vigueur, mais les moyens utilisés et les jugements exprimés sont libres.

2.7 Relation entre évaluation et processus d'apprentissage

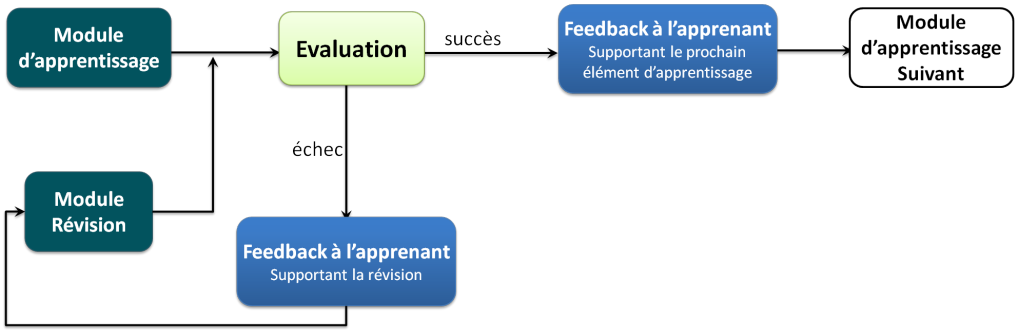

Figure 2.4 – *Relation entre l'évaluation et un processus d'apprentissage.*

Comme il est montré dans la figure 2.4, l'évaluation se situe au cœur de n'importe quel processus d'apprentissage. Elle est le moyen de réguler l'apprentissage. Dans le cas d'une évaluation réussie, un feedback est présenté à l'apprenant afin de le préparer à la prochaine séquence d'apprentissage. Sinon, un feedback est donné dans le but d'éclaircissement et la séquence d'apprentissage sera révisée puis réévaluée.

D'une autre part, l'évaluation a principalement deux objectifs : évaluation pour l'apprentissage ou de l'apprentissage. Le tableau ci-dessous explique le rôle de chacune par rapport au processus d'apprentissage.

Évaluation POUR l'apprentissage	Évaluation DE l'apprentissage
Enseignants, apprenants et parents sont les acteurs principaux	Enseignants, apprenants, superviseurs, didacticiens et pédagogues sont les acteurs principaux
Pendant le processus de l'apprentissage	Après le processus de l'apprentissage
Utilisée pour fournir des informations sur quoi et comment améliorer le succès	Utilisée pour certifier les compétences des apprenants
Utilisée par les enseignants pour identifier et répondre aux attentes des apprenants	Utilisée pour classer les apprenants
Objectif : améliorer l'apprentissage	Objectif : juger la qualité de l'apprentissage
Continue	Périodique
Exemple : évaluation par pair, un feedback descriptif	Exemple : les examens, les tests

TABLE 2.3 – *La différence entre l'évaluation de l'apprentissage et l'évaluation pour l'apprentissage.*

2.8 Objectivité et subjectivité dans l'évaluation

« J'ai été saqué ! », « Le prof s'est trompé », « Avec Mme X, j'aurais eu une meilleure note »

Ces petites phrases sont monnaie courante lors des examens des étudiants dans les différents niveaux d'apprentissage. Beaucoup d'étudiants sont donc peu convaincus de la fiabilité des notes, surtout dans l'enseignement supérieur.

Stephen et al. (2000) montrent que le changement des critères de mesure peut dramatiquement biaiser les résultats de l'évaluation tandis que Black and Wiliam (1998) ont montré qu'une bonne méthode d'évaluation formative peut apporter une amélioration au niveau d'apprentissage des apprenants. Un état de l'art de EPPI Centre Harlen and Deakin (2002) montre qu'une évaluation sommative régulière peut avoir un effet négatif sur la performance des apprenants faibles, tandis que beaucoup moins vis-à-vis les apprenants forts. De grands pas ont été faits en revanche sur la connaissance des facteurs susceptibles d'influencer le correcteur appelés effets parasites ou biais. La notion de biais, d'effet perturbateur ou parasite

suppose qu'on cherche à expliquer l'écart entre la note donnée et la note méritée.

2.8.1 Les sources de subjectivité

Il s'agit ici de montrer les différents effets qui contaminent l'évaluation des réponses des élèves et plus précisément les notes. Des études sont faites sur des examens pour identifier les biais, les effets perturbateurs qui expliqueraient les variations entre les notations. Il s'agit de trouver des lois qui rendraient compte des problèmes de la "fidélité" des notes (aboutir au même résultat quel que soit le nombre de passations ou de correcteurs), de la "validité" des examens (n'évaluer que ce qui est affiché) et de la "sensibilité" des outils d'évaluation (utiliser harmonieusement les échelles de mesure).

Effets d'ordre et de contraste

Bonniol (1986) présenta en 1972 une thèse sur l'estimation par contraste. Il proposa à des groupes d'enseignants de corriger des séries de copies identiques, mais dans des ordres différents. En répétant cette expérience un nombre de fois suffisant, il constata deux phénomènes :

- Les correcteurs notent par contraste, c'est-à-dire que la note d'une copie dépend en partie de la ou des copies précédentes et leur est en quelque sorte opposée, i.e. L'évaluateur se laisse influencer par l'ordre des copies (les premières sont surévaluées) et par la qualité de la copie précédente. Un travail moyen paraîtra bon s'il suit un travail médiocre.
- D'une manière générale, les correcteurs sont plus sévères à la fin de la série qu'au début.

A partir de cet effet de contraste, Bonniol (1986) utilise la notion d'ancre (référence à l'instrument de marine). Dans une série de copies sélectionnées après plusieurs corrections, pour leurs notes moyennes, le chercheur introduit une ou plusieurs très bonnes (ancres hautes) ou très mauvaises copies (ancres basses) pour constater leurs effets sur les copies suivantes. Il montre, par exemple, qu'en anglais, l'effet de contraste est important surtout après les ancres basses ; alors qu'en mathématiques, ce sont davantage les ancres hautes et lourdes (plusieurs copies excellentes) qui entraînent la sous-estimation.

Effets de contamination

Les notes / points attribués successivement aux différents aspects d'un même travail s'influencent mutuellement. L'avis des confrères a également un rôle influent sur le jugement du correcteur. Noizet and Caverni (1977) propose à deux groupes d'enseignants de sciences naturelles de noter quatre copies différentes. Il donne à chaque correcteur un faux dossier scolaire par copie : le premier groupe de correcteurs dispose d'appréciations antécédentes élevées et stables, le deuxième groupe se fonde sur des dossiers aux notes faibles et dispersées. Les résultats font apparaître le peu d'écart absolu (0,25/20) dû à l'influence d'un dossier pour une copie notoirement faible (de 2,75/20 à 3,00). Mais l'écart grandit au fur et à mesure que la qualité des copies augmente. Il culmine à 3,25 sous l'influence de dossiers opposés

pour des copies de haut niveau. En conclusion, un dossier ne rachète pas une mauvaise copie, alors qu'il est très influent pour un bon travail.

Effets de stéréotypie

Après quelques mois, l'enseignant connaît suffisamment les élèves pour avoir une idée de leurs compétences actuelles. L'effet de stéréotypie est une systématisation de l'appréciation établie.

Effet de halo

Gilly (1980) consacre plusieurs chapitres de son ouvrage aux représentations respectives des élèves et des maîtres, émanant de stéréotypes sociaux de natures très diverses : habillement, verbalisation, attitudes face à l'école, etc. Ces variables sont très importantes dans la relation pédagogique, et se retrouvent dans l'évaluation où l'effet de halo est à l'œuvre. Le professeur, influencé par des caractéristiques de présentation de la copie (soin, écriture, orthographe) ou de l'élève, surestime ou sous-estime la note.

Effet de fatigue ou d'ennui : peut engendrer laxisme ou sur-sévérité.

Effet d'assimilation

L'évaluation est influencée par les informations que le professeur possède de l'élève (statut scolaire de l'élève, origine socioéconomique, origine ethnique) indépendamment de la présentation de la copie de l'élève.

Effet de tendance centrale

Par crainte de surévaluer ou de sous-évaluer un élève, le professeur groupe ses appréciations vers le centre de l'échelle.

Effet de trop grande indulgence / de trop grande sévérité

Certains évaluateurs sont systématiquement trop indulgents ou trop sévères dans toutes leurs évaluations (personnalité des évaluateurs).

Effet de place des erreurs dans la copie

Les erreurs en début de copie provoquent un a priori défavorable.

Voici quelques biais et effets parasites qui ont été remarqués pendant des études et qui influencent directement l'évaluation des apprenants, donc une évaluation dévalorisante qui engendre par la suite une activité pédagogique inappropriée.

C'est à ce moment que la subjectivité implicite dans le processus d'évaluation des apprentissages prend forme, compliquant considérablement le processus. Il est intéressant d'aller un peu plus loin et de dresser un parallèle entre le processus de recherche qualitative et le processus d'évaluation des apprentissages pour clarifier d'où provient la subjectivité présente en évaluation et comment la maîtriser.

2.8.2 La subjectivité : une nature de l'évaluation

C'est cette obsession de la *vérité docimologique* qu'accusent les détracteurs : la vraie note, la note juste, n'a plus grand intérêt quand on a compris

qu'on n'arrivera jamais à neutraliser tous les biais possibles cités avant, que l'objectivité n'existe pas ; que ni les situations d'évaluation ne seront purifiées, ni les outils d'évaluation ne seront parfaits, ni l'évaluateur ne sera libéré de ses conditions humaines idiosyncrasiques.

Pour ce faire, on a remarqué que la plupart des définitions de l'évaluation tournent au tour de cette définition :

- Recueillir un ensemble d'informations suffisamment pertinentes, valides et fiables.
- Comparer cet ensemble d'informations et un ensemble de critères adéquats aux objectifs fixés au départ.
- Pour prendre une décision.

Subjectivité liée à l'objectif de l'évaluation

Est-ce que j'évalue un apprenant pour décider s'il peut passer à un niveau supérieur ou pour voir ce que je peux faire pour l'aider à progresser ?

Subjectivité liée au choix des critères

Une fois le type de décision à prendre est connu, l'évaluateur va devoir choisir les critères auxquels devra correspondre l'objet évalué. Un critère est une qualité que l'on attend de l'objet évalué, c'est un point de vue duquel on se place pour porter un regard sur l'objet.

Subjectivité liée aux choix des stratégies

Pour recueillir l'information à propos des indicateurs, l'évaluateur va devoir choisir une stratégie, parmi plusieurs méthodes de recueil d'informations. Il peut interviewer des personnes, les interroger par voie de questionnaire, étudier des documents, observer une situation, etc. Toutes ces méthodes peuvent être utilisées, mais certaines sont plus adaptées que d'autres à l'une ou l'autre situation De Ketele and Roegiers (2009). Pour garantir la rigueur du processus, le choix de la stratégie doit permettre de garantir deux caractéristiques importantes de l'information recueillie : sa validité et sa fiabilité.

Conclusion

En résumé, l'évaluation est un processus qui fait partie intégrante de la pédagogie et du contenu enseigné. Elle joue un rôle important du point de vue des apprentissages et des pédagogies appliquées. Elle constitue en quelque sorte un couloir de guidage de la progression et intervient au niveau de l'interaction entre l'enseignant et l'apprenant pour optimiser le transfert et l'acquisition du savoir, savoir-faire et des pratiques : elle dépasse le cadre théorique. En fait, elle est d'une importance capitale. L'évaluation doit être au service de la formation et non l'inverse.

Cependant, l'évaluation est fortement teintée par la personnalité de l'évaluateur engendrant, ce qui est connu en didactique par, la subjectivité.

L'évaluation dans les EIAH 3

Sommaire

3.1 L'évaluation assistée par ordinateur 31
3.2 Pourquoi évaluer dans les EIAH ? 32
3.3 L'évaluation face à la standardisation 34
3.4 Les qualités d'un outil d'évaluation 35
Conclusion 35

Avec l'apparition des nouvelles technologies, une des solutions pour réduire la subjectivité et rendre l'évaluation plus constructive est d'introduire l'évaluation automatisée. L'évaluation assisté par ordinateur est en ce sens une extension des méthodes et techniques traditionnelles d'évaluation. Les évaluations ne sont pas utiles uniquement pour mesurer les progrès réalisés par les élèves, elles peuvent aussi servir à faciliter leurs progrès à condition de pouvoir être utilisées par les enseignants comme un outil pédagogique à part entier. Aussi, des techniques en EIAH apparaissent pour réduire les divergences de notation afin de rendre les évaluations certificatives plus objectives, authentique et équitables.

3.1 L'évaluation assistée par ordinateur

Avec l'apparition des nouvelles technologies de l'information de la communication (NTIC) et leur utilisation en éducation, des méthodes d'évaluation sont implémentées avec des supports informatiques. L'évaluation dans les EIAH se retrouve face à un problème particulier. Un problème académique, social, administratif et même moral. Le besoin spécifique pour la liberté en termes de temps et de place rajoutent d'autres contraintes.

L'évaluation est utilisée essentiellement pour noter, classer et certifier les apprenants d'une part, et pour choisir les méthodes pédagogiques les plus appropriées d'autre part. Les EIAH sont maintenant largement utilisés comme un support pédagogique supplémentaire. Comme les environnements d'apprentissage continuent à développer des méthodes et des approches d'apprentissage soutenues par les nouvelles technologies, l'évaluation n'a pas eu beaucoup de développement par rapport à ses pairs à savoir : la scénarisation des cours, etc . Elle est devenue un sujet de discussion dans les communautés de recherche.

Selon Vendlinski and Stevens (2002), les nouvelles technologies de l'information et de communication offrent des nouvelles mesures pour évaluer les apprenants et permettent aux enseignants de mieux comprendre le processus d'apprentissage et de produire des données efficaces sur les apprenants.

Tout élève qui connaît sur les bouts des doigts son système éducatif sait que, s'il est facile d'éviter un mauvais cours, il est en revanche plus difficile d'éviter une mauvaise évaluation (Kalz et al. 2009). Dans le domaine de l'enseignement à distance, on évalue (ou simplement on vérifie) les cours par le biais de questionnaires dont les retours sont plus ou moins immédiats. C'est un moyen courant et maintenant bien géré par de nombreuses plates-formes d'enseignement à distance (exemple : Moodle). Il existe même des standards, permettant de transférer aisément d'une plate-forme à une autre les questionnaires réalisés (comme SCORM [1] : Shareable content object reference model, IMS QTI [2] : Question and test interoperability cf. section 3.3, ou encore AICC [3] : Aviation industry CBT committee).

Tout enseignant passe beaucoup de temps à essayer de comprendre ce que ses élèves arrivent à comprendre à propos des cours qu'il conçoit et réalise. Une partie de cette activité se passe en direct, via l'observation des élèves, leur interrogation, la correction d'exercices. Cette charge de travail est très importante, et l'enseignant peut penser à recourir à des outils informatisés qui l'aident à obtenir de telles informations. Bien évidemment, ce type d'activités existait bien avant l'invention de l'ordinateur. On peut penser aux « boîtes enseignantes » de Freinet, où un élève pouvait sélectionner un ensemble de questions sur un domaine donné, l'insérer dans une boîte laissant successivement apparaître dans une fenêtre des questions et leurs réponses, permettant à l'élève ainsi de s'auto-évaluer.

1. http ://scorm.com/
2. http ://www.imsglobal.org/
3. http ://www.aicc.org/joomla/dev/

Depuis la démocratisation de l'utilisation de l'ordinateur et de l'internet, l'évolution de ceux-ci a permis une forte apparition de l'enseignement par ordinateur. Dès lors, l'évaluation joue un rôle particulier en EIAH, elle est généralement formatrice et formative. Selon Charles Juwah (Juwah 2003) chercheur dans le domaine des EIAH, l'évaluation doit :

- être motivante pour l'apprenant,
- encourager une activité d'apprentissage soutenue,
- contribuer à la progression de l'apprenant,
- être faible en coût humain et facilement maintenable.

L'évaluation se situe au cœur de processus d'apprentissage : comment les apprenants sont-ils évalués ? En même temps, l'évaluation constitue une tâche lourde pour les enseignants et un problème pour plusieurs institutions.

Les Environnements Informatique pour l'Apprentissage Humain essayent d'assurer l'apprentissage. Pour vérifier si le but a été atteint ou pas, les apprenants ont besoin d'effectuer des tests ou des activités d'évaluation. Contrairement à l'évaluation à la main qui est un processus couteux en temps et un lourd fardeau, un gain énorme en termes de consistance peut être obtenu si le processus de l'évaluation est supporté par un système informatique.

Le processus de l'évaluation en utilisant les nouvelles technologies est appelé aussi : évaluation assistée par ordinateur (*en anglais :* Computer Assisted Assessment -CAA-, Computer Based Assessment -CBA-, Computer Assisted Testing -CAT-) et aussi évaluation électronique (*en anglais :* eAssessment). Ces appellations sont connues beaucoup plus dans la communauté anglophone.

3.2 Pourquoi évaluer dans les EIAH ?

Au même titre que les systèmes d'enseignement assistés par ordinateur, le processus de l'évaluation lui aussi a bénéficié des atouts des nouvelles technologies :

- La méthode d'écrire et de répondre aux questions,
- Les types de questions qui peuvent être posées,
- Une grande visibilité et objectivité,

Cette évaluation assistée par ordinateur améliore la détection des progrès et lacunes des apprenants suivant la rapidité et la qualité des interactions qu'ils ont avec et dans l'EIAH Bull (1999). De même que l'évaluation joue différents rôles en EIAH, elle porte sur des objets différents. Lorsqu'elle porte sur les productions des apprenants, on parle alors d'évaluation de production. En revanche, lorsque l'évaluation porte sur la démarche mise en œuvre pour réaliser des productions, on parle alors d'évaluation de la démarche (Durand 2006).

En effet, le tuteur a besoin d'informations pertinentes non seulement sur les connaissances et les compétences de l'apprenant mais aussi, plus généralement, sur le profil cognitif de l'apprenant (Labat 2002). Cette connaissance

est indispensable si on veut parler d'individualisation et permet aussi au tuteur (ou au système) d'adapter la suite de la séquence d'apprentissage en fonction des difficultés avérées de l'apprenant.

Evaluation de production : le bilan de compétences

Le bilan de compétence consiste à prendre une "photographie" du niveau de compétence d'un apprenant en évaluant ses productions. Ce type d'évaluation sommative est mené dans des environnements d'apprentissage basés sur des banques de questionnaires à choix multiples (QCM) organisés en parcours et sanctionnant l'apprenant par une note finale Green et al. (1984). Le test du TOEFL est certainement le plus connu.

Cette évaluation de compétence a été poussée à son paroxysme avec des évaluations diagnostiques, qui ne se contentent pas de corriger les productions des apprenants mais qui proposent des diagnostics. Chaque réponse de l'apprenant est analysée pour essayer de l'associer à une erreur envisagée par les concepteurs de l'exercice et à son origine potentielle. Le logiciel Pépite (Delozanne and Grugeon 2004) et ses dérivés illustrent cette évaluation diagnostique.

Évaluation de production : l'auto-évaluation

Dans certains cas, il peut être utile de proposer à l'apprenant de s'auto-évaluer afin de favoriser l'auto-régulation de ses apprentissages en s'insérant dans le cadre d'une évaluation formative. Le logiciel GenEval (David 2003) propose une liste de questions, dans lesquelles l'apprenant peut naviguer. Pour chaque question, l'apprenant peut bénéficier ou non d'indications pour répondre. Une fois la réponse donnée et corrigée, l'apprenant évalue sa maîtrise des compétences mises en jeu dans les questions en se donnant une note.

Évaluation de la démarche : l'assistance à l'évaluation

S'il est possible d'évaluer relativement finement une production, il n'en est pas de même pour une démarche. Aussi, que ce soit dans des activités individuelles ou collectives, des solutions d'assistance à l'évaluation sont proposées. Elles consistent bien souvent à fournir à l'enseignant des vues sur les activités par le biais d'indicateurs, de mesures Merceron and Yacef (2004). Ces mesures permettent à l'enseignant d'évaluer le déroulement de l'activité.

Les apprenants apprennent mieux quand :

- Ils sont engagés dans le processus d'apprentissage,
- Leurs connaissances et compétences qui existent sont mises en jeu,
- Ils relèvent des défis pédagogiques,
- Ils ont des opportunités pour dialoguer avec le tuteur et leurs pairs,
- Ils ont l'opportunité d'une évaluation formative,
- Ils reçoivent un feedback efficace dans le moment opportun,
- Ils comprennent les critères de l'évaluation,
- Ils ont la chance pour s'auto-évaluer et de réfléchir.

Les enseignants enseignent mieux quand :

- Ils ont accès aux résultats de l'évaluation formative,
- Quand les méthodes d'évaluation captent l'attention des apprenants,
- Les apprenants réagissent face aux feedback de l'enseignant et de leurs pairs.

3.3 L'évaluation face à la standardisation

IMS Question and Test Interoperability (IMS-QTI) (IMS Global Learning Consortium 2005) est issu des travaux de l'organisation IMS Global Learning Consortium dont le but est de promouvoir le développement de l'apprentissage collaboratif et coopératif à distance. En s'appuyant sur le langage XML, IMS-QTI spécifie un modèle de représentation de questions (assessment Item) et de questionnaires (assessment). Une première version de IMS-QTI a été développée dès 1999 et rendue publique en 2000.

Que ce soit dans l'expression des contenus ou le scénario des activités pédagogiques, l'évaluation devrait être largement présente. Or, IMS-QTI qui initialement devait être un format d'évaluation a manqué son objectif. IMS-QTI décrit finement les contenus mais ne propose que des préconisations pour leur évaluation.

En outre, le paradigme d'évaluation proposé est pauvre (évaluation individuelle de compétence) au regard des situations rencontrées en EIAH. Certes, il n'est pas du ressort d'un formalisme d'expression de contenu pédagogique de décrire son usage.

IMS-QTI est basé sur le modèle des quatre processus Russell et al. (2003), qui organise le cycle de vie d'un test en quatre processus distincts :

- le processus de sélection du questionnaire dans une banque de questionnaires,
- le processus de création de questionnaires avec des outils auteurs ;
- le processus de soumission du questionnaire aux apprenants dans l'ENT [4] offert par un service en charge de présenter les questionnaires aux apprenants ;
- et enfin, le processus de correction des réponses des apprenants, réalisé par le moteur d'exécution de questions.

Par ailleurs, IMS-QTI intègre une vision sommative de l'évaluation. Certes le créateur de questionnaires est libre de créer ses questionnaires et de définir un barème de correction, mais il n'est pas libre de choisir la manière dont il veut évaluer le questionnaire. Il n'est pas possible de proposer aux élèves de s'auto-évaluer, cette forme d'évaluation n'est pas supportée par les moteurs d'exécution de questionnaires.

En conclusion, l'évaluation pratiquée en EIAH est quasi absente de l'effort de formalisation et de standardisation actuel (Durand 2006).

4. Espaces Numériques de Travail

3.4 Les qualités d'un outil d'évaluation

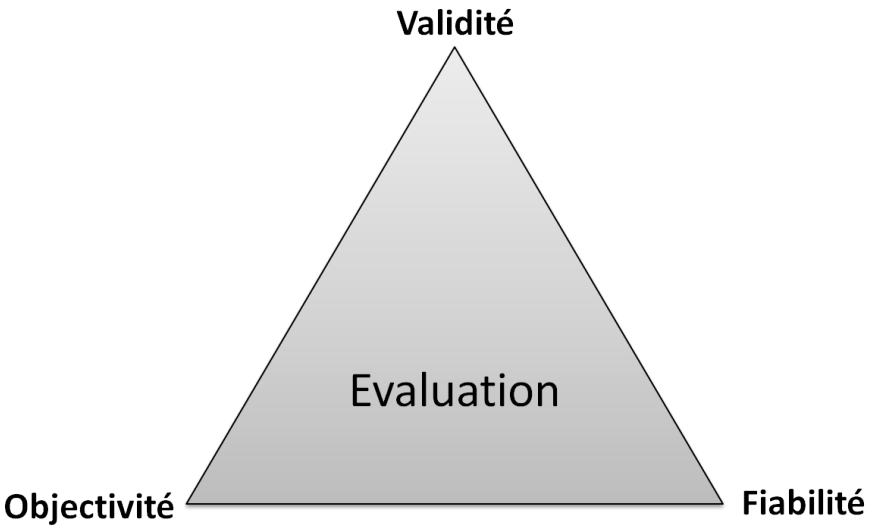

Figure 3.1 – *Critères d'efficacité des évaluations.*

Validité : Degré de précision avec lequel l'instrument mesure ce qu'il a pour objet de mesurer.

Fiabilité : Constance avec laquelle un instrument mesure une véritable donnée.

Objectivité : Degré de concordance entre les jugements portés par les examinateurs indépendants et compétents sur ce qui constitue une bonne réponse pour chacun des éléments d'un instrument de mesure.

Autre qualités :

- Pertinence,
- Clarté,
- Reproductible,
- Équité.

Un des problèmes qui peut apparaitre dans l'utilisation des systèmes informatiques pour évaluer les apprenants (évaluation formative ou sommative soit elle) est que le résultat peut être affecté par le fait que les apprenants soient technophobes.

Conclusion

L'évaluation automatisée dans les EIAH est un ensemble de méthodes et de pratiques qui utilisent les différentes technologies de l'information et de la communication. Ces méthodes sont principalement calquées sur celles mises en oeuvre par le correcteur humain.

Le processus de l'évaluation n'est pas utile uniquement pour quantifier les acquis des apprenants. Il peut servir aussi à faciliter ces progrès à condition de l'utiliser comme un outil pédagogique à part entier.

Pour que les enseignants aient confiance dans l'évaluation fournie par un système, il faut que tous les éléments de l'évaluation soient présentés d'une manière claire et précise à savoir : les critères sur lesquels l'évaluation s'appuie, le type des résultats obtenus, les référents, les référés, etc.

Le domaine d'application : L'algorithmique

4

Sommaire

4.1 Introduction 38
4.2 Etude épistémologique de l'algorithmique 38
4.2.1 L'algorithmique et la conception d'algorithme 39
4.2.2 Les différents aspects de l'algorithmique 40
4.2.3 Réflexion algorithmique et compétences en résolution de problèmes 42
4.3 Les débutants en algorithmique 42
4.3.1 Difficultés de l'algorithmique 42
4.3.2 Comportement des débutants en algorithmique 44
4.4 Systèmes dédiés à l'algorithmique 45
4.4.1 Systèmes d'apprentissage de l'algorithmique 45
4.4.2 Systèmes d'évaluation intégrant une vision sommative . . . 50
4.5 Synthèse 52
4.5.1 Les techniques d'évaluation généralement utilisées en algorithmique/programmation 54
4.5.2 Synopsis des systèmes 56
Conclusion 58

4.1 Introduction

L'algorithmique est une discipline qui favorise, notamment, l'acquisition d'une méthode de travail et de réflexion, le développement des facultés d'analyse, d'abstraction, d'anticipation et de logique Muldner and Shakshuki (2003). Elle est le noyau de toute formation d'informaticien.

Dans une formation, elle est également une matière qui a souvent été source de problèmes pour l'enseignant et l'étudiant. Pour l'enseignant, parce qu'il doit trouver les méthodes adéquates pour faire assimiler des concepts assez abstraits à des étudiants qui ne sont qu'à leur phase d'initiation. Pour l'étudiant, l'algorithmique, contrairement à d'autres sciences comme la physique, n'offre pas au débutant un modèle "naïf" viable de l'ordinateur, qui pourrait lui servir comme base pour construire des modèles plus sophistiqués. A l'inverse, son expérience avec celui-ci semble favoriser une modélisation "anthropomorphique", qui ne lui permet pas de comprendre le retour d'erreur brutal auquel il est confronté aux débuts de sa pratique de l'algorithmique (Caignaert 1988).

