MINISTERE DE L'ENSEIGNEMENT SUPERIEUR 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 : 2016/2017

THESE

Présentée en vue de l'obtention du diplôme de

Doctorat en Sciences

Technique d'évaluation pour l'apprentissage de l'algorithmique

Filière : Informatique

Spécialité: Intelligence Artificielle

par

RymAiouni

Pr Mohamed Tahar KIMOUR	Professeur à l'Université d'Annaba	Président
Pr Tahar BENSEBAA	Professeur à l'Université d'Annaba	Directeur de Thèse
Pr Balla Ammar	Professeur à l'INI d'Alger	Examinateur
Pr Yacine LAFIFI	Professeur à l'Université de Guelma	Examinateur
Pr Mahmoud BOUFAIDA	Professeur à l'Université de Constantine	Examinateur

Remerciements

« Soyons reconnaissants aux personnes qui nous donnent du bonheur ;
elles sont les charmants jardiniers par qui nos âmes sont fleuries »
Marcel Proust

A l'issue de la rédaction de cette recherche, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate de « l'apprenti-chercheur ».

En tout premier lieu, je remercie le bon Dieu, tout puissant, de m'avoir donné la force pour dépasser toutes les difficultés et de pouvoir en arriver là.

Je tiens à exprimer mes plus vifs remerciements au Professeur Bensebaa Tahar qui fut pour moi un directeur de thèse attentif et disponible malgré ses nombreuses charges. Sa compétence, sa rigueur scientifique et sa clairvoyance m'ont beaucoup appris. Ils ont été et resteront des moteurs de mon travail de chercheur.

J'exprime tous mes remerciements à l'ensemble des membres de mon jury : Pr Kimour, Pr Lafifi,

Pr balla et Pr Boufaida pour l'honneur qu'ils m'ont fait en acceptant d'être membres de mon jury de thèse. Je tiens à les assurer de ma profonde reconnaissance.

Sans oublier de remercier tout mes collègue de l'équipe LRI : Anis bey, HanaBessalem, wassilaDebabi , Meriem Abdessmad et takibelhaoues pour leur collaboration et leur soutien.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Dédicaces

A la mémoire de mon père … ne meurent que ceux que l'on oublie. Ton amour mon papa chéri, restera pour toujours gravé au plus profond de mon cœur et je sais que dans le monde ou tu es maintenant tu es fier de ta fille.

A ma très chère Mère, Affable, honorable, aimable : Tu représentes pour moi l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi. Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études. Aucune dédicace ne saurait être assez éloquente pour exprimer ce que tu mérites pour tous les sacrifices que tu n'as cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études. Je te dédie ce travail en témoignage de mon profond amour. Puisse Dieu, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur.

Merci

Résumé

L’évaluation d’algorithmes d’apprenants constitue une tâche très coûteuse en termes de temps et de réflexion. Car pour un problème donné, plusieurs solutions peuvent être élaborées. En EIAH, la majorité des environnements dédiés à l’apprentissage de l’algorithmique et de la programmation conçus présentaient certaines faiblesses au niveau du processus d’évaluation.

Dans cette thèse, nous avons adopté une méthode d'évaluation basée organigrammes. Cette méthode consiste à comparer les organigrammes des apprenants (novices) aux organigrammes solutions. Cette comparaison est faite en utilisant une méthode d’appariement structurel. eAlgo est l’outil qui implémente la méthode proposée. Les résultats des expérimentations étaient très encourageants. Ils ont montré que les notes fournies par eAlgo sont, à un niveau plus qu'acceptable, similaires aux notes attribuées par les enseignants. Mieux encore, eAlgo attribue des notes qui sont en bonne corrélation avec la moyenne des notes attribuées par les enseignants.

Ceci est certainement dû à la souplesse de gestion offerte par eAlgo et qui permet de paramétrer l'importance des éléments de réponses référencés.

Mots clés : algorithmique, évaluation dans un EIAH, compétences algorithmiques, évaluation du savoir-faire.

Abstract

Assessing students' programs by hand constitutes a burdensome task for assistant teachers in computer science course because the number of students and the variability of programs for one problem. So it will be important to support teachers and students in programming initiatives by considering new assessment approaches. In this thesis, we propose an automated flowchart algorithms scoring system called eAlgo. Recognition of solutions is based on graph matching. Given parameters to the similarity measure, this method will be able to assess automatically learners' algorithms based on predefined algorithms pre-established by the instructor. The main goal of the proposed method is to help teachers to alleviate the scoring load.

eAlgo is a tool that implement the proposed approach. Results of the experiments were very encouraging. These results showed that eAlgo assigns scores similar to teachers' scores. The most important funding is that eAlgo scores are more similar to the average scores of teachers. This automated assessment by eAlgo can also be parameterized.

Key words: Diagnostic and formative assessment, program understanding approach, problem-solving skills of algorithmic.

مـــــــــلخـــــــص

تعالج هده الاطروحة واحد من اهم مواضيع البحث في مجال التعليم الالكتروني ألا و هو التقويم الالكتروني.

يمثل التقويم أحد العناصر المهمة المكونة لمنظومة المنهج ولقد تعددت تعريفاته، فقد يعني إصدار حكم على الأشياء في ضوء استخدام محكات أو معايير معينة، أو عملية يتم من خلالها إعطاء قيمة محددة لشيء ما.

و لقد بنيت البرامج التعليمية في مجال تكنولوجيا التعليم في ضوء هذه المستحدثات وخاصة برامج التعلم الإلكتروني، والتي أصبح تقويمها ضرورة ملحة، وذلك لبيان مدى ما تحقق من أهداف هذه البرامج، وبالتالي ظهرت الحاجة إلى تقويم هذه البرامج إلكترونياً.

يعرف التقويم التعليمي الإلكتروني بأنه عملية توظيف شبكات المعلومات وتجهيزات الكمبيوتر والبرمجيات التعليمية والمادة التعليمية المتعددة المصادر باستخدام وسائل التقييم لتجميع وتحليل استجابات الطلاب بما يساعد عضو هيئة التدريس على مناقشة وتحديد تأثيرات البرامج والأنشطة بالعملية التعليمية للوصول إلى حكم مقنن قائم على بيانات كمية أو كيفية متعلقة بالتحصيل الدراسي.

هذا البحث العلمي موجه الى معالجة موضوع تقويم المهارات المعرفية للطلاب في مجال البرمجة اين تمكنا من انشاء نضام يقوم بتقويم حلول الطلاب في البرمجة و ذلك على اساس طريقة قياس اوجه الشبه الموجودة بين حلول الطلاب و حلول الموجودة في القاعدة. هذه الطريقة للتقويم الالكتروني مكنت من الحصول على نتائج فعالة تتسم الى حد كبير بالمصداقية مقارنة مع النتائج المحصول عليها من التقويم الذي يقوم به المعلم الذي لا يتسم دائما بالموضوعية اثناء تقويم اجابات الطلاب .

TABLE DES MATIERES

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

L'apprentissage de l'algorithmique et de la programmation est un grand défi. Plusieurs études ont montré que les élèves, après six à douze mois d'apprentissage, ne peuvent appliquer aisément des constructions de base telles que des boucles ; et plusieurs études reconnaissent que, même après deux années d'enseignement de la programmation à l'université, de nombreux étudiants ont seulement une compréhension rudimentaire de la programmation.

Dans ce contexte, la pensée algorithmique et les compétences en programmation jouent un rôle central. Les étudiants ont généralement besoin de se familiariser avec un grand nombre d'algorithmes et de structures de données différents. La capacité de concevoir un algorithme pour un problème donné est l'une des tâches les plus importantes et les plus difficiles dans la programmation. Cependant, la littérature montre que les novices éprouvent de sérieuses difficultés à utiliser des concepts abstraits de programmation comme des structures de données (tableau, graphiques, listes) et manquent de compétences nécessaires pour fonctionner abstraitement, pour consolider un algorithme comme entité unique, pour comprendre ses principales parties et différencier les différentes relations, et de composer de nouveaux algorithmes en utilisant leurs connaissances de programmation précédentes.

Les apprenants novices sont aisément capables de décrire leurs idées et leurs opinions en utilisant des organigrammes afin de trouver des solutions aux problèmes. Différentes tâches complexes sont visuellement présentables sous la forme d'un organigramme sans difficulté, ce qui aide les novices à acquérir une compréhension avancée (Roy et al., 1998). Les organigrammes, un type largement utilisé par les apprenants novices dans les cours d'introduction à l'algorithmique et à la programmation, pourraient être extrêmement efficaces en termes d'écriture et de compréhension des algorithmes, comme le prouve la littérature.

L'évaluation automatique sommative ou formative est déjà largement utilisée dans de nombreux environnements différents dans le domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH).

L'importance de l'évaluation en éducation a été soulevée par de nombreux auteurs (Perrenoud, Boulery& Arnaud). Les auteurs Boulery et Stephane (2005) définissent l'évaluation comme étant : « …un jugement de valeur porté sur une compétence, une

intelligence ou encore un comportement. Ainsi, évaluer, ce n'est pas seulement attribuer une note à une prestation écrite ou orale mais, plus généralement, se donner les moyens pour émettre un avis sur des critères préalablement sélectionnés ».

La problématique de cette thèse traite principalement de l'évaluation des organigrammes en algorithmique pour les apprenants novices en utilisant une méthode d'appariement structurel. La question de recherche qui a été posée durant cette thèse est : Est-ce qu'on peut évaluer automatiquement des organigrammes d'algorithmiques produits par des apprenants novices à l'aide d'une méthode d'appariement ?

Le présent manuscrit débute par une présentation de l'état de l'art sur quelques systèmes et outils qui ont été développés pour aider les apprenants à maitriser la construction des organigrammes. Cet état de l'art se termine par une conclusion qui résume les principaux avantages et inconvénients des approches proposées.

Dans le chapitre 2, une introduction aux Environnements Informatiques pour l'Apprentissage Humain (EIAH) et l'évaluation des apprenants est abordée. Dans le chapitre 3, nous présentons la méthode proposée qui représente le cœur de cette thèse : évaluation des apprenants novices en algorithmique à base des organigrammes. A la fin, nous concluons ce manuscrit par un chapitre sur l'outil développé ainsi que les résultats obtenus durant l'expérimentation de ce dernier avec des conclusions et des perspectives.

Chapitre I:Etat de l'art

I. INTRODUCTION

Les environnements de programmation qui sont basés sur la notation iconique et les environnements basés organigrammes emploient différents types d'illustrations symboliques pour les variables, les structures de contrôle et les types de données. Dans ce type d'environnements, généralement un éditeur dirigé par la syntaxe est utilisée pour effectuer un codage qui comprend des modèles et des menus, en plus de choix syntaxiquement précis pour toutes les parties inachevées du programme (Khwaja et Urban, 1993)

Sans se soucier de la syntaxe, les langages de programmation sont utilisés pour utiliser des algorithmes pour la résolution de problèmes, ce qui est la principale cible des éditeurs de syntaxe (Khwaja et Urban, 1993). En essayant de comprendre les techniques de mise en forme d'un programme afin de le rendre lisible, les modes de gestion des cycles de vie des logiciels et l'application de l'éditeur de texte ont un effet positif sur le problème et évitent de se concentrer sur les détails de bas niveau (Goldenson, 1989;.Kunimune et al, 2008). En appliquant la technique mentionnée, les apprenants sont en mesure de prêter plus d'attention aux questions de structure et de conception. Il convient de mentionner que les apprenants devraient avoir des tâches plus difficiles qui se concentrent davantage sur la conception et la structure. Sur la question des langages de programmation graphique qui sont utilisés pour élaborer des organigrammes, il faut prendre soin de diriger l'attention des apprenants débutants et leur attention vers le développement des compétences du programme de résolution (Calloni et Bagert, 1997; Ziegler et Crews, 1999). Les environnements de programmation graphique à base d'organigrammes utilisent généralement une édition syntaxique dirigée dans un sens où ils aident les apprenants à la compréhension des programmes et des concepts algorithmiques et non à la syntaxe (Price et al., 1998). Les technologies modernes des domaines de l'interaction homme machine humaine de l'ordinateur et l'infographie, la conception graphique, l'animation et des techniques de design graphique sont fondamentalement les domaines que la programmation à base d'organigrammes utilise (Price et al. , 1998). La représentation graphique des programmes, la visualisation des structures de données et variables ainsi que l'animation du programme sont parmi les caractéristiques qui aident à offrir un affichage plus clair et visuel du code source du programme en cours d'exécution. Cela permet aux apprenants d'étudier les programmes et leurs constructions connexes et, bien sûr, le programme lui-même en termes de fonctionnalité, des variables, des procédures, des mécanismes de passage de paramètres, ainsi que le flux de contrôle et de données. Un organigramme est réalisé à partir d'un certain nombre d'étapes qui

se produisent en temps réel avec leurs variables constamment mises à jour, et qui sont exécutées étape par étape. Les structures de contrôles n'apparaissent pas dans les organigrammes, elles sont représentées graphiquement. Dans la section suivante, une panoplie de systèmes et d'outils qui sont dédiés à la programmation à base d'organigrammes sont présentés.

II. Quelques systèmes et outils dédiés à la programmation visuelle

Quelques environnements de programmation basés organigramme à des fins académiques et commerciales sont examinés dans cette section. Les premières recherches dans le domaine de la programmation visuelle, à savoir, BALSA (Brown et Sedgewick, 1984), ont contribué à la conception et le développement de nombreux environnements et systèmes de visualisation graphique. Avec les capacités croissantes des Nouvelles Technologies de l'Information et de la Communication (NTIC), des recherches supplémentaires ont été rendues possibles dans ce domaine qui a permis l'introduction régulière de nouveaux systèmes dédiés à ses fins. L'ampleur de la recherche effectuée sera couverte dans ce manuscrit, tandis que les mots tels que «novice» et «visualisation» sont référencés auprès de l'IEEE, Portal Scopus et ACM. Le fait qu'une vaste zone de recherche peut être intégrée et soutenue par le champ d'application de cette thèse, la capacité de résolution de problèmes, la compréhension impérative du programme et les environnements qui sont basés sur les organigrammes ont été choisis comme le noyau de cette recherche. Dans le développement des environnements de programmation à base d'organigrammes traités par les novices, une augmentation du nombre d'outils développés est récemment devenue évidente. Plus de 17 systèmes et outils ont été créés depuis 1992.

Cependant, trois des systèmes évalués n'ont jamais été publiés dans une littérature académique, alors que seulement sept peuvent être acquis en ligne. Les systèmes évalués ci-dessous sont classés par ordre chronologique.

II.1. BACCII et BACCII++

Ben A Calloni Codage Interface Iconic (BACCII) (Figure 1) a été développé à l'Université Texas Tech dans le but de fournir un soutien dans l'enseignement de la programmation, les procédures et les concepts de la programmation orientée objet et les langages avec les applications basées sur Windows de Microsoft (MS). Pour le cours « Introduction à la programmation » (CS1), BACCII a été utilisé pour charger un paradigme de programmation impératif basé sur Pascal, tandis que BACCII ++ a été utilisé pour les «structures de données» (CSII) bien sûr basée sur le paradigme de la programmation orientée

objet et C ++. La page principale de BACCII ainsi qu'un échantillon de son code généré sont représentés sur la figure 1 (Calloni et Bagert, 1997).

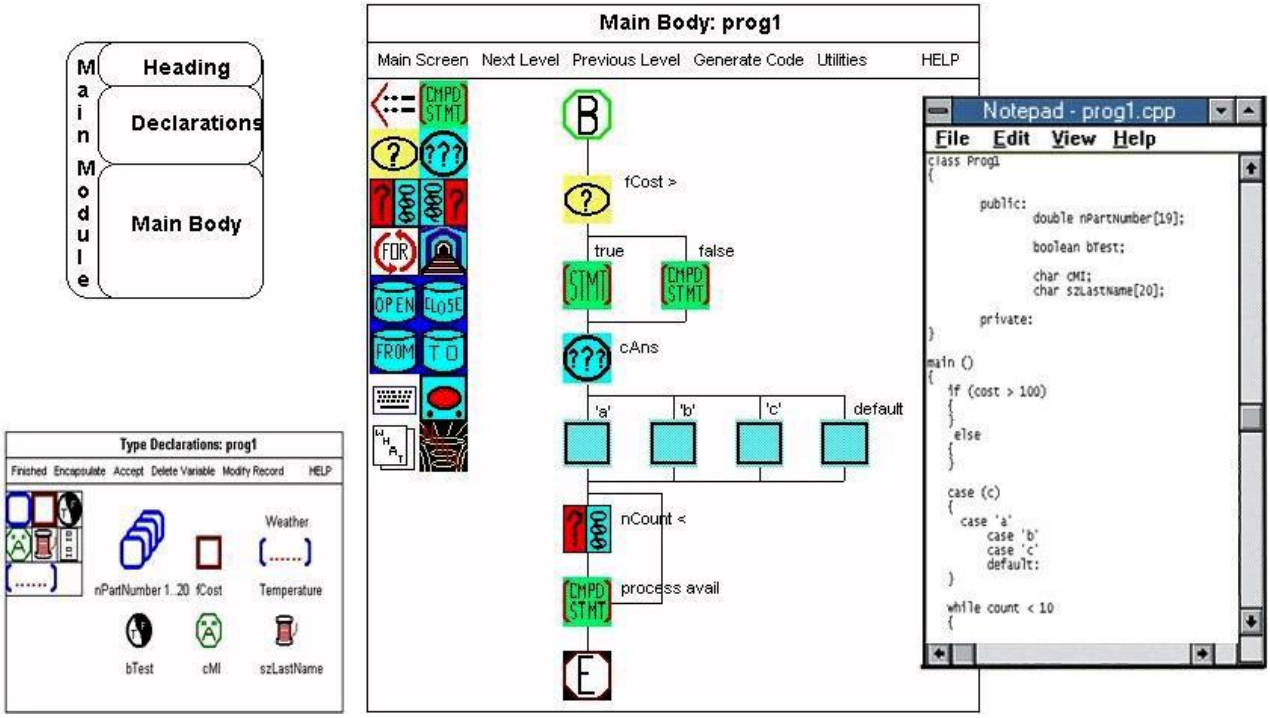


Figure N° 1: BACCII

Ce système permet aux apprenants d'exprimer leurs programmes sous forme d'organigrammes. Les novices peuvent créer des programmes à l'aide de l'interface syntaxique réalisée avec une palette qui contient des icônes pour les déclarations disponibles. L'interface permet aux apprenants d'effectuer des affectations, déclarer des variables de tous les types de données de base, appeler des sous-programmes, utiliser des fichiers de données d'écriture et de lecture, sélectionner et faire des répétitions. BACCII convertit ses organigrammes syntaxiquement au langage C++, Fortran ou en un code Pascal, qui sont uniquement visibles et exécutable en présence d'environnements de programmation supplémentaires. Cet outil améliore la compréhension conceptuelle des élèves de la programmation dans un cours d'algorithmique et de programmation (Calloni et Bagert, 1994). L'intention exprimée par les créateurs de BACCII était la diffusion d'un système commercial en 1999, mais il est toujours indisponible sur le web jusqu'à aujourd'hui (Bagert et Calloni, 1999).

II.2. FLINT

Ce système a été conçu pour fournir un soutien pour le cours " Introduction à la programmation (CS1)" élaboré à l'Université Western Kentucky. L'organigramme interprète (FLINT) et utilise les notations d'organigramme en combinaison avec la structure "Jackson diagrammes" pour permettre aux apprenants débutants de résoudre des problèmes dans l'algorithmique et la méthodologie de conception. Le système est basé sur MS Windows et sert à enseigner aux apprenants la conception de l'algorithme, le débogage, le test et une façon d'apprendre sans la syntaxe. L'interface utilisateur de FLINT est représentée sur la figure 2.

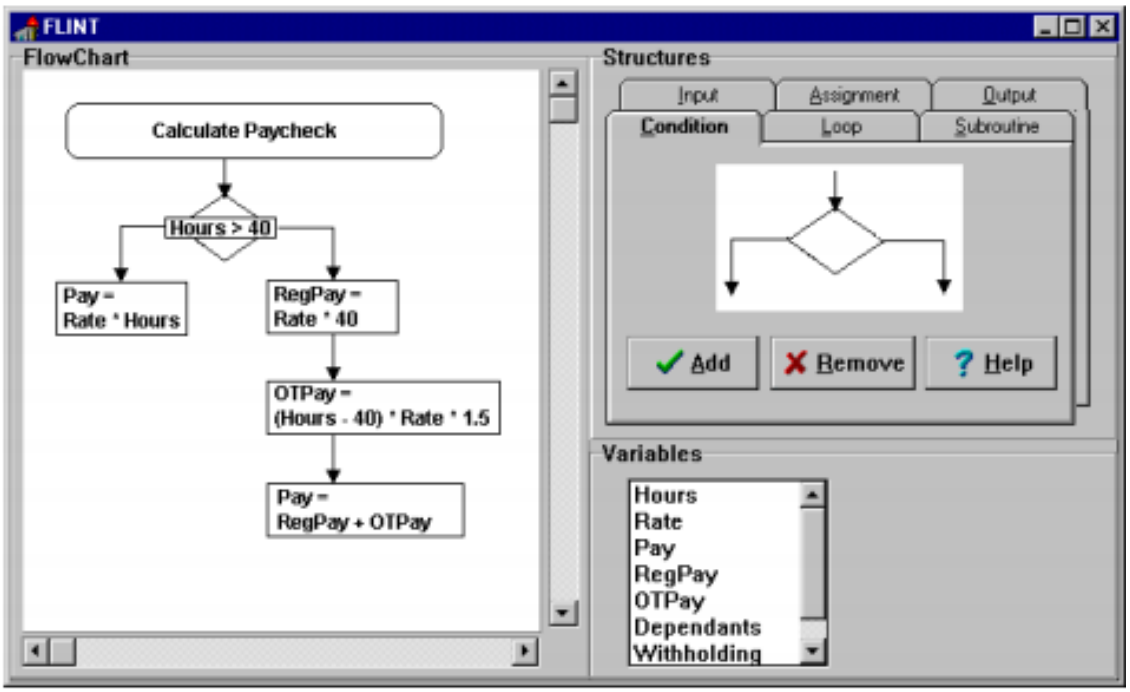


Figure N° 2: FLINT

FLINT demande à ses utilisateurs de construire d'abord un schéma structuré pour esquisser des problèmes de fonctionnalité clés (Ziegler et Crews, 1999). L'utilisateur, sous la forme d'un organigramme structuré (SFC), définit chaque nœud du diagramme de structure avant de représenter le processus. Cela permet aux apprenants d'être plus exposés à la décomposition fonctionnelle avant la séquence de structure fondamentale ainsi que des structures d'itération et de sélection. Le système applique l'utilisation de différentes couleurs pour développer l'organigramme et ses symboles. Le système FLINT dispose des instructions d'entrée/sortie, de la sélection (si-sinon seulement), des variables, de l'affectation, des boucles (boucle while uniquement) et des procédures. Il prend en charge l'exécution visuelle des programmes en fonction des variables mises en évidence car elles sont en cours d'exécution ainsi que chaque composant schématique (Crews et Ziegler, 1998). Ainsi, les apprenants vont mieux comprendre le programme et le processus de développement de stratégies de débogage.

Une évaluation de FLINT indique que l'utilisation d'un organigramme à l'étape initiale a un effet positif sur les apprenants en ce qui concerne la compréhension logique de programmation et l'acquisition de compétences de composition par rapport à un groupe d'apprenants de contrôle qui ont été pris en charge avec un enseignement plus traditionnel(équipages, 2001).

II.3. Empirica Control

L'outil Empirica Control ou CE, a été développé sur la base de l'application MS Windows pour l'enseignement de la technologie. Afin de contrôler un système simple et représenter un affichage graphique de l'exécution d'un programme, les novices peuvent utiliser la fonction visuelle de cet outil pour construire des programmes à l'aide d'un organigramme. L'environnement de contrôle Empirica est représenté sur la figure 3 (Lavonen et al., 2002).

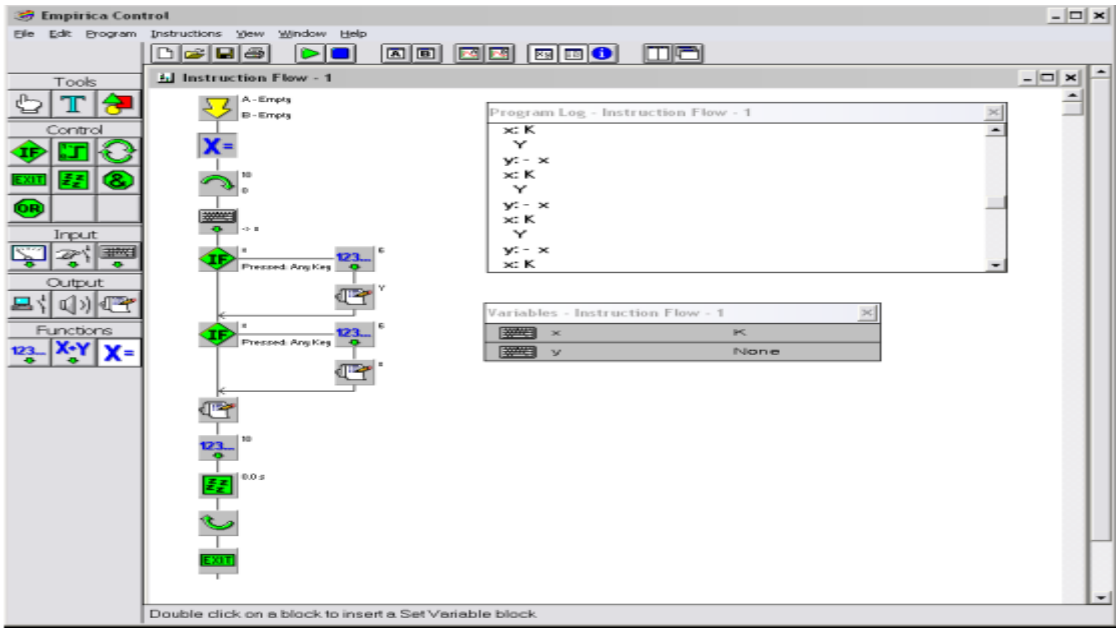


Figure N° 3: Empirica Control

Pour créer et développer un programme facilement, cet outil propose quelques options pour la sélection des outils à utiliser, ainsi que la zone du diagramme de flux de contrôle où les nouveaux composants peuvent être insérés. Par conséquent, le programme est exécutable visuellement une fois que les composants ajoutés sont choisis et que les paramètres sont définis. Ses fonctionnalités de base regroupent des entrées, des sorties, des boucles et des structures de décision et une variété de caractéristiques pour interagir avec différents périphériques externes. Les symboles et les représentations utilisés dans son organigramme ne sont pas en ligne avec des diagrammes communs et standards cependant ils sont un peu plus

semblables aux symboles de l'ingénierie électronique plutôt qu'à ceux des organigrammes de programmation. En outre, les expressions booléennes ne sont pas prises en charge par des structures de décision ou des boucles. Malgré cela, il semble relativement approprié pour l'enseignement de la logique des systèmes de commande (Lavonen et al., 2001).

II.4. FlowchartInterpreter (FCI)

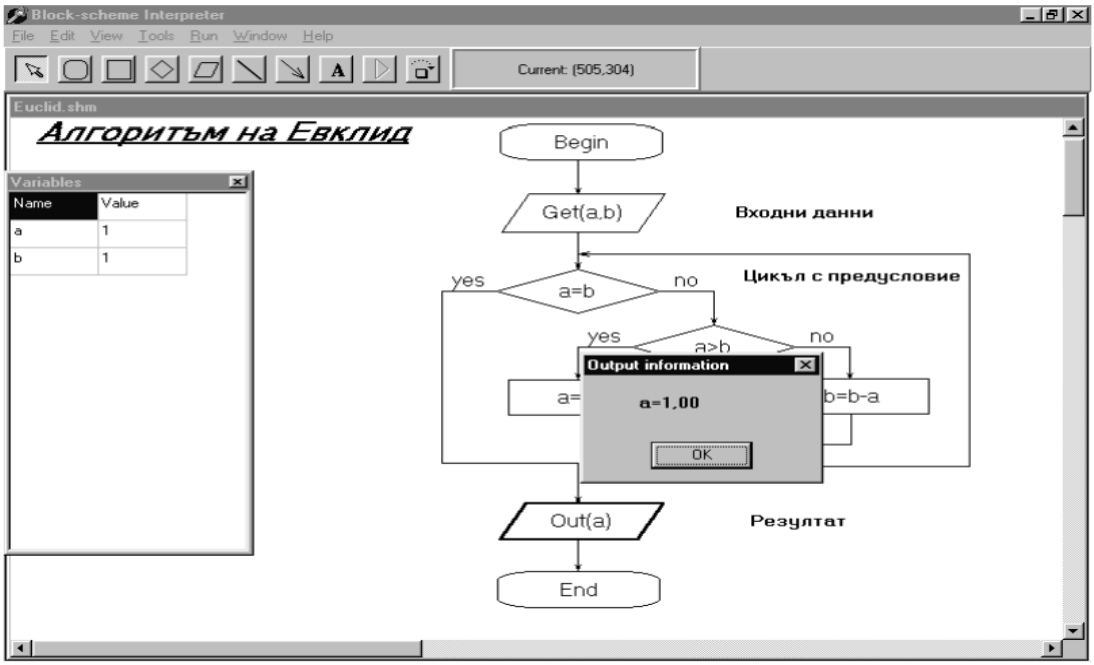


Figure N° 4: FlowchartInterpreter

FlowchartInterpreter a été conçu et développé sur la base de l'application MS Windows à l'Université Rousse en Bulgarie. Des organigrammes noir et blanc sont utilisés dans FCI avec des notations primitives et simples. L'interface de l'interpréteur des organigrammes est représentée sur la figure 4.

En ajoutant chaque morceau de l'organigramme, une expression correcte est entrée par les utilisateurs et vérifiée pour son exactitude syntaxique. L'organigramme est mis à jour et les nouveaux composants sont acceptés aussi longtemps que cela fonctionne. Les utilisateurs reçoivent des commentaires sous la forme d'une boîte de message pop-up si une erreur se produit. Dès que le programme est terminé, il peut être exécuté visuellement. En décrivant l'instruction de l'organigramme, le flux d'exécution du programme peut être visualisé. L'exécution peut également être effectuée automatiquement du début jusqu'à la fin. Il y a un tableau qui répertorie les valeurs des variables sur le côté gauche, où les changements de programme sont affichés pendant que le programme est en cours d'exécution. Ses fonctions d'animation permettent aux apprenants débutants de pratiquer leurs compétences de débogage,

ainsi que de fournir un modèle d'exécution réelle. Étant donné que les symboles primitifs sont appliqués à la construction d'un programme avec cet outil, un large éventail de programmes, dont la majorité sont non-procéduraux, peuvent être construits en utilisant des entrées, des sorties, des variables, des boucles et des structures de décision. Par conséquent, les novices perdent plus de temps sur la modification de l'organigramme au lieu de passer plus de temps sur la résolution des problèmes et la compréhension des concepts essentiels de la programmation. Une enquête de FCI montre qu'il n'y a pas de publication ou de recherche empirique sur cet outil jusqu'à présent (Atanasova et Hristova, 2003).

II.5. Raptor

Raptor a été développé dans le but d'aider à la programmation en cours de calcul d'introduction à l'académie de l'Armée de l'Air USA. Raptor est basé sur MS Windows avec son principal accent sur le développement des compétences de résolution de problèmes tout en évitant simultanément les problèmes de syntaxe. RAPTOR ainsi qu'un exemple de son code généré est représenté sur la figure 5.

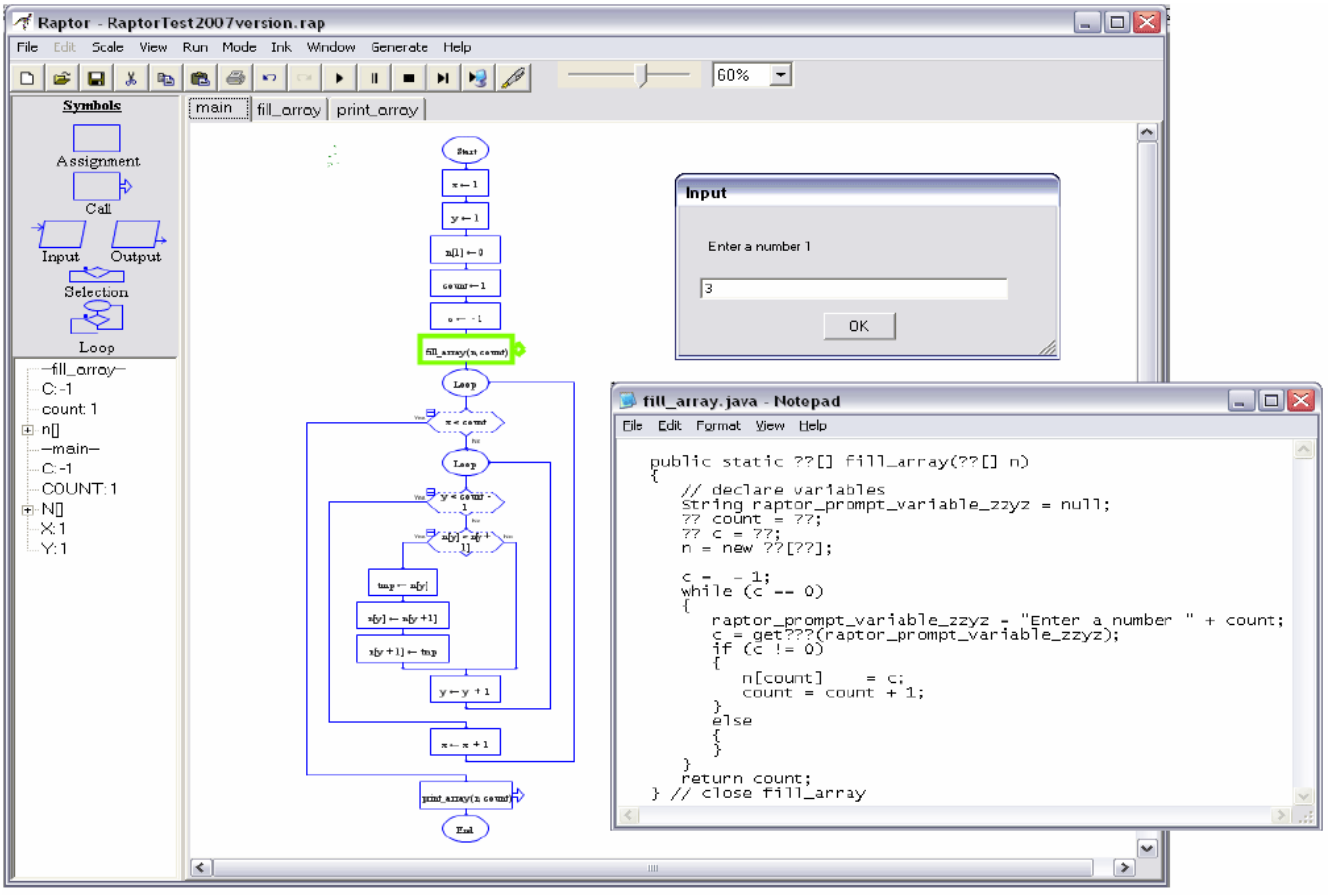


Figure N° 5: Raptor

En termes de génération et d'exécution d'un organigramme en noir et blanc, Raptor est similaire à FCI. Toutefois, il convient de mentionner que, plutôt que d'utiliser des formes et des flèches individuelles, les programmes sont construits et auto-structurés par "glisser-déposer" des structures complètes sur l'organigramme. L'un des principaux points forts de cet outil est qu'il se redessine sans l'intervention des utilisateurs, ce qui contribue considérablement à l'amélioration de l'efficacité cognitive dans le maintien de l'attention des utilisateurs sur la programmation et la résolution de problèmes au lieu de la construction d'un organigramme. Une autre amélioration de Raptor par rapport aux outils précédents est l'aide à la création des procédures et les tableaux, offrant une bibliothèque de procédures pour effectuer des entrés/sorties avec des fichiers textes et définissant les types de variables de données. En outre, la version la plus récente de Raptor développée en 2007 offre la possibilité de traduire son organigramme à un code, offrant aux utilisateurs des différents langages de programmation tels que Ada, C ++ et Java, ainsi que la création de programmes exécutables autonomes. Une évaluation empirique de Raptor a révélé près de 5% de croissance chez l'élève dans son apprentissage des exercices de résolution de problèmes par rapport à un élève utilisant un organigramme traditionnel, Ada, ou le langage Matlab (Carlisle et al., 2004, 2005). Il est également intéressant de noter que cet outil est accessible sur Internet gratuitement.

II.6. The SFC editor

L'outil de l'éditeur SFC est fondé sur MS Windows et a été développé à l'Université d'État de Sonoma en Californie. L'interface de l'éditeur SFC est illustrée dans la figure 6.

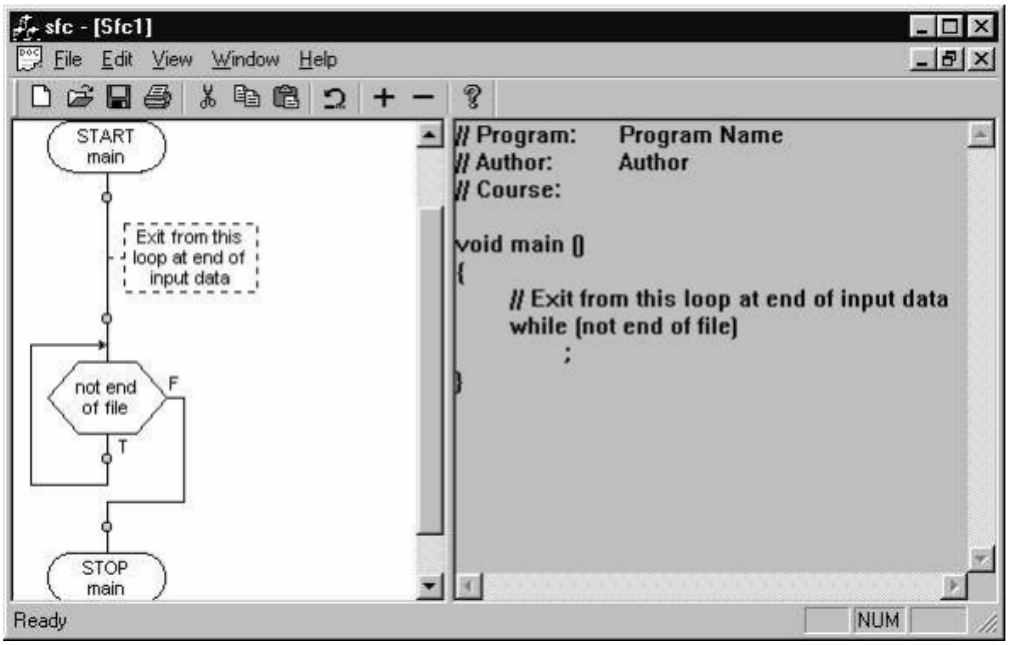


Figure N° 6: SFC Editor

Semblable à d'autres travaux connexes examinés dans cette section, l'éditeur SFC emploie un organigramme en plus d'avoir une fonction d'auto-structuration pour la construction d'organigrammes. Toutefois, afin d'ajouter des commandes et des structures de contrôle, l'éditeur SFC utilise un point d'insertion d'organigramme, un menu contextuel ainsi que la méthode de "glisser-déposer". En outre, les commandes séquentielles, boucles et structures sélectives, ainsi que des procédures sont pris en charge par l'éditeur SFC, ce qui signifie que cet outil a de meilleures caractéristiques que les autres travaux connexes examinés jusqu'ici. L'ajout d'éléments à l'organigramme génère un code en C++ ou un pseudo-code générique d'une manière automatisée. L'une des caractéristiques importantes que l'éditeur SFC a par rapport à d'autres outils est la représentation d'un code en liaison avec l'organigramme. En outre, une ligne de code est mise en surbrillance lorsqu'un composant d'organigramme est sélectionné. Bien que cette synchronisation n'est pas bidirectionnelle, il permet aux apprenants débutants de voir et de comprendre la relation entre l'organigramme et le code généré. À l'heure actuelle, aucune recherche empirique qui évalue et prouve l'efficacité de l'éditeur SFC n'existe (Watts, 2004).

II.7. SICAS

SICAS (Marcelino et al., 2004) fait référence à un système interactif pour le développement d'algorithmes et de simulations en portugais. Il est basé sur MS Windows et vise à couvrir et soutenir les activités de résolution de problèmes selon les théories constructivistes. Un exemple de conception d'algorithmes utilisant SICAS est représenté dans la figure 7.

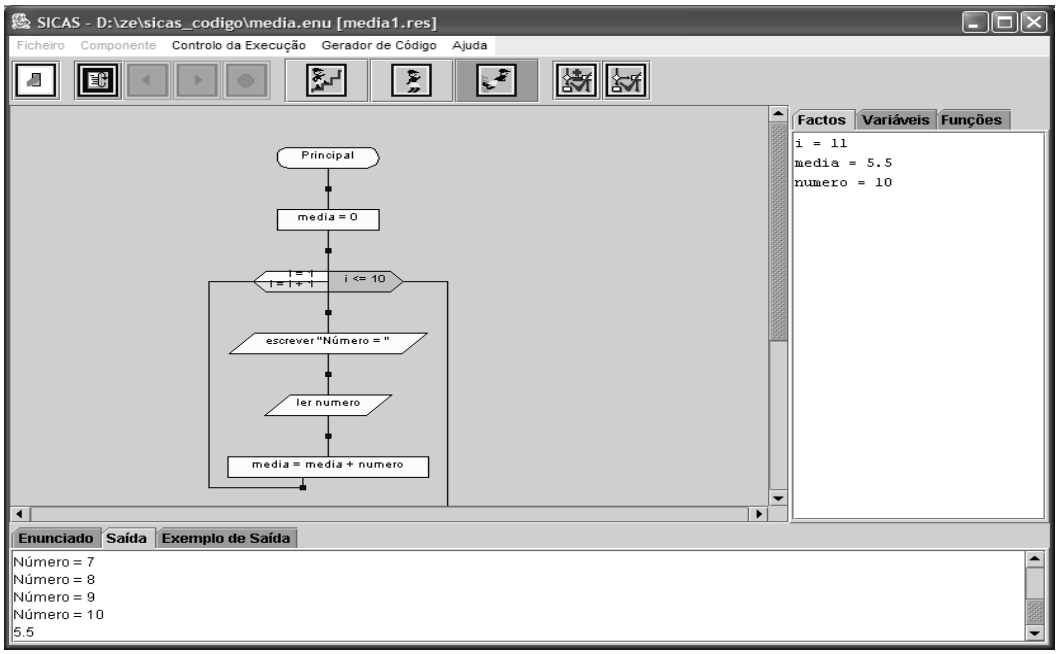


Figure N° 7: SICAS

En SICAS, afin de développer et d'entrer un organigramme, des symboles et des options à partir d'une barre d'outils sont sélectionnés et le point désiré est indiqué sur la zone de conception une fois que les boîtes de dialogue sont remplies d'informations minimales. Les lignes de connexion des différents composants d'organigramme sont ensuite construites de manière automatisée. Cet outil prend en charge et maintient l'affectation, l'entrée et la sortie, la sélection des fonctions et des variables numériques et alphanumériques et des tableaux (Marcelino et al., 2004; Santos et al., 2010). L'exécution du programme se fait dans SICAS étape par étape et les variables sont mises à jour automatiquement. En outre, chaque étudiant peut valider et évaluer son programme lorsque l'instructeur lui fournit un ensemble de données d'entrée/sortie pour tester sa solution. Il y a un mode professeur dans SICAS, ce qui est important lorsque les élèves souhaitent tester leurs programmes pour des cas spécifiques. Dés que les apprenants commencent à exprimer des solutions aux problèmes en Java, le code Java généré à partir de l'organigramme de cet outil peut être utilisé par les utilisateurs (Mendes et al., 2005). La simulation et la conception de procédures sont couvertes par l'outil Picasso alors que le contenu du langage de programmation tels que les entrées et les sorties peut être décrit par les apprenants.