Cette situation a poussé plusieurs spécialistes à étudier et essayer de comprendre les raisons d'une telle situation (Soloway et al. 1983, McCracken et al. 2001, Lister et al. 2004, Lahtinen et al. 2005). Il en est ressorti que les difficultés sont liées principalement à la nature même de la matière et se situent principalement au niveau de l'abstraction de ses concepts. Cette abstraction empêche les apprenants de s'appuyer sur des modèles réels pour les assimiler. A cela, il est important de rajouter que, souvent, l'enseignant percevant mal ce problème agrandit alors, de par sa façon d'enseigner la matière, le fossé séparant l'apprenant de la matière.

Dans les sections suivantes, nous allons voir les différentes caractéristiques de l'algorithmique et de la programmation.

4.2 Etude épistémologique de l'algorithmique

Il faut bien rappeler que durant ce manuscrit on parle de l'algorithmique procédurale (ou dite fonctionnelle) et non de l'algorithmique déclaratif.

L'algorithmique constitue le cœur du domaine de l'informatique. Malgré la simplicité de sa définition : « l'algorithmique permet de préparer la résolution d'un problème en détaillant les opérations élémentaires à accomplir. Il reste ensuite à traduire ces opérations en instructions pour l'ordinateur » ; cette dernière est connue par sa difficulté pour l'appréhender et de l'enseigner aux débutants. Le choix des opérations élémentaires pour résoudre un problème donné dépend du problème lui-même et de la précision souhaitée pour la description de sa résolution. Donc la problématique principale se résume dans le fait de savoir comment utiliser ces opérations élémentaires, les combiner, pour faire sortir la solution, c'est le savoir-faire de la résolution. Des études confirment qu'il faut 10 ans d'expérience pour qu'un novice devienne expert (Winslow 1996).

A travers cette étude synthétique, nous allons essayer de relever tous

les aspects liés à l'algorithmique à savoir : les différents comportements des apprenants, les difficultés, etc.

4.2.1 L'algorithmique et la conception d'algorithme

L'algorithmique est la science qui étudie l'application des algorithmes à l'informatique tandis qu'un algorithme est défini comme la transformation de la connaissance que l'humain possède sur un problème (i.e. description des étapes de la méthode utilisée) en actions élémentaires exécutées par l'ordinateur. D'une autre manière, l'algorithmique est un ensemble de règles opératoires dont l'application permet de résoudre un problème au moyen de nombre fini d'opérations (ou d'actions).

Pour un problème donné à résoudre, la méthodologie la plus standard et la plus connue est la suivante :

(1) Analyse du problème,

(2) Conception d'une solution algorithmique et le choix de la représentation des données,

(3) Développement : programmation et choix du langage de programmation,

(4) Tests.

Si le problème est trop complexe, réaliser une analyse méthodique descendante :

- Décomposer en sous-problèmes,
- Appliquer, pour chacun, la méthodologie simple.

Décomposer un problème revient à en fournir une description de plus en plus détaillée jusqu'à obtenir un ensemble d'opérations élémentaires traductibles en langage de programmation.

L'algorithmique ne doit pas être confondu avec la programmation proprement dite, comme il est bien montré dans la méthodologie de résolution d'un problème. Un programme est la réalisation (l'implémentation) d'un algorithme au moyen d'un langage donné. Il s'agit de la mise en œuvre du principe. Par exemple, lors de la programmation on s'occupera parfois explicitement de la gestion de la mémoire (allocation dynamique en C) qui est un problème d'implémentation ignoré au niveau algorithmique.

Autrement dit, un algorithme est indépendant du langage de programmation dans lequel on va l'exprimer et de l'ordinateur utilisé pour le faire tourner. C'est une description abstraite des étapes conduisant à la solution d'un problème. Un algorithme est une partie conceptuelle d'un programme (indépendante du langage) tandis qu'un programme est une implémentation (réalisation) de l'algorithme, dans un langage de programmation donné, sur une architecture spécifique et sur un système particulier.

La plupart des environnements qui existent dont le but est l'apprentissage de l'algorithmique ne distingue pas la phase conception de l'algorithme du programme (Lane et al. 2005). La phase de la conception d'algorithme

est très importante pour réussir un programme car elle dessine la solution, si cette solution n'est pas bien conçue dès le départ, le programme va inévitablement être erroné.

Concevoir et établir un algorithme pour un problème donné est très important. L'un des avantages de cette étape permet d'examiner l'algorithme établi d'une manière rationnelle. L'identification des actions à faire pour résoudre un problème donné réduit la tâche à une série de sous problèmes plus faciles à gérer. Un problème qui serait difficile ou impossible à résoudre d'un seul bloc, peut être considéré comme un ensemble de sous problèmes résolvables.

En utilisant un algorithme, le processus de résolution devient plus efficace et consistant. Il devient un outil mnémonique qui aide à s'assurer que les variables ou des parties du problème ne sont pas ignorées. Présentant la solution sous forme algorithmique assure une communication meilleure et précise.

Dans le cas où l'apprenant ne sait pas ce qu'il a fait, il ne va pas donc savoir ce qu'il a commis comme erreur et à quel niveau. L'existence d'une solution algorithmique permet donc l'identification des points faibles et des erreurs dans la solution. La réduction de la tâche de résolution à une suite d'étape formant l'algorithme est important dans l'analyse, le contrôle et l'évaluation.

4.2.2 Les différents aspects de l'algorithmique

Il est remarqué chez les débutants des déficits de compréhension des notions d'algorithmique (Soloway and Spohrer 1988). Par exemple, l'initialisation d'une variable est plus difficile à appréhender qu'une mise à jour ou un test d'une variable. Ainsi, des bugs au niveau des boucles, des tests et des actions liés à ces derniers comme l'initialisation d'une variable d'une boucle 'For' sont fréquents chez les apprenants.

Milne and Rowe (2004) montrent que les pointeurs aussi constituent un obstacle pour les apprenants d'une part, et pour les enseignants de l'autre part. Pour cette raison, il est souhaitable d'introduire la notion d'allocation dynamique de mémoire, le passage de paramètres (par référence, par valeurs) et d'autres notions sur la mémoire. D'autres problèmes liés aux caractéristiques du langage de programmation existent déjà par centaines, avec de nombreuses variations et ils font l'objet de recherches très actives.

Cependant, la véritable difficulté ne dépend pas de la syntaxe ou la compréhension des concepts, mais de la constitution d'un plan de la solution. Il est nécessaire de distinguer entre les connaissances des concepts de programmation et les stratégies de programmation. Un apprenant peut comprendre un concept (exemple : qu'est ce qu'un pointeur ?), mais il reste incapable de l'utiliser dans un programme. Winslow (1996) indique que les apprenants peuvent connaitre la syntaxe et la sémantique des différents concepts, mais ils ne connaissent pas comment les combiner pour avoir un programme valide. Même quand ils savent résoudre un problème, ils trouvent des problèmes pour le convertir à un code source.

Aujourd'hui, la plupart des universités utilisent pour l'apprentissage de la programmation des outils similaires à ceux utilisés dans les industries. Ces outils ne sont pas basés sur des démarches pédagogiques bien spécifiques.

Weigend (2006) montre à travers une expérience faite au niveau des écoles l'existence des connaissances intuitives sans savoir même le domaine de l'informatique. Ce papier exhibe quatre obstacles qui apparaissent lors du passage de l'intuition à la programmation.

Difficulté de passage de l'intuition à l'algorithme

Lors de la résolution d'un problème donné, généralement, une description de la solution proposée est faite avec le langage naturel (ex : ce programme fonctionne comme une ruche d'abeilles, aller chercher du pollen et retourner vers la ruche). Pour passer à implémenter une telle idée, il faut casser ce *Gestalt*, en le rendant plus spécifique et plus concret.

Spohrer et al. (1985) modélise le processus de résolution des problèmes dans l'algorithmique en utilisant GAP-trees (Goal-Plan-trees). La racine de l'arbre est un plan qui représente l'idée sur la solution de problème. Un ensemble de sous-buts constitue le plan et tous ces sous-buts doivent être atteints pour réaliser ce plan. Les chercheurs ont analysé des bugs sémantiques sur 46 programmes (produits par des novices) qui sont syntaxiquement correctes. Deux observations sont faites. La première, 103 sur 549 erreurs sont dues au non contrôle des données en entrées. L'explication donnée est que les novices essayent de résoudre une version simple du problème puis ils pensent à l'étendre progressivement. La deuxième observation est que les apprenants mélangent les objectifs du problème à résoudre en utilisant un seul plan pour atteindre plusieurs buts d'une manière intégrée. Par conséquent, l'utilisation d'un seul plan pour atteindre plusieurs objectifs rend le problème beaucoup plus compliqué, ce qui génère beaucoup de bugs.

Difficulté d'implémentation

Il se peut qu'une intuition soit inappropriée pour l'implémenter. Pour être spécifique, elle peut inclure des aspects que l'apprenant n'est pas capable d'implémenter (il ne connait pas le concept approprié).

Manque de connections entre les intuitions et le programme

Cette difficulté se traduit par l'inexistence des connections mentales entre les modèles intuitifs et les programmes correspondants. Beaucoup d'apprenants échouent lors de l'écriture d'un programme de tri d'un tableau d'entiers même s'ils connaissent les différents concepts de programmation (condition et boucle). Ils maitrisent bien la solution mais sont incapables de l'exprimer en utilisant un langage de programmation.

Erreurs de conception

Cette difficulté est rencontrée lors de l'utilisation des solutions d'autres problèmes comme des modèles pour résoudre un nouveau problème donné avec quelques modifications, sans avoir bien compris le fonctionnement de ces modèles.

4.2.3 Réflexion algorithmique et compétences en résolution de problèmes

La réflexion algorithmique est l'ensemble des compétences qui sont connectées pour construire et comprendre des algorithmes :

- Compétence d'analyse des problèmes,
- Compétence de la spécification précise des problèmes,
- Compétence de trouver les actions basiques qui sont adéquates pour la résolution d'un problème donné,
- Compétence de construire un algorithme correct pour un problème donné,
- Compétence de réflexion sur tous les cas normaux et spéciaux possibles du problème,
- Compétence d'amélioration de l'efficacité d'un algorithme.

Il faut noter que ces compétences sont nécessaires pour construire un bon algorithme. Ce sont les plus importants ingrédients dans le domaine de l'algorithmique.

Savoir programmer c'est être capable d'analyser un problème de petite taille et d'en formuler une solution algorithmique, de représenter cette solution aussi bien en pseudo code qu'au moyen d'un morphogramme afin de traduire cette solution dans un langage informatique en respectant les standards et en utilisant un outil de développement intégré.

Par conséquence, cette misconception commise au début de la résolution d'un problème en algorithmique engendre une variété de problèmes :

- Une phase de codage assez compliquée,
- Des étudiants débordés et démotivés,
- Une surcharge cognitive,
- La phase de codage de l'algorithme devient une tâche laborieuse.

4.3 Les débutants en algorithmique

4.3.1 Difficultés de l'algorithmique

L'importance de l'algorithmique, noyau de l'informatique, est proportionnelle à la difficulté de son acquisition. Ce phénomène, difficulté de l'algorithmique, n'est pas nouveau. Soloway et al. (1983) confirment que 38% des étudiants en algorithmique peuvent écrire un programme qui calcule une moyenne. McCracken Group (McCracken et al. 2001), Bootstrapping (Petre et al. 2003), Leeds Group (Lister et al. 2004), Scaffolding Fincher et al. (2004) et BRACE Fincher et al. (2005) et les impressions aussi bien des élèves que des enseignants soulignent toutes les difficultés à apprendre ou à faire apprendre l'algorithmique. En effet, un taux de réussite ne dépasse pas 25% de par le monde.

Résoudre un problème en algorithmique relève d'une double problématique :

Les compétences en résolution de problèmes	Activité d'apprentissage
Compréhension de problème	Reformulation de problème en termes de état initial, hypothèse et contrainte
Décomposition du problème	Identifier, nommer et lister les sous tâches
Raisonnement par analogie	Identifier les similarités entre les problèmes.Distinguer entre similarité structurelle et similarité superficielle. Identifier les erreurs communes entre les problèmes
Généralisation et abstraction	Extraire les prototypes du problème dans des contextes différents
Identifier le prototype du problème	Catégorisation du problème. Identification de la structure de l'algorithme.
Identifier la structure du problème	Identifier la relation entre les sous tâches. Schématiser la structure du problème en utilisant les diagrammes
Evaluation et appréciation	Comparer les solutions en tenant compte de l'efficacité.
Réflexion et conclusion	Tenir compte de la stratégie de résolution
Verbalisation des idées	Formulation précise de l'idée . Différenciation entre l'idée et son implémentation

TABLE 4.1 – *Les différentes compétences en résolution de problèmes et les activités d'apprentissage liées en algorithmique.*

1. Trouver une méthode de résolution (exacte ou approchée) du problème,
2. Trouver une méthode efficace (savoir résoudre un problème est une chose, le résoudre efficacement en est une autre).

Ces difficultés sont traduites principalement par la nature même de la matière ou comme l'appellent les didacticiens "*le flou didactique du domaine*", qui se situe au niveau de l'abstraction des concepts. Ainsi, la manière dont les enseignants appréhendent ses fondements est généralement floue. Par conséquent, les apprenants ne trouvent pas des modèles réels pour assimiler ses concepts.

L'algorithmique joue un rôle important en informatique et elle est définie comme étant un ensemble d'instructions complètes et consistantes pour résoudre un problème. À cause de cette importance, beaucoup de chercheurs ont essayé de trouver la meilleure façon d'apprendre et d'enseigner l'algorithmique (Muldner and Shakshuki 2003). Des enseignants chevronnés dans différentes universités, ont été depuis longtemps, malgré leur expérience, confrontés aux difficultés de leurs étudiants face à cette matière. Kaasboll (2002) confirme que le taux d'échec ou d'abandon aux cours d'initiation à la programmation, en premier cycle universitaire, varie de 25% à 80% de par le monde.

Les apprenants en algorithmique, pendant leur formation, se retrouvent face à plusieurs choses à apprendre : l'environnement de développement,

la syntaxe et la sémantique du langage de programmation, les concepts de programmation et aussi les paradigmes spécifiques (ex : orienté objet). Ils ont aussi besoin de compétences qui sont primordiales en algorithmique mais qui sont fréquemment négligées : des compétences en résolution de problèmes et les stratégies dans le but de résoudre efficacement un problème donné. Généralement, les apprenants ont des difficultés dans la formulation d'une solution, la reconnaissance des similarités parmi les problèmes, et l'identification des sous tâches dans un problème complexe d'où l'échec et l'inconsistance des algorithmes construits (Benedict 1986, Joni and Soloway 1986, Robins et al. 2003).

Robins et al. (2003) ont montré en étudiant les problèmes d'apprentissage et d'enseignement de l'algorithmique que la différence entre un bon et un mauvais étudiant en algorithmique est liée à la stratégie employée lors de la résolution. Bien plus clair, ils ont du mal à regrouper les solutions des sous problèmes pour exprimer la solution générale du problème. Sandy et al. (2005) indiquent aussi que le problème de la conception d'un programme vient en second lieu juste après les problèmes de syntaxe.

4.3.2 Comportement des débutants en algorithmique

Il est connu que les débutants n'ont pas les connaissances spécifiques et les compétences des experts. Plusieurs études menées par Winslow (1996) affirment que les débutants sont limités à des connaissances superficielles (ces connaissances sont organisées sur des similarités superficielles), plus, l'absence des modèles mentaux détaillés, l'incapacité de l'application des connaissances pertinentes, l'utilisation des stratégies de résolution de problème très générales. Contrairement aux experts, les débutants passent peu de temps dans la planification de la solution d'un problème et aussi dans le test du code et peu de tentatives de reformulation de programme.

Des problèmes aussi sont remarqués au niveau de la compréhension de l'exécution d'un programme et surtout au niveau de la séquentialité des actions où parfois les débutants oublient que certaines instructions dépendent des instructions précédentes.

Un nombre important d'études ont conclu que les débutants connaissent la syntaxe et la sémantique des instructions individuellement mais ils ne sont pas capables de les combiner pour l'obtention d'un programme valide. Même quand ils savent résoudre le problème par une description avec le langage naturel, ils trouvent des obstacles pour le traduire à un programme (codage).

Ginat (2006) traite plus particulièrement le comportement des apprenants face à des problèmes algorithmiques. L'expérience est faite sur des exemples bien précis. L'auteur à travers cette étude montre trois modèles de comportement :

- Sous structure locale : la pensée des apprenants sur la solution est partielle, c.à.d. ils proposent des solutions qui ne traitent pas tous les cas possibles,

- Optimisation du programme : signifie le besoin de l'optimisation et son danger,
- Modèle inapproprié : signifie l'utilisation des modèles (programmes connus ou des actions stéréotypés) dans des situations inadaptées.

Examinant les comportements des novices avec le modèle cognitive de Schoenfeld pour la résolution des problèmes, l'auteur note une inadéquation sur le contrôle et les croyances des apprenants sur un problème donné. Pour le contrôle, les apprenants montrent rapidement leurs convictions sur l'inexistence des caractéristiques en se basant sur des données locales et partielles. En plus, ils n'ont pas cherché à présenter ni une preuve ni des justifications. Pour les croyances, il parait que les apprenants se comportent naturellement face un problème d'une manière inconsciente sur l'effet de l'erreur et le rôle important des solutions alternatives.

4.4 Systèmes dédiés à l'algorithmique

Un nombre considérable d'outils destinés à aider les novices à apprendre la programmation ont été développés, e.g. Jeroo (Sanders and Dorn 2003), Alice (Moskal et al. 2004), JPie (Goldman 2004), Iconic Programmer (Chen 2005). L'ensemble de ces outils ont évité l'algorithmique pour se consacrer à la programmation. Ils sont destinés à l'apprentissage de la programmation et non pas à l'algorithmique (Smith et al. 2000, Ben-Ari 2001) et ils dépendent généralement d'un langage de programmation et la validation des solutions se fait à base de solutions stéréotypées.

Pour cet état de l'art, nous avons choisi des systèmes qui ont pour objet l'évaluation des productions des apprenants en algorithmique. Il ne faut pas confondre entre l'évaluation d'un algorithme pour le valider et l'évaluation d'une solution algorithmique produite par un apprenant afin de connaitre ses compétences en matière de résolution de problèmes algorithmiques qui est l'objectif de cette thèse.

Nous avons donc sélectionné quelques outils à visée pédagogique et qui sont classés selon deux catégories et selon leurs objectifs : (1) des systèmes dédiés à l'apprentissage de l'algorithmique et de la programmation et qui intègrent un processus d'évaluation généralement formative et qui sont généralement des Systèmes Tuteurs Intelligents et (2) les systèmes dédiés à l'évaluation des productions des apprenants en algorithmique et programmation qui sont purement des outils d'évaluation.

4.4.1 Systèmes d'apprentissage de l'algorithmique

Bridge

L'un des premiers systèmes qui distinguent la conception d'algorithme était Bridge (Bonar and Cunningham 1988). Pendant l'utilisation de ce système, l'étudiant entre via un menu, une méta-solution du problème exprimée en langage naturel. Avec un support d'aide à tous les niveaux de résolution,

cette solution est itérativement écrite sous une forme de plus en plus détaillée donnant à la fin un programme Pascal. La réaction des étudiants était positive mais il n'y a pas eu une conclusion sur les atouts pédagogiques vu que l'utilisation de Bridge n'a jamais été publiée. Il est possible que l'utilisation des menus au lieu du langage naturel ne contribue pas dans la compréhension de l'objectif visé par ces derniers.

Proust

Dans le but d'aider les apprenants à corriger leurs programmes, Johnson développe PROUST pour fournir un feedback sur ce que les apprenants construisent comme programmes (Johnson 1990). Ce système utilise une bibliothèque de plans de programmes pour construire la décomposition du but du problème, puis générer un arbre qui représente la solution. Après, le programme de l'apprenant est comparé avec la solution générée pour déterminer si elle est juste ou pas. Quand deux programmes se diffèrent, la différence est calculée sur les opérateurs des plans pour aider le système à caractériser ce qui est erroné et fournir un feedback.

PROUST était capable de localiser des erreurs avec un grand degré de précision (94%), cependant il n'était pas capable d'identifier des erreurs précises comme les erreurs sémantiques.

GPCeditor

GPCeditor (Goal-Plan-Code, (Guzdial 1998)) est un système qui aide les apprenants à résoudre le problème en décomposition puis composition (figure 4.1). Il fournit des outils pour identifier les buts et les plans nécessaires pour la solution, qui sont ensuite regroupés par des opérateurs comme (en anglais : 'abut' et 'nest'). L'évaluation de GPCeditor a montré que les apprenants qui ne sont pas bons produisent des programmes de qualité alors que les apprenants qui ont un bon niveau de programmation ne sont pas contents avec les restrictions ajoutées. Ainsi, un atout positif sur le plan stratégique de résolution de problème algorithmique est que les apprenants réutilisent les plans prédéfinis et appliquent des techniques pour les combiner.

SOLVEIT

Deek and McHugh (2000) a développé SolveIt comme un système complet de résolution de problème et de la programmation pour les débutants (figure 4.2). Les différentes étapes de la programmation (compréhension de problème, conception de l'algorithme, implémentation et test) sont prises en charge dans cet outil. Le système ne fournit aucune analyse intelligente de programmes des apprenants. Cependant, l'évaluation de SolveIt a révélé des différences en termes de compétences en résolution de problèmes entre les apprenants qui ont utilisé le système et le groupe de contrôle.

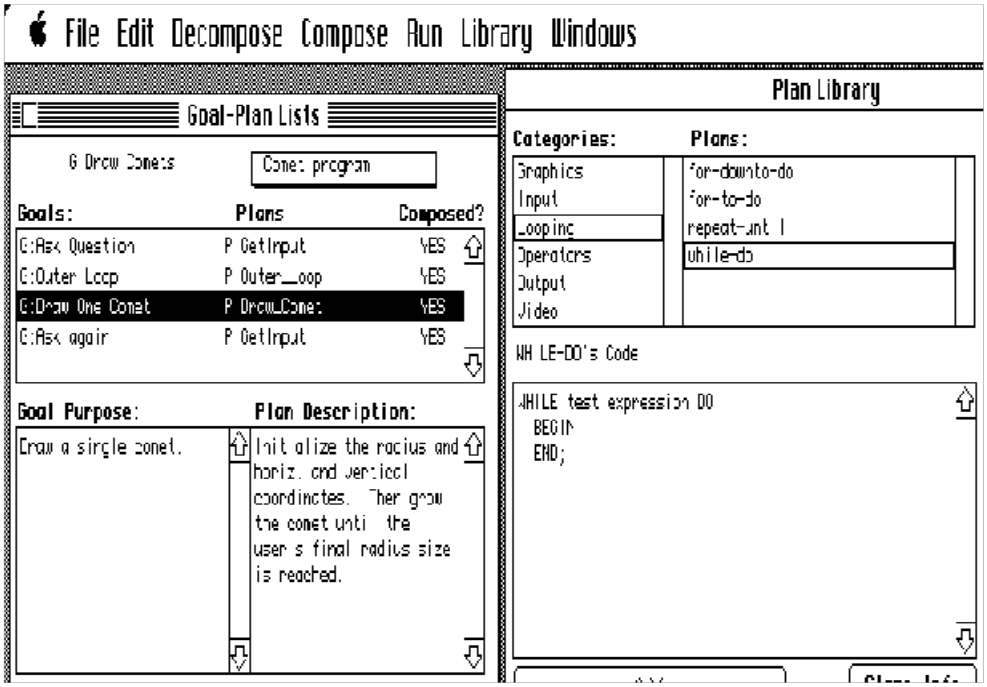

FIGURE 4.1 – *Interface de GPCeditor.*

Discover

DISCOVER (Ramadhan et al. 2001) se base sur un modèle de traçage pour traquer les apprenants. Ces derniers construisent des solutions en pseudo code sous le contrôle du tuteur et doivent suivre la solution prescrite par le modèle. Le pseudo code est construit à partir des boutons qui, lorsque l'apprenant appuie dessus, produisent des expressions prédéfinies de pseudo code qu'il doit compléter. Par conséquence, cette manière de résolution qui pousse l'apprenant à reconnaitre la solution va l'empêcher d'apprendre.

SICAS

SICAS (Marcelino et al. 2004) est un outil interactif d'organigramme basé sur un générateur de code pour les débutants en algorithmique (figure 4.3). L'idée de l'outil est d'utiliser les organigrammes pour développer des solutions visuelles des problèmes algorithmiques simples. Un code correct est généré automatiquement à partir de l'organigramme. L'animation et l'interaction entre la représentation graphique et le code généré avec, renforce la compréhension et la solution schématisée par l'organigramme et le code approprié.

SICAS dispose de deux types d'activités conception/édition d'algorithme et exécution/simulation d'algorithmes. Dans le premier cas, l'utilisateur construit un algorithme en utilisant les organigrammes (figure 4.3). Dans le second cas, l'utilisateur peut simuler et voir une version animée de l'exécution de l'algorithme.

SICAS a été expérimenté dans les classes en remplaçant la méthode papier-crayon pour construire les algorithmes. Durant la première utilisa-

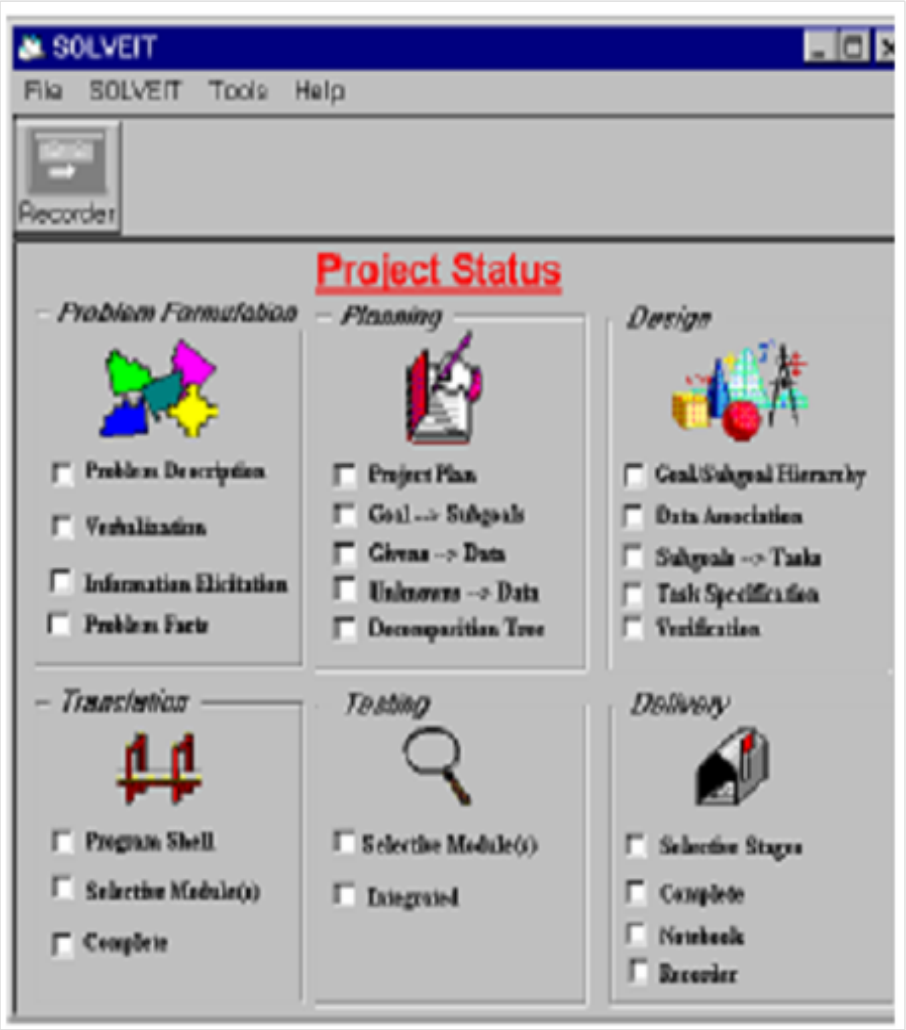

FIGURE 4.2 – *Interface de Solveit.*

tion du système, les étudiants ont utilisés principalement les menus pour sélectionner les opérations à introduire au lieu des organigrammes qui ne sont pas très sollicités. Ceci est du au fait que la plupart des étudiants sont habitués à l'utilisation du pseudo code pour exprimer les algorithmes et non les organigrammes.

Propl

PROPL (Lane et al. 2005) est considéré parmi les systèmes les plus récents dont l'objectif est d'aider les apprenants à apprendre la conception d'algorithme. PROPL essaye d'enseigner et de transmettre les compétences de conception d'algorithme. Il est basé sur les systèmes de dialogue en langage naturel et utilise plusieurs stratégies de tutorat qui renforcent la compréhension intuitive des apprenants face à un problème donné (comment doit il être résolu ?) et soulignent les concepts à implémenter. Ce système est conçu pour faire apprendre comment préconcevoir un algorithme et souligner les aspects importants de son élaboration ainsi que les connaissances

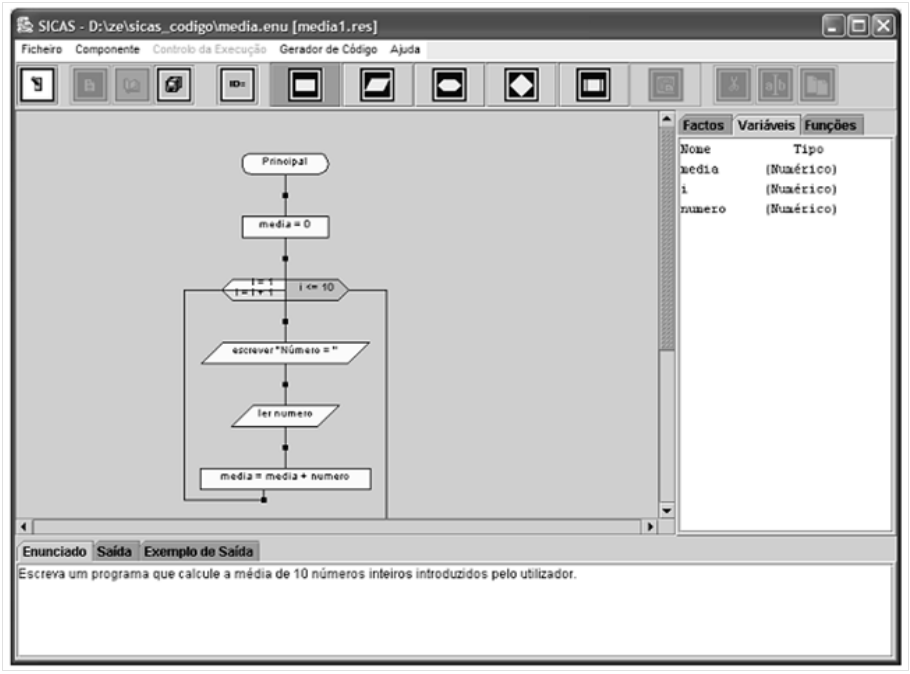

FIGURE 4.3 – *Interface de SICAS.*

tacites en exploitant les propriétés des approches de dialogue en langage naturel.

PROPL montre que les étudiants qui utilisent le dialogue (Figure 4.4) pour élaborer et planifier une solution algorithmique réalisent un remarquable progrès de leurs compétences de conception d'algorithme et de représentation abstraite des connaissances sans aucune amélioration pour l'habilité de décomposition du problème. Après, le plan construit en dialoguant se traduit automatiquement en pseudo code.

Cependant, PROPL est limité à quelques solutions stéréotypées où l'apprenant est guidé dans l'élaboration de sa solution par le dialogue, d'où l'absence de la liberté de conception d'une solution.

AlgoBox

Algobox [1] est un logiciel libre et gratuit d'initiation à la programmation qui permet d'élaborer et exécuter des algorithmes du programme de la classe de seconde en mathématiques (figure 4.6). Dès avril 2009, après la publication du projet de programme pour la seconde, Pascal Brachet, professeur de mathématiques au lycée Bernard Palissy à Agen, pense à créer un logiciel dont le cahier de charges serait le suivant :

- Facile d'utilisation pour tous les débutants (élèves et professeurs),
- Logiciel qui se concentre sur l'apprentissage des structures de base de l'algorithmique,
- Logiciel simple mais assez complet pour traiter tous les algorithmes de niveau lycée.

1. http ://www.xm1math.net/algobox/

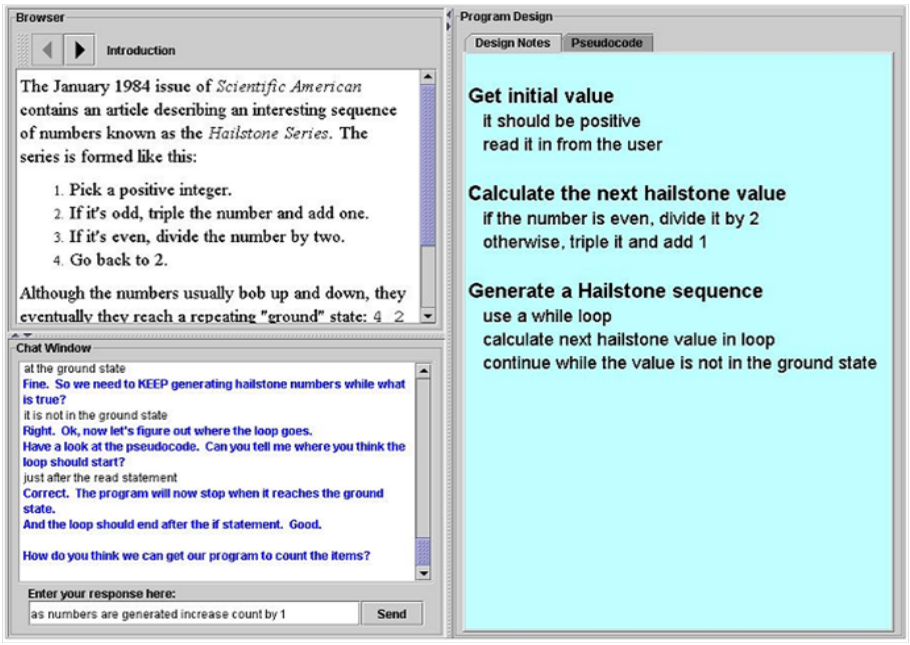

FIGURE 4.4 – *Interface principale de PROPL.*

Algobox permet à l'évidence de traiter complètement les algorithmes simples mais quelques manquements :

- Pas de sous programmes,
- Pas de récursivité,
- Écran graphique limité : Peu de couleurs, pas de gomme, ...

4.4.2 Systèmes d'évaluation intégrant une vision sommative

ASSYST

ASSYST (ASsessment SYSTem) est l'un des premiers systèmes d'évaluation automatique des programmes. Il dispose d'une interface graphique pour définir tout les aspects du processus d'évaluation dont les critères pour une évaluation automatique. Le système ASSYST développé par Jackson and Usher (1997) analyse les programmes en utilisant un nombre de critères par une comparaison des opérations du programme avec un ensemble de donnée prédéfinies.

La contribution la plus importante dans ASSYST était la compréhension qu'un système d'évaluation automatique peut aussi devenir un système de notation. Ce système fournit un mécanisme pour contrôler les programmes à évaluer, créer et générer les rapports de l'évaluation, et permettre la pondération des aspects particuliers. Une expérimentation d'ASSYST a été encourageante mais les résultats n'ont jamais été publiés.

TRAKLA 2

TRAKLA 2 (Korhonen et al. 2003) est un environnement basé web pour l'enseignement de l'algorithmique et les structures de donnée. Le système

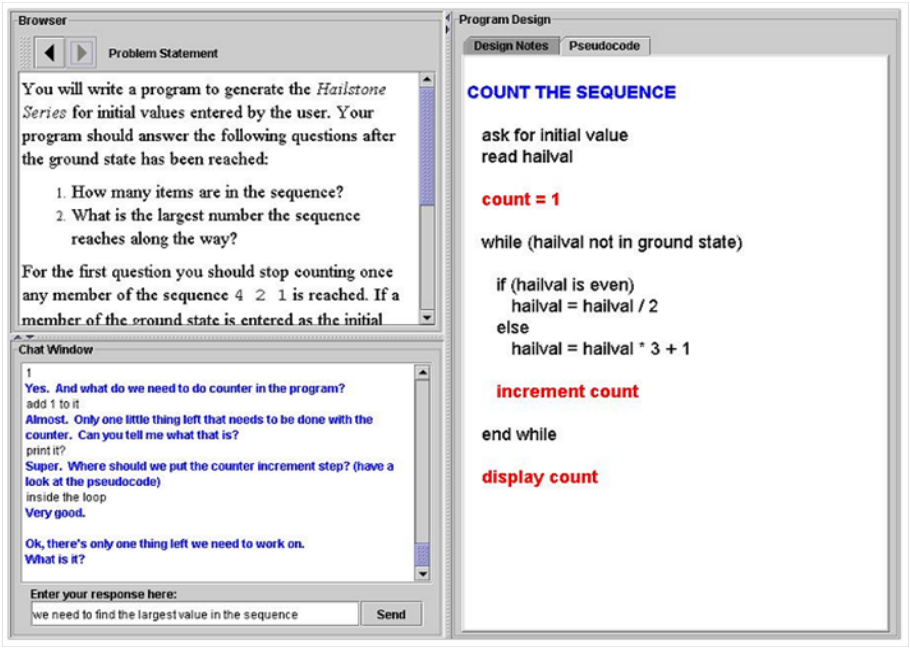

FIGURE 4.5 – *Interface de l'algorithme en pseudo code.*

permet à l'utilisateur de contrôler la manipulation de la structure de donnée à travers l'interaction GUI (Graphical User Interface) en utilisant une méthode appelée simulation visuelle des algorithmes (figure 4.7). Cette méthode permet aux étudiants de manipuler les structures de donnée pour simuler l'exécution de l'algorithme. L'outil est unique parmi les outils qui offrent la simulation des algorithmes. Il facilite l'apprentissage et fournit une évaluation sommative (notation automatique des solutions) et formative. Il fournit un feedback automatique et il permet à l'étudiant de resoumettre sa solution. Cette manière lui permet de corriger ses erreurs sur place. Les auteurs montrent dans leurs expériences sur l'utilisation de l'outil dans la classe que l'outil donne des évaluations positives.

BOSS

Le système BOSS (Joy et al. 2005) est originaire de l'université de Warwick en UK. Sa spécification initiale était similaire à celle d'ASSYST (Jackson and Usher 1997) qui était l'analyse des programmes. La première version de BOSS a été limitée à une suite de commandes faciles à utiliser. L'étudiant peut évaluer un programme pour déterminer s'il est correct et un autre programme soumis au tuteur pour l'évaluer.

Le système BOSS a continué à se développer (figure 4.8). Avec une interface utilisateur graphique pour les étudiants et les tuteurs, la dernière version de BOSS inclut aussi un composant serveur web. Ceci permet au tuteur d'évaluer les solutions en utilisant un navigateur web.

Ce système introduit aussi la notion de détection de plagiat. Il est utilisé actuellement sous deux différentes formes : une version open source qui peut être installée sur un système utilisant une plate forme Java et une implémentation spécifique à l'université Warwick.

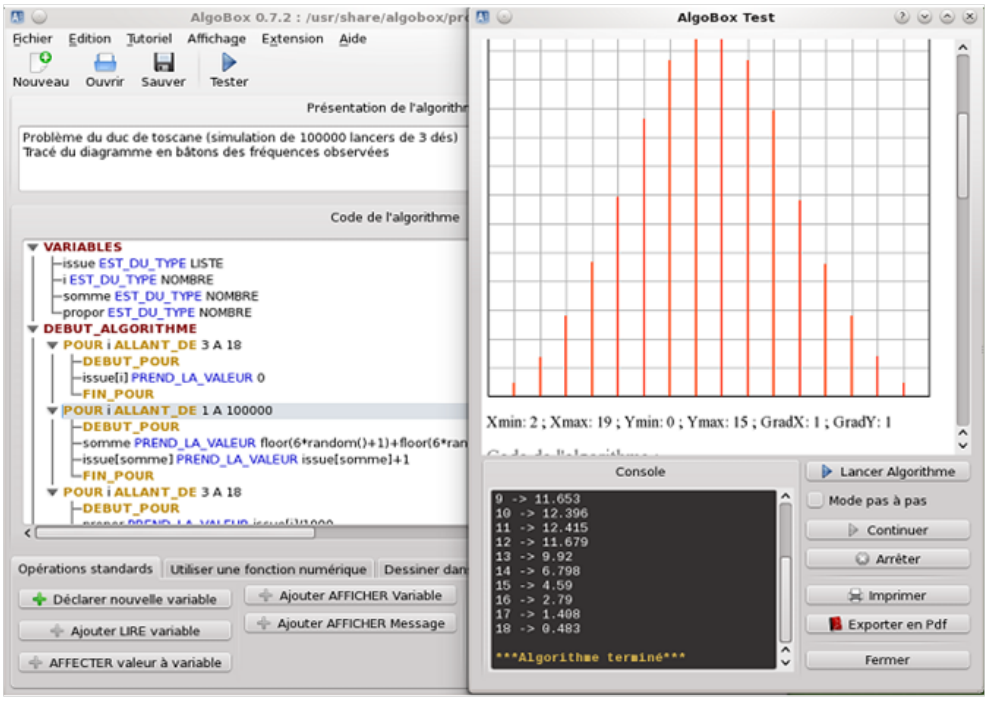

FIGURE 4.6 – *Interface principale d'AlgoBox.*

Web-CAT

Web-CAT (Edwards 2003) est un système innovant d'évaluation automatique (figure 4.9). L'approche adoptée est une approche pour enseigner le test de programme pour encourager les apprenants à pratiquer les compétences de test de programme et leurs fournir un feedback formatif.

Sa stratégie est de mettre à la disposition des apprenants une manière simple pour le développement et le test de programme et après, fournir un outil qui va évaluer les productions des apprenants automatiquement en fournissant un feedback explicatif. Cette approche a été expérimentée avec le cours des étudiants gradués. Les résultats ont été positifs avec les appréciations des étudiants. L'analyse expérimentale des programmes des étudiants montre 28% de réduction d'erreurs par mille lignes de code.

4.5 Synthèse

Notre analyse des systèmes et des outils dédiés à l'algorithmique et à la programmation nous a amenés au constat que le processus de l'évaluation est omniprésent. Malgré que nous nous intéressons à l'algorithmique et non à la programmation, la prise en compte des outils traitant la programmation a pour objectif de comprendre comment les programmes sont évalués et si l'étape de conception d'algorithmique est respectée ou non.

Les deux classes de systèmes présentés dans la section précédente ont été divisées selon deux objectifs constatés durant notre étude : la première classe de systèmes représente les systèmes qui sont dédiés à l'apprentissage de l'algorithmique et la programmation et qui ont forcément une étape d'évalua-

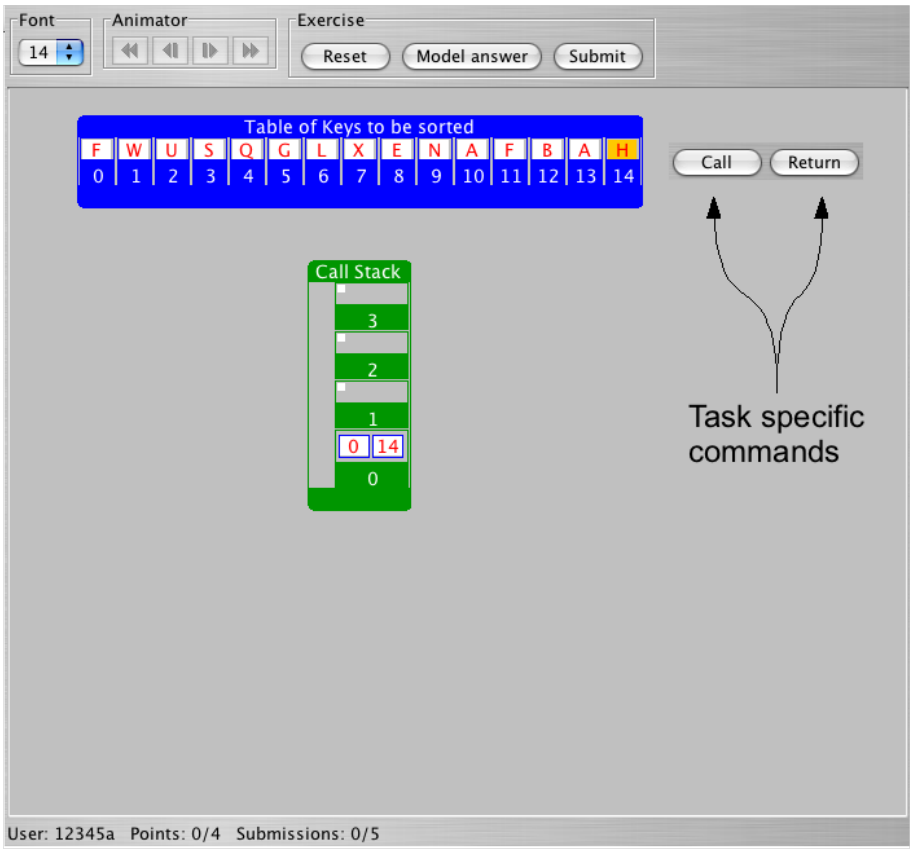

FIGURE 4.7 – *Interface TRAKLA 2.*

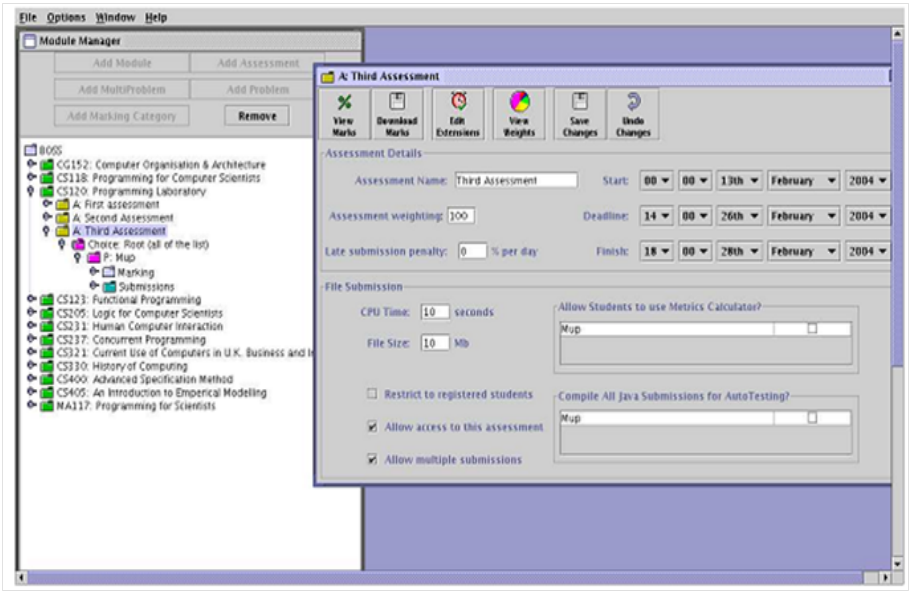

FIGURE 4.8 – *Interface BOSS.*

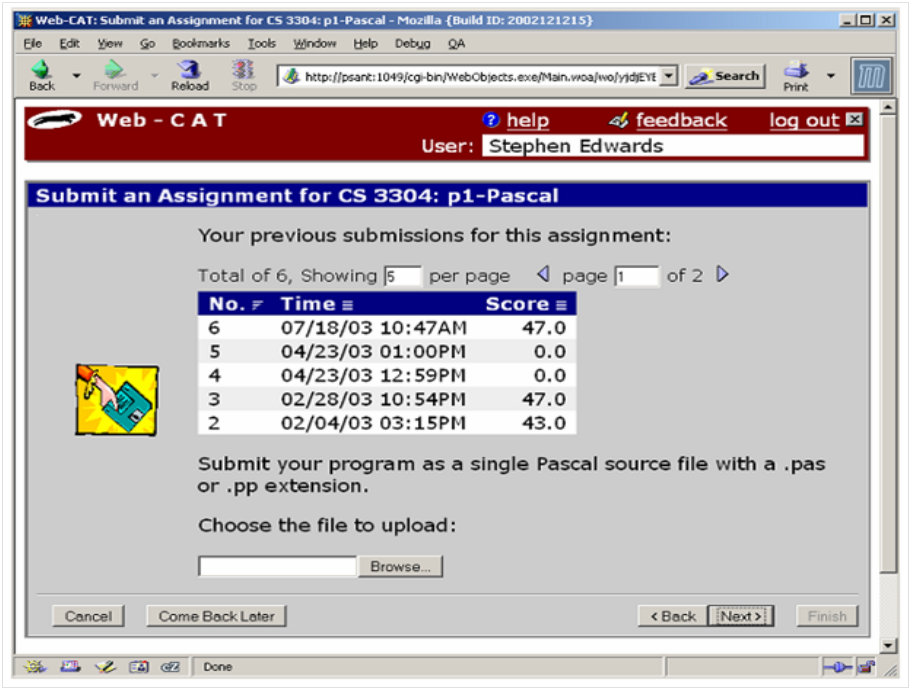

FIGURE 4.9 – *Interface Web-CAT.*

tion sur laquelle le processus d'apprentissage s'appuie. La deuxième classe regroupe les outils intégrant une vision sommative de l'évaluation et parfois une rétroaction personnalisée. Ces outils peuvent être utilisés de différentes façons. Certains enseignants les utilisent pour noter automatiquement, alors que d'autres utilisent les outils pour générer une évaluation initiale qui peut indiquer les erreurs dans le programme de l'apprenant. Les outils d'évaluation automatique (Automated assessment tools) peuvent aussi réduire la subjectivité de l'évaluation des enseignants.

Certains outils utilisent les différentes notes données par les assistants pour faire sortir une note finale et permettent à l'enseignant de la modifier si une inconsistance est trouvée. D'un point de vu apprenant, ces outils peuvent fournir une évaluation formative permettant un apprentissage par l'erreur. Cependant, la disponibilité de tel feedback encourage une approche d'apprentissage par essai et erreur.

En résumé, les systèmes étudiés sont des systèmes dans lesquels l'évaluation est soit une partie du processus d'apprentissage, soit elle représente le cœur de l'outil.

4.5.1 Les techniques d'évaluation généralement utilisées en algorithmique/programmation

Évaluation par l'approche boite noire

Concernant le type d'évaluation, la manière standard d'évaluer un programme est de le compiler puis de l'exécuter avec un jeu de tests comme entrées et sorties (approche boite-noire). Le programme est classé comme

accepté s'il est compilé sans erreurs et si les sorties du programme sont identiques aux valeurs de sorties attendues.

Évaluation par l'approche boite blanche

Cette stratégie avait montré des problèmes pédagogiques indésirables comme la frustration et la confusion de l'étudiant (Tang et al. 2009*b*;*a*). Plusieurs systèmes (Blumenstein et al. 2004, Jackson and Usher 1997, Juedes 2003, Amit et al. 2006, Pisan et al. 2003, Saikkonen et al. 2001) évaluent non seulement le comportement d'un programme mais aussi analysent la structure du code source (approche boite blanche 'white-box approach'). Cette approche permet de savoir si le programme a été écrit avec une façon bien particulière selon un algorithme particulier ou même en utilisant certaines structures de donnée.

Edwards and Pugh (2006) et Auffarth and L (2008) utilisent des unités de tests prédéfinies par les enseignants pour valider les productions des étudiants.

Un autre problème réside dans la non détermination des sorties du programme où les différentes solutions (correcte ou plus au moins acceptable) pour le même exercice, peuvent ne pas produire exactement la même sortie (Tang et al. 2009*b*). José and Fernando (2003) résout ce non déterminisme en utilisant des correcteurs dynamiques comme étant des évaluateurs qui sont sollicités après chaque test. Par exemple, si la solution est un ensemble de valeurs qui peuvent être présentées dans n'importe quel ordre, le correcteur dynamique peut être utilisé pour réduire la sortie du programme à une forme normale.

Les techniques utilisées dans la plupart des outils :

- La comparaison de résultat de sortie du programme avec les résultats attendus est l'approche traditionnelle utilisée par plusieurs systèmes. Un état de l'art présenté par Ala-Mutka (2005) montre les différentes variations de la technique de comparaison de sortie y compris l'exécution de la solution code de l'apprenant et l'utilisation des expressions régulières pour reconnaitre la sortie.
- Le scénario (Scripting) d'évaluation est la technique la plus utilisée pour définir les tests. Par exemple, un script peut être un script qui compile le programme, l'exécute puis comparer le résultat du programme à un fichier contenant le résultat attendu.
- Des approches expérimentales comme la comparaison des graphes de programme d'une solution d'un apprenant par rapport à un ensemble de solutions justes Naudé et al. (2010), Wang et al. (2007) sont aussi utilisées.

Les QCM aussi sont parfois utilisés pour évaluer des connaissances en programmation qui sont généralement des connaissances liées au langage de programmation comme il est montré dans les deux figures 4.10, 4.11.

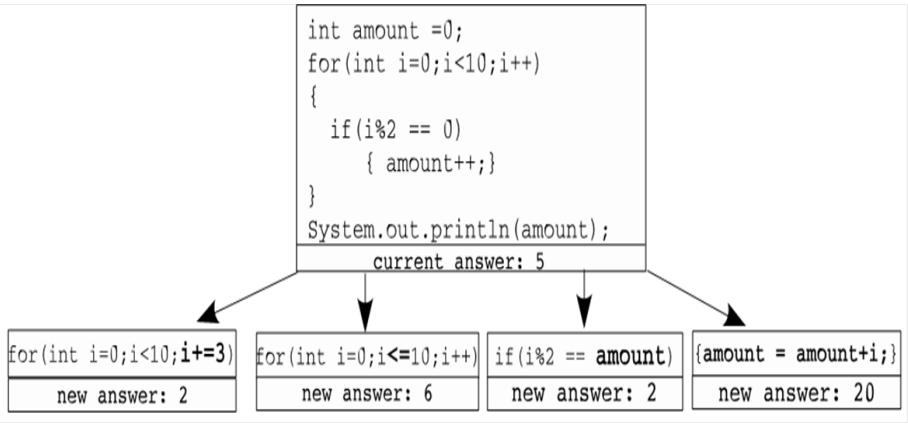

FIGURE 4.10 – *Exemple 1 d'utilisation des QCM en programmation.*

Quelle est la déclaration correcte de la fonction main en Java ?

a) Begin Program ()
b) Public start main()
c) Public static void main(String args[])
d) int program;
e) System.out.println(''Hello World'');

FIGURE 4.11 – *Exemple 2 d'utilisation des QCM en programmation.*

4.5.2 Synopsis des systèmes

Dans ce tableau comparatif des systèmes discutés dans les sections 4.4.1 et section 4.4.2 on va projeter ces outils par rapport à un ensemble de critères qui sont classés selon des catégories.

Résolution du problème

La première catégorie de critères représente l'étape de développement d'un programme traité par le système à savoir :

- Compréhension du problème,
- Conception d'algorithme,
- Programmation,
- Test.