II.8. Visual logic

L'outil Visual Logic ou VL, a été fait et mis au point pour des cours d'introduction à la programmation basée sur Windows. La figure 8 montre un organigramme et son code généré (Crews et Murphy, 2004).

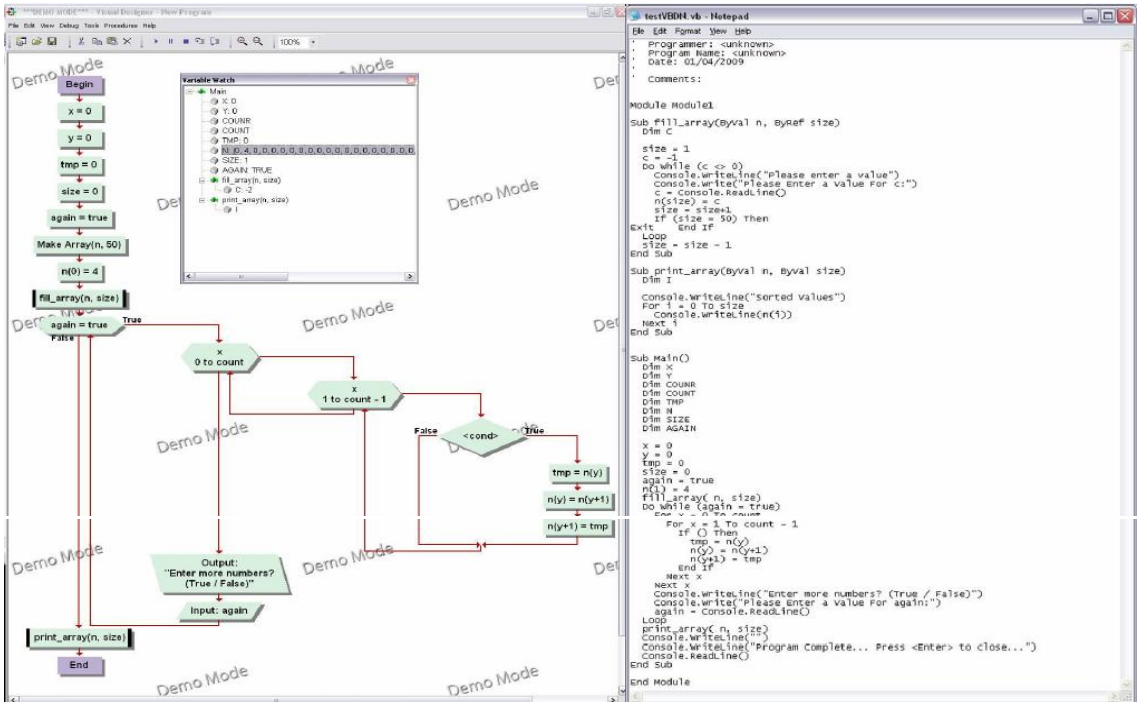


Figure N° 8: Visual Logic

Les structures de contrôle et d'itération, entrée et sortie, les tableaux et variables, ainsi que les procédures et les affectations sont entièrement pris en charge par cet outil. Un menu d'options permet de construire un organigramme. La fonction de structuration automatique est appliquée aux diagrammes VL et divers éléments de l'organigramme utilisent la même couleur. Ainsi, certaines différences sont visibles parmi les formes graphiques comme des boucles et des décisions structurées. Un organigramme peut être réalisé soit directement avec un diagramme ou sous forme non visuelle (Crews, 2009). L'exécution d'un programme visuel est effectuée en mettant en évidence l'ombre du composant d'organigramme. VL fournit aux utilisateurs une fenêtre de contrôle de variable ainsi qu'un état des tableaux et des variables lors de l'exécution du programme pour améliorer la capacité de débogage (Crews, 2001). Dans VL, convertir l'organigramme en fichiers VB.Net, VB et Pascal est considéré comme un atout majeur. D'autre part, le code et l'application résultante sont séparés, ce qui signifie que l'environnement extérieur est nécessaire pour visualiser et exécuter le code. Dans VL, semblable aux outils précédents, le passage de l'organigramme au code résultant est une tâche ardue pour les novices, ce qui indique que cet outil ne peut pas améliorer les compétences de traçage. Tout comme Raptor, le code source généré doit être modifié de manière significative avant exécution en raison de l'absence de types de déclarations des procédures et des variables données. Une évaluation de VL a été effectuée sur les étudiants par le biais d'un questionnaire, où les résultats montrent que les apprenants débutants l'ont trouvé agréable, utile et facile à utiliser.

II.9. The Iconic Programmer

Le système Iconic Programmer ou IP en bref, est un système basé sur le système d'exploitation Windows et a été conçu et développé dans le but de donner aux apprenants débutants la possibilité de générer des organigrammes et des codes exécutables. L'interface du système IP est illustrée dans la figure 9.

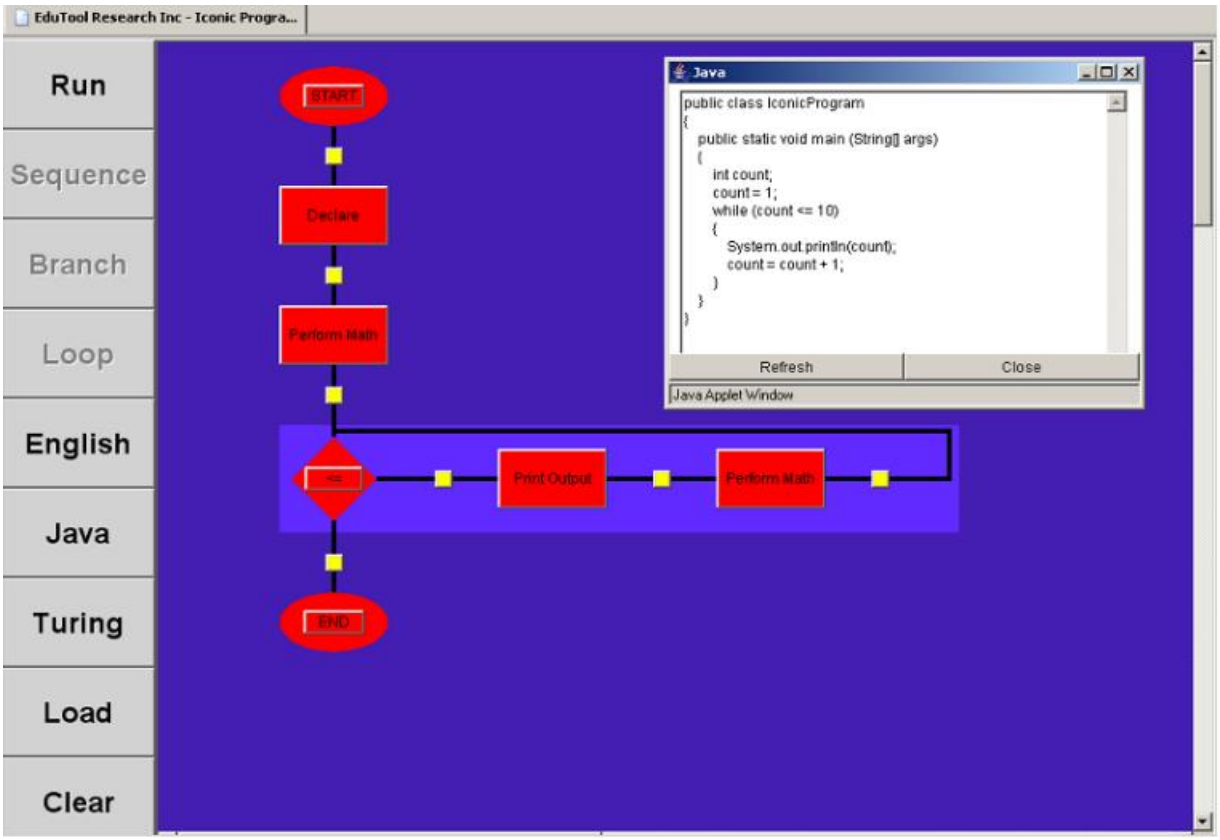


Figure N° 9: The Iconic Programmer

Sur le plan élaboration du programme et exécution, en cliquant sur le menu, les composants nécessaires peuvent être ajoutés à l'organigramme. Dans IP, la structuration automatique est utilisée par l'organigramme. Les apprenants sont autorisés à exécuteur le programme conçu de manière automatisée, du début jusqu'à la fin et de suivre l'exécution de chaque instruction. La conversion de l'organigramme en Java, C ++ et en pseudo-code est un atout majeur dans Iconic Programmer. Afin d'identifier la zone des structures imbriquées dans l'organigramme, des couleurs de fond sont utilisées. Celles-ci aident beaucoup surtout dans le cas d'un organigramme avec plusieurs éléments : ceci rend l'organigramme trop long et conduit les apprenants à la non compréhension des limites réelles de chaque structure (Chen et Morris, 2005).

II.10. Dev flowcharter

DF, ou Dev FlowCharter, est basé sur le système d'exploitation Windows. En DF, les utilisateurs peuvent faire leurs organigrammes alors un code source Pascal est automatiquement généré. L'interface DF est montrée dans la Figure 10.

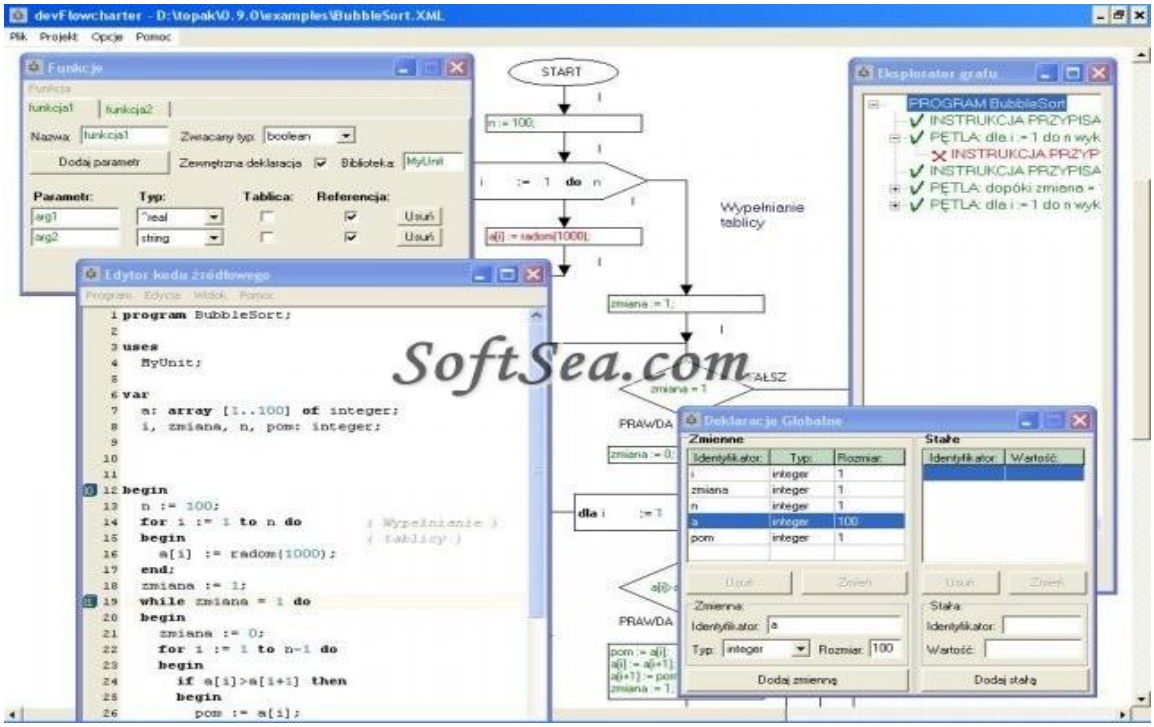


Figure N° 10: Dev Flowchart

Les caractéristiques de DF couvrent les affectations, les variables, les boucles et les tests, mais pas les tableaux et les procédures. Pour construire un organigramme, l'utilisateur choisit à partir d'un menu d'options les composants à ajouter. Le code source Pascal généré par DF peut être compilé et un compilateur Pascal supplémentaire est également fourni et installé par DF. A ce jour, aucune publication ne mentionne une quelconque évaluation de cet outil (Domagala, 2006).

II.11. Outil sans nom

Différentes méthodes ont été utilisées pour développer l'environnement de visualisation basé organigramme par Arai et Yamazaki de l'Université Tokyo, Japon. L'outil d'Arai et l'environnement Yamazaki sont représentés dans la figure 11.

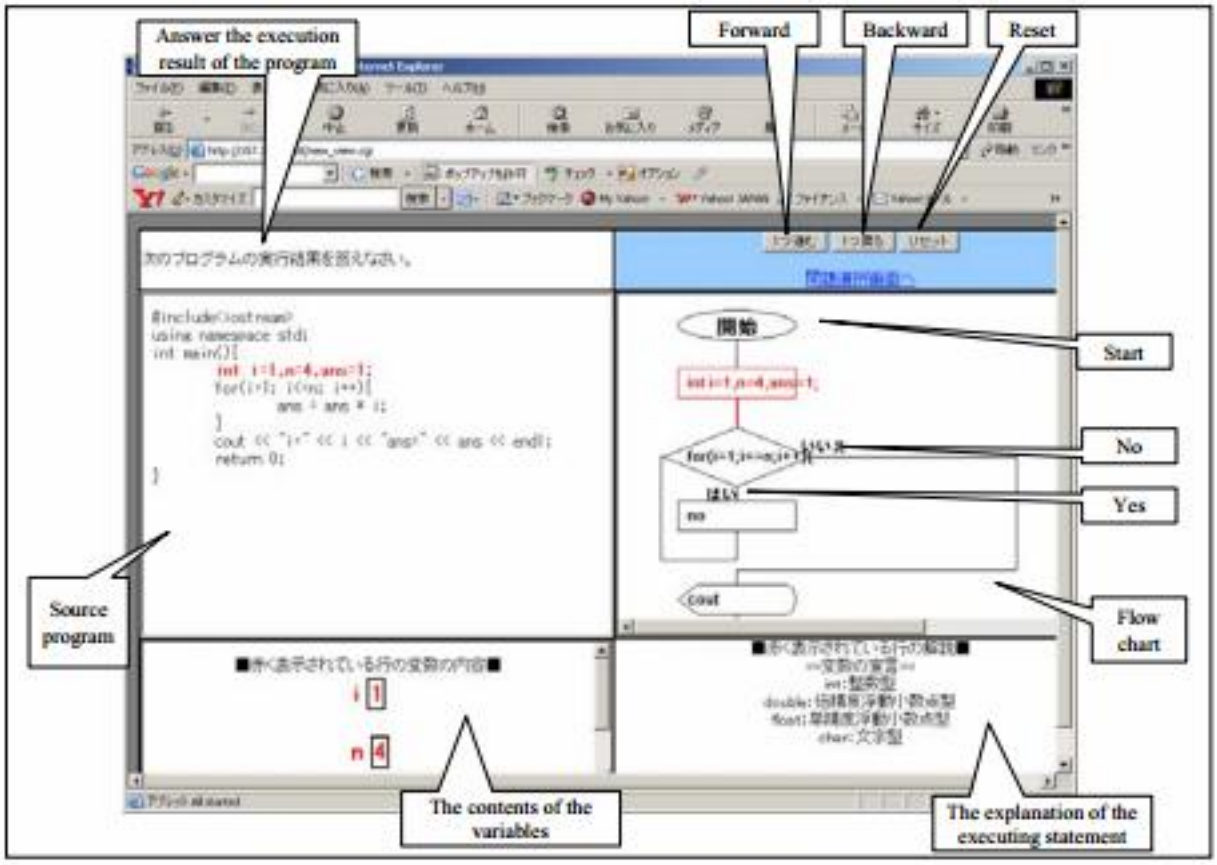


Figure N° 11: Outil inconnu

Semblable à des œuvres déjà examinées, cet environnement emploie également un organigramme exécutable et un code, mais sous la forme d'une application basée sur le Web pour fournir une plate-forme indépendante accessible par les navigateurs web modernes. Cela peut être considéré comme une amélioration par rapport à d'autres plates-formes indépendantes. Dans cette application, l'organigramme est affiché à côté du code. L'exécution du programme se fait étape par étape manuellement, mais il est bidirectionnel, en allant vers l'avant et vers l'arrière. Voir aussi l'organigramme et le code aide à synchroniser et à permettre aux novices de comprendre le lien entre les différents flux de donnée et de contrôle. En offrant un modèle d'exécution dans l'organigramme et le code, l'outil d'Arai et Yamazuki peuvent améliorer le traçage des compétences des apprenants. Une recherche a montré qu'aucune publication ou large recherche empirique n'a été réalisée pour évaluer cette application . En outre, il est largement accessible (Arai et Yamazaki, 2006).

II.12. B#

L'outil nommé B # a été développé à l'Université de Port Elizabeth et est entièrement basé sur le système d'exploitation Windows. Avec B#, la création d'un organigramme est réalisable par des novices et le code correspondant est automatiquement généré en Pascal. L'environnement de programmation à base d'organigramme B# est représenté dans la figure 12 :

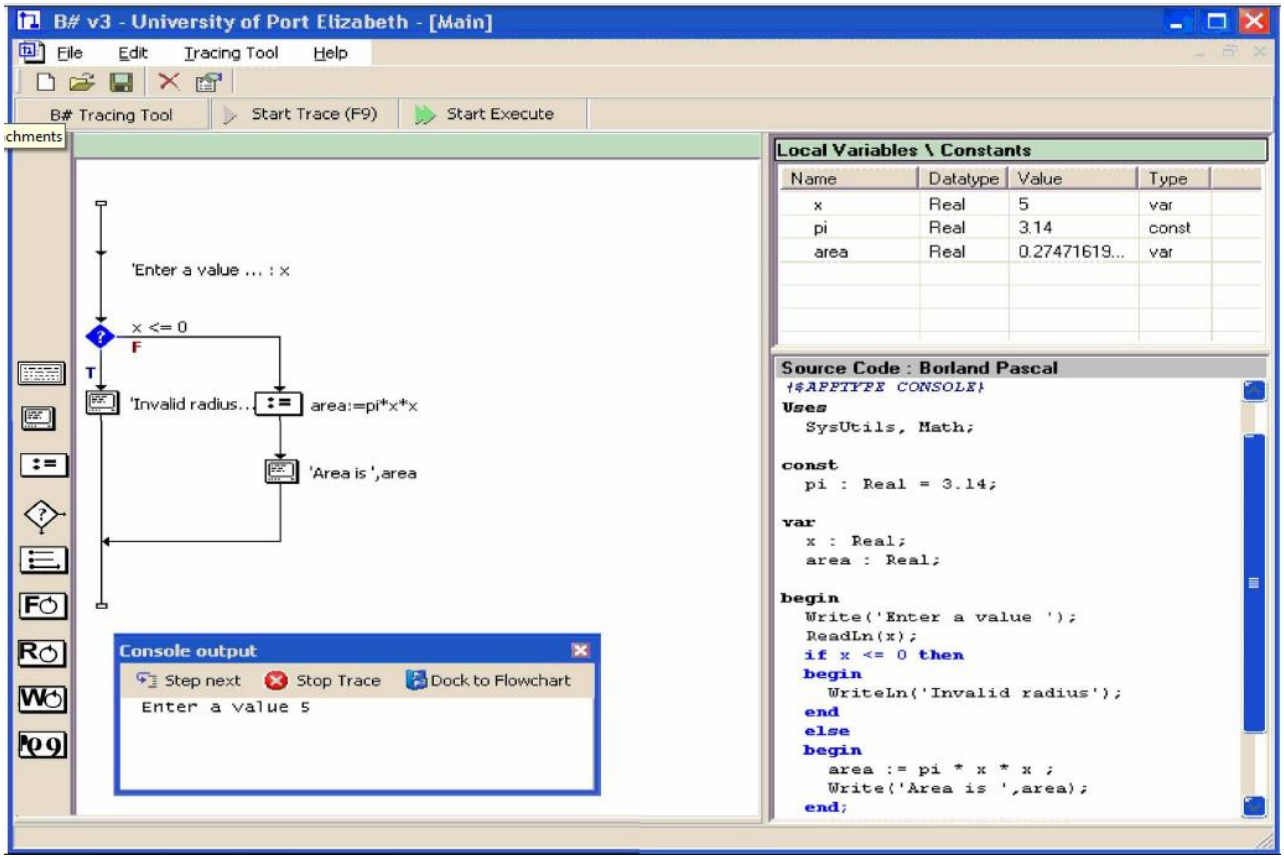


Figure N° 12: B#

Cet outil dispose d'auto-structuration dans la construction d'organigrammes. les apprenants débutants peuvent simplement faire glisser les options vers l'emplacement désiré dans l'organigramme (Greyling et al., 2006). Les paramètres de l'organigramme doivent être déterminés et ensuite vérifiées dans un mode syntaxique, après quoi les novices seraient alertés de toutes les erreurs possibles. Une fois que les composants sont ajoutés par glisser-déposer, le code Pascal lié sera généré de manière automatisée et l'organigramme sera restructuré et redessiné. En outre, B # fournit aux novices la possibilité de voir et ainsi de comprendre la connexion côte à côte entre le code et l'organigramme. Par conséquent, une force clé de B # est sa capacité à synchroniser l'organigramme avec le code généré. Il

convient cependant de noter que, en sélectionnant un composant de l'organigramme, sa ligne respective de code serait mise en évidence et non inversement. Cette synchronisation offre la possibilité à l'apprenant de visualiser l'exécution du programme. Les modèles graphique et textuel des exécutions sont bien représentés dans B #, comme les entrées, les sorties et l'état changeant des variables en mémoire. Une recherche sur ce produit a montré que la capacité des apprenants débutants qui utilisent cet outil a été considérablement améliorée par rapport aux autres élèves. Il est à noter que cet outil n'est pas accessible sur l'Internet (Greyling et al., 2006).

II.13. ProGuide

À l'Université de Coimbra au Portugal, un outil appelé ProGuide (PG) a été conçu pour améliorer / promouvoir les compétences de résolution de problèmes de novices en algorithmique. Cet outil permet de construire des organigrammes exécutables pour transmettre des solutions à une bibliothèque de problèmes de programmation. ProGuide, l'environnement est représenté dans la figure 13.

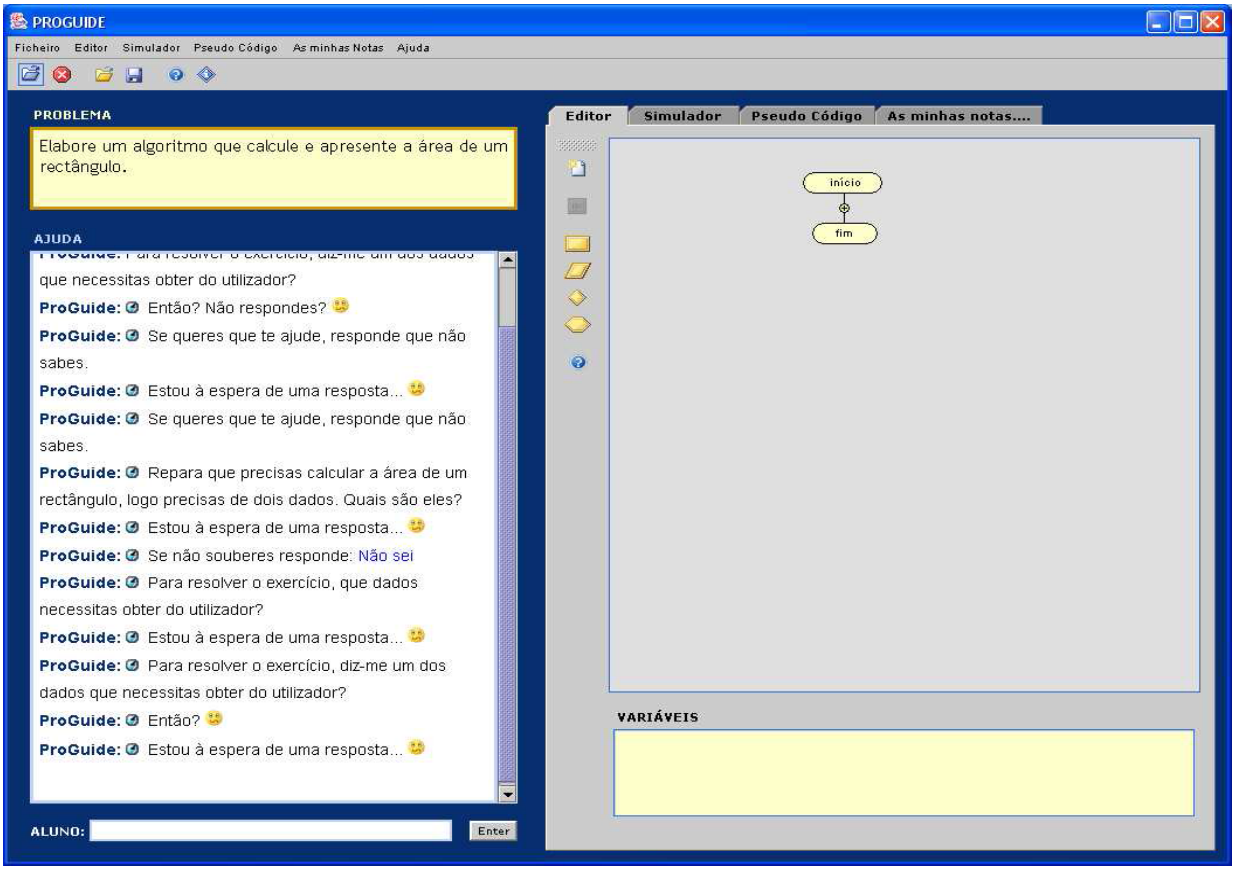


Figure N° 13: ProGuide

Dans PG, les boucles, les entrée/sortie, l'affectation et les expressions sont prises en charge. Bien que similaire à d'autres environnements examinés jusqu'à présent, cet outil dispose d'animation des programmes d'organigrammes construits au sein de son espace de travail ainsi que l'utilisation d'une approche d'auto-structuration pilotée par menu pour la construction d'un organigramme. Cela permet à l'apprenant de se concentrer sur la tâche à accomplir et ne pas avoir à lutter avec la réorganisation et la modification de l'organigramme. Une caractéristique unique de cet outil est qu'il représente un support qui fournit une assistance de programmation aux novices dans les activités de raisonnement. Cependant, il n'y a pas de soutien du programme général et il est dépendant de sa collection intégrée des activités de programmation. L'exécution de l'organigramme est prise en charge par cet outil visuel, offrant aux apprenants débutants un bon modèle d'exécution ainsi que la possibilité de vérifier l'exactitude du programme concerné. Une recherche a révélé qu'aucune évaluation accessible de PG n'existe dans la littérature, malgré de nombreux articles publiés par ses auteurs (Areias et Mendes, 2007).

II.14. UsingMicroworlds Pro

Cet outil a été conçu et développé par Glezou et Grigoriadou de l'Université d'Athènes dans le but d'enseigner la condition IF et les structures IF-ELSE aux enfants d'environ 14 ans dans un tutoriel sur la base d'organigramme. Il est basé sur les systèmes d'exploitation MS Windows et Macintosh. Glezou et Grodoriadoi est un didacticiel plutôt qu'un environnement, mais avec des caractéristiques qui surpassent celles des autres systèmes d'organigrammes de visualisation examinés.

L'organigramme joue un rôle énorme dans la transmission des concepts de structures de séquence et de sélection dans ces tutoriels. Les programmes sont présentés côte à côte dans des formes spécifiques, y compris pseudo-code, avec la description verbale et l'organigramme. L'exécution du programme modélisé est animée par ces quatre éléments par des moyens de synchronisation. De là, le novice est capable de comprendre le lien réel entre l'organigramme et le code généré, ainsi que la sémantique de la structure. La présentation multi-modèle indique également que plusieurs styles d'apprentissage ont une influence sur l'apprenant. Ce tutoriel conçu fournit la preuve que l'application des organigrammes pour enseigner les concepts fondamentaux de la programmation est une méthode appropriée et motivante pour les jeunes générations. Une étude a révélé qu'il n'y a pas d'évaluation empirique de Microworlds Pro et que ses travaux ne sont pas facilement accessibles.

Cependant, ses auteurs ont indiqué qu'une amélioration positive dans les performances des élèves était évidente lors de la mise en œuvre de l'outil (Glezou et Grigoriadou, 2007).

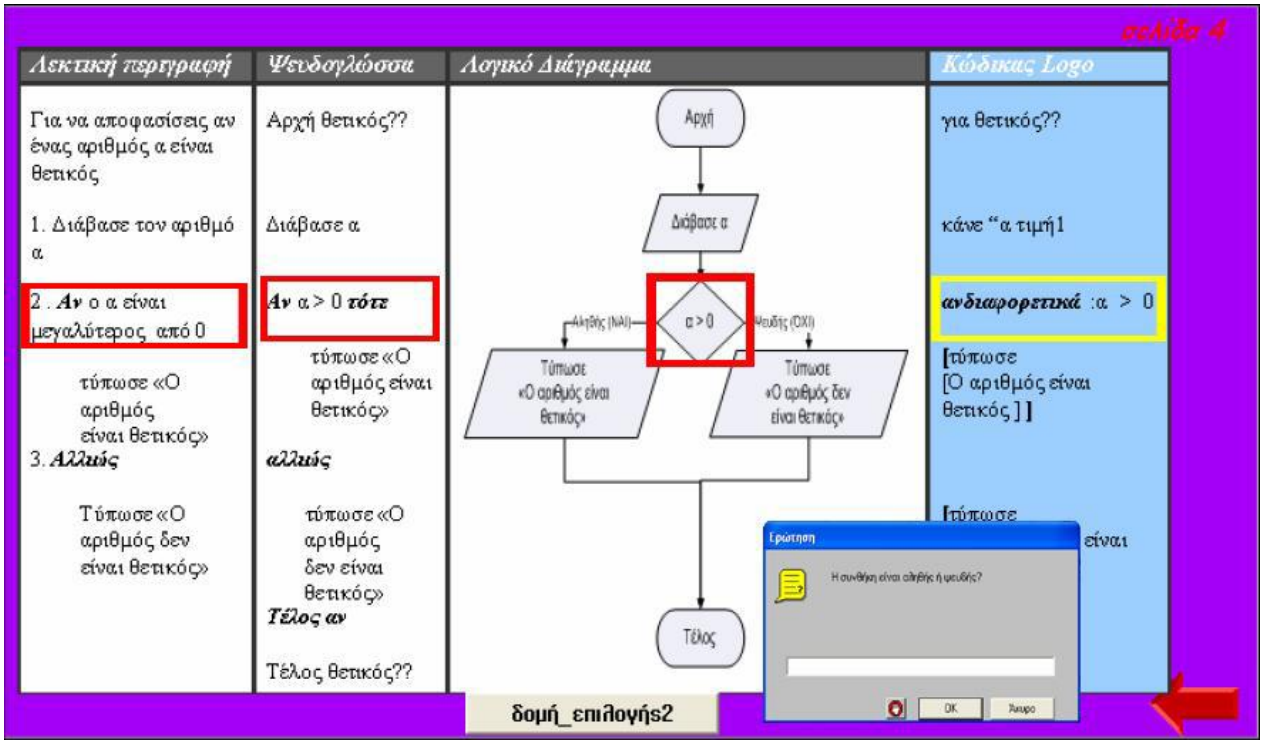


Figure N° 14: MicroWorlds Pro

II.15. Code visual to flowchart

Ce système, appelé code visuel à l'organigramme (CVF), est un projet commercial qui a été créé par Fatesoft. L'objectif principal est de permettre aux novices d'être en mesure de convertir leur programme en un organigramme afin de mieux comprendre et documenter le code généré. L'interface de CVF est représentée dans la figure 15.

Un organigramme généré pour chaque ligne de code entrée dans l'éditeur de texte de CVF. Ainsi, la méthode d'obtention de l'organigramme d'un programme est faiteée par ce code afin de réduire les mauvaises habitudes de programmation et pour supprimer le programme dans lequel la sélection n'a pas été effectuée avec soin. La synchronisation bidirectionnelle est prise en charge par CVF et cette caractéristique est la force majeure de cet outil. Cela signifie que l'organigramme et son code sont présentés côte à côte et une fois une ligne de code est cliquée, son composant organigramme respectif est mis en surbrillance et vice versa. Cela permet aux novices de voir et de comprendre le véritable lien entre un organigramme et son code source et comment exactement l'organigramme et le code se

rapportent les uns aux autres. Il est cependant, un produit commercial, donc il n'y a aucune forme de recherche universitaire sur cet outil. Ainsi, aucune évaluation formelle sur CVF n'existe et il ne semble pas exister dans les milieux éducatifs (FateSoft, 2009).

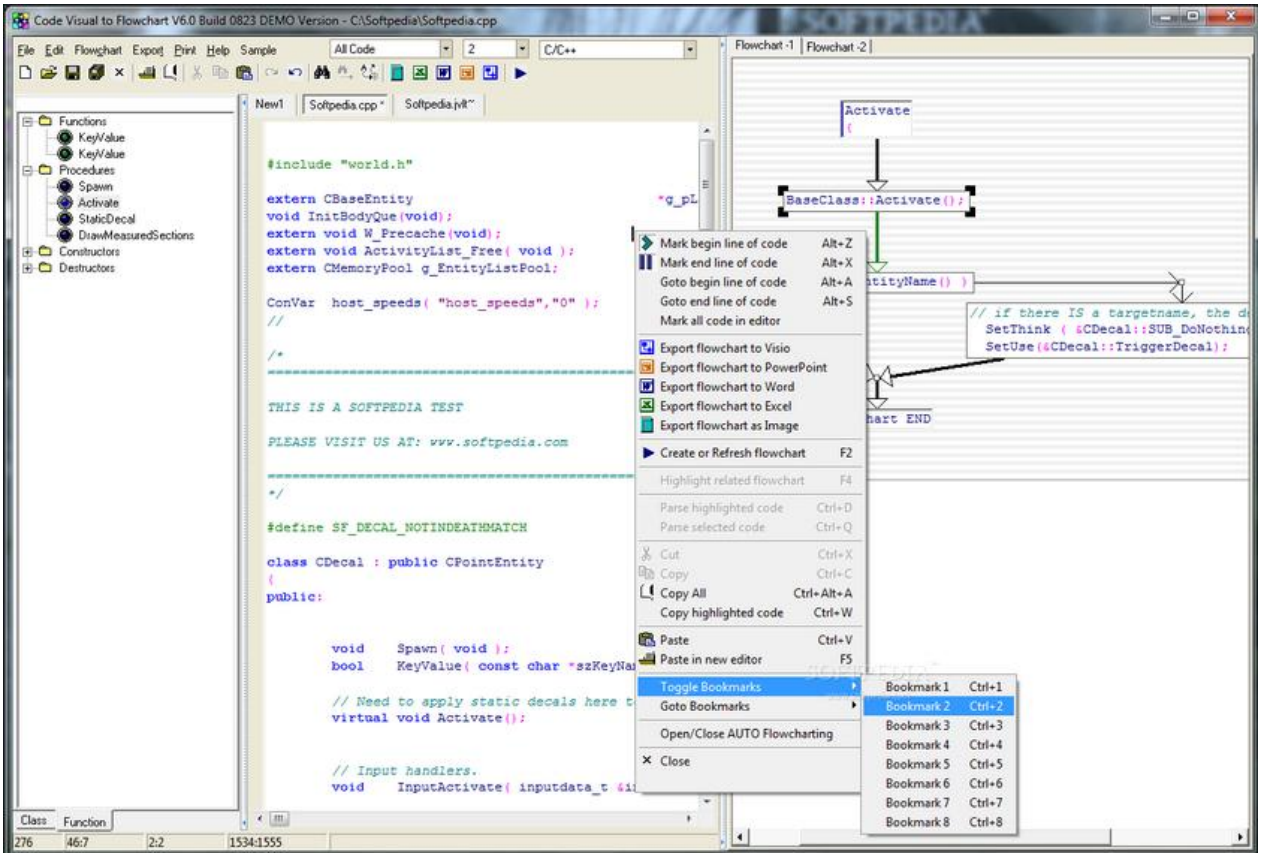


Figure N° 15: Code Visual to flowchart

II.16. The web-based flowchart system

Le système d'organigramme Web base flowchart est un outil d'aide à la programmation en organigrammes basé sur le Web. Différentes animations, des symboles et des images statiques sont produites par ses kits LED basés sur l'organigramme. Il peut être mis en œuvre dans Microsoft Visual Basic et HTML. Interface Web-organigrammes est représentée sur la figure 16.

Les utilisateurs sont autorisés à changer les paramètres d'un programme par l'intermédiaire d'une interface graphique. Comme pour la plupart des travaux évalués à ce jour, les concepts de programmation de base tels que des tableaux, des fonctions, des variables, des boucles et les conditions IF / Structure ELSE sont couverts par ce web-organigramme. Deux cours sont offerts au sein de ce système qui vise les apprenants

débutants et utilisateurs expérimentés. Une animation simple peut être créée par les apprenants avec cette configuration de base et un système de minuterie peut contrôler la vitesse du mouvement du point en millisecondes. Les étudiants ne sont pas tenus de se familiariser avec toute la syntaxe de commande d'impression particulière afin de coder les animations générées. Ainsi, la logique de l'algorithme derrière l'animation permet aux étudiants de se concentrer et de recevoir une rétroaction instantanément et ne pas négliger de se familiariser avec les principes fondamentaux des concepts de programmation (Chun et Ryoo, 2010).

Figure N° 16: Web base Flowchart

II.17. Programinate

Une application basée sur le Web appelé Programinate a été créée pour concevoir et exécuter des organigrammes dans un environnement e-learning web intégré. L'environnement de programmation Programinate est représenté sur la figure 17.

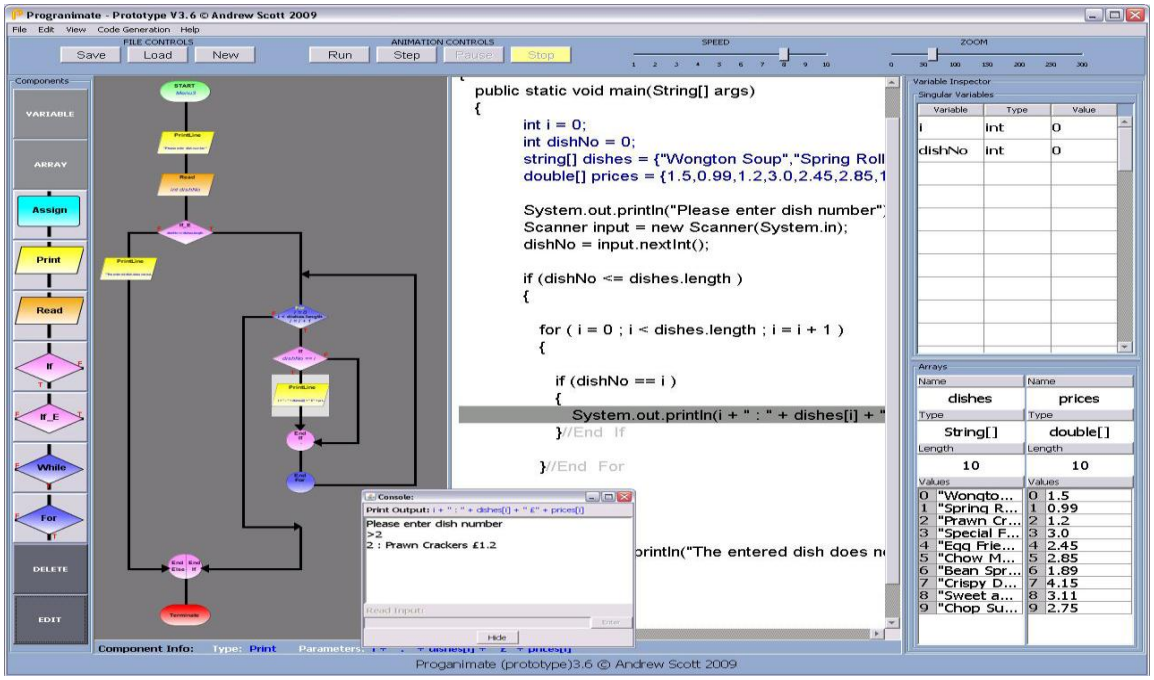


Figure N° 17: Programinate

Comme dans Raptor, l'auto-structuration est utilisée dans la construction d'organigramme. Cependant, dans Programinate, les structures de contrôle et les commandes sont ajoutées en cliquant, alors que dans Raptor, ils doivent être glissés et déposés. Les organigrammes peuvent être construits en interagissant avec le code choisi ou l'organigramme (Scott et al., 2008a). Contrairement à d'autres œuvres examinées jusqu'ici, cet outil prend en charge les commandes les plus séquentielles, structures et procédures sélectives et en boucle et fonctions. Les représentations sont standards et les différentes couleurs servent à éviter toute confusion. En outre, l'extrémité de chaque structure de commande est désignée par une représentation particulière pour éviter toute confusion quant à la portée de la structure. Dans Programinate, les novices sont pourvus d'une opération de traçage de l'organigramme étape par étape et par la visualisation simultanée des données. L'exécution synchronisée des organigrammes et du code source d'une manière automatique est considérée comme la principale force de cet outil, ce qui permet aux novices de voir et de comprendre la relation directe entre un composant d'un organigramme et sa ligne connexe de code source. Ainsi, cet outil est considérablement utile tant pour les apprenants visuels et verbaux. Depuis la syntaxe entrée par la structure de Programinate, cet outil donne aux novices un feedback utile et parfait en ce qui concerne la correction du programme. Il convient de noter que, avec Programinate, les novices peuvent obtenir un code Java, VisualBasic.NET, Pascal ou JavaScript avec un organigramme.