Représentation de la solution à évaluer

Ce critère montre la manière dont la solution est exprimée par l'apprenant pour que le système puisse l'évaluer. Une solution peut être :

- *Informelle :* une représentation informelle veut dire une description de la solution en langage naturel et qui n'a aucune relation avec les concepts de l'algorithmique.

Critères Systèmes	Bridge	Proust	GPC-Editor	Solvit	Discover	Sicas	Propl	AlgoBox	Assyst	Trakla	Boss	Web-CAT
Résolution du problème												
Compréhension du problème			√			√						
Conception d'algorithme	√		√	√	√	√	√	√				
Programmation	√	√	√	√	√	√			√	√	√	
Test				√								√
Représentation												
Informelle	√					√						
Organigramme					√				√			
Pseudo code	√	√	√	√	√			√		√		
Évaluation												
Objective												
Subjective	√	√	√	√	√	√	√	√	√	√	√	
Types d'évaluation												
Formative	√		√	√	√	√	√			√	√	
Diagnostic des erreurs sémantiques		√								√		√
Sommative							√		√	√		
Approche de l'évaluation												
Exécution de programme	√		√	√		√		√		√	√	√
Analyse de programme		√			√		√		√			
Langage de programmation												
Dépendance de langage		√							√		√	√
Indépendance du langage	√		√	√	√	√	√	√		√		
Variabilité des solutions	√		√	√		√		√		√	√	√

TABLE 4.2 – *Synopsis des systèmes dédiés à l'algorithmique et à la programmation.*

- *Pseudo code :* Une description informelle de haut niveau de l'algorithme en utilisant les concepts de la programmation.
- *Organigramme :* une représentation schématique des liens fonctionnels, organisationnels et hiérarchiques d'un programme.

Subjectivité et objectivité

L'objectivité et la subjectivité de l'évaluation fournie par le système est aussi un critère important. Ici, on mesure la subjectivité par rapport aux facteurs qui influencent les résultats de l'évaluation, par exemple : l'intervention humaine et l'évaluation par rapport à une solution stéréotypée.

Types d'évaluation

Les types d'évaluation que le système fournit : formative, diagnostic des erreurs sémantiques ou sommative.

Approche d'évaluation

Parmi les approches d'évaluation des algorithmes ou de programmes qu'on peut trouver dans la plupart des systèmes, les approches par exécution de programme quand il s'agit de programme ou par analyse quand il s'agit d'algorithme.

Dépendance et indépendance du langage de programmation

Ce critère représente une restriction par rapport à l'outil d'évaluation parce que l'utilisateur va être obligé à apprendre le langage afin d'utiliser ce système.

Variabilité des solutions

Cette dimension qui caractérise le domaine de l'algorithmique est souvent absente dans les systèmes qui analysent et comparent la solution à évaluer par rapport à des solutions prédéfinies. Contrairement aux systèmes qui utilisent l'approche par exécution à l'aide des compilateurs, on peut dire que la variabilité est prise en compte mais reste à prouver que la solution est algorithmiquement bonne.

Conclusion

Dans ce chapitre nous avons étudié les caractéristiques du domaine de l'algorithmique et entre autre la programmation ainsi qu'une étude panoramique sur quelques systèmes existants qui traitent la problématique de l'apprentissage et de l'évaluation des acquis des apprenants en algorithmique.

A partir de l'étude faite sur les plus importants systèmes qui traitent notre problématique, nous pouvons constater que quand il s'agit d'une approche d'évaluation par exécution, la variabilité des solutions est assurée par le biais d'un compilateur. Cette approche est connue par son ignorance à la conception du programme s'il est algorithmiquement bon (boite noire). Tandis que l'approche d'évaluation par analyse statique, la variabilité des solutions n'est pas assurée vu qu'elle est basée sur des solutions prédéfinies en amont.

Aussi, l'objectivité caractérise les outils des deux approches ainsi que le diagnostic des erreurs sémantique et l'évaluation sommative.

Une approche d'évaluation des compétences algorithmique

5

Sommaire

5.1 Modèle global de la méthode proposée 61
5.2 Stratégie de conception d'une solution 62
5.2.1 Définition d'un Plan de solution 65
5.2.2 La bibliothèque d'opérations de base 66
5.3 Une base de solutions correctes et de solutions erronées 67
5.4 L'évaluation à l'aide d'un processus de reconnaissance 69
5.5 Boucle d'alimentation de la base de solutions . . . 69
Conclusion 70

5.1 Modèle global de la méthode proposée

Le but de cette thèse, comme il a été indiqué dans les sections précédentes, est d'élaborer un système d'évaluation des compétences en résolution de problèmes algorithmiques. Afin d'atteindre cet objectif, nous avons commencé par mettre en avant la philosophie de notre système d'évaluation. Nous entendons par philosophie du système, le modèle général de l'évaluation proposée.

L'architecture décrite dans la figure 5.1, représente l'idée générale de notre méthode.

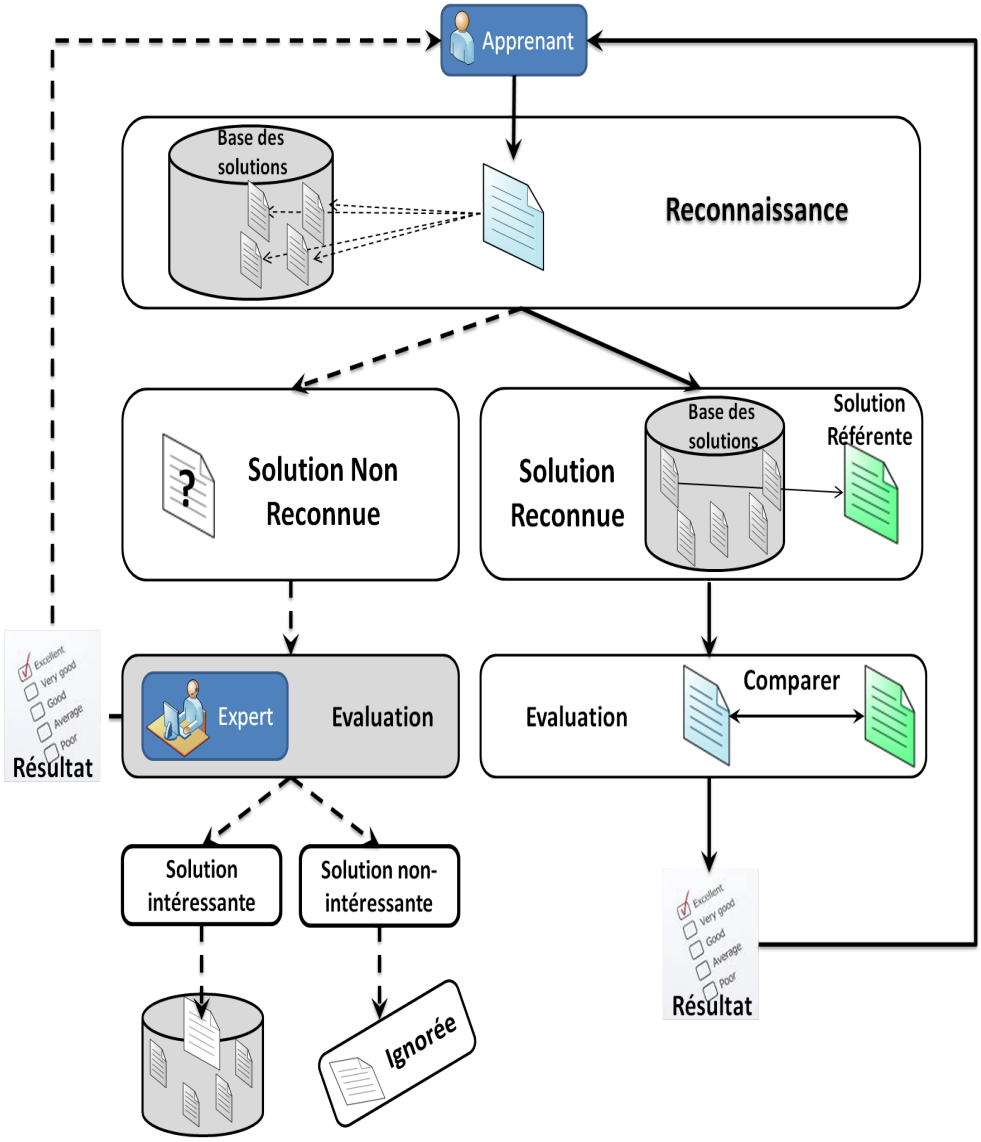

Figure 5.1 – *Architecture générale de la méthode.*

Cette architecture descriptive de la méthode est basée essentiellement sur trois aspects, qui constituent notre contribution et qui sont liés à notre objectif : l'évaluation du savoir-faire algorithmique, (1) comment résoudre un problème algorithmique et comment l'exprimer d'une manière indépendante de tout langage de programmation ?, (2) comment évaluer l'algorithme

solution du problème ?, (3) comment assurer que n'importe quelle solution proposée par l'apprenant soit évaluée ?

Présentons ci-dessous l'algorithme qui décrit la méthode générale. A partir de cet algorithme, nous allons discuter les fonctions principales sur lesquelles se base notre méthode.

Algorithme 1: Méthode générale d'évaluation

Données : S :Solution,BS :Solutions prédéfinies, Reconnu :Booleen
Résultat : Évaluation formative/sommative de la solution S
début
  $S \longleftarrow Concevoir.une.solution$;
  $Reconnu \longleftarrow Reconnaitre(S, BS)$;
  si *Reconnu* **alors**
    retourner *(Résultat : Note,Feedback)*
  sinon
    Solution S évaluée par un enseignant ;
    retourner *(Résultat : Note,Feedback)*
  fin si
fin

5.2 Stratégie de conception d'une solution

La question essentielle de ce travail est de développer une méthode d'évaluation d'une solution algorithmique produite par un apprenant. Cette question, comment évalue-t-on une solution algorithmique automatiquement, nous a poussé en premier lieu à poser la question suivante : comment modéliser une solution algorithmique pour qu'on puisse l'évaluer.

Pour répondre à cette question, on est parti de la définition même de l'algorithmique :

« L'algorithmique permet de préparer la résolution d'un problème en détaillant les opérations élémentaires à accomplir. Il reste ensuite à traduire ces opérations en instructions pour l'ordinateur ». On entend par préparer la solution, l'application de la décomposition du problème à résoudre à des sous-problèmes faciles (figure 5.2).

Il est connu en algorithmique que pour exécuter des tâches complexes, chaque tâche doit être décomposée en une suite de tâches plus simples. Cette décomposition est répétée jusqu'à arriver à des tâches élémentaires. Le nombre d'étapes de décomposition dépend de la complexité du problème à résoudre. Le but est de passer du problème abstrait à des sous tâches plus au moins concrètes en passant par des niveaux de décomposition (figure 5.3).

Cette approche descendante (dite également par raffinements successifs), source de notre inspiration, permet de passer progressivement et avec un maximum de chances de réussite, de la description abstraite de la solution du problème (par une opération complexe) à l'algorithme qui permettra sa

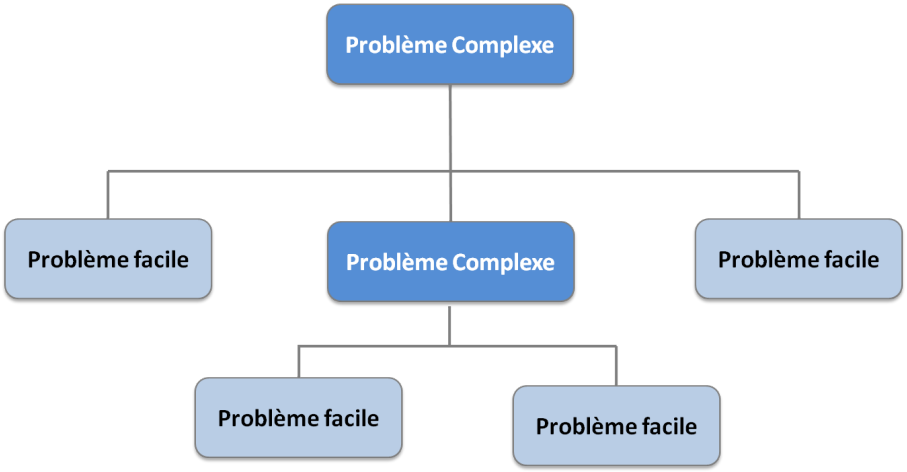

FIGURE 5.2 – *Décomposition d'un problème algorithmique.*

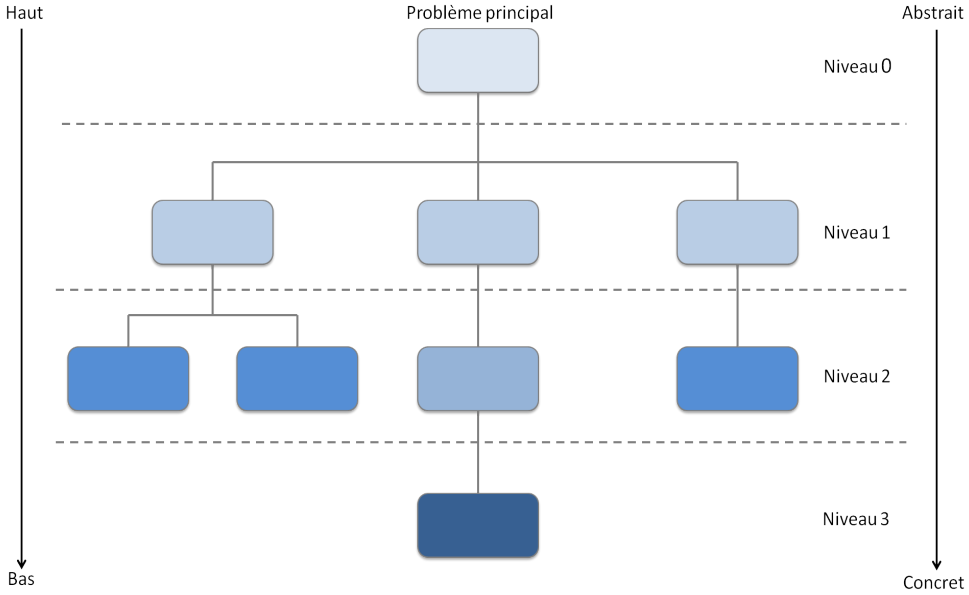

FIGURE 5.3 – *Décomposition d'un problème de l'abstraction au concret.*

résolution. Concrètement, on peut alors dire qu'un algorithme est au dernier niveau de raffinement lorsqu'il ne comporte que des opérations élémentaires, des opérations de base (opérations connues) et des structures de contrôle.

Tout algorithme est caractérisé par :

- Un ensemble d'actions ou d'opérations à exécuter,
- Un ordre d'exécution de ces différentes opérations déterminé par la logique d'enchaînement et conditionné par les structures mises en œuvre,
- Un début et une fin.

Pour fournir une solution au problème, il faut donc un moyen d'exprimer cette solution à l'aide d'un langage qui doit être :

- Formel (pas d'ambigüités),
- Lisible et concis,
- Indépendant de tout langage informatique,
- Qui reprend les concepts des langages impératifs,

- Les entités manipulées sont désignées et typées.

La figure 5.4 illustre comment un problème est décomposé et quelles sont les éléments utilisés pour exprimer une solution algorithmique.

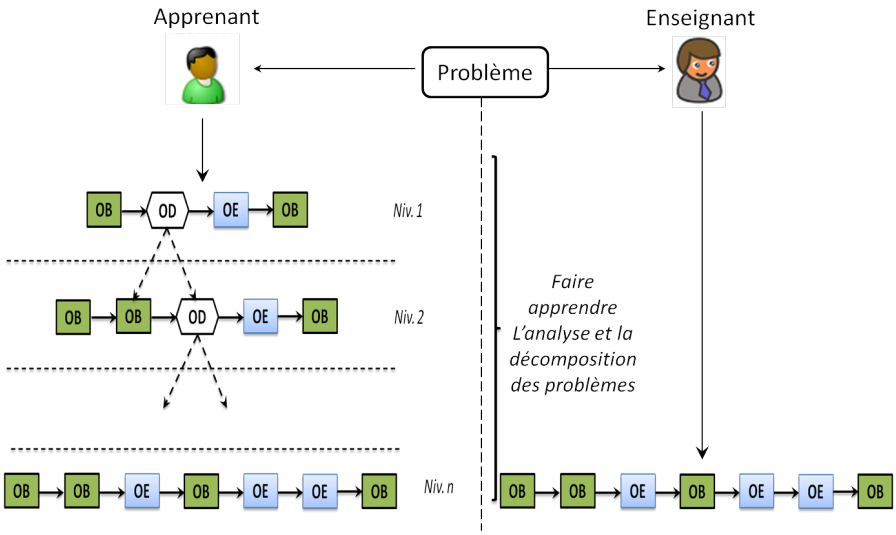

FIGURE 5.4 – *La décomposition d'un problème : vue par l'apprenant et par l'enseignant.*

La figure 5.4 illustre la démarche de résolution vu par l'apprenant en utilisant des éléments qui sont définis comme suit :

Définition 5.1 *Une opération de base (OB) est une opération qui s'opère sur un type abstrait de données spécifique telle que : la recherche d'un élément dans une liste, la suppression d'un élément, l'ajout, etc.*

Définition 5.2 *les opérations élémentaires (OE), sont en fait des instructions simples telles que l'affectation, lecture, écriture, etc.*

Définition 5.3 *les opérations décomposable (OD) désignent un état intermédiaire et abstrait qui n'est pas encore définit par l'apprenant. Ce genre d'opération aident l'apprenant à décomposer successivement et définir au fur et à mesure les opérations résolvant le problème.*

Au niveau 1, le problème est décomposé en un ensemble d'opérations décomposables (OD), d'opérations de base et d'opérations élémentaires liées par des structures de contrôle si c'est nécessaire (conditions, répétitions).

Dans le niveau 2, chaque opération décomposable du niveau 1 est décomposée, soit en opérations encore décomposables, soit en opérations de base, soit en opérations élémentaires. Cela dépend de sa complexité.

Etc.

Ainsi, dans un contexte d'évaluation, l'apprenant peut soumettre sa solution, à un problème donné, en termes d'opérations décomposables (par la suite), d'opérations de base et d'opérations élémentaires. Ces opérations pouvant être liées par des structures de contrôle. Cette ébauche graduelle de la solution et l'utilisation d'un niveau méta (opérations de base) permet-

tra une évaluation d'abord efficace, ensuite totalement *adaptée à la matière* (Bey et al. 2010*c*).

Cette façon de faire vise deux objectifs :

- Forcer l'apprenant à apprendre et à décomposer un problème. Ceci lui permettra d'apprendre à diminuer graduellement la complexité d'un problème et lui évitera de se noyer dans les détails dès le départ,
- Permettre à l'apprenant de focaliser ses efforts sur le problème et non pas sur des questions secondaires (opérations de base). Exemple : dans un problème nécessitant un tri, ce qui nous intéresse, c'est si l'apprenant opte pour un tri et non comment il réalise le tri. Car les opérations de base peuvent elles mêmes faire l'objet d'un apprentissage.

A partir de cet ensemble de caractéristiques d'un algorithme, nous avons défini une représentation d'une solution algorithmique nommée : Plan de Solution.

5.2.1 Définition d'un Plan de solution

Définition 5.4 *Un plan : « Projet élaboré, comportant une suite ordonnée d'opérations, de moyens, destinée à atteindre un but » (petit Robert)*

Tandis que dans le domaine des environnements informatiques pour l'apprentissage humain, cette approche de modélisation est conçue dans le cadre de la planification et la reconnaissance de plan. Les connaissances sont modélisées sous forme de stratégies de résolutions de problème appelées plans. Le système cherche dans cette approche à découvrir le plan mis en œuvre par l'apprenant à partir de l'observation de ses actions. Cette approche est utilisée dans les domaines d'apprentissage qui nécessitent que l'apprenant effectue un choix d'actions adaptées pour la résolution, comme par exemple en algèbre ou en géométrie (Mufti-Alchawafa 2008). Dans cette approche, pour chaque problème il y a plusieurs plans possibles et non pas une seule stratégie correcte. Du point de vue du modèle de l'apprenant, elle permet de déduire les plans de la résolution choisie par l'apprenant pour un problème donnée et non pas les connaissances de l'apprenant selon les thèmes ou les concepts du domaine, comme dans la première approche. « La reconnaissance de plan permet d'élaborer un modèle local à une session, ou un problème, puisqu'elle vise à inférer la stratégie mise en œuvre pour atteindre un but donné.» (Dominique 1998).

Un exemple de cette approche de modélisation est le tuteur Mentoniezh dans le domaine de la démonstration en géométrie élémentaire (Dominique 1996). Dans ce tuteur, les plans sont un ensemble des démonstrations possibles produites automatiquement par un démonstrateur pour un problème donné. Après chaque action effectuée par l'élève pour constituer sa preuve, le système élimine les plans qui ne sont pas cohérents par rapport à la résolution, afin de déterminer les plans qui expliquent le mieux les actions observées.

Dans notre contexte, on définit un plan de solution de la manière suivant :

Définition 5.5 *Un plan de solution noté PS, est une suite d'opérations (de base ou élémentaires) et des structures de contrôle (des tests et des boucles) qui résout un problème donné.*

Soit par exemple l'énoncé suivant : *Écrire un algorithme qui remplace toutes les occurrences de X dans la pile P par Y.*

L'algorithme suivant représente un exemple d'un plan de solution pour le problème donné.

Algorithme 2: Un exemple d'un Plan de Solution (PS)

```
Données : P,P1 :Pile,X,Y,VAR :élément
début
    tant que NotEst_Vide(P) faire
        VAR ⟵ Sommet(P) ;
        si VAR<>X alors
            Empiler(Y,P1)
        sinon
            Empiler(VAR, P1);
        fin si
        Dépiler(P);
    fin tq
fin
```

5.2.2 La bibliothèque d'opérations de base

Ce qui est enseigné en algorithmique aux étudiants ce sont les structures de données et les opérations de base qu'on peut effectuer sur ces structures de données. Un exemple, la plupart des apprenants en informatique doivent suivre : Algorithmique et Structures de données. Dans ce module, les notions de base en algorithmique sont enseignées ainsi que les différentes structures de données. Pour la structure de données Pile par exemple, les différentes modélisations de la structure Pile sont introduites à savoir : avec les pointeurs ou avec tableau, ainsi que les opérations de base qu'on peut effectuer sur cette structure comme : Empiler, Dépiler, Sommet, etc. Le but de ces opérations enseignées est de les réutiliser pour résoudre d'autres problèmes algorithmiques plus complexes.

Ces opérations de base, utilisées dans la conception d'une solution, sont regroupées dans une bibliothèque. Une opération de base a des paramètres et un résultat qui sont utilisés dans la reconnaissance lors de l'évaluation de la solution.

Le tableau suivant montre un exemple des opérations de base de la structure de la pile (voir annexe A.2).

Opération	Paramètre	Résultat	Description
New		P : pile	crée une nouvelle pile p
InitPile	P : pile		initialise la pile à vide
Empiler	e : élément, P : pile		dépose l'élément e au sommet de la pile
Dépiler	P : pile		enlève l'élément qui se trouve au sommet de la pile
Sommet	P : pile		élément retourne la valeur de l'élément qui est au sommet de la pile
estVide	P : pile	booléen	retourne vrai si la pile est vide
Profondeur	P : pile	entier	retourne la profondeur de la pile

TABLE 5.1 – *Ensemble d'opérations de base de la structure Pile.*

5.3 Une base de solutions correctes et de solutions erronées

Rappelons qu'en algorithmique, pour un problème donné, il existe une multitude de solutions. Afin de satisfaire cette contrainte spécifique à ce domaine, nous avons choisi de prévoir un ensemble des solutions plausibles dans une base de solutions afin de les utiliser dans notre évaluation.

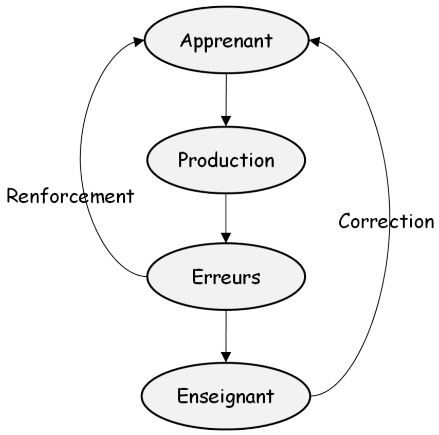

FIGURE 5.5 – *Rôle pédagogique de l'erreur.*

Avant les années 80, les interprétations des erreurs situaient celles-ci hors des processus d'apprentissage. Depuis, on les considère comme un élément du processus didactique, c'est-à-dire comme une information dont il faut élucider les composants (origines) pour construire une connaissance correcte

(figure 5.5). Là est le rôle de l'enseignant : il doit situer les erreurs dans leur diversité afin de déterminer les modalités de l'intervention didactique à mettre en œuvre. La prise en compte des erreurs positivement (pas d'une manière négative) est nécessaire dans un apprentissage et il ne suffit pas de citer les erreurs mais de les interpréter et les analyser pour qu'elles soient un repère pour l'enseignant afin d'adapter sa stratégie en fonction de ces erreurs-ci.

Puisque l'erreur est révélatrice d'une authentique activité intellectuelle de l'élève (stratégie d'appropriation par élaboration progressive de schémas de représentation), elle n'est pas blâmable : elle n'est pas de la faute de l'élève, ce n'est pas une faute. Ce n'est pas l'indice d'un défaut de connaissance, mais celui de l'inadéquation des connaissances de l'élève à rendre compte du réel.

Il faut analyser la valeur des erreurs en essayant de déterminer leurs origines. Mais la prise en compte ne s'arrête évidemment pas là. Il faut ensuite que les élèves prennent conscience de leurs erreurs.

En effet, Stella BARUK (mathématicienne et écrivaine) explique que lorsque l'apprenant identifie lui-même l'erreur, la confusion cesse au moment même où il en prend conscience.

Par exemple dans Buggy (Brown and Burton 1978) et Pépite (Jean-Daubias et al. 1998), une approche d'évaluation diagnostic des connaissances procédurales en intégrant les erreurs a été utilisée et a montré son efficacité puisqu'elle a débouché sur un modèle cognitif de l'apprentissage des connaissances procédurales. Cette approche apparaît trop lourde à mettre en œuvre car, établir un catalogue de bugs pour un domaine donné demande un travail considérable d'analyse de domaine à enseigner, et un nouveau catalogue de bugs devra donc être reconstruit à chaque changement de domaine.

Dans le cas de l'algorithmique, il est impossible de construire une bibliothèque de bugs complète puisqu'on ne peut pas prévoir tous les bugs que peuvent commettre les apprenants.

Cette approche cognitive place l'erreur au centre de la démarche pédagogique. Dans cette perspective, l'évaluation n'est plus une sanction mais une aide précieuse à l'apprentissage et l'erreur devient un outil pour enseigner, un tremplin susceptible de débloquer les démarches d'apprentissages (Astolfi 1997, Tagliante 1994). C'est sur ces principes que repose la pédagogie de l'erreur.

Dans cette optique, on peut classer les erreurs en deux grandes catégories :

- Les erreurs de compétence, récurrentes, que l'apprenant ne peut rectifier parce qu'il ne possède pas les savoirs nécessaires,
- Les erreurs de performances, occasionnelles, assimilables à la faute, que l'apprenant peut rectifier parce qu'il dispose des savoirs adéquats et que ces erreurs sont dues à une distraction passagère.