Tableau N° 1: tableau comparatif des systèmes (1)

Features	*BACCII*	*FLINT*	*EC*	*FCI*	*Raptor*	*SFC*	*SICAS*	*VL*	*IP*	*DF*	*A&Y*	*B#*	*PG*	*G&G*	*CVF*	*Web-flowchart*	*Programimate*
Year published	*1992*	*1999*	*2002*	*2003*	*2004*	*2004*	*2004*	*2004*	*2005*	*2006*	*2006*	*2006*	*2007*	*2007*	*2009*	*2010*	*2010*
Flowchart	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Flowchart-based programming	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓
Flowchart generates code	✓				✓	✓	✓	✓	✓	✓		✓					✓
Structural rules enforced	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓		✓		✓
Colour used to fully differentiate components and structures	✓	✓	✓							✓							
Code	✓				✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓
Code-based programming															✓		
Code generates flowchart															✓		
Code and flowchart displayed concurrently						✓					✓	✓		✓	✓		✓
Synchronised highlighting of flowcharts and code											1/2	1/2		✓	✓		✓

Tableau N° 2: Tableau comparatif des systèmes (2)

Features	*BACCII*	*FLINT*	*EC*	*FCI*	*Raptor*	*SFC*	*SICAS*	*VL*	*IP*	*DF*	*A&Y*	*B#*	*PG*	*G&G*	*CVF*	*Web-flowchart*	*Programimate*
Year published	*1992*	*1999*	*2002*	*2003*	*2004*	*2004*	*2004*	*2004*	*2005*	*2006*	*2006*	*2006*	*2007*	*2007*	*2009*	*2010*	*2010*
Synchronised visual execution of flowcharts and code											✓			✓			✓
Non visual execution				✓				✓									
Visual execution		✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓			✓
Variable inspector	✓	✓	✓	✓	✓		✓		✓		✓	✓					✓
Error feedback	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓			✓
Java support					✓		✓	✓	✓						✓		✓
Variables	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sequence	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Selection	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Iteration	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
Arrays	✓				✓	✓	✓	✓							✓	✓	✓
Procedures	✓	✓			✓		✓	✓							✓	✓	
Empirically evaluated	✓	✓			✓			✓				✓					✓
Associated pedagogy		✓											✓	✓			✓
OS dependency	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Any	Win	Win	Win/ mac	Win	Win	Independent

III. DISCUSSIONS

Dans cette thèse, les organigrammes sont considérés comme une représentation graphique d'un algorithme, donc la problématique principale de cette thèse est l'évaluation des organigrammes en algorithmique. Nos constatations sont relatives à des environnements de programmation basés organigrammes et leur impact dans l'amélioration de la compréhension et la résolution de problèmes pour les apprenants débutants sont :

• Chaque fois qu'un organigramme est traduit en code, les apprenants débutants sont exposés à une représentation textuelle, contribuant à mettre davantage l'accent sur les compétences de résolution de problèmes et les concepts essentiels de la programmation en réduisant le travail de syntaxe.

- En réduisant l'accent sur la syntaxe, les apprenants débutants sont capables d'augmenter l'effort consacré à l'analyse des problèmes et la conception de la solution.
- Les organigrammes sont une «représentation universelle», ils permettent une représentation de l'information clairement et sans ambiguïté. Bien qu'il y ait eu un intérêt croissant dans les techniques de programmation visuelle, aucun système visuel n'a obtenu l'acceptation universelle des organigrammes.
- Les apprenants débutants qui utilisent un outil d'apprentissage basé organigrammes dans un cours d'Introduction à la programmation peuvent montrer des améliorations significatives dans la logique de programmation et les compétences d'écriture de code. Ceci suggère que l'utilisation d'un outil de schématisation approprié avec rétroaction automatique dans une pédagogie conçue de manière appropriée peut améliorer l'expérience d'apprentissage de résolution de problèmes pour les apprenants débutants.
- Les organigrammes représentent une séquence d'actions. Ces actions sont les compétences de base que les apprenants débutants ont besoin d'acquérir pour résoudre un problème en algorithmique.

En évaluant les systèmes et les outils précédents, il est évident que fournir un environnement de résolution de problèmes en algorithmique contribue à l'amélioration des compétences de résolution de problèmes chez les apprenants débutants.

Comme un organigramme peut être rapidement compris sans aucune exigence et avec juste une petite quantité d'information, les apprenants débutants focalisent sur les impératifs fondamentaux (structures d'itération, de séquence et de sélection) tout en mettant l'accent sur la composition du programme de flux de donnée et d'exécution.

Parceque les organigrammes sont capables d'être utilisés dans l'enseignement de stratégies de simulation mentale, ils peuvent considérablement aider à améliorer les compétences de résolution de problèmes et la modélisation de la composition du programme pour les apprenants débutants.

Dans la majorité de ces outils, l'auto-structuration sert à l'élaboration de l'organigramme. Même si un modèle virtuel du programme écrit est fourni par CVF (Code visual to flowchart) ainsi que la production automatique du code de l'organigramme, cet outil ne peut rien faire pour résoudre la surcharge syntaxique de la programmation. Sur la totalité des systèmes examinés, seulement neuf ont la capacité de générer du code à partir de leur organigramme, à savoir BACCII, Raptor, SFC, VL, IP, B #, SICAS, DF et Progranimate; dont trois sont capables de représenter simultanément un organigramme à côté du code généré (SFC, B # et Progranimate). Cela ne signifie pas que d'autres systèmes ne peuvent pas générer du code à partir d'un organigramme ; certains le font mais mal tandis que d'autres insèrent le résultat dans un fichier texte indépendant qui est sans rapport avec l'environnement de l'organigramme. Cela conduit à une perte de temps car les novices doivent chercher entre les deux représentations afin de convertir l'organigramme en code plutôt que de se concentrer sur la compréhension des concepts de programmation. Pour enseigner les compétences de traçage de code par l'exemple et pour construire des modèles plus complets de l'exécution, un système devrait être mis au point pour prendre en charge les deux modèles d'exécution : visuel et textuel.

A titre d'exemple, le travail de Arai et Yamazakia a démontré la représentation d'un programme dans les formes visuelles et verbales, synchronisées côte à côte. Cependant, le développement du programme ne peut pas être pris en charge par cet outil. Même si la génération de code de l'organigramme , la présentation d'un organigramme et son code source sont disponibles avec l'éditeur SFC; il est en mesure d'exécuter ses programmes. Bien que dans l'environnement de B#, les apprenants sont autorisés à accéder au développement et à l'exécution visualisées ainsi qu'à la représentation visuelle et verbale d'un programme, B# ne parvient pas à offrir d'autres langage tel que Pascal. Un environnement, à savoir

Programinate, facilite la programmation via une convention standard de schématisation, la génération de code et la présentation d'organigramme à côté du code dans les deux formes visuelle et textuelle. Malheureusement, il n'y a pas d'environnements avec une plate-forme indépendante capable d'accepter tout programme.

Dans ce qui suit, les avantages et les atoûts considérables des travaux examinés sont énumérés ainsi que certains de leurs inconvénients.

III.1. Avantages

- Raptor est basé sur les paradigmes de programmation orientés objet de BACCII
- B # permet aux novices d'intégrer et de parcourir le programme qui a été automatiquement généré à partir de l'organigramme,
- Programinate fournir aux novices l'exécution synchronisée d'un organigramme et son code respectif,
- IP et Programinate offrent une visualisation explicative pour augmenter la compréhension des apprenants novices, de la sémantique des structures de programmation ainsi que du flux de contrôle,
- Web-organigramme et Programinate ont été conçues et développées comme des applications basées sur le Web pour développer un système d'e-learning amélioré,
- SICAS et ses différentes versions (SICAS-COL, H-SICAS) a comme avantage l'utilisation de la fonctionnalité dans les appareils mobiles, ainsi que des activités de collaboration qui peuvent considérablement contribuer à l'amélioration de la programmation en groupe.
- ProGuide fournit aux novices un modèle de tutorat pour guider et aider dans le processus d'élaboration d'une solution pour un problème donné.

III.2. Inconvénients

Comme certains systèmes ne suivent pas les conventions d'organigramme standard, les systèmes ne sont pas entièrement équipés pour former les apprenants sur les méthodologies de conception des programmes conventionnels ce qui décourage et réduit l'efficacité de l'apprenant dans l'apprentissage de la programmation.

Étant donné que certains systèmes ne fournissent pas la capacité d'exécuter son programme, ils sont incapables de fournir aux apprenants un modèle adéquat de l'exécution des codes le privent ainsi du soutien pour le développement de tests de programme, le déboggage, les compétences de traçage ou des formes d'organigramme.

La plupart des systèmes n'utilisent pas la couleur dans leurs organigrammes , rendant ainsi difficile la distinction entre les symboles. Cette limitation peut conduire à la confusion et peut entraver le processus d'apprentissage.

Certains systèmes ne fournissent pas un support complet pour examiner l'énoncé du problème et le traduire en un organigramme et procéder à l'exécution du code. Par conséquent, les utilisateurs sont amenés à surfer d'un environnement à un autre.

Dans certains des systèmes examinés, des expressions mathématiques et les types de données booléens ne sont pas entièrement pris en charge rendant impropre la construction d'une gamme de programmes.

Quelques-uns des outils ne supportent pas les apprenants verbaux. Cela peut conduire à un retard dans l'apprentissage de la syntaxe.

Les fonctions de réglage de programmation supplémentaires comme l'affichage de l'organigramme et le code exécutable lié d'une manière synchronisée doivent être pris en charge de manière à aider les utilisateurs à développer leur capacité de traçage de code.

La plupart des outils étudiés ne fournissent pas la capacité de produire des codes exécutables obligeant les utilisateurs à plus d'efforts pour convertir le pseudo-code résultant dans des codes de langages de programmation. La plupart de ces outils ne supportent que Pascal , Par conséquent, ces outils ne sont pas appropriés que pour l'enseignement de cours d'introduction à la programmation.

D'autres outils ne fournissent pas de conversion automatique de l'organigramme en code exécutable. Cette lacune empêche les élèves de se concentrer sur l'amélioration de leurs compétences de résolution de problèmes, et les pousse à se concentrer davantage sur les détails de la conversion de la syntaxe.

Un autre problème est qu'aucun outil ne fournit un feedback sur l'organigramme produit par l'apprenant.

IV. CONCLUSIONS

Cette partie a examiné les recherches récentes dans le domaine des environnements de programmation basés sur la représentation graphique des algorithmes et des programmes sous formes d'organigrammes spécifiquement destinés aux apprenants novices. Cet état de l'art a présenté une vue d'ensemble des nombreuses approches et techniques utilisées dans différents environnements et outils pour acquérir une meilleure compréhension de la recherche entamée intéressée par cette thèse. En outre, grâce à l'analyse et la comparaison entre les différentes catégories de recherche, ce chapitre a identifié les tendances et les développements récents dans les approches et techniques pour les environnements de programmation à base d'organigrammes destinés aux apprenants novices. Bien que nous avons essayé rigoureusement d'explorer les recherches qui ont eu lieu dans le domaine des environnements informatiques pour l'apprentissage de la programmation à base d'organigrammes, spécifiquement pour les apprenants novices, il semble encore difficile d'appréhender le vaste éventail des problèmes et des tentatives de recherche même si ce domaine de recherche est très ciblé par la communauté des chercheurs en e-learning.

L'objectif principal de cette thèse est d'étudier les moyens qui servent à améliorer la capacité de programmation des novices qui suivent des cours en programmation en fournissant un modèle mental exact et précis et une compréhension conceptuelle des concepts de programmation essentiels et d'améliorer ainsi leurs compétences en résolution de problèmes. Par conséquent, cette étude conclue que l'organigramme est une aide visuelle efficace pour le développement de la programmation. L'accent est mis sur les impératifs fondamentaux ainsi que les compétences de composition de programmes et flux d'exécution.

Bien que la programmation à l'aide des organigrammes puisse aider un novice à apprendre la programmation, un système complètement visuel peut avoir un impact négatif sur les apprenants verbaux. En offrant un programme à la fois textuel et visuel, ce problème peut être résolu. En outre, un système qui est uniquement visuel peut éliminer les problèmes d'apprentissage de syntaxe du langage de programmation, mais il ne doit pas diminuer l'influence de la syntaxe lorsqu'elle est finalement introduite dans l'apprentissage des novices. Les systèmes qui génèrent du code, mais n'oblige pas les utilisateurs à écrire du code connexe, n'offriront qu'une introduction légère à la syntaxe du langage de programmation tout

en maintenant leur attention sur la question de la résolution de problèmes, la compréhension des fondamentaux sous-jacents et le flux d'exécution.

Cette étude critique a montré qu'un grand nombre d'auteurs croit aux organigrammes qui sont des représentations influentes dans l'apprentissage de notions de programmation. Bien que ces systèmes inclus des caractéristiques pratiques, ils étaient incomplets ou mal conçus et pas un seul d'entre eux n'est véritablement complet. L'une de nos principales recommandations est de concevoir et de développer un environnement simple et complet offrant une plate-forme indépendante pour représenter toutes les étapes à partir de l'énoncé du problème de programmation en langage naturel, qui présente des informations utiles, à son organigramme correspondant avec un engagement des apprenants débutants dans le processus d'analyse et de construction de l' organigramme, puis la composition et l'exécution du programme.

C'est dans cette démarche que nous nous sommes engagés à travers cette thèse.

Chapitre II :

Environnements Informatiques pour l'Apprentissage Humain (EIAH) et Evaluation des apprenants

I. Introduction

Historiquement, les outils informatiques utilisés dans le cadre scolaire ou de la formation ont toujours intégré des moyens d'évaluer les apprenants. Les Environnements Informatiques pour l'Apprentissage Humain (EIAH) (Tchounikine, 2002) ne dérogent pas à cette règle.

L'évaluation joue un rôle particulier en EIAH. Selon Charles (Juwah, 2003), chercheur dans le domaine des EIAH, l'évaluation doit :

– Etre motivante pour l'apprenant ;

– Encourager une activité d'apprentissage soutenue ;

– Contribuer à la progression de l'apprenant ;

– Etre faible en coût humain et facilement maintenable.

Son point de vue est que l'évaluation joue un rôle primordial dans l'activité pédagogique, non seulement comme le moyen de vérifier les acquisitions mais aussi comme le moyen de motiver ces apprentissages et d'inciter les élèves à progresser. C'est la volonté d'obtenir de bons résultats qui doit inciter l'élève à apprendre et c'est la satisfaction qu'il éprouve à réussir qui doit lui donner l'envie de persévérer et d'aller plus loin. L'enseignant n'est plus seulement confronté à la nécessité de comprendre ce qui fait obstacle à la réussite de ses élèves mais il peut aussi, lorsqu'il utilise certains outils informatiques, se servir de l'évaluation comme un "moteur" pour les apprentissages (Dintilhac& Rac, 2005), l'évaluation devenant un moyen pédagogique. CommeditPapert, "It was hard but it was fun" (Jaillet, 2003). La dimension ludique est exploitée, au risque du consumérisme et de l'abandon des objectifs pédagogiques. Les études réalisées par exemple sur l'outil TDmaths(Dinet et Rouet, 2002), (Dinet et Rouet, 2003) montrent que les apprenants apprécient l'aspect ludique, le fait par exemple de gagner des récompenses lorsqu'ils réussissent une tâche (les bons points de notre enfance). Pourtant, les enseignants utilisent assez peu les scores réalisés par les élèves pour noter leurs compétences, comme s'ils ne prenaient pas ces évaluations au sérieux (Dinet et Rouet, 2002), (Dinet et Rouet, 2003). Les méthodes de calcul des scores sont bien souvent opaques et pas toujours adaptées aux besoins de l'enseignant.

Les pratiques d'évaluation en EIAH sont cependant plus variées qu'il n'y paraît. Dans ce chapitre, nous tentons de présenter et commentons une vue exhaustive de ces pratiques.

II. Les pratiques d'évaluation

Pour mener les nécessaires évaluations (Dintilhac et Rac, 2005) liées à ses enseignements, l'enseignant dispose de plusieurs types d'évaluation :

– l'évaluation pronostique ;

– l'évaluation formative ;

– l'évaluation formatrice ;

– l'évaluation diagnostique ;

– l'évaluation sommative.

Sont absentes de cette typologie les évaluations normatives (Hadji, 1990) et critériées (Hadji, 1997). L'évaluation normative consiste à comparer les résultats d'un apprenant par rapport à ceux d'autres apprenants. En outre, nous pensons que tout type d'évaluation repose sur des critères, explicites ou non. Les évaluations normatives et critériées sont donc transversales aux cinq autres présentées ici.

II.1. L'évaluation pronostique

L'évaluation pronostique "fonde des décisions de sélection ou d'orientation en fonction de l'aptitude présumée à suivre un nouveau cursus, par exemple telle filière du secondaire ; elle se situe en amont d'un cursus et sous-tend un choix" (Perrnoud, 2001). L'évaluation pronostique 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.

II.2. L'évaluation formative

L'évaluation formative est une "évaluation dont l'ambition est de contribuer à la formation" (Hadji, 1990). Dans ce contexte, elle a pour but de réguler l'enseignement. L'évaluation fournit des informations permettant à l'enseignant d'adapter son enseignement aux particularités de l'apprenant, elle entre dans le cadre d'un enseignement différencié (Allal, Cardinet et Perrnoud, 1979). Comme l'explique P. Perrenoud (1997), "toute différenciation appelle une évaluation formative". Pédagogie différenciée et évaluation formative sont intimement liées. L'évaluation formative joue le rôle d'une "discrimination positive" qui vise à emmener chaque apprenant, en tenant compte de ses différences, à un

niveau de connaissances. La pédagogie différenciée s'inscrit dans un processus égalitaire d'acquisition d'un savoir. En ce sens, cette pédagogie et l'évaluation formative qui l'accompagne ont du mal à faire leur place dans l'enseignement scolaire encore habitué à une pédagogie élitiste qui vise à sélectionner les élèves. 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. (Perrnoud, 1997).

II.3. L'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 régulation est assurée par l'apprenant. En ce sens, l'activité d'auto-évaluation, qu'elle soit individuelle, mutuelle ou collective, est une évaluation formatrice. L'auto-évaluation est une "évaluation interne conduite par le sujet de sa propre action et de ce qu'elle produit. C'est un processus d'altération de son référentiel d'action au cours de confrontations entre son propre référentiel et celui ou ceux d'autrui" (Campanale, 1997). L'auto-évaluation ne peut donc pas être contrainte, elle est tributaire du bon vouloir de l'évalué.

II.4. L'évaluation diagnostique

L'évaluation formative s'appuie en partie sur l'évaluation diagnostique. 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.

La notion de compétence abordée ici est récurrente dans la littérature sur l'évaluation. Selon Chomsky (1965), la compétence en linguistique désigne le système de règles intériorisées qui permet de comprendre et de produire un nombre infini de phrases inédites. Tandis que la performance désigne la manifestation de la compétence des locuteurs et réfère à la diversité des actes de langage et des contextes d'énonciation et de communication. Ainsi, la performance peut être considérée comme la mise en application, en pratique d'une compétence. C'est donc par la performance qu'on évalue la compétence de l'apprenant.

En sciences de l'éducation, une compétence est souvent vue sous l'angle d'un ensemble de savoir-faire conceptualisés (Malglaive, 1990), (vergnaud, 1995).

II.5. L'évaluation sommative

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).L'évaluation met donc l'accent sur les performances, elle contrôle les connaissances. Elle est en opposition avec l'évaluation formative, elle ne régule pas l'apprentissage, elle le contrôle. 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 performances, l'évaluation sommative s'intéresse essentiellement aux productions réalisées par les apprenants (Campale, 2001).

III. Pratiques d'évaluation en EIAH

L'évaluation sommative est la plus pratiquée, compte tenu de la facilité de sa mise en œuvre. Elle repose bien souvent sur des questionnaires à choix multiples ou fermés relativement faciles à évaluer automatiquement. Les systèmes de tests adaptatifs (CAT : Computer Adaptive Test) (Green, Bock, Humphreys et al, 1984) supportent ce type d'évaluation et permettent d'envisager leur usage dans le contexte d'une évaluation pronostique.

L'évaluation diagnostique a fait l'objet de recherches bien connues en France, avec entre autres le logiciel Pépite (Delozanne et Grugeon, 2004). Contrairement à l'évaluation sommative ou pronostique rencontrée dans les CAT, l'évaluation diagnostique de Pépite permet d'obtenir un bilan de compétences de l'apprenant, et non un simple score.

L'évaluation sommative qui a pour mission de certifier les connaissances de l'apprenant et l'évaluation diagnostique peuvent être associée. Dans la classe, l'enseignant fait souvent appel aux deux, l'évaluation diagnostic permettant à l'enseignant de réguler l'apprentissage. Certains EIAH, tels que TDmaths3 essaient d'intégrer ces deux types d'évaluation.