Notre base contient et les solutions correctes les solutions erronées. Nous entendons par solutions erronées, les solutions qui contiennent des erreurs

de compétence et qui sont sont récurrentes. Le but est d'expliciter ces erreurs à partir des solutions erronées afin de les exploiter comme des sources d'apprentissage à fournir aux apprenants.

5.4 L'évaluation à l'aide d'un processus de reconnaissance

Après la résolution du problème selon la méthodologie citée, la solution exprimée en fonction d'opérations (opération de base et élémentaire), sera soumise au module évaluation.

Selon Hadji : « Évaluer, c'est mettre en relation des éléments issus d'un observable (ou référé) et un référent pour produire de l'information éclairante sur l'observable, afin de prendre des décisions. » (Hadji 1990).

Dans notre cas, on veut évaluer une production de l'apprenant, qui est une solution algorithmique à un problème donnée. Si on applique la définition précédente à propos de l'évaluation et ce qu'on veut évaluer, on obtient :

- La solution algorithmique de l'apprenant est l'objet à évaluer : **le référé**,
- Une solution qui représente l'idéal attentes et intentions de l'enseignant : **le référent**.

Donc pour évaluer, on doit comparer la solution de l'apprenant (le référé) avec une solution modèle (le référent). Or dans la base de solutions, il y a plusieurs solutions pour un problème donné. Donc, on doit comparer la solution produite par l'apprenant avec toutes les solutions de la base pour reconnaitre la solution référente.

5.5 Boucle d'alimentation de la base de solutions

Sous la lumière de l'étude épistémologique de l'algorithmique présentée dans le chapitre 4, on a bien compris que la variété de solutions ou la multiplicité des solutions possibles pour un problème donné en algorithmique est une source de problème remarquable quand on veut automatiser la tâche de l'évaluation d'une solution quelconque.

Par exemple, pour un problème donné X, l'enseignant peut prévoir N solutions dans la base afin de les utiliser comme référents possibles pour juger la solution de l'apprenant. Mais un apprenant donné peut proposer une solution qui n'étais pas prévue parmi les N solutions de la base par oubli ou par le fait que cette solution est une nouvelle solution non connue chez l'expert ou l'enseignant qui a défini ces solutions, car il est difficile de prévoir toutes les solutions possibles d'un problème afin de les intégrer dans une éventuelle base de solutions.

Ce phénomène d'évaluation des productions des apprenants par rapport

à des solutions prédéfinies et stéréotypées est nuisible et il engendrer des effets indésirables comme :

- Une bonne solution produite par un apprenant évaluée par rapport à des solutions stéréotypées engendre une sous évaluation, c.-à-d. si la solution de l'apprenant est identique à l'une des solutions, elle est jugée juste, sinon, elle est jugée moins bonne,
- Limiter la créativité des apprenants dans la conception d'algorithmes, les apprenants vont apprendre que, pour un problème donné, il y a juste quelques solutions qu'il faut apprendre.

Pour éviter ce problème de multiplicité des solutions qui représente le maillon faible de tous les systèmes d'évaluation en algorithmique, on a suggéré l'évaluation d'un expert humain dans le cas d'une production inattendue par le système. Cet enseignant est sollicité pour analyser cette solution puis l'ajouter à la base. Aussi, les solutions récurrentes non reconnues sont automatiquement ajoutées.

Conclusion

Dans ce chapitre, nous avons présenté le modèle de la méthode d'évaluation proposée avec tous ses éléments de base : la stratégie de résolution de problème, la bibliothèque des opérations de base, l'intervention de l'expert et le processus de l'évaluation.

En résumé, une solution algorithmique proposée par un apprenant est évaluée de la manière suivante. Une fois la solution est établie en fonction d'opérations, cette dernière est soumise à un mécanisme de reconnaissance par rapport à une base de solutions préétablies. Dans le cas où la solution n'a pas été reconnue, un expert humain est sollicité pour l'évaluer et l'ajouter dans la base si elle est jugée intéressante.

Une méthode d'évaluation basée sur une mesure de similarité proposée

6

Sommaire

6.1 Conception d'un Plan de Solution 72
6.1.1 Anatomie des Plans de Solution de la base 72
6.1.2 Feedback associé à chaque plan de solution 73
6.2 Scénario de l'évaluation 74
6.3 Proposition d'une mesure de similarité 75
6.3.1 Caractéristiques d'un plan de solution 75
6.3.2 Mesure de similarité proposée 76
6.4 Recherche d'un Plan de Solution référent 78
6.4.1 Cas d'un plan de solution référent trouvé 78
6.4.2 Cas d'un plan de solution référent non trouvé 79
6.5 Algorithme du processus d'évaluation 80

Après avoir défini les grandes lignes de notre méthode d'évaluation : la stratégie de conception d'une solution algorithmique, le mécanisme d'évaluation et le traitement des solutions nouvelles qui n'ont pas été reconnues parmi les solutions de la base, nous allons maintenant nous baser sur le processus de reconnaissance où on va proposer notre propre mesure de similarité entre plan de solution.

Dans ce chapitre nous commençons par l'anatomie d'un plan de solution dans la base des solutions et puis nous définirons notre mesure de similarité et nous terminerons par l'algorithme détaillé du processus d'évaluation.

6.1 Conception d'un Plan de Solution

6.1.1 Anatomie des Plans de Solution de la base

Dans la partie précédente (section 5.2), nous avons défini les composantes d'une solution algorithmique qui sont les opérations (de base et élémentaire) et les structures de contrôles.

Pour un problème donné, l'enseignant doit définir les solutions plausibles (correctes et erronées). Lors de la définition d'une solution quelconque, il doit annoter aussi l'importance d'une opération dans la solution. Le rôle de l'annotation avec des valeurs binaires est d'exprimer l'objectif visé par cette solution. L'annotation vaut 1 pour importante,et vaut 0 pour non importante. Par exemple, dans un exercice où on veut évaluer l'habilité de l'apprenant à faire une boucle, on attribue la valeur 1 pour la boucle. Et pour ignorer d'autres opérations par exemple une affectation, on attribue la valeur 0. Ces annotations représentent les poids sur ces opérations. Aussi, une note et un feedback explicatif sur la solution sont ajoutés à la solution (Bey et al. 2010*b*).

La figure suivante représente un exemple illustratif de ce qui a été dit.

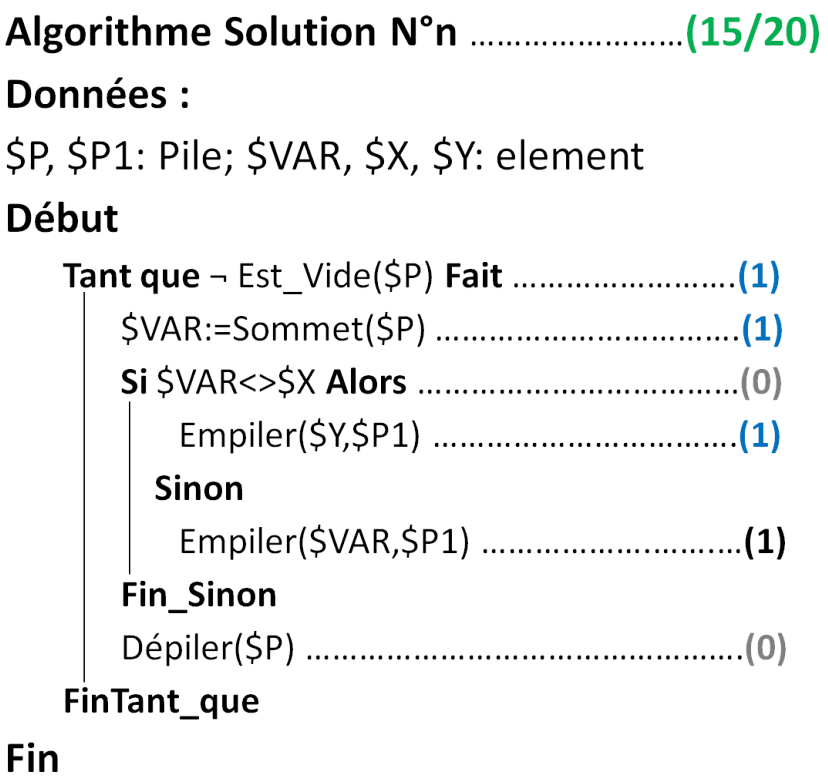

FIGURE 6.1 – *Anatomie d'un plan de solution dans la base.*

La variété des solutions possibles à un problème algorithmique fait que, lors de la résolution de ce problème, l'apprenant peut prendre un chemin parmi plusieurs conduisant à une solution (figure 6.2). Une solution est un chemin qui est constitué d'opérations de base et d'opérations élémentaires (Bey et al. 2010*a*). On distingue deux types de solution : solution correcte, et

solution erronée qui décrit une erreur pédagogiquement intéressante. L'ensemble des solutions constitue la Carte Descriptive (CD) d'un problème donné (figure 6.2).

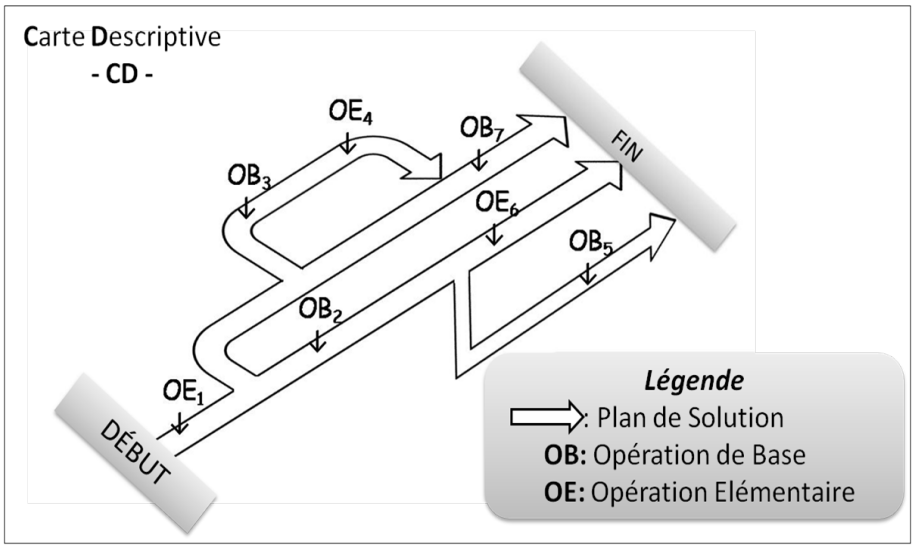

FIGURE 6.2 – *Carte descriptive d'un problème.*

Cette carte descriptive permet de localiser l'apprenant et de reconnaître sa démarche qu'elle soit bonne ou erronée.

Les cartes descriptives sont évolutives. Elles peuvent, ainsi, s'enrichir avec d'autres solutions. Ceci permettra d'intégrer une solution proposée par l'apprenant mais non prévue par l'enseignant. En effet, toute solution non reconnue verra son évaluation suspendue jusqu'à l'intervention de l'enseignant humain qui, s'il la juge intéressante (qu'elle soit correcte ou pas), l'ajoutera à la CD.

Le choix de l'utilisation du pseudo code pour la définition d'une solution est justifié par le fait que l'apprenant se concentre mieux sur la conception et non sur la syntaxe quand il s'agit d'un langage de programmation. Après, la migration vers des langages sera facile à adopter.

Néanmoins, des recherches suggèrent que l'utilisation du pseudo code pour la conception d'algorithme aide les apprenants à mieux programmer.

Ramadhan (2000) a montré dans un travail de synthèse sur les résultats obtenus que :

- Les compétences de conception d'algorithme peuvent être apprises indépendamment de n'importe quel langage de programmation,
- L'utilisation du pseudo code pour l'apprentissage des bases de la programmation facilite le transfert des connaissances d'un langage à un autre et réduit la surcharge cognitive des apprenants.

6.1.2 Feedback associé à chaque plan de solution

Générer un feedback sur une production d'un apprenant est un ingrédient essentiel pour une bonne activité d'apprentissage. Pour chaque plan

de solution dans la base, une rétroaction produite et recommandée par un enseignant est associé. Cette rétroaction donne une explication sur la solution tout en mentionnant les compétences utilisées dans la solution ainsi que les erreurs s'ils existent.

6.2 Scénario de l'évaluation

Après la modélisation des solutions algorithmiques avec les différentes caractéristiques ajoutées qui vont aider à l'évaluation, nous allons passer au scénario d'évaluation d'une solution algorithmique.

Ce scénario (figure 6.3) est composé de deux modules : un module de reconnaissance et un module d'évaluation.

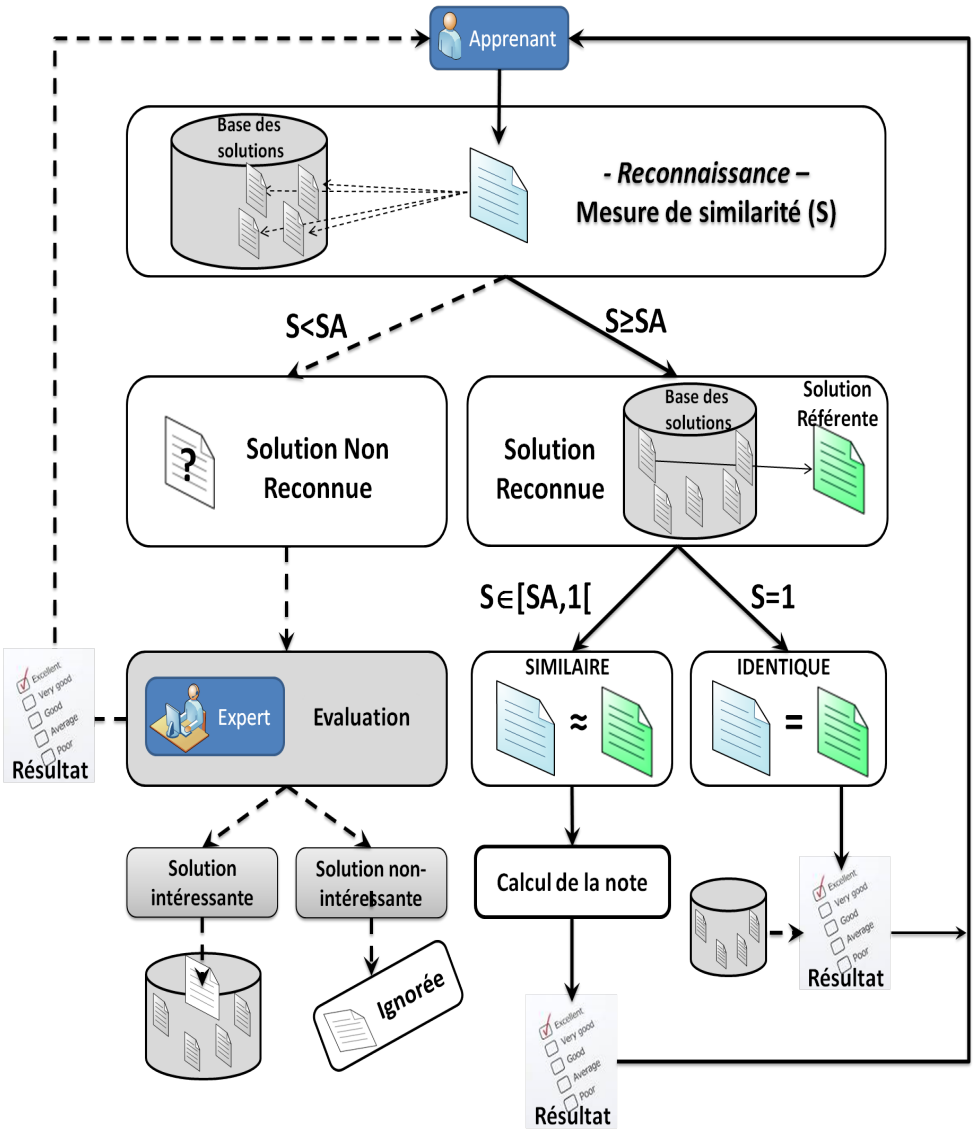

Figure 6.3 – *Scénario de l'évaluation.*

La première étape du processus consiste à reconnaitre le plan de solution

proposé dans la base des solution. La reconnaissance a pour objectif de repérer le plan de solution référent à l'aide d'une mesure de similarité.

6.3 Proposition d'une mesure de similarité

Une mesure de similarité est un concept de base qui est très largement utilisées dans des domaines très divers. Sa terminologie varie considérablement (ressemblance, proximité, ...) et l'auteur d'une mesure particulière n'est pas forcément unique. Malgré la confusion qui règne dans la littérature, certaines conditions sur les mesures de similarité sont plus fréquemment mentionnées que d'autres. En général, c'est une fonction qui quantifie le rapport entre deux objets comparés en fonction des points de ressemblance et de différence.

Le choix d'une mesure de similarité est tout à fait crucial (Hanson 1990) pour la bonne exécution d'une tâche. Il s'agit en effet de trouver la meilleure adéquation entre le but qui est poursuivi et le comportement effectif de la mesure. Dans le domaine des Sciences Cognitives ou plus exactement celui de la psychométrie, on constate que les propriétés vérifiées par les distances mathématiques (minimalité, symétrie et inégalité triangulaire) ne sont pas toujours respectées dans la manière dont les sujets humains perçoivent la notion de ressemblance. De manière synthétique, ces différences s'expliquent par le fait que les critères sur lesquels les êtres humains jugent la similarité ne sont pas stables et qu'ils varient dynamiquement en fonction du contexte dans lequel on se trouve, des connaissances *a priori* que l'on possède sur le domaine, du degré de typicité des exemples, etc.

En effet, le choix d'une mesure de similarité dépend essentiellement des caractéristiques des deux objets à mesurer. Dans notre cas se sont les plans de solution.

6.3.1 Caractéristiques d'un plan de solution

Comme il est défini dans la section 5.2.1, nous appelons un plan de solution, une composition d'un ensemble fini d'*opérations* de base et élémentaire dans un ordre déterminé. Celui-ci est donc caractérisé par : la concordance, la discordance et l'inutilité des opérations (tableau 6.1).

Deux mesures sont, dans la littérature, essentiellement utilisées pour évaluer la corrélation entre deux ordres : le coefficient de corrélation de Spearman et celui de Kendall. Ces degrés permettent de comparer des ordres qui présentent des concordances, des discordance et des ex-æquo mais ils doivent lister les mêmes objets. Cette dernière propriété est nécessaire car, dans le cas d'une mesure de similarité entre un plan de solution et le plan de solution référent, il se peut que le plan de solution ait des opérations qui n'existent pas dans le plan de solution référent.

Dans le contexte général de notre problématique, aucune mesure de similarité ne peut satisfaire pleinement nos besoins en termes de contraintes à

Cas	Exemple PS Référent	Exemple PS proposé
Concordance	**Début** $i \longleftarrow 1;$ $j \longleftarrow 1;$ $x \longleftarrow 2i + j;$ Écrire(x) ; **Fin**	**Début** $i \longleftarrow 1;$ $j \longleftarrow 1;$ Écrire(x) ; $x \longleftarrow 2i + j;$ Écrire(j) ; **Fin**
Discordance	**Début** $i \longleftarrow 1;$ $j \longleftarrow 1;$ $x \longleftarrow 2i + j;$ $Écrire(x);$ **Fin**	**Début** $i \longleftarrow 1;$ $j \longleftarrow 1;$ $Écrire(x);$ $x \longleftarrow 2i + j;$ Écrire(j) ; **Fin**
Absence	**Début** $i \longleftarrow 1;$ $j \longleftarrow 1;$ $x \longleftarrow 2i + j;$ Écrire(x) ; **Fin**	**Début** $i \longleftarrow 1;$ $j \longleftarrow 1;$ Écrire(x) ; $x \longleftarrow 2i + j;$ $Écrire(j);$ **Fin**

TABLE 6.1 – *Les différentes situations : Concordance, Discordance, Absence.*

considérer pour l'évaluation par similarité d'où la proposition de la mesure de similarité entre plans de solution.

6.3.2 Mesure de similarité proposée

Vu les attentes particulières que nous avions par rapport à l'évaluation d'un plan de solution, notamment la prise en compte de la position (concordance et discordance) ainsi que la prise en compte des opérations supplémentaires (des opérations présentes dans un plan de solution et absentes dans l'autre), nous avons opté à associer deux paramètres de pénalité P_1 et P_2.

Le premier paramètre P_1 est utilisé quand l'enseignant veut évaluer l'ordre des opérations. Dans ce cas, il peut pénaliser le fait de mettre des opérations en désordre avec des valeurs allants de 0 à 1 (0 : pas de pénalité, 1 : pénalité maximale).

Tandis que le deuxième paramètre P_2 a comme but de pénaliser les opérations jugées supplémentaires dans le PS quand elles sont comparées avec le PS référent.

On note $DS \in]-\infty, 1]$ le degré de similarité entre deux plans de solution PS_i (plan de solution de la base) et PS (à évaluer) est décrit respectivement

par les ensembles d'opérations A et B, à partir des termes suivants : A, $A \cap B$, $(A \cap B)$Dis (les opérations en discordance)et B-A (absence).

Le degré de similarité s'écrit comme :

$$DS(A,B) = \frac{A \cap B}{A} - (P_1 \frac{(A \cap B)Dis}{A} + P_2 \frac{B-A}{A}) \tag{6.1}$$

Où :

$\frac{A \cap B}{A}$: représente la similarité opération par opération et il désigne '*la récompense*',

$\frac{(A \cap B)Dis}{A}$: représente la pénalité pour les OB/OE qui sont en discordance,

$\frac{B-A}{A}$: représente la pénalité pour les OB/OE supplémentaires.

Où :

- A : Nombre d'OB/OE du plan de solution de la CD,
- $A \cap B$: Nombre d'OB/OE du plan proposé, qui sont similaires aux OB/OE du PS de la base,
- $(A \cap B)$ Dis : Nombre d'OB/OE du plan de solution proposé qui sont en discordance,
- B-A : Nombre d'OB/OE du plan de solution proposé qui sont supplémentaires,
- P_1 et P_2 sont les paramètres de pénalité.

Selon la manière dont les paramètres P_1 et P_2 sont instanciés, on peut exprimer nos objectifs pédagogiques visés sachant que $P_1, P_2 \in [0,1]$. Par exemple, quand on veut évaluer que le choix des opérations dans une solution donnée, P_1 vaut 0.

Soit par exemple les deux plans de solution présentés dans le tableau 6.2.

PS Référent	**PS proposé**
Début $i \longleftarrow 1$; $j \longleftarrow 1$; $x \longleftarrow 2i + j$; Écrire(x) ; **Fin**	**Début** $i \longleftarrow 1$; $j \longleftarrow 1$; Écrire(x) ; $x \longleftarrow 2i + j$; Écrire(j) ; **Fin**
A=4 ; $A \cap B$=4 ; $(A \cap B)$Dis=2 ; B-A=1	

TABLE 6.2 – *Exemple de mesure de similarité entre deux plans de solution : Référent et proposé.*

En appliquant la formule 6.1 et en choisissant arbitrairement les deux paramètres $P_1 = 0.5$ et $P_2 = 0.2$, on obtient le $DS(PS_{Référent}, PS_{proposé})$ du tableau 6.2 comme suit :

$DS(PS_{Référent}, PS_{proposé}) = \frac{4}{4} - (0.5\frac{2}{4} + 0.2\frac{1}{4})$=0.7

Nous avons donc proposé une nouvelle mesure de similarité paramétrable entre plans de solution qui tient compte d'une certaine spécificité du contexte, à savoir que la similarité doit dépendre de la taille relative du plan de solution référent par rapport à la taille du plan de solution proposé ainsi qu'aux paramètres de pénalisation P_1 et P_2.

6.4 Recherche d'un Plan de Solution référent

Pour comparer le plan de solution proposé avec les plans de solution de la base, on devra mesurer le degré de similarité et retenir le PS le plus proche du PS proposé par l'apprenant.

Cette première étape dans le processus de l'évaluation consiste à trouver le plan de solution le plus proche du plan de solution à évaluer, dit plan de solution référent. Ce plan de solution va constituer le référent avec lequel on va comparer le plan de solution de l'apprenant.

Pour évaluer le plan de solution proposé par l'apprenant, il faut reconnaitre, dans la carte descriptive (CD) du problème traité, le plan de solution référent. Lorsque les degrés de similarités sont calculées, on retient le plan de solution ayant le plus grand degré de similarité. Ce plan de solution peut être un plan de solution référent, comme il peut ne pas l'être. Ceci est lié au seuil d'acceptabilité.

Le seuil d'acceptabilité (SA), appelé aussi *Seuil de signification* exprime à partir de quelle performance nous pouvons évaluer un plan de solution par rapport au référent. Autrement dit, à partir de quelle performance nous pouvons accepter le plan de solution de la base ayant le plus grand degré de similarité en tant que référent. C'est cette définition d'une norme correspondant à nos exigences que nous appelons seuil d'acceptabilité ou seuil de signification.

En comparant le degré de similarité $DS(PS, PS_i)$ avec le seuil d'acceptabilité SA, sachant que PS et PS_i sont les solutions de l'apprenant et la solution la plus proche respectivement, on obtient trois cas possibles suivants.

6.4.1 Cas d'un plan de solution référent trouvé

C'est le cas d'un plan de solution prévu par l'enseignant. Un plan de solution référent trouvé peut se présenter sous deux formes : identique ou similaire.

Plans de solution identiques (PS=PS_i)

Dans ce cas, le degré de similarité DS=1. Autrement dit, le plan de solution proposé a été totalement reconnu, il est donc identique à un plan de solution de la base. La note attribuée est la note du plan de solution.

Le résultat de cette évaluation est présenté à l'apprenant sous forme d'une évaluation sommative commentée.

Plans de solution similaires ($PS \simeq PS_i$)

$SA \leq DS < 1$, le PS proposé a été partiellement reconnu mais il est similaire et comparable au PS trouvé dans la base. Dans ce cas, il n'est pas possible d'utiliser la note du PS trouvé car le PS à évaluer n'est pas identique à 100% à ce PS. Pour cette raison, ce dernier devient un PS référent et on doit calculer la note selon l'étape suivante.

Évaluation sommative

Pour calculer la note dans le cas où le plan de solution proposé n'a pas été identifié comme similaire au plan de solution référent, on se base sur la formule de calcul de similarité 6.1 mais en prenant en compte les poids des opérations constituants le PS afin de l'évaluer selon les objectifs désignés par l'enseignant au départ. Plus précisément, on va pondérer la note du PS référent par une similarité calculée en fonction des poids affectés à ses opérations notée S_{pond} pour une similarité pondérée.

La formule 6.2 calcule la similarité pondérée entre le plan de solution référent représenté par A et le plan de solution proposé représenté par B.

$$S_{pond} = \frac{\sum pds(A \cap B)}{A} - (P_1 \frac{\sum pds(A \cap B)Des}{A} + P_2(1 - \frac{\sum pds(A \cap B)}{A}) \frac{\sum pds(A \cap B)Des}{A}) \tag{6.2}$$

$$Note = Note_A \times S_{pond} \tag{6.3}$$

Où :

- $Note_A$: la note du PS référent,
- pds($A \cap B$) : poids d'une OB/OE en concordance,
- pds(($A \cap B$)Des) : poids d'une OB/OE en discordance.

6.4.2 Cas d'un plan de solution référent non trouvé

Dans ce cas, le plan de solution est considéré comme non-reconnu vu que le degré de similarité DS<SA. La recherche du PS référent était donc infructueuse et l'évaluation du PS est suspendu jusqu'à intervention de l'expert humain. Cette intervention commencera par évaluer le PS de l'apprenant. Ensuite, si celui-ci est estimé pédagogiquement intéressant, il viendra enrichir la carte descriptive associée au problème. Ainsi, toute solution de l'apprenant est forcément évaluée (en direct ou en différé).