Toujours dans un souci de régulation des apprentissages, l'évaluation formatrice est aussi rencontrée en EIAH. L'auto-évaluation avec le logiciel GenEval (David, 2003) et l'évaluation par pairs ou co-évaluation avec des systèmes tels que OASYS (Bhalerao et Ward, 2001), constituent les deux formes les plus connues. L'évaluation formatrice est souvent utilisée lorsque l'évaluation n'est pas automatisable (productions libres des apprenants) (Maor, 1998), (Juwah, 2003).

L'évaluation formative est présente sous l'angle de l'assistance à la régulation des activités par l'enseignant. L'objectif est alors de fournir des indicateurs à l'enseignant lui permettant de mener des évaluations formatives en relation avec une activité d'apprentissage. Ces dernières années, de nombreux systèmes de "monitoring" de l'activité pédagogique sont apparus, tels que FORMID SUIVI (Gueraud, Adam, Perin et al, 2005), GISMO (Mazza et Milani, 2005), mais aussi Aplusix (Sander, Nicaud, chachoua et Croset, 2005) qui enregistre la totalité des actions des apprenants.

D'autres évaluations connexes pouvant être utilisées dans une démarche formative ou diagnostique existent en EIAH. C'est le cas de l'évaluation de la participation et de l'évaluation par portfolio.

L'évaluation de la participation s'appuie généralement sur des outils de communications qu'il s'agisse de forums (S.Fujitania, T.Mochiduki ,Y.Isshiki et al, 2003), (Mochizuki, Fujitani, Hisamatsu et al, 2004) (Bratitsis et Dimitracopoulou, 2005)ou de chats (Georgr, 2001).

L'évaluation par portfolio permet d'évaluer la progression de l'apprenant dans ses apprentissages (Ministère de l'Education du Québec, 2002), (Eyssautier et Bavay, 2004).

Dans la suite, nous allons détailler chaque type d'évaluation en EIAH en l'illustrant par un exemple.

III.1. L'évaluation sommative et pronostique

L'évaluation sommative est souvent menée dans des environnements d'apprentissage basés sur des banques de questionnaires à choix multiples (QCM). Chaque questionnaire est organisé en parcours et sanctionne l'apprenant par une note finale. Le test du TOEFL qui permet d'évaluer le niveau d'anglais d'une personne étrangère est certainement le plus connu.

Ce test fait partie de ce qu'on appelle les tests adaptatifs par ordinateurs ou CAT pour "Computer Adaptive Test". Ces tests permettent de certifier des connaissances de base (Biggs, 1999). Outre une évaluation sommative, un CAT peut aussi offrir une évaluation pronostique comme le test du GRE qui conditionne l'accès aux formations universitaires par les bacheliers aux États-Unis.

Les tests du TOEFL et du GRE reposent sur ce système. Ils sont administrés par le service des tests éducatifs, qui est une organisation privée à but non lucratif travaillant en étroite collaboration avec les services d'éducation américains.

Le principe de base d'un CAT est simple. Un élève se place seul devant un ordinateur et répond à une liste de questionnaires à choix multiples. Pour chaque question, si la réponse est correcte, la question suivante sera un peu plus difficile. Dans le cas contraire, la question suivante sera un peu plus facile (Auger et Séguin, 1992). Il en va ainsi tout au long du test, ce qui permet au logiciel de déterminer le niveau du candidat et de lui attribuer un score. Les systèmes de CAT s'appuient sur le modèle de la théorie de réponse aux items (Item Response Theory) (Lord, 1980). Ce modèle guide le concepteur dans la construction d'un système de CAT. Le but de ces modèles est d'optimiser la cohérence entre les scores obtenus et les réelles compétences de l'évalué.

III.1.1. L'évaluation dans les CAT

Dans un CAT, on évalue des réponses à un QCM. A chaque question, la réponse est juste ou fausse, c'est une notation binaire qui conduit à une note finale après évaluation de la dernière réponse. Plus le nombre de questions est important plus la note finale est précise. L'évaluation globale menée par un CAT est sommative et individuelle. Elle appartient au registre du "testing". Un CAT permet de certifier un niveau de compétences, que ce soit en anglais pour le TOEFL ou sur les connaissances pré-universitaires pour le GRE.

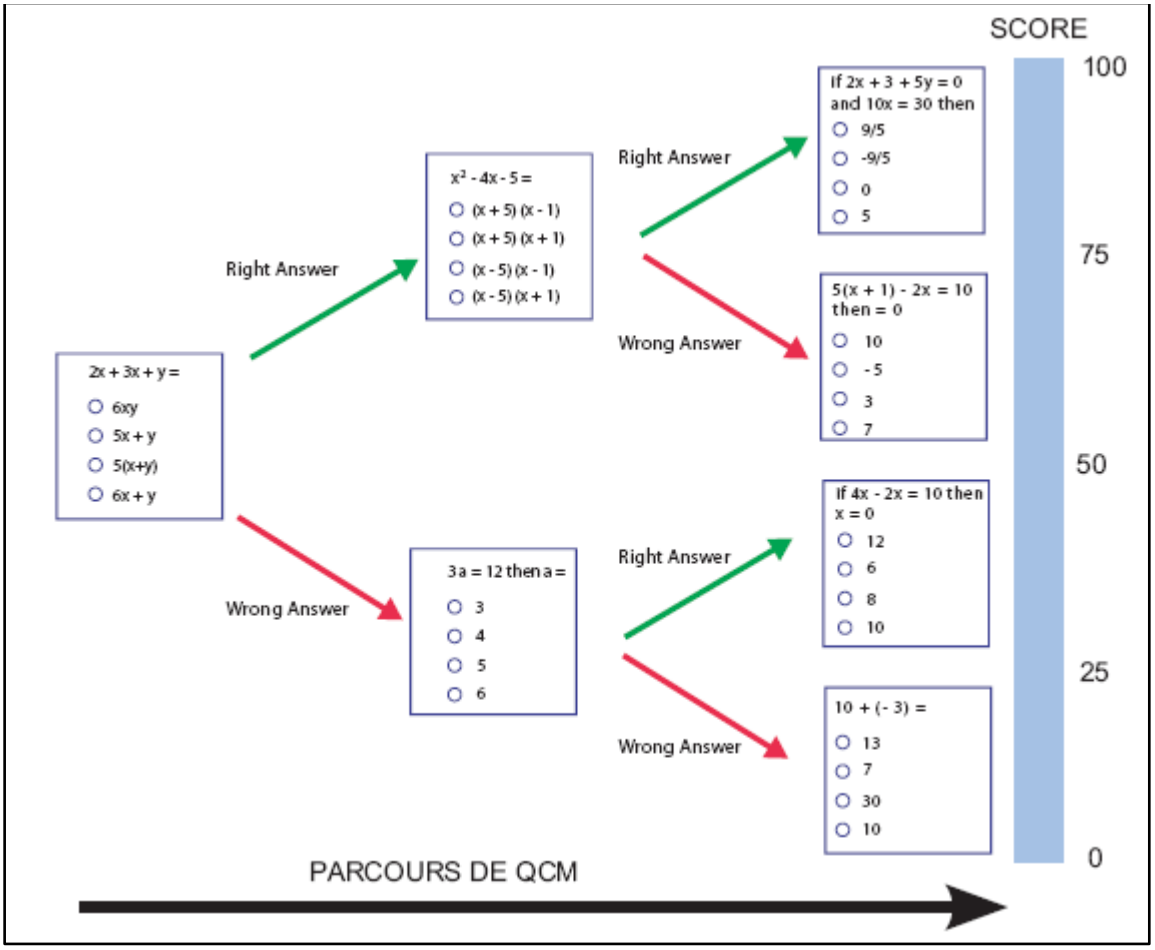


Figure N° 18: Principe

III.2. L'évaluation diagnostique

Contrairement aux évaluations sommatives et pronostiques des CAT, les systèmes d'évaluation diagnostique ne se contentent pas de corriger les productions des apprenants et de comptabiliser des points, mais proposent des diagnostics de compétences de l'apprenant. Chaque réponse de l'apprenant est alors 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 et Grugeon, 2004) et ses dérivés illustrent la mise en œuvre de cette évaluation diagnostique en EIAH. Pépite (Jean-Daubias, 2000), (Delozanne et Grugeon, 2004) est un logiciel qui a pour objectif d'aider les enseignants à diagnostiquer les compétences de leurs élèves en algèbre élémentaire. Pépite a été réalisé dans le cadre du projet pluridisciplinaire Lingot (Delozanne, B.Rugeon, Artigue et Rozalski, 2003), dont l'objectif est de concevoir des logiciels qui aident les enseignants à réguler les apprentissages en algèbre. Pépite est l'axe diagnostique du projet Lingot. C'est un outil logiciel composé de trois parties :

– PepiTest qui est destiné aux élèves. Il permet à ces derniers de résoudre une liste de 22 exercices constitués de questions ouvertes et fermées en algèbre ;

– PepiDiag qui est le module d'analyse des réponses. Il interprète les réponses des élèves à partir des analyses didactiques élaborées en amont du projet ;

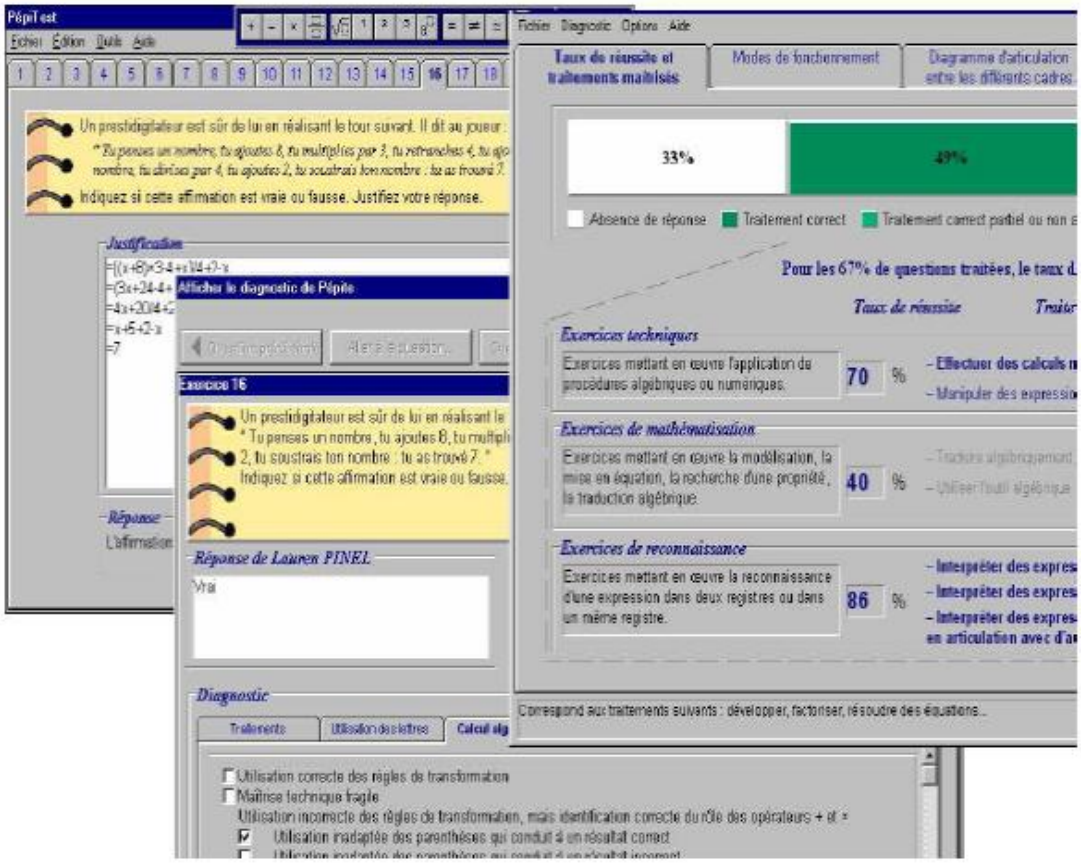


Figure N° 19: Pépite : de gauche à droite, les interfaces de PepiTest, PepiDiag et PepiProf

– PepiProf qui est l'outil enseignant. Il établit le profil de chaque élève et permet de modifier les réponses apportées par les élèves dans le cas où l'enseignant les trouverait contestables.

III.2.1. L'évaluation dans Pépite

Pépite permet d'évaluer les compétences des apprenants en algèbre. Cette évaluation sommative et individuelle de l'apprenant est automatique. Il fournit grâce à PepiDiag une synthèse en pourcentages d'acquisition des compétences mises en jeux par le questionnaire. L'enseignant obtient pour chaque apprenant un diagnostic composé de quatre parties :

- La partie "Taux de réussite et traitements maîtrisés" indique le taux de réussite par type d'exercice ;

- La partie "Modes de fonctionnement" détaille la manière de justifier les réponses et entre plus en détail dans la résolution des exercices. On s'attache aux compétences transversales de l'apprenant ;
- La partie "Diagramme d'articulation entre les différents cadres" représente graphiquement les modes de fonctionnement de l'apprenant ;
- La partie "Résumé du profil" propose un résumé textuel du diagnostic.
- Voici ci-dessous un extrait d'un résumé de profil PepiDiag.
- Utilisation des lettres pour faire du calcul algébrique avec des règles fausses dans 40% des réponses. La maîtrise du calcul algébrique reste insuffisante (maîtrise correcte dans seulement 45% des questions). Les difficultés sont liées à :
- Règles de transformation non maîtrisées, mais identification correcte du rôle des opérateurs + et x dans 20% des réponses (utilisation inadaptée des parenthèses qui conduit à un résultat correct, utilisation de règles de transformation fausses identifiées).
- Identification incorrecte du rôle des opérateurs + et x dans 5% des réponses (les règles de transformation utilisées "assemblent" les termes).

Conversion correcte dans 72% des réponses. Justification de type scolaire dans 50% des réponses.

Pépite permet aussi de réaliser un profil collectif des apprenants. Il est possible de visualiser le diagnostic "moyen" d'une classe.

Une première version de Pépite a été développée en Delphi en 2000, suivie en 2002 par une nouvelle version Java. Pépite a été très largement expérimentée. Cependant, les résultats de ces expérimentations sont assez mitigés. PepiDiag montre ainsi quelques inconsistances dans la grille d'analyse didactique lorsqu'elle est appliquée par le logiciel, c'est-à-dire d'une manière très rigide et sans discernement. En outre la grille d'évaluation présentée hors contexte induit de nombreuses difficultés : par exemple, certains enseignants ne comprennent pas les items du diagnostic (Jean-Daubias, 2000) [DG04].

III.3. L'évaluation sommative et diagnostique

Certains exerciseurs commerciaux tels que TDmaths intègrent les deux types d'évaluation, sommatifs et diagnostiques. TDmaths prend aussi en compte l'évaluation transversale évoquée plus haut (cf. II) : l'évaluation normative.

TDmaths est un exerciseur de mathématiques destiné aux élèves de niveau collège. Il équipe près de400 collèges, soit un minimum de 12000 utilisateurs. TDmaths est une application client-serveur java.

Les élèves progressent en autonomie, encouragés et guidés par leur enseignant. L'ensemble du programme d'algèbre du collège est présenté sous forme de cours synthétiques, illustrés par des exemples et structurés en chapitres. Les liens entre les chapitres dessinent la carte des mathématiques. Cette carte présente les différents parcours possibles dans le programme, y compris les passages entre les niveaux (6ème, 5ème, 4ème, 3ème).

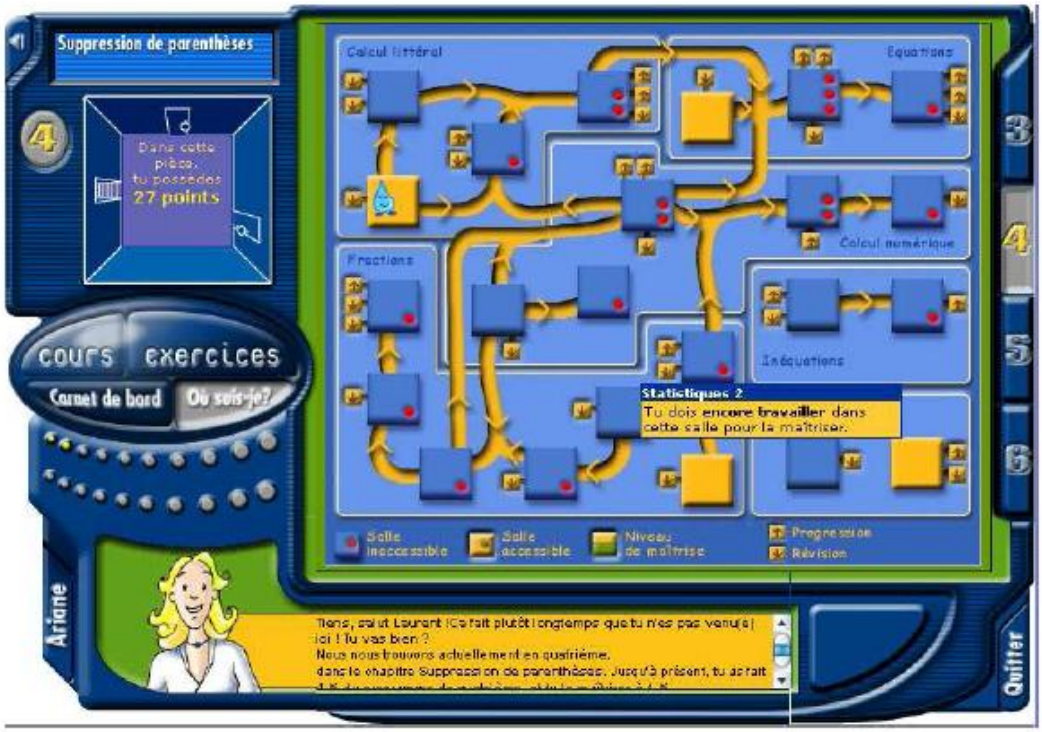


Figure N° 20: TDmaths : parcours de l'apprenant

En suivant ces liens, l'élève progresse au fur et à mesure de sa maîtrise des notions, validée par ses succès aux exercices. Les exercices, de difficulté croissante, ont des énoncés aléatoires. TDmaths offre ainsi plus de 600 exercices, une grande variété de formes (QCM, saisie alphanumérique, calcul mental chronométré, représentation graphique) et une infinité

d'énoncés. A chaque nouvelle connexion, l'élève peut reprendre son travail là où il l'a laissé. Les caractéristiques de suivi du travail de l'élève par l'enseignant sont les suivants :

- évaluation permanente des élèves conforme aux objectifs des évaluations nationales;
- analyse précise des connaissances de chaque élève : notions travaillées, maîtrise de ces notions;
- historique des sessions de travail et des résultats obtenus.

III.3.1. L'évaluation dans TDmaths

Dans TDmaths, trois types d'évaluation sont intégrés :

– Une évaluation sommative individuelle des performances sanctionnée par des scores obtenus dans les exercices de mathématiques ;

– Une évaluation diagnostique des apprenants. Les compétences transversales des apprenants sont évaluées et un diagnostic graphique est ensuite proposé à l'enseignant. Il peut alors envisager des remédiations à partir de cette évaluation. Dans ce cas, l'évaluation s'inscrit alors dans un cadre formatif ;

– Une évaluation normative des apprenants. Que ce soit au niveau des scores ou des compétences transversales, le logiciel propose de positionner les résultats de ces évaluations par rapport à la classe.

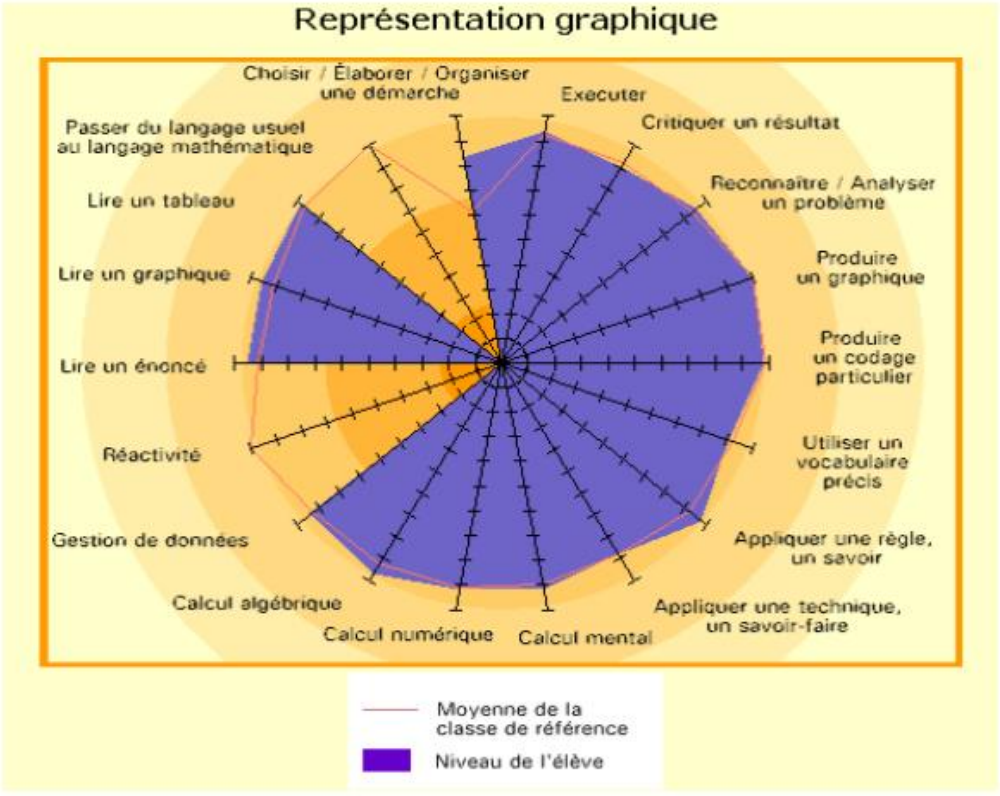


Figure N° 21: TDmaths : interface enseignant, carte des compétences transversales

Les compétences transversales évaluées dans TDmaths sont des savoir-faire tels que "lire un énoncé", "critiquer un résultat", etc.

Dans le sens où TDmaths est un outil commercial largement diffusé, il est représentatif des pratiques en EIAH dans les collèges français. Du point de vue de l'évaluation, TDmaths est dual. D'une part, il offre une évaluation de type "testing" partagée par le plus grand nombre d'EIAH, d'autre part il introduit l'évaluation diagnostique des compétences. Celle-ci est plus souvent rencontrée dans des outils de recherche tels que Pépite (Delozanne et Grugeon, 2004). Il confirme donc l'attente des enseignants d'une évaluation fine des compétences des apprenants.

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 laquelle 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.

GenEval (David, 2003) est un logiciel qui permet de générer des exercices d'auto-évaluation. Ce logiciel a été développé par le Centre d'Auto-Formation et d'Innovation Multimédia (CAFIM) dans le cadre du projet européen ARIADNE (Alliance of RemoteInstructionalAuthoring and Distribution Networks for Europe).

Le projet européen ARIADNE regroupe une quinzaine d'universités européennes et a pour objectifs de favoriser et mutualiser la production de documents pédagogiques hypermédias et d'expérimenter des scénarios pédagogiques avec des étudiants en situation d'apprentissage autonome. Le CAFIM peut se définir à la fois comme une unité de réflexion sur les usages pédagogiques de l'ordinateur et comme un centre de développement des outils et des ressources nécessaires à la mise en place d'enseignements médiatisés.

Un exercice d'auto-évaluation créé avec GenEval offre aux étudiants un cadre pour résoudre un exercice, avec des accès hypermédias à des informations pertinentes pour la compréhension des concepts étudiés, avec des orientations et des aides méthodologiques progressives. GenEval propose à l'apprenant une suite de questions auxquelles il doit répondre. Il dispose dans cet exercice d'indices qu'il peut ou non utiliser pour répondre aux

questions. Il répond à toutes les questions puis évalue sa maîtrise des compétences mises en jeux dans le questionnaire.

III.3.2. L'évaluation dans GenEval

Dans GenEval, deux types d'évaluation sont menés conjointement :

- Une évaluation individuelle sommative. Lorsque l'apprenant répond à une question, le système GenEval donne une correction ;
- Une auto-évaluation. L'apprenant évalue sa maîtrise des compétences. Cette évaluation favorise l'auto-régulation de ses apprentissages en s'insérant dans le cadre d'une évaluation formative (Schunk, 1990).

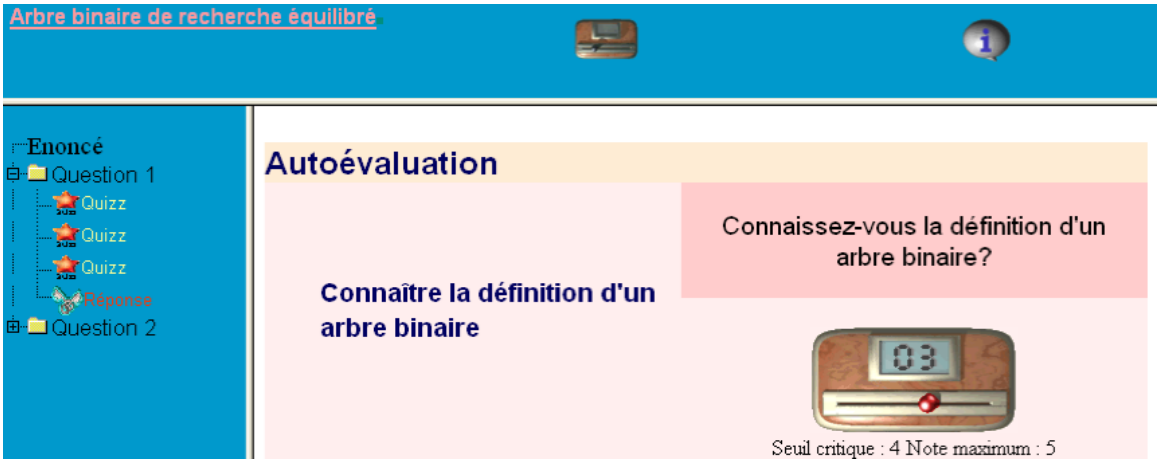


Figure N° 22: GenEval : phase d'auto-évaluation de l'apprenant

Contrairement aux autres pratiques d'évaluation, l'auto-évaluation est peu représentée. Pourtant, comme le soulignent certains psychologues (Schunk, 1990), cette évaluation s'intègre idéalement dans une logique formative de l'évaluation, en responsabilisant l'apprenant dans sa construction du savoir. C'est donc une pratique d'évaluation légitime et intéressante dans cette optique.

III.3.3. L'évaluation par pairs

Une des premières pratiques d'évaluation des activités collaboratives en EIAH est l'auto-évaluation par pairs (Maor, 1998). Cette évaluation peut être :

– Individuelle. Un apprenant évalue un autre apprenant (Bhalerao et Ward, 2001).

– Collective. Un groupe d'apprenants évalue un apprenant ou un groupe d'apprenants (Ward, Sitthiworachart et Joy, 2004), (juwah, 2003).

Dans le cas où l'évaluation est menée par un groupe d'apprenants et porte sur une production réalisée par un autre groupe, une des difficultés peut provenir de la difficulté pour

les membres du groupe qui évaluent à se mettre d'accord sur les résultats de l'évaluation [Juw03].

Le développement d'OASYS (Bhalerao et Ward, 2001) (On-line Assessment System) a débuté en 2000 et s'est terminé en 2002 avec une nouvelle version OASYS2 (Ward, Sitthiworachart et Joy, 2004).Cet outil a été conçu par des chercheurs de l'Université de Warwick à Coventry en Angleterre. OASYS est un outil WEB d'évaluation par pairs. Il permet de créer et diffuser auprès d'apprenants des questionnaires. Les questionnaires sont composés de questions à choix multiples et de questions ouvertes.

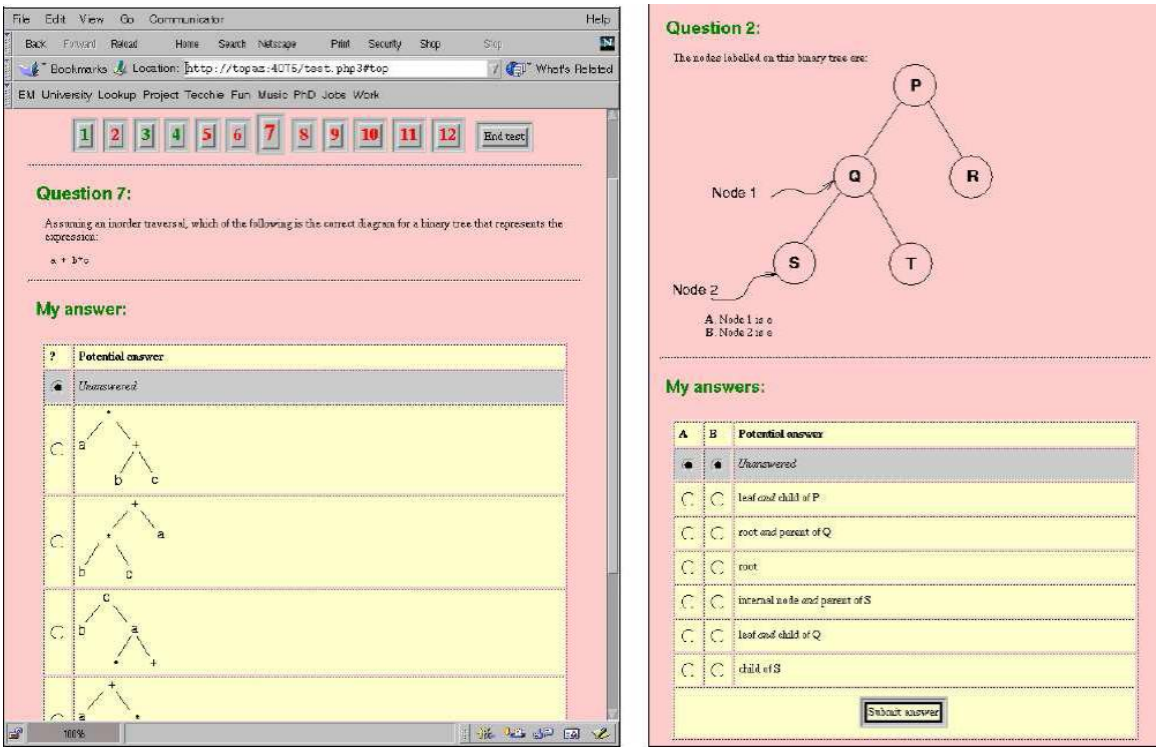


Figure N° 23: OASYS : exemple de question fermée

Les concepteurs ont choisi l'évaluation par pairs pour évaluer les réponses aux questions ouvertes dont l'évaluation est difficilement automatisable. OASYS a été expérimenté en 2002 dans le cadre de cours en programmation. Dans cette expérimentation, la possibilité d'utiliser des questions ouvertes était indispensable pour permettre aux apprenants d'écrire des programmes informatiques.

III.3.4. L'évaluation dans OASYS

Dans OASYS, chaque apprenant répond tout d'abord aux questions qui lui sont posées avant de devenir évaluateur. L'apprenant doit ensuite évaluer trois autres apprenants dont les copies lui sont automatiquement attribuées. Pour l'évaluation de chaque question, l'enseignant a préalablement défini les critères d'évaluation, c'est-à-dire les éléments sur lesquels l'apprenant va devoir se focaliser pour évaluer la réponse. Pour évaluer, l'apprenant dispose d'un questionnaire qui apprécie le respect des critères sélectionnés par l'enseignant. Dans le processus d'évaluation, l'enseignant joue le rôle de modérateur. Il est prévenu si un écart entre les différentes évaluations d'une même copie est important.

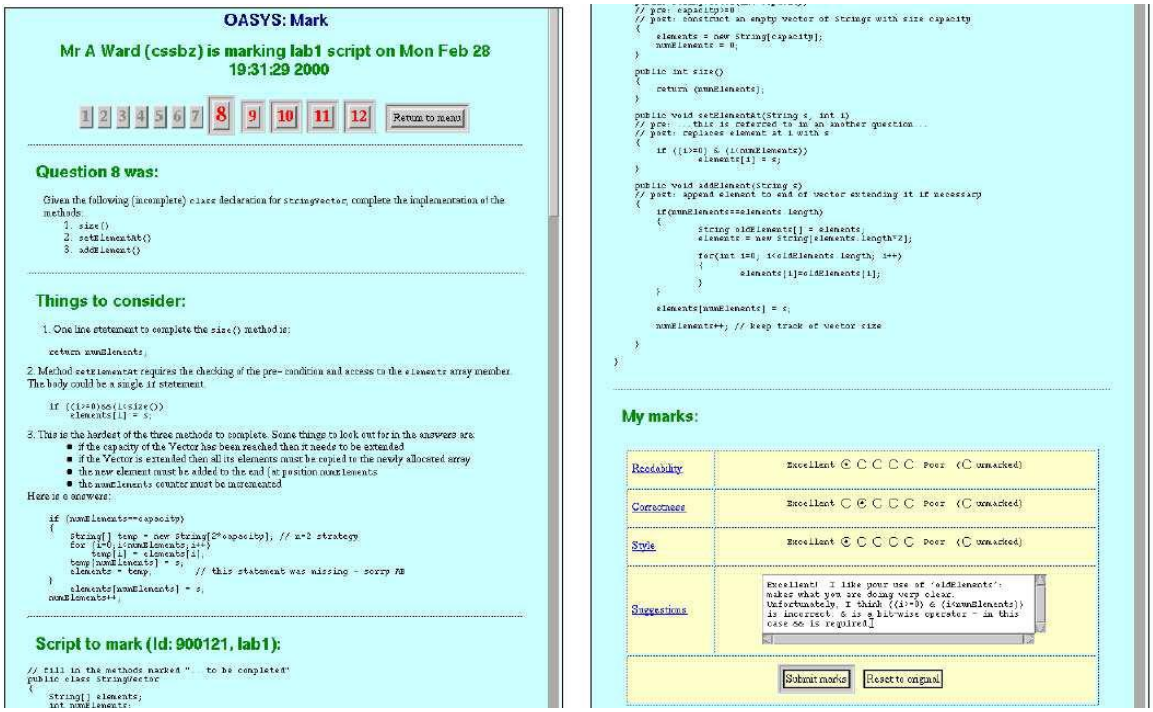


Figure N° 24:OASYS : interface de correction

La version OASYS2 intègre de nouvelles fonctionnalités. Dans la phase d'évaluation, les apprenants peuvent être groupés. Ils évaluent ensemble les réponses d'autres groupes en échangeant leurs points de vue dans un chat. A la fin du processus d'évaluation, chaque apprenant reçoit son résultat.

OASYS est un système évolué d'évaluation par pairs. Il apporte une solution originale à l'évaluation des questions ouvertes. En outre, le fait d'avoir été utilisé par les concepteurs avec leurs élèves a permis l'émergence de la version 2 qui intègre une vision collective de

l'évaluation par pairs. Les résultats issus des expérimentations montrent aussi que l'enseignant est assez rarement sollicité dans son rôle de modérateur.

Cela témoigne d'une bonne conception de ce système d'évaluation et d'une réelle autonomie des apprenants.

III.3.5. 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, qu'il s'agisse d'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 et Yacef, 2004) qui permettent à l'enseignant d'évaluer le déroulement de l'activité. Dans ce contexte, plus que des outils de mesure (Mazza et Milani, 2005), il existe des systèmes qui proposent de superviser l'activité.

Dans la configuration de la classe, l'enseignant a la possibilité d'observer les réactions de ses élèves et ainsi d'adapter son cours. GISMO (Graphical Interactive Student Monitoring System) (Mazza et Milani, 2005) permet à l'enseignant de maîtriser ce qui se déroule durant l'activité pédagogique, et ce par le biais d'interfaces de visualisation d'indicateurs (Mazza, 1998). Il répond à la question du "retour pour l'enseignant" (Labat, 2002) en lui fournissant des visualisations d'indicateurs. C'est un outil dit d'"awareness".

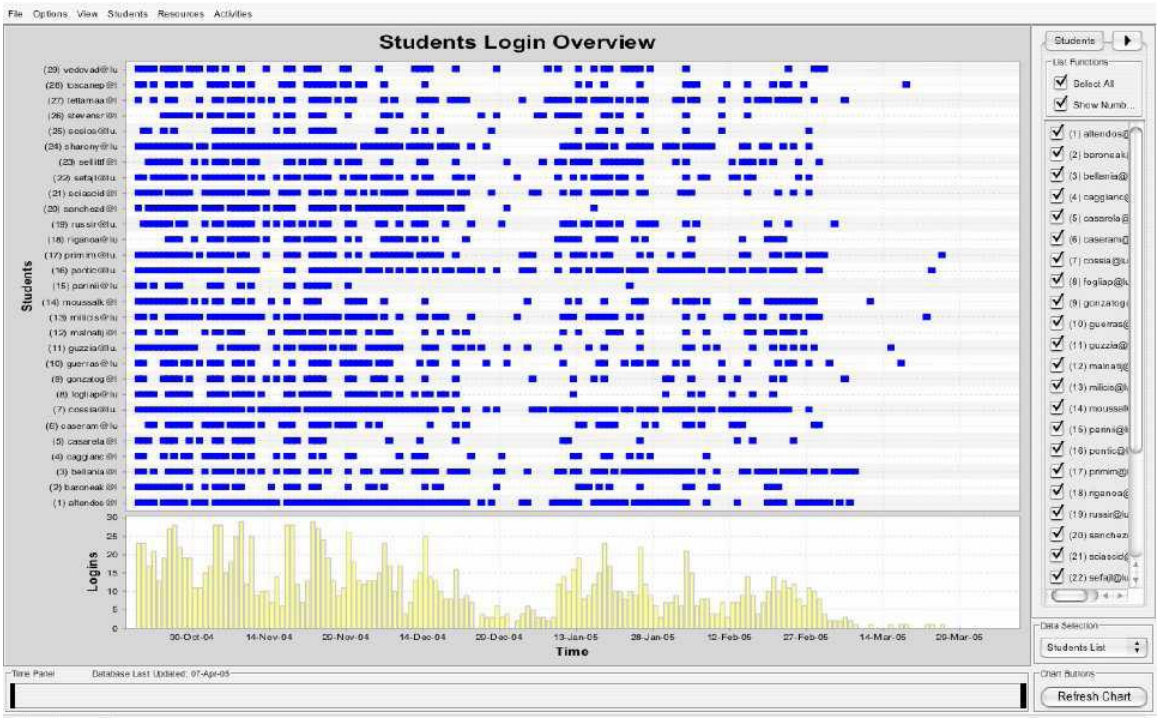


Figure N° 25: GISMO: visualisation des connexions des apprenants au système

Le système de gestion d'apprentissage utilisé par GISMO est Moodle. Moodle est un Espace Numérique de Travail (ENT) particulier. C'est un Système de Gestion de l'Apprentissage (SGA). Cette sous-catégorie très spécialisée d'ENT permet de créer et de jouer des questionnaires et des cours. Moodle dispose d'outils de communication et d'espaces partagés. GISMO permet de visualiser les connexions à Moodle, les accès aux ressources contenues dans Moodle, les contributions dans le forum (création d'un sujet de discussion, postage d'uncommentaire, etc.)et l'état des exercices proposés aux apprenants (en cours, terminé, etc.).

III.3.6. L'évaluation dans GISMO

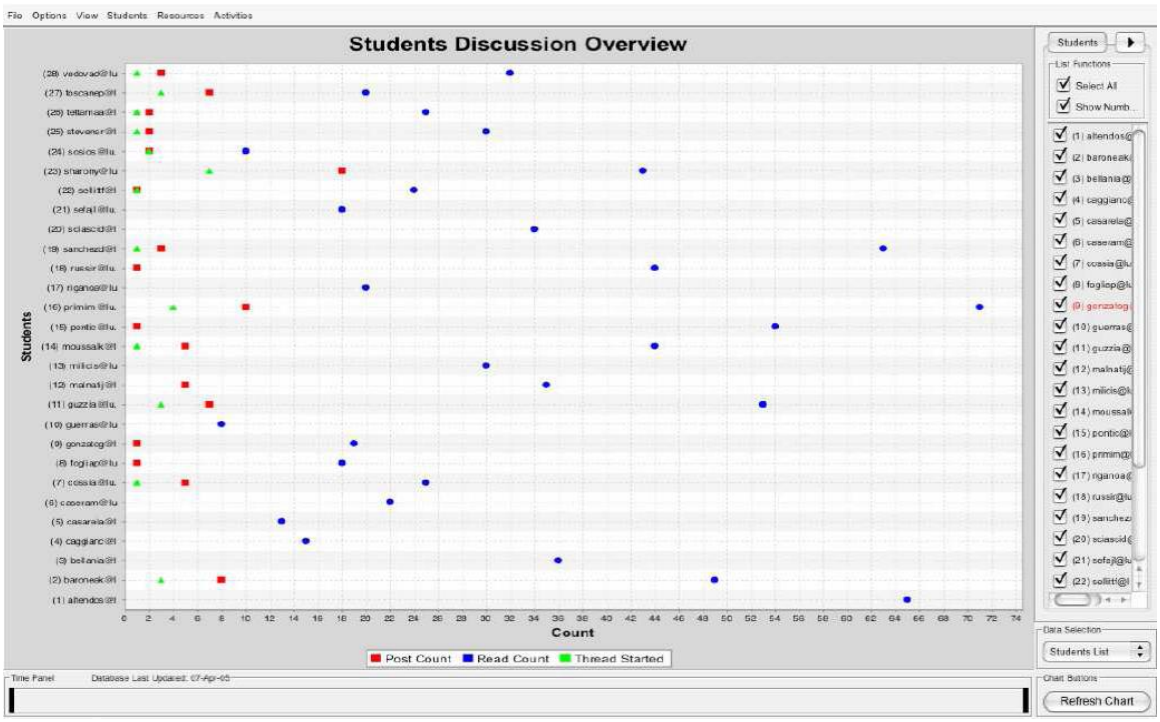


Figure N° 26: GISMO : visualisation des interventions des apprenants dans le forum

GISMO fournit à l'enseignant des indicateurs qui l'aident dans sa tâche d'évaluation et de régulation du travail des apprenants dans Moodle. Mis à part les tests qui sont corrigés de manière automatique dans Moodle, l'évaluation formative des apprenants est à la charge de l'enseignant. A partir des observations fournies par GISMO, l'enseignant peut interagir et guider les apprenants. GISMO est un outil de visualisation de l'activité à travers des traces prédéfinies par les concepteurs du logiciel. Il n'est pas possible pour l'enseignant de visualiser d'autres traces. C'est pourquoi GISMO offre une vision orientée de l'activité même si les concepteurs ont défini les observables avec des enseignants. Cependant, GISMO atténue

l'effet "boîte noire" des EIAH et est un premier pas certain pour aider l'enseignant à superviser une activité.