6.5 Algorithme du processus d'évaluation

L'algorithme suivant résume le processus d'évaluation de solution algorithmique décrit dans les sections précédentes. Cet algorithme est paramétrable par l'enseignant qui doit définir : le seuil d'acceptabilité SA et les paramètres de pénalités P_1 et P_2.

Algorithme 3: Méthode d'évaluation

Données : PS :plan de solution, PS_i :
$[1..n] lesPSprédéfinis, P_1, P_2, SA, DSmax, DS, S_{pond}$:Réel
Résultat : Évaluation formative/sommative de PS
début
  $DS \longleftarrow 0$;
  pour $i \longleftarrow 0$ *à* n **faire**
    $DS \longleftarrow \frac{PS_i \cap PS}{PS_i} - (P_1 \frac{(PS_i \cap PS)Dis}{PS_i} + P_2 \frac{PS - PS_i}{PS_i})$;
    si *DS>DSmax* **alors**
      $DSmax \longleftarrow DS$
    fin si
  fin pour
  si *DSmax=1* **alors**
    PS est IDENTIQUE à un PS_i ;
    retourner $(Résultat : Note_{PS_i}, Feedback_{PS_i})$
  sinon si $Seuil \leq DSmax < 1$ **alors**
    Le PS est SIMILAIRE à un PS de la base ;
    $S_{pond} \longleftarrow \frac{\sum poids(PS_i \cap PS)}{PS_i} - (P_1 \frac{\sum poids(PS_i \cap PS)Dis}{PS_i} +$
    $P_2(1 - \frac{\sum poids(PS_i \cap PS)}{PS_i}) \frac{\sum poids(PS_i \cap PS)Dis}{PS_i})$
    $Note \longleftarrow Note_{PS_i} \times S_{pond}$;
    retourner $(Résultat : Note, Feedback_{PS_i})$
  sinon
    PS évalué par un enseignant ;
    retourner *(Résultat : Note,Feedback)*
    si *PS est intéressant* **alors**
      Ajouter PS à la base ;
    fin si
    sinon PS Ignoré
  fin si
fin

Algo+, Un système d'évaluation des solutions algorithmiques

7

Sommaire

7.1 Démarche et objectifs pédagogiques d'Algo+ 82
7.2 Architecture d'Algo+ 82
7.2.1 Architecture logicielle 82
7.2.2 Architecture fonctionnelle 83
7.3 Expérimentation d'Algo+ 84
7.3.1 Résultats 85
7.3.2 Discussion et Analyse 90

7.1 Démarche et objectifs pédagogiques d'Algo+

ALGO+ repose sur une démarche pédagogique totalement constructiviste, destinée à l'évaluation des compétences algorithmiques dans un EIAH. La situation d'apprentissage est présentée sous forme de résolution de problèmes. Elle consiste à évaluer une solution algorithmique pour un problème donné, cette évaluation étant formative et sommative.

L'approche déjà proposée dans le chapitre précédent part de l'idée que pour résoudre un problème, il faut le décortiquer en opérations et de répéter ce processus jusqu'à arriver à des opérations connues. Il faut alors savoir combiner celles-ci pour obtenir la solution. Cette approche apporte à l'apprenant un nombre considérable d'avantages dont les plus importants sont :

- Apprentissage de la décomposition : une solution n'est acceptée que si elle est composée de constituants connus,
- Apprentissage de la combinaison d'une solution à partir de constituants connus,
- Concentration sur le problème donné et non sur les sous-problèmes que représentent les opérations connues contribuant à la solution. Celles-ci peuvent, d'ailleurs, faire l'objet d'un apprentissage à part.
- Etc.

7.2 Architecture d'Algo+

7.2.1 Architecture logicielle

Algo+ est un outil d'évaluation des productions d'apprenants en algorithmique. Son objectif final est le développement des compétences algorithmiques sur le terrain.

Il est basé sur une architecture trois tiers. Cette architecture, appelée aussi architecture client/serveur deuxième génération ou client serveur distribué, est caractérisée principalement par l'invariance à l'échelle (elle peut supporter un nombre important de clients). Une telle architecture applique les principes suivants :

- Les données sont toujours gérées de façon centralisée,
- La représentation est toujours prise en charge par le poste client,
- Le traitement est pris en charge par un serveur intermédiaire.

Pour l'architecture, le premier tiers est l'interface qui représente le point d'accès des apprenants et de l'enseignant responsable à travers des applets. Le deuxième tiers s'occupe de l'exécution des applets et de la représentation des pages HTML. Le dernier tiers, les données (partie serveur), doté d'une application serveur qui traite toutes les requêtes transmises par les clients à travers RMI (Remote Method Invocation). Cette technique d'invocation des objets à distance permet, dans notre cas, à l'applet d'utiliser des objets se trouvant sur le serveur.

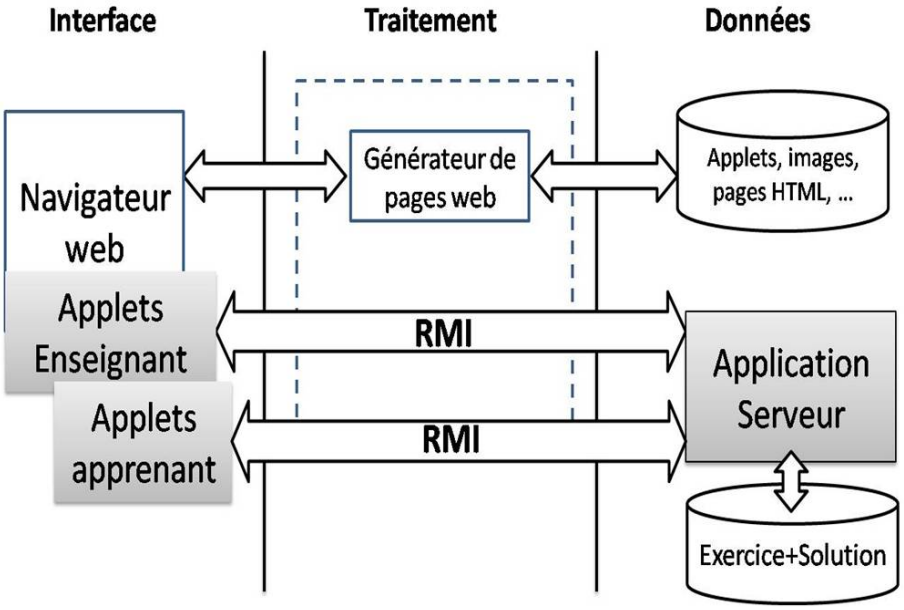

FIGURE 7.1 – *Architecture logiciel d'Algo+.*

7.2.2 Architecture fonctionnelle

Algo+ est un outil d'évaluation des compétences en algorithmiques permettant à l'élève de concevoir des solutions algorithmiques et de les évaluer. Pour cela, l'outil comporte cinq composants : interfaces utilisateurs (apprenant et expert, un gestionnaire d'exercices), un gestionnaire des solutions non-reconnues, un module de reconnaissance et un module d'évaluation (figure 7.2).

Le système est doté principalement de deux interfaces gérées par des Applets Java téléchargées par le navigateur. Une interface apprenants gérée par une Applet contenant l'ensemble des exercices et un éditeur de plans de solutions (figure 7.3).

Un plan de solution est représenté graphiquement sous forme de blocs d'opérations exprimées en pseudo code. Cette représentation, dite verbale sous forme graphique, est proche de la programmation avec un langage informatique et il suffit de traduire les pseudos codes en langage de programmation voulu (avec la définition des opérations de bases avec ce langage) pour que le plan devienne exécutable.

Le composant 'Bloc' (figure 7.3) de l'éditeur de l'apprenant représente l'opération décomposable qui permet, à l'apprenant, d'affiner graduellement sa solution. Notons que le plan de solution ne peut être évalué que s'il est exprimé en fonction d'opérations de base/élémentaires et de structures de contrôles, si nécessaire. Les opérations décomposables sont temporaires et sont remplacées, au fur et à mesure du raffinement, par des opérations de base/élémentaires.

Qu'est ce qu'il se passe dans une session d'évaluation ?

En premier lieu, l'apprenant devrait choisir un exercice et une structure de donnée appropriée pour le résoudre. Une fois l'énoncé est lu et la structure

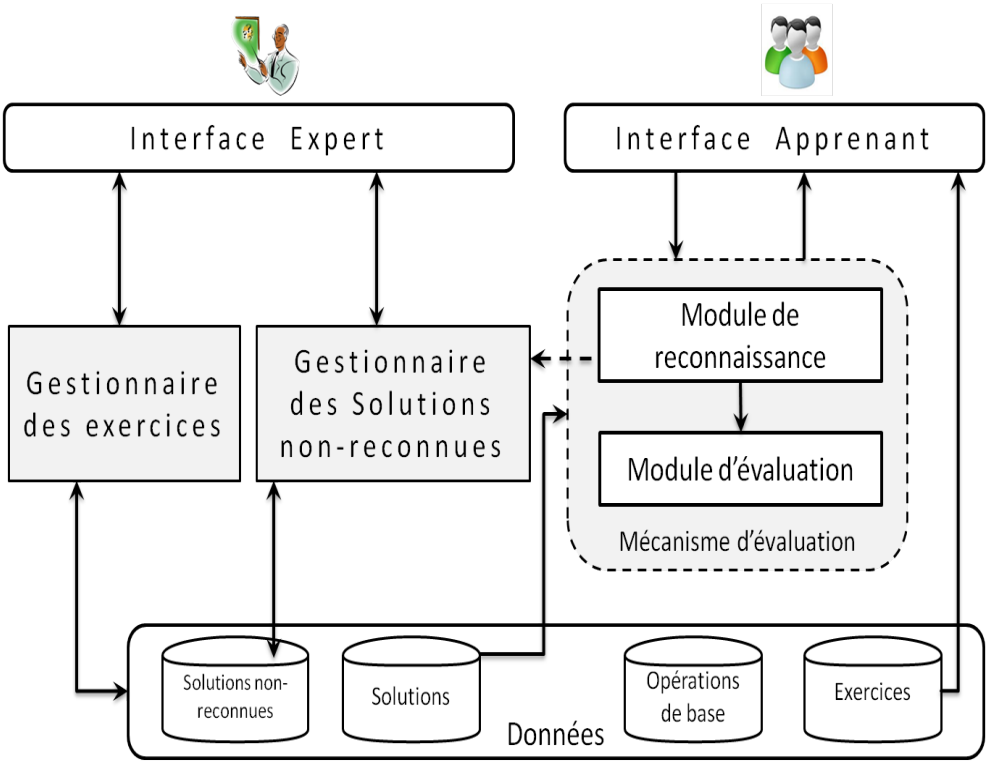

FIGURE 7.2 – *Architecture fonctionnelle d'Algo+.*

de donnée est choisie, l'apprenant peut construire son plan de solution en terme d'opérations de base en utilisant la bibliothèque et des opérations élémentaires à saisir. Une opération de base peut avoir des paramètres (figure 7.4).

Après, dès que l'apprenant finit la construction du plan de solution, il le soumet pour une évaluation. Le résultat retourné est, soit un résultat immédiat sous forme d'une note et un feedback, soit un résultat que l'apprenant recevra en différé.

Qu'est ce qu'il se passe dans une session d'un expert ?

Le rôle de l'expert est de définir les exercices ainsi que leurs solutions. Ainsi, il gère les solutions non-reconnues qui sont classées selon leurs fréquences d'apparition. L'enseignant expert analyse et ajoute les solutions et les erreurs fréquentes à la base. Les résultats des évaluations sont envoyés automatiquement aux apprenants par le système juste après la validation des évaluations par l'expert.

7.3 Expérimentation d'Algo+

Afin de valider la méthode proposée qui consiste à évaluer automatiquement une solution algorithmique, nous l'avons confrontée à l'évaluation humaine (Bey and Bensebaa 2011). L'expérimentation s'est déroulée sur 29 étudiants durant le deuxième semestre 2010. Un enseignant expert en algorithmique est désigné pour évaluer les réponses des étudiants. Ce même enseignant configure le système Algo+. On entend par configurer, la défini-

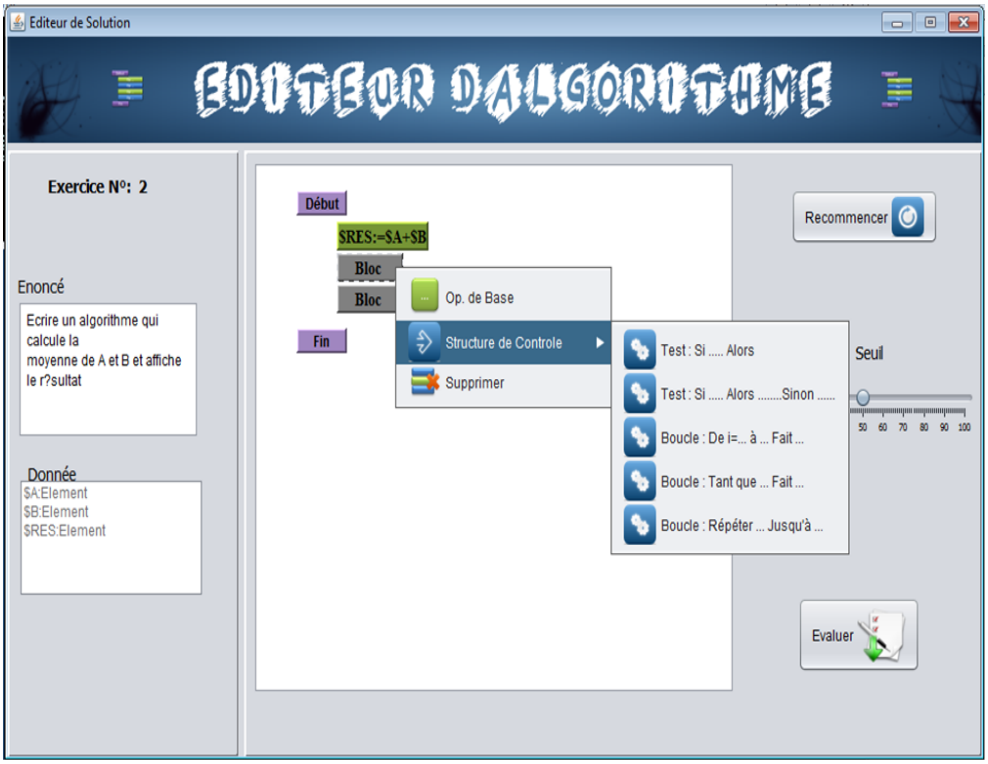

FIGURE 7.3 – *Éditeur de plan de solution.*

tion des notes des solutions, les paramètres P_1 et P_2 ainsi que le seuil SA. De plus, quelques solutions types ont été ajoutées à la base des solutions.

Les étudiants ont été invités à utiliser le système Algo+ pour résoudre un exercice sur les listes. Les mêmes réponses (29 réponses) ont été transférées à l'enseignant pour les corriger.

7.3.1 Résultats

Dans les 29 solutions données par les étudiants, seulement 24 solutions ont pu être évaluées par Algo+, 5 solutions n'ont pas été reconnues par le système. Donc, nous disposons pour valider la méthode d'évaluation, un échantillon total de 48 notes :

1°. Notes de l'enseignant expert (24 notes),

2°. et notes du système (24 notes).

Le tableau suivant 7.1 présente les différents taux de reconnaissance des solutions par le système Algo+.

État	Reconnu	Non-reconnu
-	82.76%	17.24%
Identique	29.17%	-
Similaire	70.83%	-

TABLE 7.1 – *Taux de reconnaissance des solution par Algo+.*

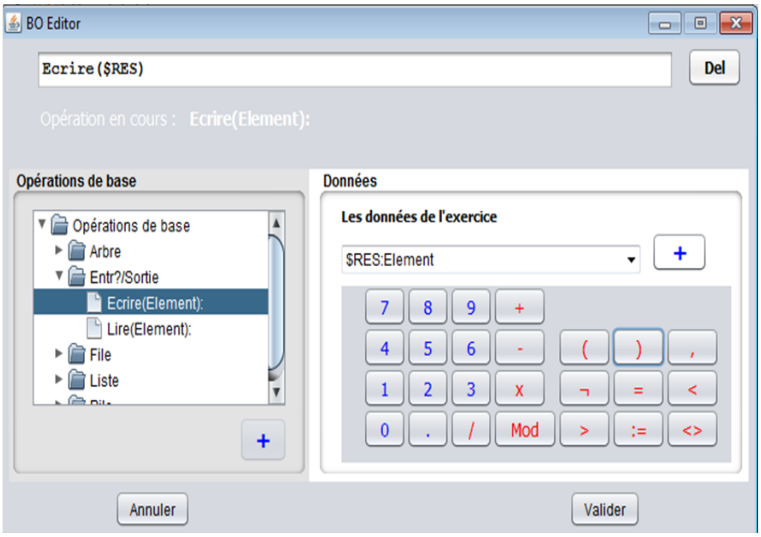

FIGURE 7.4 – *Éditeur des paramètres d'une opération de base.*

Le tau de reconnaissance observé est de 82.76% dont 29.17% identique et 70.83% similaire. Le tau de reconnaissance change en fonction des deux paramètres de pénalités et le seuil. Afin de pouvoir observer l'effet des deux paramètres sur le tau de reconnaissance des plans de solution, on va faire varier des valeurs différentes. Par exemple, pour un seuil de 0.52, la figure 7.5 montre le changements des taux selon une variation des deux paramètres P_1 et P_2 avec des valeurs minimales ($P_1 = 0$ et $P_2 = 0$), des valeurs maximales ($P_1 = 1$ et $P_2 = 1$) et des valeurs choisies par l'expert ($P_1 = 0.2$ et $P_2 = 0.05$).

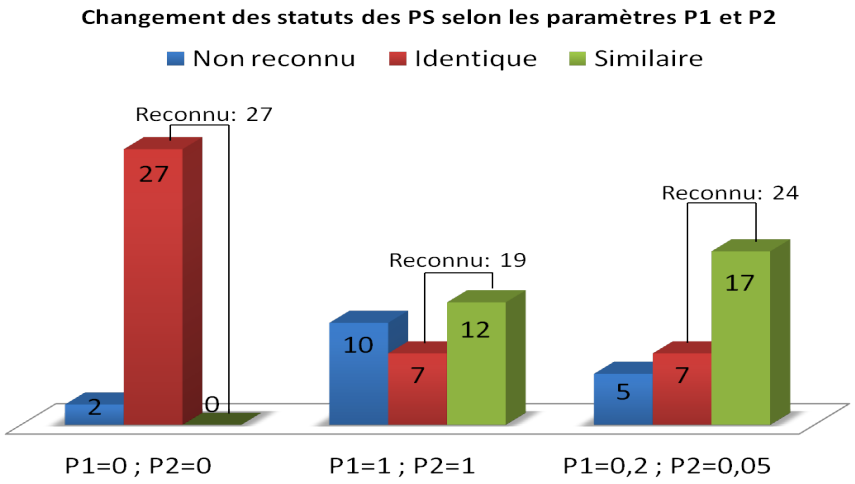

FIGURE 7.5 – *Processus du reconnaissance de PS en fonctions de P1 et P2 avec un SA=0.52.*

La première question à laquelle nous cherchons à répondre en terme statistique concerne la corrélation entre les notes d'Algo+ et les notes de l'expert.

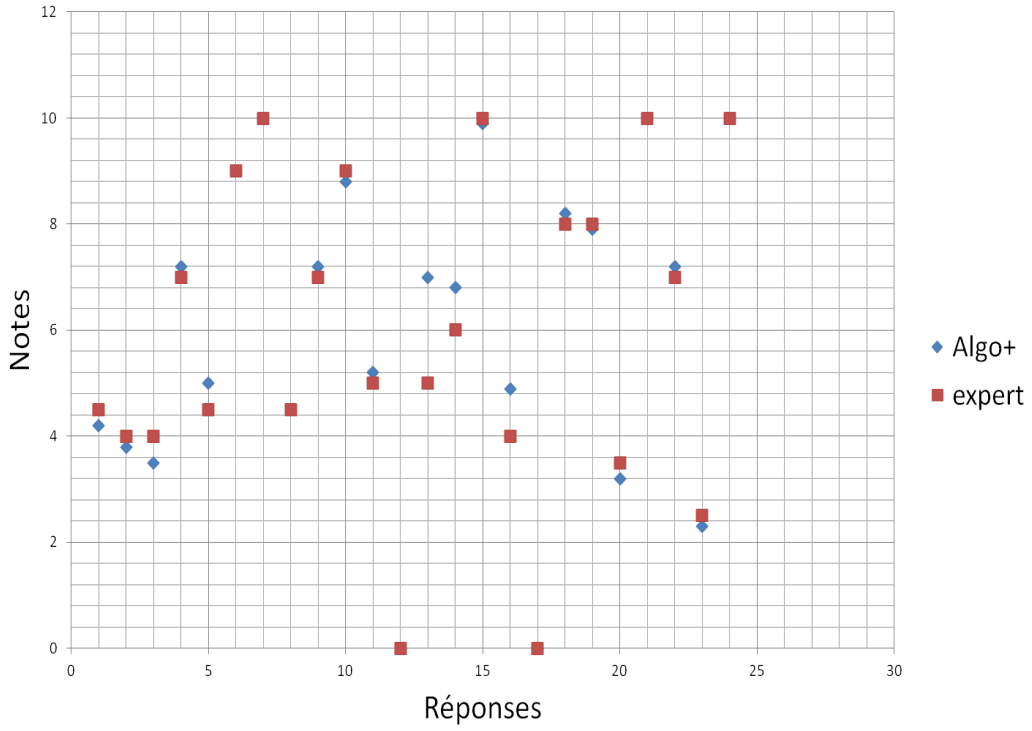

FIGURE 7.6 – *Nuage des deux notes : Expert VS Système.*

Un coefficient de corrélation affichant une valeur élevée (figure 7.7) signifierait que l'enseignant expert et le système Algo+ notent de la même manière pour les 24 réponses corrigées.

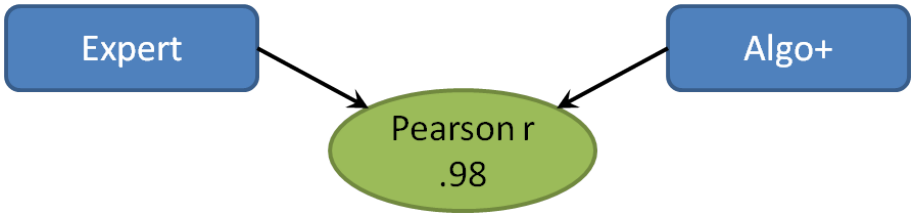

FIGURE 7.7 – *Efficacité des notes d'Algo+/Expert.*

Les résultats obtenus auprès de l'expert ont été comparé aux résultats fournis par Algo+. Les différents paramètres des distributions des 48 notes attribuées par Algo+(24 notes) et l'Expert(24 notes) sont mentionnés dans le tableau ci-après (Tableau 7.2).

Notes	Moyenne	Écart type	Médiane	Min	Max	Étendu
Algo+	6.08	3	6.9	0	10	10
Expert	5.94	2.98	5.5	0	10	10

TABLE 7.2 – *Paramètres des distributions des notes attribuées par Algo+ et l'Expert.*

Les répartitions reflètent tout d'abord un phénomène statistique classique puisque les distributions se rapprochent de la loi gaussienne avec une concentration des notes autour de la moyenne et de la médiane.

Un deuxième constat concerne les deux moyennes (Moy_système=6.08, Moy_expert=5.94). On peut clairement remarquer qu'il n'y a pas une différence entre la moyenne des notes d'Algo+ et la moyenne des notes de l'expert.

Les écart-types, qui fournissent une mesure statistique de la dispersion moyenne des notes, ne présentent pas également des valeurs élevées (de 2.98 à 3 points).

Par ailleurs, les écarts entre les deux notes varient entre -2 et 0.5 (figure 7.8) avec une concentration des écarts qui se rapprochent du 0. Un seul écart de 2 points (4.16%) est observé.

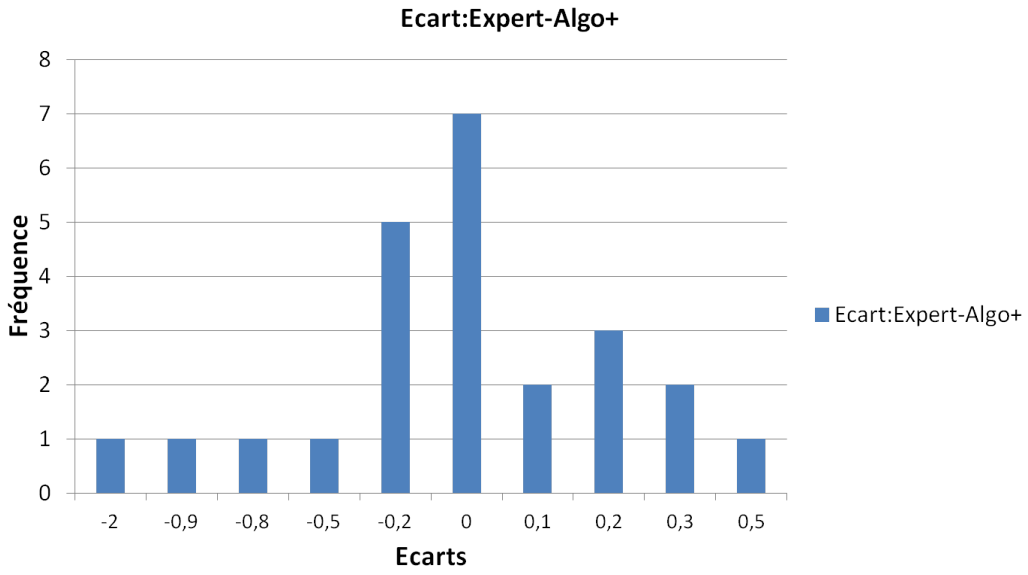

FIGURE 7.8 – *Distribution des écarts Expert/Algo+.*

On remarque ici que les écart-types estimés des deux notes d'Algo+ et de l'expert sont respectivement 3 et 2.98. Aussi, pour certaines notes attribuées par Algo+ sont supérieures aux notes attribuées par l'expert. Mais cette différence est-elle significative ? Sachant qu'il n'y pas une différence significative entre les deux moyennes (Moy_système=6.08, Moy_expert=5.94).

Il est en pratique insuffisant de comparer ces deux valeurs et nous proposerons un test plus loin. Ce test a pour objectif de tester deux hypothèses :

H0 : il n'y a pas une différence significative entre notes attribuées par Algo+ et les notes attribuées par l'expert,

H1 : il y a une différence significative entre notes attribuées par Algo+ et les notes attribuées par l'expert

Autrement dit, soit μ la médiane de la population des mesures *Note_Expert - Note_Algo+*. On testera l'une contre l'autre les hypothèses :

(H 0) : $\mu = 0$

(H 1) : $\mu < 0$

Vu que notre échantillon est apparié et de taille petite (n=24, les solutions qui ont été reconnues) avec des variables ordinales, nous avons effectué le test de Wilcoxon.

Le test de Wilcoxon est une adaptation à la comparaison de deux moyennes, pour deux échantillons appariés. Il calcule les différences (positives et négatives), les traduit en rang, puis compare la distribution de la somme des rangs pour les deux échantillons, et estime la différence observée.

Le test prend en compte la grandeur des différences entre les deux échantillons des notes et donne plus de poids à une paire qui montre une large différence entre les deux conditions qu'à une paire ayant une faible différence.

Nous rangeons par ordre croissant les écarts entre les notes de l'expert et les notes d'Algo+ (écarts en valeurs absolue), puis nous associons à chaque écart son signe. Nous calculons la somme S_p des rangs des écarts positifs et la somme S_n des rangs des écarts négatifs. Si H0 est vraie, on s'attend à ce que ces deux sommes soit presque égales. La statistique de test est la plus petite des deux sommes. Pour évaluer la signification, nous utilisons la table des signes et rangs de Wilcoxon (annexe A.4) qui donne le seuil de la région de rejet.

Après le calcul, nous avons obtenu 7 paires de note qui sont égaux sur un total de 24. De telles paires sont abandonnées. N est alors égal au nombre de paires dont la différence entre les traitements n'est pas nulle (n=17) avec un S_p=65.5. Or dans la table des signes et rangs de Wilcoxon, nous voyons que si n = 17, le test bilatéral de niveau 5% ne rejette pas H0.

Un autre constat concerne l'échelle de notations utilisé par l'enseignant expert. Par exemple dans le cas de ces notes, l'enseignant expert a utilisé un échelle de 0.5 pt (i.e. 0.5 ;1 ;1.5 ;2 ;2.5 ;...) contrairement aux notes du système qui ne cadrent pas avec cette échelle. Pour obtenir des notes qui ressemblent aux notes de l'expert, on peut effectuer des arrondissements de ses notes au multiples de 0.5. Par exemple une note de 7.9 donnée par le système devient 8, etc.

En appliquant ce traitement, on aperçoit un changement dans les taux des notes exactes (table 7.3)

Algo+	Exacte	plus que l'expert	moins que l'expert
Notes obtenues	29.17%	37.5%	33.33%
Notes arrondies	70.83%	16.67%	12.5%

TABLE 7.3 – *Notation du système Algo+ par rapport à l'expert.*

Par contre, ce traitement n'affecte pas la signification des résultats obtenues au début. Les résultats se trouvent en annexe A.3.

7.3.2 Discussion et Analyse

Nous évoquons dans cette partie quelques questions et réflexions soulevées par l'application de notre modèle aux données disponibles. Nous comptons ainsi montrer les avantages et les limites de cette approche, pour ouvrir la voie sur les solutions que nous voyons pour l'améliorer.

La comparaison des deux groupes de notes n'a pas relevé une différence significative entre la moyenne des notes attribuées par l'expert humain (moyenne=6.08,écart-type=3) et la moyenne des notes attribuées par Algo+ (moyenne=5.94, écart-type=2.98).

D'une autre part, les résultats de l'analyse de corrélation en utilisant le coefficient de Pearson (r=.98) indique une forte association linéaire positive entre les deux groupes des notes.

Le test de Wilcoxon a confirmé l'hypothèse et a montré qu'il n'existe pas une différence significative entre les notes attribuées par Algo+ et les notes attribuées par l'Expert humain pour p = .05.

En général, les résultats de cette étude montre l'efficacité de la méthode d'évaluation dans le contexte de résolution du problème en algorithmique. Cette efficacité revient à la possibilité de paramétrer le système de l'évaluation et de donner à l'enseignant la main à exprimer ses mesures en termes de :

- Paramètres P_1 et P_2 pour calculer le degré de similarité et la note,
- Poids des Opérations selon les objectifs voulus par les exercices,
- Seuil d'acceptabilité d'une solution,
- Notes des solutions de la base.

De plus, autre que le fait qu'elle soit sommative, cette méthode d'évaluation est formative. Elle permet à l'apprenant de s'exercer et de s'entrainer à concevoir des algorithmes et d'apprendre à travers les erreurs récurrentes et les feedbacks fournis.

En ce qui concerne le mécanisme de l'évaluation, l'expérimentation de cette approche a révélé une certaine anomalie dans la recherche de la solution la plus proche. En effet, deux plans solutions comportant des opérations identiques et indépendantes dans un ordre différent constituent deux plans de solutions différents. Ceci a eu pour conséquence d'accroître exponentiellement la taille de la base des plans de solution.

Pour bien comprendre ce phénomène, l'exemple suivant représente un plan de solution qui remplace toutes les occurrences de X de la liste L par Y tout en calculant la somme des valeurs de X.

Pour ce plan de solution, on utilise une opération de base Remplacer(L(i),Y).

Ces deux exemples représentent deux algorithmes qu'on peut trouver dans la base des solutions et qui sont considérés comme distincts alors qu'ils sont équivalents.

La distinction entre deux algorithmes équivalents est due à la modélisation d'une solution algorithmique qui n'apporte pas beaucoup d'information.

```
début
    sum ⟵ 0 ;
    pour i ⟵ 0 à n faire
        si L(i)=X alors
            sum = sum + X ;
            Remplacer(L(i), Y);
        fin si
    fin pour
fin
```

Algorithme 4: Plan de solution version 1°.

```
début
    sum ⟵ 0 ;
    pour i ⟵ 0 à n faire
        si L(i)=X alors
            Remplacer(L(i), Y);
            sum = sum + X ;
        fin si
    fin pour
fin
```

Algorithme 5: Plan de solution version 2°.

Elle est seulement basée sur la concordance et la discordance des opérations ainsi que les opérations supplémentaires, donc, dès que deux opérations changent de positions dans un autre algorithme, ce dernier est considéré comme différent même si ses deux opérations sont indépendantes l'une de l'autre. Ce problème se traduit par le fait que les dépendances entre les opérations dans une solution ne sont pas représentées dans la modélisation ni dans le processus d'évaluation.

Conclusion

Parmi les objectifs de cet outil est l'amélioration des compétences de résolution des problèmes en algorithmique à travers une évaluation formative.

Avec cet outil, on peut évaluer des productions d'apprenants et assurer l'objectivité dans la correction ainsi qu'une grande fidélité à l'évaluation, même pour des grands groupes d'apprenants, contrairement à l'évaluation de l'enseignant humain qui est toujours biaisée par des facteurs liés à l'évaluateur (sa personnalité, etc.), ou à l'état de la copie.

Notons toutefois que l'enrichissement de la base avec de nouvelles solutions et d'erreurs récurrentes sera primordiale pour assurer une évaluation plus efficace et une sollicitation de l'enseignant expert moins fréquentes.

Conclusion et perspectives

7.4 Conclusion

Bilan

L'arrivée massive de l'internet et le grand essor de l'utilisation des nouvelles technologies de l'information et de la communication pour renforcer l'apprentissage humain a engendré une énorme production des Environnements Informatiques pour l'Apprentissage Humain (EIAH). Les auteurs de ces environnements s'intéressent généralement à faire apprendre une masse de connaissances sans s'assurer de son acquisition. L'évaluation, cet acteur fréquemment manquant dans plusieurs activités pédagogique et souvent mis à part, est une véritable garante de réussite et de mesure d'efficacité. Elle est l'instrument d'orchestration de n'importe quelle formation. Son rôle est de promouvoir l'apprentissage(figure 7.9).

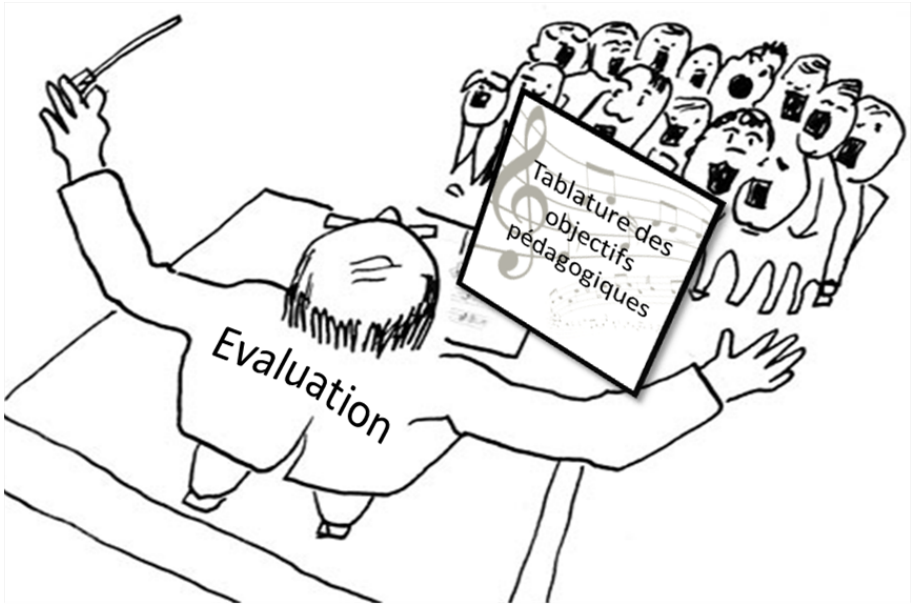

Figure 7.9 – *L'évaluation : le chef d'orchestre d'un processus d'apprentissage.*

L'évaluation automatique permet des retours immédiats. Des logiciels y recourent donc pour seconder les enseignants dans leurs corrections ou pour assister les apprenants.

Généralement, les apprenants avaient connaissance du corrigé type dont l'intérêt pédagogique est limité. En effet, l'enseignant ne leur précise pas ce qui est erroné dans leurs réponses mais il leur indique simplement ce qu'il fallait faire : méthodes utilisées et résultats.

Il sera alors difficile à l'élève, qui a utilisé une démarche différente de celle du corrigé-type, de trouver ses erreurs notamment celles d'ordre conceptuel et procédural.

L'évaluation des connaissances procédurales et des compétences dans les milieux professionnels ou éducatifs n'est guère la même que l'évaluation des simples savoirs théoriques et connaissances déclaratives.

Dans la littérature, la plupart des systèmes d'évaluation des productions des apprenants en algorithmique et en programmation se basent sur deux approches : exécution de programmes (approche dynamique) ou analyse de programme (approche statique). La première consiste généralement à comparer le résultat de sortie avec un résultat défini à priori ou par un jeu d'essai. La principale limite de ces approches est qu'on ne peut pas évaluer les compétences de conception d'algorithme étant donné qu'on ne voit que le résultat du programme et non sa conception (boite noire). La deuxième approche, quant à elle, consiste à analyser le programme par rapport à des programmes prédéfinis. Cette technique favorise l'occurrence des solutions stéréotypées ce qui réduit la liberté des apprenants dans la conception des solutions et les prive de l'espace large et riche des solutions possibles.

Ce challenge nous a conduits à proposer une approche d'évaluation du savoir-faire.

L'approche proposée a été conçue dans le but d'évaluer n'importe quelle solution algorithmique proposée par l'apprenant et indépendamment de n'importe quel langage de programmation. Cette prise en compte de la variabilité de solutions correctes et erronées pour un problème donné permet non seulement d'évaluer n'importe quelle solution proposée mais aussi d'offrir un feedback instantané sur la solution reconnue dans la base.

L'approche d'évaluation proposée assure une évaluation trilatérale : évaluation sommative (quantifier les compétences et les connaissances réellement possédées), formative (aider l'apprenant à progresser) et diagnostique (diagnostiquer les lacunes).

Contribution #1 : *un modèle d'évaluation du savoir-faire*

Le modèle a été conçu dans le but d'évaluer n'importe quelle production d'apprenant. Il est basé principalement sur une stratégie de résolution de problème en fonction d'opérations ou d'actions connues dans le domaine de l'algorithmique. L'évaluation consiste à reconnaitre la solution de l'apprenant parmi les solutions de la base. Cette dernière contient des solutions correctes et erronées qui sont déjà évaluées par un expert humain. Cette conception socio constructiviste de prise en compte des erreurs favorise l'acquisition des compétences et des incompétences de résolution de problème algorithmique qui sont conceptualisées à travers les solutions. Dans le cas où une solution proposée par un apprenant n'est pas reconnue dans la base, un expert du domaine, l'enseignant, est sollicité pour l'évaluer et l'ajouter à la base sil elle est jugée intéressante. La base des solutions est donc évolutive.

Cette technique, « l'intervention humaine », est un remède au problème

de variabilité des solutions. Et le taux de sollicitation de l'expert humain diminue au fur et à mesure de l'enrichissement de la base de solutions.

En effet, l'avantage est que toute démarche proposée par l'apprenant et reconnue par le système se verra enrichie par des explications, démonstrations, etc.

Contribution #2 : *une évaluation sommative basée sur une mesure de similarité*

En plus du feedback fourni lors de la reconnaissance d'une solution, une note est calculée automatiquement en fonction d'une note prédéfinie de la solution de la base et selon le degré de similarité qui joue le rôle de pondérateur de la note prédéfinie.

Contribution #3 : *un système d'évaluation paramétrable des productions des apprenants en algorithmique*

Le système d'évaluation du savoir-faire algorithmique Algo+ est un système qui se base essentiellement sur deux formules. Une formule de reconnaissance définie en fonction de deux paramètres : le choix des éléments de la solution et l'enchainement de ces derniers. Une deuxième formule est utilisée afin de calculer la note en fonction des poids attribués à la solution de la base.

La validation du système a montré des résultats encourageants :

- L'utilisation des opérations de haut niveau (les opérations de base) lors de la conception d'une solution permet aux apprenants de mieux concentrer sur la conception de la solution,
- Un autre atout du à l'utilisation des opérations de base est de nous permettre d'évaluer non seulement le choix de l'opération pour la résolution du problème mais aussi son paramétrage (passage de paramètres),
- L'évaluation sommative est très proche de l'évaluation de l'enseignant (98.7%), cela s'explique par le fait que le système est un système mono expert, i.e. un seul expert qui définit les paramètres des formules.

Un des problèmes rencontrés est que la modélisation d'une solution algorithmique dans Algo+ n'est pas assez riche. Plus précisément, dans Algo+, il n'y a que le choix et l'enchainement des opérations qui sont pris en compte, alors que les dépendances entre ces opérations ne sont pas prises en considération. Par conséquent, des solutions équivalentes sont considérées comme différentes.

7.5 Perspectives

Notre modèle d'évaluation du savoir-faire fait que notre travail ne s'arrête pas là et que des perspectives expérimentales et de recherche présentant

plusieurs facettes, plus ou moins abouties, seront poursuivies à court et à long terme.

7.5.1 Perspectives expérimentales

Nous menons actuellement une étude de multi corrections avec plusieurs correcteurs de différentes institutions nationales et internationales. L'objectif de cette étude est de valider la méthode à grande échelle en comparant les évaluations données par plusieurs enseignants avec l'évaluation donnée par l'outil Algo+. Les résultats obtenus vont nous permettre d'étudier le comportement d'Algo+ face aux différences de corrections : intra-correcteur, inter-correcteurs et interinstitutionnelles. Autrement dit, on va étudier les résultats donnés par Algo+ par rapport aux effets idiosyncrasiques liés aux différences personnelles, culturelles et institutionnelles.

7.5.2 Perspectives de recherche

A court terme

Notre travail et les résultats que nous avons obtenus montrent qu'il est important et utile de s'intéresser aux deux aspects relevés pendant l'expérimentation de la méthode proposée : la différenciation entre deux algorithmes équivalents et le feedback explicatif.

- **Une représentation interne d'une solution algorithmique**

  L'indifférenciation du mécanisme de reconnaissance entre deux algorithmes équivalents est due à la représentation d'une solution algorithmique qui n'apporte pas beaucoup d'informations. Elle est seulement basée sur l'ordre et l'enchainement des opérations. Dans cette représentation, dès que deux opérations changent de positions dans un autre algorithme, ce dernier est considéré comme différent même si ses deux opérations sont indépendantes l'une de l'autre. Ce problème est du au fait que les dépendances entre les opérations dans une solution ne sont pas représentées dans la modélisation ni dans le processus d'évaluation.
  Cependant, l'utilisation d'une représentation interne d'une solution algorithmique en utilisant le formalisme du plan (plan canonique) inspiré du domaine de compréhension de programmes (Bey Bensebaa, 2011), nous permet une meilleure explicitation des dépendances (dépendances de donnée et dépendances de contrôle) qui n'apparaissent pas dans la représentation externe d'une solution algorithmique (figure 7.10).
  Plus précisément, quand une solution algorithmique est soumise à une éventuelle évaluation, elle sera désormais transformée à une forme canonique puis évaluée.

- **Un diagnostic plus précis**

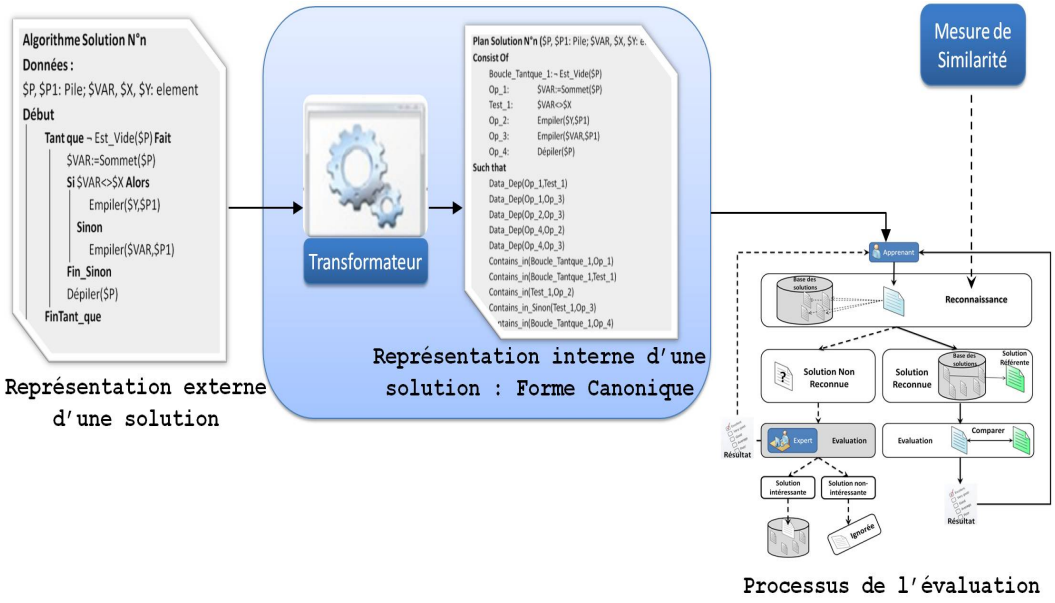

FIGURE 7.10 – *L'évolution de la méthode proposée.*

En effet, pour une solution reconnue, le résultat d'une évaluation est une note plus un feedback explicatif. Ce feedback restreint offre une explication globale de la solution, i.e. l'apprenant reçoit un commentaire sur la solution dans sa totalité et non pas sur ce qu'il a commis comme erreur exactement. A cet effet, une évaluation diagnostic plus précise s'avère très utile afin de situer les erreurs et aider l'apprenant à les franchir d'une part, et d'enrichir le modèle élève d'une autre part. Cette solution est réalisable grâce à la représentation interne ajoutée. Grâce à cette forme canonique, les informations tirées de la solution proposée (opérations utilisées, paramètres des opérations, structures de contrôle, dépendances de données et dépendances de contrôles) vont nous offrir aussi la possibilité de diagnostiquer la solution proposée et savoir exactement le type d'erreurs commises : erreur de choix d'opération ou de structures de contrôles, ou encore de passage de paramètres des opérations (Bey Bensebaa, 2013).

- **Une nouvelle mesure de similarité**

  Sur le plan processus d'évaluation, en modifiant la représentation des solutions à comparer, lors du processus de reconnaissance, il sera nécessaire de changer la mesure de similarité proposée au départ. Puisque cette dernière a été conçue par rapport à un formalisme bien précis (l'ordre et l'enchainement des opérations). On peut se référer aux mesures de similarité dans le domaine des sciences cognitive et choisir la mesure de similarité qui nous offre une meilleure adéquation entre le but poursuivi et le comportement effectif de la mesure.

A long terme

Comme il a été montré, malgré que cette approche a été conçue pour l'évaluation des productions en algorithmiques, elle peut, très facilement, être adaptée à n'importe quel domaine manipulant des connaissances procédurales.

Nous envisageons de poursuivre notre travail vers une méthode générique d'évaluation du savoir-faire. Nous faisons l'hypothèse que l'utilisation du formalisme du plan proposé en compréhension de programme associé à une bonne mesure de similarité peut nous permettre de créer des systèmes d'évaluations formative, diagnostic et même sommative.

La méthode générique peut comporter les éléments suivants :

- **Une bibliothèque d'opérations** : ces opérations sont des actions connues dans le domaine d'application. Cette bibliothèque est établie par les experts du domaine.
- **Une base de contraintes** : les différentes dépendances et contraintes qui peuvent exister entre les opérations du domaine d'application.
- **Un transformateur** : qui transforme la solution en une forme canonique et génère les différentes contraintes qui expriment les dépendances entre les opérations d'une solution.

Ceci constitue bien évidemment une liste non exhaustive des perspectives possibles, l'objectif étant d'améliorer et de faire évoluer la méthode tout en gardant son ossature et aussi les avantages précédemment acquis.

Bibliographie

Abdourahmane, M. (2008), Les modalités d'évaluation et de controle des apprentissages, Technical report, Atelier TRANSFER 3.2 FLSH.

Ala-Mutka, K. M. (2005), 'A survey of automated assessment approaches for programming assignments', *Computer Science Education* **15**(2), 83–102.

Amit, K. M., Chittaranjan, M. and Christopher, M. P. R. (2006), Architecture of an automatic program evaluation system, *in* 'CSIE Proceedings'.

Angelo, T. A. and Cross, K. P. (1993), *Classroom Assessment Techniques : A Handbook for College Teachers*, San Francisco : Jossey-Bass.

Ardoino, G. and Berger, J. (1989), *D'une évaluation en miette à une évaluation en actes, le cas des université*, Paris, ANDSHA-Matrice.

Astolfi, J.-P. (1997), *L'erreur, un outil pour enseigner*, Paris : ESF.

Auffarth, B. and L, M. (2008), System for Automated Assistance in Correction of Programming Exercises, *in* 'V International Congress University Teaching and Innovation', Lleida (Spain), pp. pp. 104 (1–9).

Ben-Ari, M. (2001), 'Constructivism in computer science education', *Journal of Computers in Mathematics and Science Teaching* **20**(1), 45–73.

Benedict, D. B. (1986), 'Some difficulties of learning to program', *Journal of Educational Computing Research* **2**(1).

Bensalem, H. and Bensebaa, T. (2010), Contribution to the improvement of learning algorithmic, *in* 'Proceeding of the 10th International Educational Technology Conference, Turkey', IETC'10, Istanbul, Turkey, pp. 457–521.

Berry, R. (2008), *Assessment for Learning*, Hong Kong University Press.

Bey, A. and Bensebaa, T. (2011), Algo+, an assessment tool for algorithmic competencies, *in* 'IEEE Engineering Education 2011 Learning Environments and Ecosystems in Engineering Education', EDUCON'11, Amman, Jordan, pp. 941–946.

Bey, A., Bensebaa, T. and Bensalem, H. (2010*a*), Assessment of algorithmic competences in tel environment, *in* '10th International Educational Technology Conference', IETC'10, Istanbul, Turkey.

Bey, A., Bensebaa, T. and Bensalem, H. (2010*b*), 'Easel : Evaluation of algorithmic skills in an environment learning', *World Academy of Science, Engineering and Technology* **42**, 64–67.

Bey, A., Bensebaa, T. and Bensalem, H. (2010*c*), Evaluation des compétences algorithmiques dans un environnement d'apprentissage, *in* 'Proceeding of LEAFA 2010, The First International Conference in E-LEArning For All', LEAFA'10, Hammamet, Tunis.

Black, P. and Wiliam, D. (1998), 'Assessment and classroom learning', *Assessment in Education* **5**(1), 7–74.

Bloom, B. S., Engelhart, M. B., Furst, E. J., Hill, W. H. and Krathwohl, D. R. (1956), *Taxonomy of educational objectives. The classification of educational goals. Handbook 1 : Cognitive domain*, Longmans Green.

Blumenstein, M., Green, S., Nguyen, A. and Muthukkumarasamy, V. (2004), An experimental analysis of game : a generic automated marking environment, *in* 'Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2004, Leeds, UK', ACM, pp. 67–71.

Bonar, G., J. and Cunningham, R. (1988), *Bridge : Tutoring the Programming Process*, Lawrence Erlbaum Associates, chapter Intelligent Tutoring Systems : Lessons Learned.

Bonniol, J. (1986), *Recherches et formations : pour une problématique de l'évaluation formative*, De Boeck, chapter L'évaluation : approche descriptive ou prescriptive ?

Brown, J. S. and Burton, R. R. (1978), 'Diagnostic models for procedural bugs in basic mathematical skills', *Cognitive Science* **2**, 155–192.

Bull, J. (1999), 'Computer-assisted assessment : Impact on higher education institutions', *Journal of Educational Technology and Society* **3**(2), 123–126.

Caignaert, C. (1988), 'Etude de l'évolution des méthodes d'apprentissage et de programmation'.

Chen, S. (2005), Iconic programming for flowcharts, java, turing, etc, *in* 'Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2005, Caparica, Portugal', ACM, pp. 104–107.

David, J.-P. (2003), 'Modélisation et production d'objets pédagogiques', *Sciences et Techniques éducatives* .

De Ketele, J. and Roegiers, X. (2009), *Méthodologie du recueil d'informations*, Bruxelles : De Boeck, chapter Collection Pédagogie en développement, Méthodologie de la recherche, p. 226.

De Landsheere, G. (1979), *Dictionnaire de l'évaluation et de la recherche en éducation : avec lexique anglais-francais*, Presses universitaires de France, Paris.

Deek, F. and McHugh, J. (1999), 'A survey and critical review of tools for learning programming', *Journal of Computer Science Education* **8**(2), 130–178.

Deek, F. and McHugh, J. (2000), Prototype tools for programming, *in* 'Proceedings of the ED-MEDIA World Conference on Educational Multimedia, Hypermedia and Telecommunications, Canada', pp. 255–259.

Delozanne, E. and Grugeon, B. (2004), 'Pépites et lingots : des logiciels pour faciliter la régulation par les enseignants des apprentissages en algèbre', *Cahiers éducation et Devenir Hors série, « Les TIC à l'école : miracle ou mirage ? »* pp. 82–92.

Dominique, P. (1996), 'Aide à la démonstration en géométrie : le projet mentoniezh', *Sciences et Techniques Educatives* **3**(2), 227–256.

Dominique, P. (1998), 'Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève', *Sciences et Techniques Educatives* **5**(2).

Durand, G. (2006), Vers une scénarisation de l'évaluation en eiah.l'évaluation comme activité scénarisable dans un dispositif de scénarisation pédagogique, *in* 'RJC-EIAH'.

Edwards, S. H. (2003), 'Improving student performance by evaluating how well students test their own programs', *ACM Journal of Educational Resources in Computing* **3**(3), 1–24.

Edwards, S. H. and Pugh, W. (2006), 'Toward a common automated grading platform', *In SIGCSE'06 : technical symposium on Computer science education, ACM* .

Fincher, S., Lister, R., Clear, T., Robins, A., Tenenberg, J. and Petre, M. (2005), 'Multi-institutional, multi-national studies in csed research : Some design considerations and trade-offs'.

Fincher, S., Petre, M., Tenenberg, J., Blaha, K., Bouvier, D., Chen, T.-Y., Chinn, D., Cooper, S., Eckerdal, A., Johnson, H., McCartney, R., Monge, A., Mostrom, J. E., Powers, K., Ratcliffe, M., Robins, A., Sanders, D., Schwartzman, L., Simon, B., Stoker, C., Elliott Tew, A. and VanDeGrift, T. (2004), A multi-national, multi-institutional study of student-generated software designs, *in* 'Proc. 4th Finnish/Baltic Sea Conference on Computer Science Education : Koli Calling', Helsinki University of Technology, pp. 20–27.

Gilly, M. (1980), *Maitre-élève, roles institutionnels et représentations*, Paris : Presses universitaires de France.

Ginat, D. (2006), On novices local views of algorithmic characteristics, *in* 'Informatics Education - The Bridge between Using and Understanding Computers, International Conference in Informatics in Secondary Schools - Evolution and Perspectives, ISSEP 2006, Vilnius, Lithuania, November 7-11, 2006, Proceedings', Lecture Notes in Computer Science, Springer, pp. 127–137.

Goldman, K. J. (2004), A concepts-first introduction to computer science, *in* 'SIGCSE', ACM, pp. 432–436.

Green, B. F., Bock, R. D., Humphreys, L. G., Linn, R. L., Lord, F. M. and Reckase, M. D. (1984), 'Technical guidelines for assessing computerized adaptive tests', *Journal of Educational Measuremen* pp. 347–360.

Guzdial, M. L. (1998), 'Supporting programming and learning-to-program with an integrated cad and scaffolding workbench', *Interactive Learning Environment* **6**(1), 143–179.

Hadji, C. (1977), *L'évaluation démystifiée*, EME Editions Sociales Francaises (ESF).

Hadji, C. (1990), *Evaluation, les règles du jeu*, ESF.

Hanson, S. J. (1990), Conceptual clustering and categorization : Bridging the gap between induction and causal models, *in* Y. Kodratoff and R. S. Michalski, eds, 'Machine Learning : An Artificial Intelligence Approach', Vol. 3, Kaufmann, pp. 235–268.

Harlen, W. and Deakin, R. (2002), 'A systematic review of the impact of summative assessment and tests on students' motivation for learning', *London : EPPI-Centre, Social Science Research Unit, Institute of Education, University of London* .

IMS Global Learning Consortium (2005), 'IMS Question and Test Interoperability Version 2.0 Final Specification'.

Jackson, D. and Usher, M. (1997), 'Grading student programs using ASSYST', *SIGCSE Bull.* **29**(1), 335–339.

Jean-Daubias, S., Delozanne, E., Jacoboni, P. and Grugeon, B. (1998), 'Cognitive profile in elementary algebra : the PéPITE test interface', *IFIP TC-3 Official Journal special issue Education and Information Technology* (3), 291–305.

Johnson, W. L. (1990), 'Understanding and debugging novice programs', *Artif. Intell.* **42**(1), 51–97.

Joni, S. A. and Soloway, E. (1986), 'But my program runs : discourse rules for novice programmers', *Journal Educational Computing Research* **2**(1), 95–126.

José, P. L. and Fernando, M. A. S. (2003), 'Mooshak : a web-based multisite programming contest system', *Software : Practice and Experience* **33**(6), 567–581.

Joy, M., Griffiths, N. and Boyatt, R. (2005), 'The BOSS online submission and assessment system', *Journal on Educational Resources in Computing* **5**(3), 2.

Juedes, D. W. (2003), Experiences in web-based grading, *in* '33rd ASEE/IEEE Frontiers in Education Conference. Boulder, C'.

Juwah, C. (2003), 'Using peer assessment to develop skills and capabilities', *United States Distance Learning Association* **17**(1), 39–50.

Kaasboll, J. (2002), *Learning Programming*, University of Oslo.

Kalz, M., Koper, R. and Hornung-Prahauser, V. (2009), 'Technology support for self-organized learners', *Educational Technology & Society* **12**(3).

Korhonen, A., Malmi, L. and Silvasti, P. (2003), TRAKLA2 : A framework for automatically assessed visual algorithm simulation exercises, *in* 'Proceedings of Kolin Kolistelut / Koli Calling - Third Annual Baltic Conference on Computer Science Education', Helsinki University Printing House, Helsinki, Finland, pp. 41–49.

Labat, J. (2002), Eiah : Quel retour d'information pour le tuteur ?, *in* 'Actes du colloque Technologies de l'information et de la Communication dans les enseignements d'ingénieurs et dans l'industrie EIAH', EIAH'02.

Lahtinen, E., Ala-Mutka, K. and Järvinen, H.-M. (2005), 'A study of the difficulties of novice programmers', *SIGCSE Bull.* **37**(3), 14–18.

Lane, H. C., Vanlehn, K. and Lane, H. C. (2005), 'Teaching the tacit knowledge of programming to novices with natural language tutoring', *Computer Science Education* **15**, 183–201.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Mostrom, J. E., Sanders, K., Seppala, O., Simon, B. and Thomas, L. (2004), 'A multi-national study of reading and tracing skills in novice programmers', *SIGCSE Bulletin* **36**(4), 119–150.

Marcelino, M., Gomes, A., Dimitrov, N. and Mendes, A. (2004), Using a computer-based interactive system for the development of basic algorithmic and programming skills, *in* 'Proceedings of the 5th international conference on Computer systems and technologies', CompSysTech '04, ACM, pp. 1–6.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. (2001), A multi-national, multi-institutional study of assessment of programming skills of first-year cs students, *in* 'ITiCSE-WGR '01', ACM, pp. 125–180.

Merceron, A. and Yacef, K. (2004), Train, store, analyse for more adaptive teaching, *in* 'Proceedings of International Symposium Information and Knowledge Technologies in Higher Education and Industry', TICE'04, Technical University of Compigne, Compiegne, France, pp. 52–59.

Mickael, R. (2002), Evaluation de l'apprenant dans les hypermédias éducatifs, *in* 'Système multi-agent', pp. 411–412.

Milne, I. and Rowe, G. W. A. (2004), 'OGRE : Three-dimensional program visualization for novice programmers', *Journal of Education and Information Technologies* **9**(3), 219–237.

Morris, P. and Adamson, B. (2010), *Curriculum, schooling and society in Hong Kong*, Hong Kong : Hong Kong University Press.

Moskal, B., Lurie, D. and Cooper, S. (2004), Evaluating the effectiveness of a new instructional approach, *in* 'SIGCSE', ACM, pp. 75–79.

Mufti-Alchawafa, D. (2008), Modélisation et représentation de la connaissance pour la conception d'un système décisionnel dans un environnement informatique d'apprentissage en chirurgie, PhD thesis, Univeristé Joseph Fourier Grenoble.

Muldner, T. and Shakshuki, E. (2003), Teaching students to implement algorithms, Technical report, Jodrey School of Computer Science, Acadia University.

Naudé, K. A., Greyling, J. H. and Vogts, D. (2010), 'Marking student programs using graph similarity', *Comput. Educ.* **54**(2), 545–561.

Noizet, G. and Caverni, J.-P. (1977), *Psychologie de l'évaluation scolaire*, Paris : PUF.

Perrenoud, P. (2001), 'Les trois fonctions de l'évaluation dans une scolarité organisée en cycles', *Educateur* **2**, 19–25.

Petre, M., Fincher, S., Tenenberg, J., Anderson, R., Anderson, R., Bouvier, D., Fitzgerald, S., Gutschow, A., Haller, S., Jadud, M., Lewandowski, G., Lister, R., McCauley, R., McTaggart, J., Morrison, B., Murphy, L., Prasad, C., Richards, B., Sanders, K., Scott, T., Shinners-Kennedy, D., Thomas, L., Westbrook, S. and Zander, C. (2003), My criterion is : Is it a boolean ? : A card-sort elicitation of students' knowledge of programming constructs, Computing Laboratory Technical Report 6-03, University of Kent.

Pisan, Y., Richards, D., Sloane, A., Koncek, H. and Mitchell, S. (2003), Submit ! a web based system for automatic program critiquing, *in* 'Proceedings of the fifth Australasian conference on Computing education', Vol. 2 of *ACE'03*, Australian Computer Society, Inc., pp. 59–68.

Ragonis, N. and Ben-Ari, M. (2005), 'A long-term investigation of the comprehension of oop concepts by novices', *Computer Science Education* **15**, 203 – 221.

Ramadhan, H. A. (2000), 'Programming by discovery', *J. Comp. Assisted Learning* **16**(1), 83–93.

Ramadhan, H. A., Deek, F. F. and Shihab, K. (2001), 'Incorporating software visualization in the design of intelligent diagnosis systems for user programming', *Artif. Intell. Rev.* **16**(1), 61–84.

Rivers, K. and Koedinger, K. R. (2012), A canonicalizing model for building programming tutors., *in* S. A. Cerri, W. J. Clancey, G. Papadourakis and K. Panourgia, eds, 'ITS', Vol. 7315 of *Lecture Notes in Computer Science*, Springer, pp. 591–593.

Robins, A., Rountree, J. and Rountree, N. (2003), 'Learning and teaching programming : A review and discussion', *Computer Science Education* **13**(2), 137–172.

Russell, G. A., Linda, S. S. and Robert, J. M. (2003), A four-process architecture for assessment delivery, with connections to assessment design, Technical report, University of Illinois.

Saikkonen, R., Malmi, L. and Korhonen, A. (2001), Fully automatic assessment of programming exercises, *in* S. Fincher, B. J. Klein, F. Culwin and M. McCracken, eds, 'ITiCSE', ACM, pp. 133–136.

Sanders, D. and Dorn, B. (2003), 'Classroom experience with jeroo', *J. Comput. Sci. Coll.* **18**(4), 308–316.

Sandy, G., Y, G., Patricia, H. and Anthony, R. (2005), My program is correct but it doesn't run : A preliminary investigation of novice programmers' problems, *in* 'Proceedings of Australasian Computing Education Conference', pp. 173–180.

Shaw, J. and Chen, J. (2012), 'Transactional distance and teaching presence in e-learning environments', *International Journal of Innovation and Learning* **11**(1), 44–59.

Smith, P. A., Webb, G., Webb, G. I. and Visualisation, S. (2000), 'The efficacy of a low level program visualisation tool for teaching programming concepts to novice c programmers'.

Soloway, E., Bonar, J. and Ehrlich, K. (1983), 'Cognitive strategies and looping constructs : An empirical study', *Commun. ACM* **26**(11), 853–860.

Soloway, E. and Spohrer, J. C. (1988), *Studying the Novice Programmer*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Spohrer, J. C., Soloway, E. and Pope, E. (1985), 'A goal/plan analysis of buggy pascal programs', *Human Computer Interaction* **1**(2), 163–207.

Stephen, K., Laura, H., Daniel, M. and Brian, S. (2000), 'What do test scores in texas tell us ?', *education policy analysis archives* **8**.

Tagliante, C. (1994), *La classe de langue*, Paris : Clé International.

Tang, C. M., Yu, Y. T. and Poon, C. K. (2009*a*), An approach towards automatic testing of student programs using token patterns, *in* 'Proceedings of the 17th International Conference on Computers in Education', ICCE 2009, pp. 188–190.

Tang, C. M., Yu, Y. T. and Poon, C. K. (2009*b*), Automated Systems for Testing Student Programs : Practical Issues and Requirements, *in* 'Workshop Proceedings of the 17th International Conference on Computers in Education', Vol. 2, pp. 132–136.

Thomas, L., Ratcliffe, M., Woodbury, J. and Jarman, E. (2002), Learning styles and performance in the introductory programming sequence, *in* 'SIGCSE', ACM, pp. 33–37.

Vendlinski, T. and Stevens, R. (2002), 'Assessing student problem-solving skills with complex computer based tasks', *Journal of Technology, Learning, and Assessment* **1**(3).

Vinh-Bang, V. (1989), *Bases psychologiques de l'initiation scientifique aux enfants de 7 à 12 ans*, P. Lang, chapter Psychologie génétique et didactique des sciences, pp. 25–57.

Wang, T., Su, X., Wang, Y. and Ma, P. (2007), 'Semantic similarity-based grading of student programs', *Information & Software Technology* **49**(2), 99–107.

Weigend, M. (2006), From intuition to programme, *in* R. Mittermeir, ed., 'ISSEP', Vol. 4226 of *Lecture Notes in Computer Science*, Springer, pp. 117–126.

Winslow, L. E. (1996), 'Programming pedagogy psychological overview', *SIGCSE Bull* **28**(3), 17–22.

Annexes

A

Sommaire

A.1 Quelques interfaces du système Algo+ 107
A.2 Les opérations de base 109
A.3 Résultats expérimentaux 110
A.4 Valeurs critiques du test des rangs pour échantillons appariés de Wilcoxon 110

A.1 Quelques interfaces du système Algo+

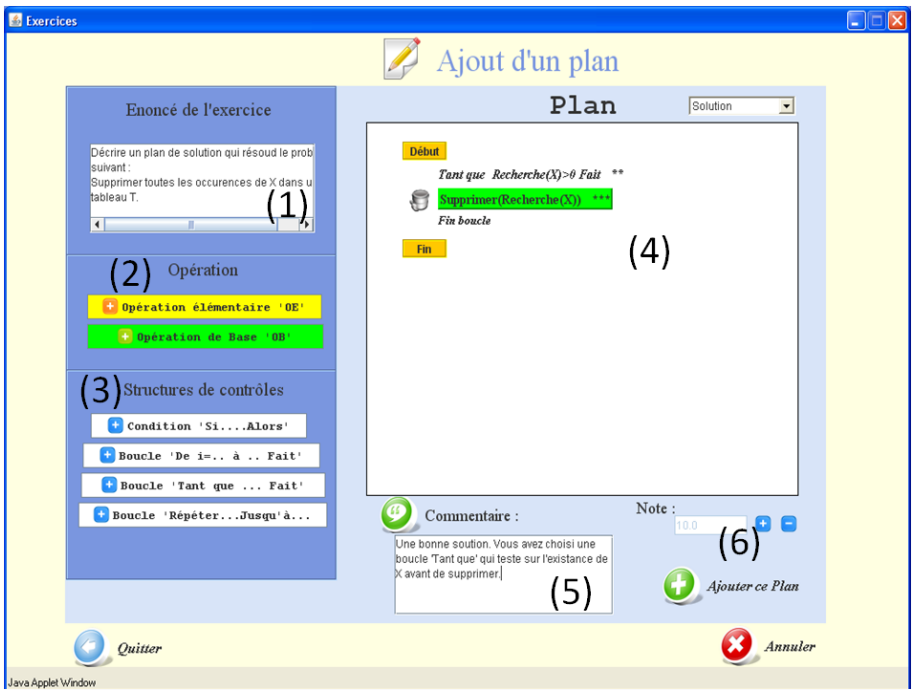

FIGURE A.1 – *Ajout d'un plan de solution*

1. Énoncé
2. Opération de base/élémentaire
3. Structures de contrôle
4. Plan de Solution
5. Commentaire/feedback
6. Note

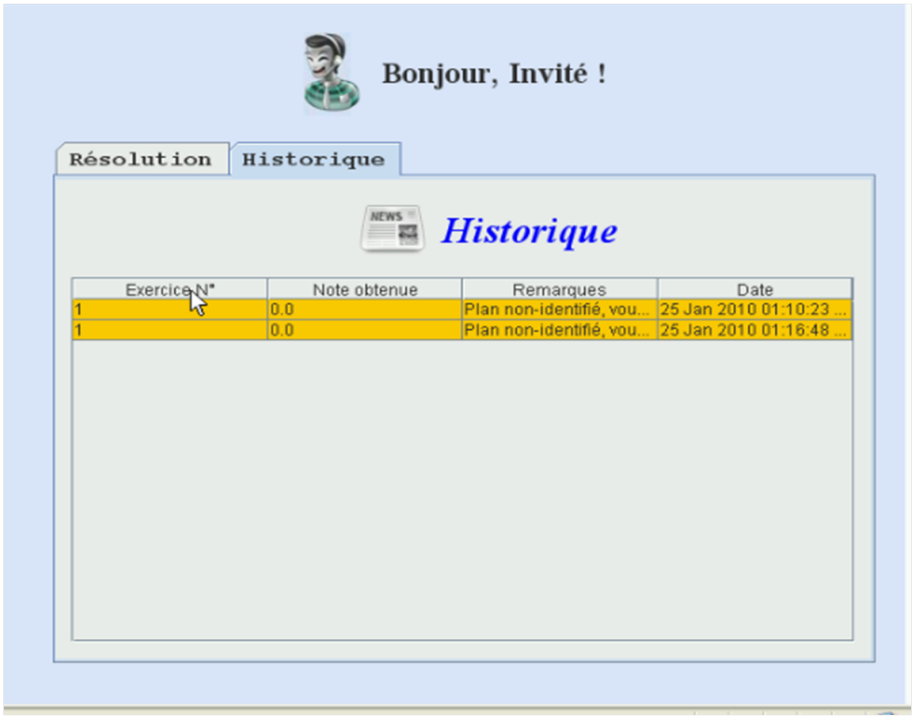

FIGURE A.2 – *L'historique*

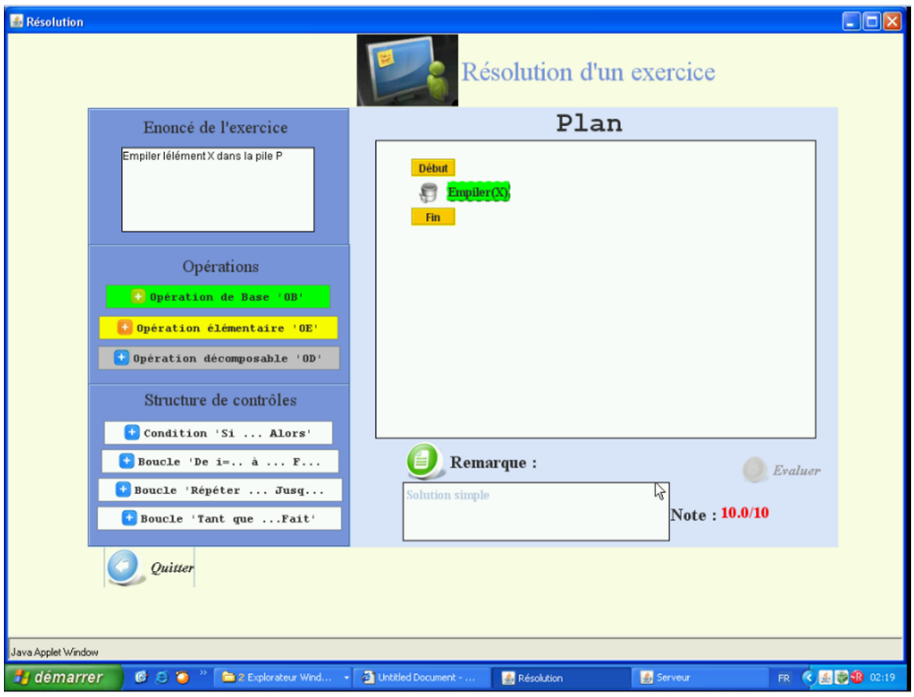

FIGURE A.3 – *Une solution reconnue*

A.2 Les opérations de base

Opération	Paramètre	Résultat	Description
new		p : Pile	crée une nouvelle pile p
Init_Pile	P :pile		initialise la pile à vide
Empiler	e : élément, P :pile		dépose l'élément e au sommet de la pile P
Dépiler	P :pile		enlève l'élément qui se trouve au sommet de la pile
Sommet	P :pile	élément	retourne la valeur de l'élément qui est au sommet de la pile
Est_Vide	P :pile	booléan	retourne vrai si la pile est vide
Profondeur	P :pile	entier	retourne la profondeur de la pile
Créer_FileVide	F :file		On construit une file vide
Enfiler	e :élément, F :file		On ajoute un élément e dans la file F (par la tête)
Défiler	F :file		On enlève un élément de la file F (par la tête)
Tête	F :file	e :élément	On récupère l'élément en premier dans la file
Est_Vide	F :file	booléan	retourne vrai si la file F est vide
Est_Vide	L :liste	booléan	retourne vrai si la file F est vide
Supprimer	i :position, L :liste		supprimer le ième élément d'une liste L
Ajouter	e :élément, i :position, L :liste		ajouter un élément e à une liste L dans la ième position
listeVide	L :liste		construire une liste L vide
Est_vide	a :arbre	booléen	teste si a est vide ou non
G	a :arbre	arbre	renvoie le sous-arbre gauche de a (si a est non vide)
D	a :arbre	arbre	renvoie le sous-arbre droit de a (si a est non vide)
ArbreVide	a :arbre		créer un arbre vide
FixerAG	a,g :arbre		remplace le sous-arbre gauche de a par g
FixerAD	a,d :arbre		remplace le sous-arbre droit de a par d

Table A.1 – *Table des opérations de base*

A.3 Résultats expérimentaux

	Différence de note Expert-Algo+									
	-2	**-0.9**	**-0.8**	**-0.5**	**-0.2**	**0**	**0.1**	**0.2**	**0.3**	**0.5**
Fréquence	1	1	1	1	5	7	2	3	2	1
%	4.17	4.17	4.17	4.17	20.83	29.17	8.33	12.5	8.33	4.17

Table A.2 – *Différences de notes entre l'Expert et Algo+*

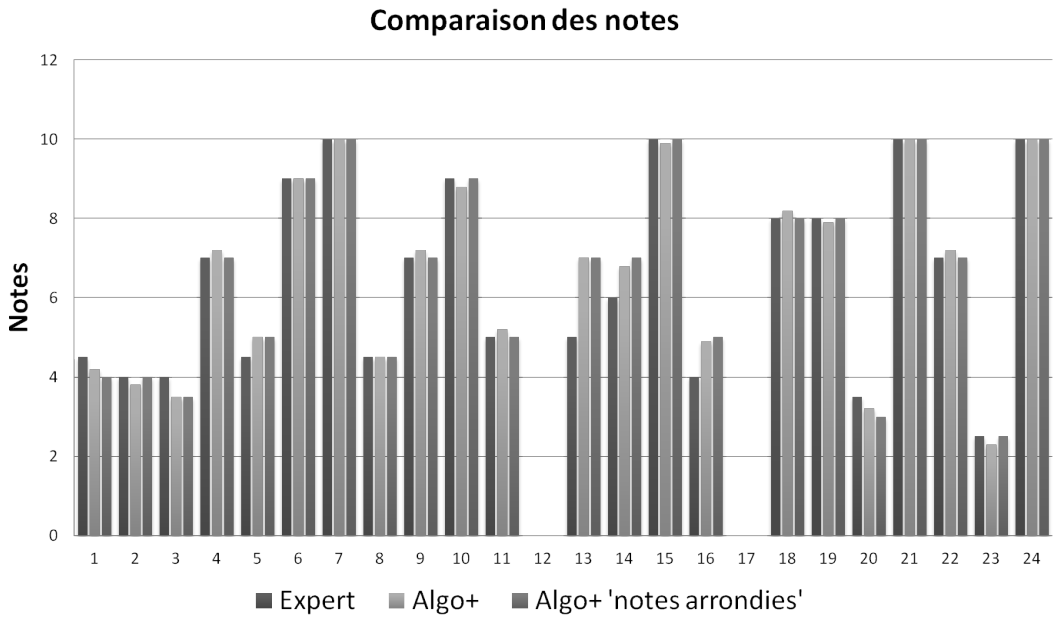

Figure A.4 – *Comparaison des notes : Expert VS Algo+ VS Algo+ 'arrondies'*

Notes	Moyenne	Écart type	Médiane	Min	Max	Étendu
Expert	5.94	2.98	5.5	0	10	10
Algo+	6.08	3	6.9	0	10	10
Algo+ 'arrondies'	6.06	3	7	0	10	10

Table A.3 – *Paramètres des distributions des notes : Expert VS Algo+ VS Algo+ 'arrondi'*

A.4 Valeurs critiques du test des rangs pour échantillons appariés de Wilcoxon

-	Niveau de signification, test unilatéral		
	0.025	0.01	0.005
-	Niveau de signification, test bilatéral		
N	0.05	0.02	0.01
6	0	-	-
7	2	0	-
8	4	2	0
9	6	3	2
10	8	5	3
11	11	7	5
12	14	10	7
13	17	13	10
14	21	16	13
15	25	20	16
16	30	24	20
17	35	28	23
18	40	33	28
19	46	38	32
20	52	43	38
21	59	49	43
22	66	56	49
23	73	62	55
24	81	69	61
25	89	77	68

TABLE A.4 – *Valeurs critiques du test des rangs pour échantillons appariés de Wilcoxon*

Notations

CAA	Computer Assisted Assessment
CAT	Computer Assisted Testing
CBA	Computer Based Assessment
CD	Carte Descriptive
DS	Similarité entre deux PS
EIAH	Environnement Informatique pour l'Apprentissage Humain
ITS	Intelligent Tutoring System
OB	Opération de Base
OE	Opération Élémentaire
P1	Paramètre de pénalité pour les Opérations en désordre
P2	Paramètre de pénalité pour les Opérations inutiles
PS	Plan de Solution
PS_i	Plan de Solution de la base
pt	Point
QCM	Question à Choix Multiples
SA	Seuil d'Acceptabilité d'un plan de solution
S_{pond}	Similarité pondérée

Publications réalisées au cours de la thèse

Publications et Conférences internationales

- Bey, A. and Bensebaa, T. (2013), "Assessment makes perfect : improving student's algorithmic problem solving skills using plan-based programme understanding approach", Int. J. Innovation and Learning 14(2), 162-176.
- Anis Bey and Tahar Bensebaa "An e-Assessment Approach of Algorithmic Problem-Solving Skills". International Journal of Computer Applications 25(10) :15-21, 2011. Published by Foundation of Computer Science, New York, USA.
- Bey, A. and Bensebaa, T. (2011), "Algo+, an assessment tool for algorithmic competencies", in IEEE Engineering Education 2011 Learning Environments and Ecosystems in Engineering Education, EDUCON'11, Amman, Jordan, p. 941-946.
- Bey, A. and Bensebaa, T. (2011), "An innovative method to assess algorithmic problem-solving skills", in GUIDE INTERNATIONAL CONFERENCE, E-learning innovative models for the integration of education technology and research, Italie : Università degli Studi
- Bey, A., Bensebaa, T. and Bensalem, H. (2010), "Assessment of algorithmic competences in tel environment", in 10th International Educational Technology Conference, IETC'10, Istanbul, Turkey.
- Bey, A., Bensebaa, T. and Bensalem, H. (2010), "Easel : Evaluation of algorithmic skills in an environment learning", World Academy of Science, Engineering and Technology 42, 64-67.
- Bey, A., Bensebaa, T. and Bensalem, H. (2010), Evaluation des compétences algorithmiques dans un environnement d'apprentissage, in Proceeding of LEAFA 2010, The First International Conference in E-LEArning For All', LEAFA'10, Hammamet, Tunis.

Conférences nationales

- Anis Bey, Tahar Bensebaa, Hana Bensalem "Une nouvelle approche d'évaluation des compétences algorithmiques dans un EIAH", JED'2010, Journées d'Etudes Doctorales 2010, Université Badji Mokhtar, Annaba.
- Anis Bey, Tahar Bensebaa, Hana Bensalem, "Evaluation des compétences de résolution des problèmes en algorithmique", Séminaire Inter Laboratoires des Doctorants en Informatique, SILDI 2011, Constantine, Algérie

Autres travaux réalisés au cours de la thèse

Collaborations :
2012-2014 Chercheur Junior dans le projet de coopération ROSE : Raisonnements Ontologiques pour les Systèmes d'Enseignements (projet de coopération Accord CMEP). Ce projet rassemble trois équipes de recherches sur les EIAH de deux laboratoires nationals (Ecole Supérieur d'Informatique, Alger Équipe EIAH, Université Badji-Mokhtar d'Annaba, Équipe Enseignement Intelligemment Assisté par Ordinateur & e-learning) et l'équipe Metah du LIG-France.

Co-encadrement :
Master de Recherche Ingénierie de Connaissances (Janvier 2011-Juin 2011)

Bendjedid Hana, "*Une Approche de Résolution Collective des Problèmes d'Algorithmique dans un EIAH*", un master portant sur la mise en œuvre du modèle d'évaluation proposé dans cette thèse dans une situation collaborative.

Cours présenté :
Janvier 2011 et Janvier 2012

Intitulé du cours : *Evaluation dans les EIAH*

Niveau : Master Ingénierie des Connaissances