III.3.7. L'évaluation de la participation

Bien que commune à toute activité collective, l'évaluation de la participation se retrouve quasi essentiellement dans le cadre d'activités utilisant des outils de communication. Deux cas de figures sont rencontrés :

– L'évaluation porte sur la qualité des discussions grâce à des techniques de "textmining" qui analysent le rapport entre mots clés (ceux attendus par l'enseignant) et nombre d'échanges. C'est le cas par exemple du logiciel Ibee (S.Fujitania, T.Mochiduki, Y.Isshiki et al, 2003), qui assiste l'enseignant pour gérer l'activité des apprenants dans un forum. En évaluant la qualité du discours des participants, le logiciel aide à l'évaluation de la maîtrise de compétences attendues par l'enseignant.

– L'évaluation est menée au sein d'un outil de communication, par l'analyse des actes de langage des apprenants. C'est le cas dans SPLACH (George, 2004), où à chaque communication, le participant doit au préalable pré-sélectionner un acte de langage (Question, Réponse, etc) parmi la liste proposée. Le système est alors capable d'évaluer le profil de l'apprenant, c'est-à-dire s'il est actif, en retrait etc.

Dans ce cas la qualité du discours n'est pas évaluée. On ne cherche pas à vérifier des compétences mais l'engagement (Durand et Vignollet, 2003) des participants dans l'activité. SPLACH (Support d'une pédagogie de Projet pour l'Apprentissage Collectif) est un environnement informatique support d'une pédagogie par projet permettant l'apprentissage collectif à distance (George, 2001). Il fait partie des Environnements Interactifs d'Apprentissage à Distance (EIAD). SPLACH alterne des phases de travail synchrones et asynchrones. Deux prototypes ont vu le jour : le premier a permis à des collégiens d'établissements différents de réaliser la construction de robots. Les collégiens étaient en concurrence pour fournir le meilleur robot possible. Le second permettait à des étudiants de la Télé université du Québec (Téluq) de développer un programme informatique en commun.

SPLACH dispose de différents outils de communication : forum, webmail, espace de réunion. L'espace de réunion est un chat amélioré, fonctionnant par tour de parole et requérant des actes de langage (répondre, demander,..) pour obtenir une conversation structurée. Les apprenants rédigent différents documents et questionnaires qu'ils sauvegardent dans un espace "Documents", qui est partagé en "documents de l'équipe", "documents personnels" et "documents des équipiers". Suivant les prototypes, différents outils ont été

ajoutés. Dans le premier prototype, on trouve un outil de description de robot, un outil de programmation et une interface de pilotage de robot. Dans le second, un outil permettant de programmer en Pascal dans l'environnement SPLACH a été développé.

III.3.8. L'évaluation dans SPLACH

L'évaluation des apprenants dans SPLACH est formative. L'enseignant supervise l'activité et agit sur son déroulement. Il a ainsi la possibilité de vérifier la participation au projet et d'évaluer les productions des apprenants. Le point fort de SPLACH est son outil d'évaluation des profils des apprenants basé sur les conversations dans un chat structuré. Chaque apprenant intervient dans le chat en utilisant un acte de langage. A partir de la théorie sur les actes de langage de Pléty (1996), SPLACH (George, 2004) permet de profiler l'apprenant comme animateur, vérificateur, quêteur ou indépendant de l'activité de groupe.

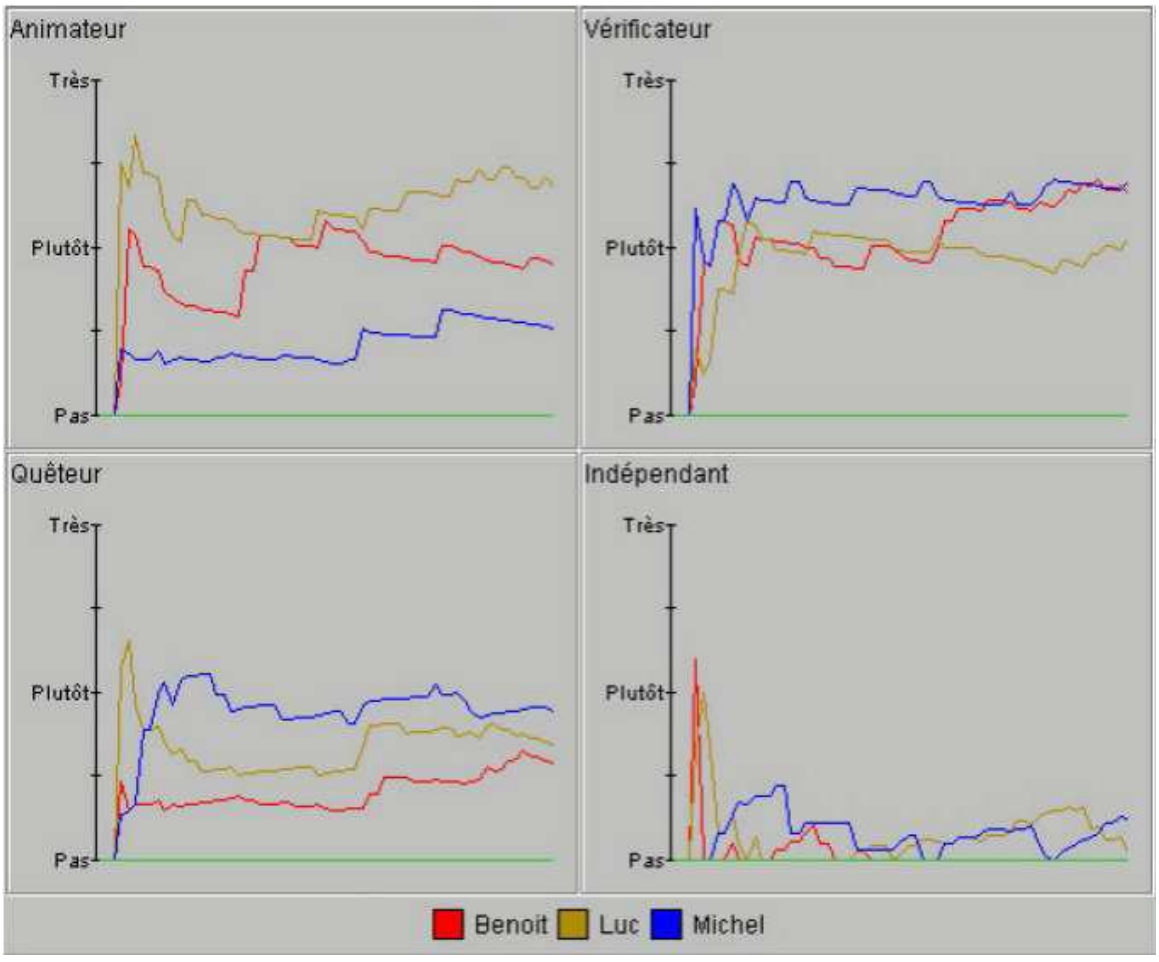


Figure N° 27: SPLACH : outil de visualisation des profils d'apprenant

L'intérêt de SPLACH réside dans l'assistance à l'évaluation, qu'il propose en permettant à l'enseignant de superviser le déroulement du projet mais aussi en permettant de connaître le rôle social joué par chaque apprenant.

III.3.8.1. Le portfolio

Les systèmes de portfolio sont très répandus en Amérique du Nord. Un groupe de travail de la commission scolaire des Premières-Seigneuries au Québec définit le portfolio comme étant un outil d'évaluation des apprentissages qui permet de recueillir et de conserver des échantillons des réalisations de l'élève. Il s'inscrit dans une démarche d'évaluation formative continue et est réalisé en collaboration avec l'élève.

Nous nous intéressons ici au portfolio numérique de l'apprenant, système logiciel permettant le stockage et l'annotation de ressources numériques issues des réalisations de l'apprenant. Charge est à l'apprenant de choisir les réalisations qu'il ajoute à son portfolio, ce qui l'aide à s'auto-évaluer, observer ses progrès, s'impliquer dans une réelle démarche de réflexion métacognitive.

Selon l'usage qui en est fait, on distingue trois types de portfolio de l'apprenant selon (Eyssautier-Bavay, 2004), (Ministère de l'Education du Québec, 2002) :

– le portfolio d'apprentissage, qui centralise les travaux de l'apprenant et ses commentaires sur ses propres travaux ;

– Le portfolio de présentation, qui répertorie les meilleurs travaux de l'apprenant. C'est l'équivalent du "book" de l'artiste ;

– Le portfolio d'évaluation, qui permet d'évaluer la progression de l'apprenant dans ses apprentissages.

Il contient des productions, des résultats d'évaluation, les observations de l'apprenant et parfois ses auto-évaluations.

Ces types de portfolio ne sont pas exclusifs les uns par rapport aux autres. En général, un portfolio numérique peut engendrer ces trois stéréotypes. C'est l'enseignant qui choisit l'usage du portfolio. Le développement du portfolio APOM a débuté en 1998 par un groupe de travail de la Coopération régionale de développement pédagogique (CDR) de Montérégie (Québec). Les concepteurs ont voulu développer un portfolio numérique en ligne, simple, convivial et ne nécessitant pas de compétences techniques.

En 2003, APOM est devenu un service du système de gestion de l'apprentissage COLLABA et en 2005 d1'EDU-GROUPE. Dans APOM, l'apprenant gère des fiches de

1COLLABA, http ://www.collaba.ca/fr/

travail (voir figure ci-dessous) dans lesquelles il indique une description du travail réalisé, les compétences mises en jeu, ses remarques et observations ainsi que la date et la discipline. Il est possible d'associer aux fiches un document numérique (pdf, text). L'enseignant peut consulter les fiches et les modifier. L'apprenant peut décider de partager ses fiches avec d'autres apprenants.

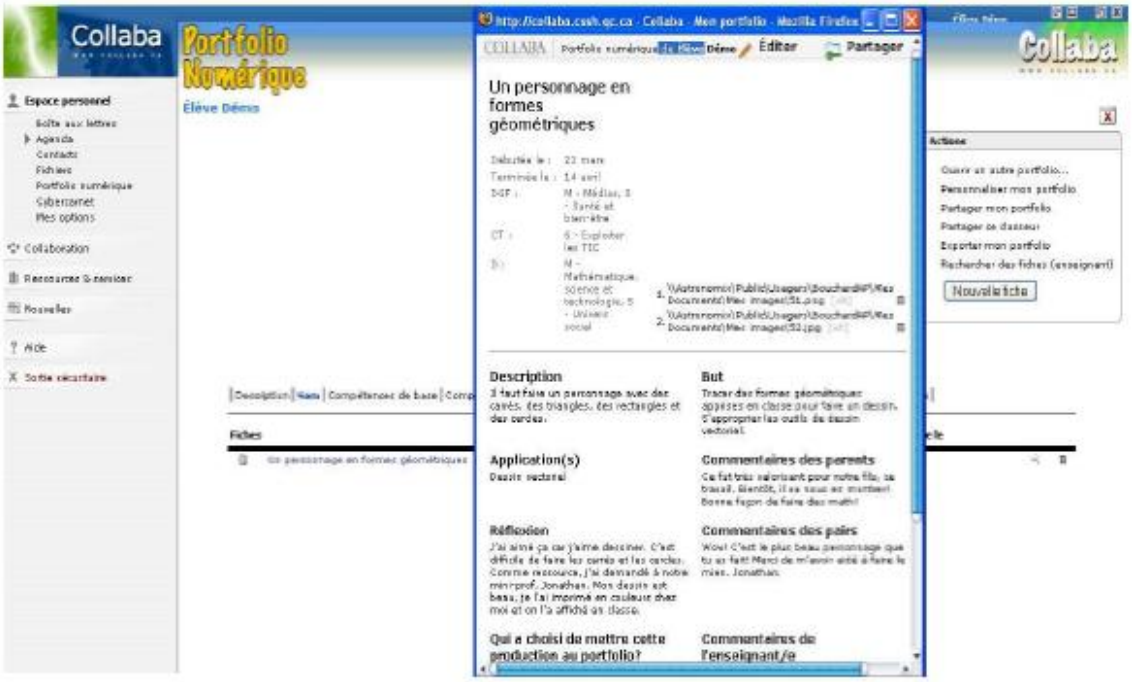


Figure N° 28: APOM: interface élève

III.3.8.2 L'évaluation dans APOM

L'évaluation dans un portfolio peut être de nature différente selon les désirs de l'enseignant (Ministère de l'Education du Québe, 2002). Dans le portfolio d'apprentissage et de présentation, une large place est laissée à l'apprenant qui autoévalue ses productions en les choisissant et en les commentant. En revanche, dans le cas du portfolio d'évaluation, l'évaluation principale est celle de l'enseignant qui évalue les compétences de l'apprenant grâce aux documents qu'il a ajoutés. Un portfolio d'évaluation peut être utilisé, par exemple, pour l'obtention d'un diplôme.

Les systèmes de portfolio très répandus en Amérique du Nord, offrent un contexte d'apprentissage riche. Avec ces outils, les situations d'apprentissage envisageables sont diverses. Du point de vue de l'évaluation, le portfolio permet de mener des évaluations formatrices dans lesquelles l'apprenant prend conscience de ses progrès. Cela lui permet de mieux s'impliquer dans ses apprentissages.

Mais ces types sont rarement présents simultanément dans un même EIAH. Or, un enseignant peut avoir besoin, sur une même activité pédagogique, de prévoir et de mettre en place des évaluations de différents types. S'il veut, par exemple, constituer des groupes de lecture en les fondant sur la capacité de l'élève à lire un texte, il aura besoin d'introduire une méthode d'évaluation de type diagnostique. Si sur la lecture proposée de ce même texte, il s'aperçoit que les élèves ont du mal à comprendre une notion, il devra introduire une méthode d'évaluation de type formatif. Si enfin il veut s'assurer que le sujet dont traite le texte a été compris et qu'il n'a pas l'intention d'y revenir, il devra prévoir une évaluation de type sommative.

Les évaluations ne sont pas utiles uniquement pour mesurer les progrès réalisés par les élèves. Elles peuvent aussi servir à faciliter ces progrès à condition de pouvoir être utilisées par les enseignants comme un outil pédagogique à part entière, ce qui implique qu'ils puissent les organiser à leur guise.

Pour que les enseignants aient confiance dans les évaluations qui leurs sont proposées, il faut qu'ils comprennent comment elles sont bâties, sur quels critères elles s'appuient, par quels moyens les résultats qu'elles proposent sont obtenus, quels sont leurs référents, leurs référés et les relations qui existent entre les deux, pour employer la terminologie utilisée en Sciences de l'Éducation.

Les élèves, eux aussi, ont le droit de connaître les modalités d'une évaluation. Par exemple, si la grammaire et la forme générale d'un devoir écrit revêtent de l'importance pour les objectifs que l'enseignant s'est assigné, les élèves doivent être informés de ces critères lorsque le travail à faire leur est donné. Si leur degré d'autonomie est un élément qui compte dans l'évaluation de leurs travaux, il importe qu'ils le sachent. Si l'enseignant a l'intention d'observer ses élèves au cours de l'activité, il vaut mieux le leur indiquer.

D'une manière générale, le "scénario" prévu pour l'évaluation doit être compréhensible et communicable.

2.5 Limites des évaluations dans les EIAHs

La plupart des produits EIAHs souffrent toujours de quelques déficits. Voici d'une manière succincte quelques limites :

2.5.1 Limites par rapport à l'outil d'évaluation

- ***Unicité du modèle d'évaluation***

Chaque EIAH utilise au moins un modèle d'évaluation qui est généralement en adéquation avec son utilité pédagogique. Ainsi, il existe des EIAH qui utilisent la co-évaluation dans les activités de groupe, de l'auto-évaluation pour stimuler l'implication des apprenants, de l'évaluation sommative pour certifier des compétences, etc. (Campanale, 2001). Mais dans tous les cas de figure, l'enseignant est contraint par l'EIAH dans le choix du modèle d'évaluation. Les EIAH ne permettent pas de changer de modèle d'évaluation. On peut dès lors se demander si le modèle d'évaluation peut être au moins paramétré.

- ***Fixité de la méthode d'évaluation***

Le modèle d'évaluation utilise une méthode. Mais, là encore cette méthode est statique. Une critique récurrente faite aux exerciseurs de mathématiques est bien souvent de mal-corriger. De circulaires en réformes, ce qui est juste à une époque ne l'est plus toujours à une autre. Il est alors possible qu'un enseignant considère la méthode de correction fausse sans pouvoir la modifier. Toujours dans la méthode d'évaluation, le barème est aussi problématique. L'enseignant n'est pas libre de choisir les caractéristiques qu'il souhaite évaluer et encore moins de leur associer des coefficients.

- ***Résultats d'évaluation peu compréhensibles***

En outre la méthode d'évaluation de l'EIAH, utilise bien souvent un algorithme de calcul de score. Dans TDmaths le score final est pondéré par la difficulté des questionnaires ainsi que par le nombre de tentatives. Dans Pépite (Delozanne et Grugeon 2004), l'enseignant dispose d'un bilan très complet des compétences en algèbre de l'apprenant sans pour autant connaître la manière dont ce bilan est obtenu. Lors de la publication des résultats, l'EIAH n'exprime pas cet algorithme. Il est alors difficile de donner du sens à un score lorsque sa méthode de calcul n'est pas explicite.

- ***Interopérabilité des EIAH***

Enfin, l'EIAH comme entité logicielle autonome et très spécialisée se transforme de plus en plus en une simple fonctionnalité d'un Espace Numérique de Travail (ENT) ou d'un Système de Gestion de l'Apprentissage (SGA). Dans ces dispositifs, de nouveaux acteurs apparaissent tels que les parents mais aussi les services administratifs. Les résultats

d'évaluation produits par l'EIAH doivent donc être utilisables par d'autres services. Par exemple, les scores obtenus en mathématiques sont nécessaires à l'enseignant mais aussi au service de scolarité. Du point de vue de l'évaluation, ces services doivent être interopérables. Cette interopérabilité passe par un format de résultat commun qui n'existe pas aujourd'hui.

2.5.2 Limites par rapport à la connaissance évaluée

En pédagogie, on distingue habituellement savoirs déclaratifs (savoir) et savoirs procéduraux (savoir-faire), bien que cette distinction soit parfois controversée (Bentolila, 1995). En termes de connaissance, on parle de connaissances déclaratives et de connaissances procédurales. Les connaissances déclaratives sont beaucoup plus faciles à apprendre que les connaissances procédurales, car apprendre à un apprenant à réfléchir, à analyser et résoudre des problèmes, est plus délicat qu'à lui apprendre à appliquer une règle bien définie.

Évaluer des savoirs faire est une tâche délicate. Le savoir-faire est une mise en œuvre d'un savoir et d'une habileté pratique maîtrisée dans une réalisation spécifique. Évaluer le savoir-faire d'un apprenant, c'est évaluer sa créativité…imaginons un peu la difficulté. Si on rajoute à cette difficulté, les limites de l'évaluation dans un EIAH (déjà citées), on se rend compte de la complexité à évaluer des connaissances procédurales dans un EIAH.

IV. Conclusion

L'évaluation automatisée recouvre un large éventail de techniques. Elle se base sur des outils assez rudimentaires comme les questionnaires à choix multiples (QCM) ou les textes à trous aussi bien que sur des techniques plus avancées comme les programmes de test de codes ou d'évaluation automatisée de copies. On peut distinguer les exercices d'application du cours des tests de remémoration de contenu. Ces derniers ont pour fonction de s'assurer qu'une information a été mémorisée. Dans les MOOC, on les retrouve fréquemment au sein même des vidéos de cours, les séquences de cours magistraux de quelques minutes alternant avec des séquences de test.

Les exercices d'application ont une fonction tout autre. Leur difficulté n'est pas nécessairement liée à la simplicité apparente du type de test. Un simple QCM à quatre solutions peut nécessiter des heures de calculs. Parmi les exercices d'application, les applications numériques sont particulièrement utilisées dans les cours de sciences fondamentales (physique, statistiques, etc.). La réponse donnée par le participant n'est considérée comme correcte que si elle se trouve dans une gamme de valeurs fixée par l'équipe

pédagogique. L'inconvénient de cette approche est qu'elle permet de considérer comme correct un résultat juste même si le raisonnement suivi pour l'obtenir est faux, et un résultat comme faux même si le raisonnement suivi est le bon. Ce sont les limites de l'évaluation automatisée.

L'évaluation par des programmes de test est une forme d'évaluation automatisée utilisée dans les cours impliquant la rédaction de programmes informatiques. Les programmes de test permettent d'analyser de manière automatique le contenu d'un code, d'en détecter les erreurs et de faire des retours éventuels à son concepteur. Ce type de programme est bien antérieur à l'avènement des MOOC ; les plates-formes comme edX ou Coursera n'ont fait qu'intégrer ce concept. Les enseignants sont donc libres de rédiger leurs propres programmes de test ou d'utiliser des programmes existants, selon leurs besoins.

Pour terminer sur la question de l'évaluation automatisée, nous aimerions souligner que ces techniques connaissent un essor rapide au sein des plateformes, qui inclue désormais de petits laboratoires virtuels. Les exercices basés sur l'apprentissage par essai et erreur vont sans doute se multiplier dans les années à venir.

Nous nous sommes placés jusqu'à présent dans le cadre de l'évaluation de productions individuelles, les productions collectives peuvent également être notées via l'évaluation par les pairs. Et ce pour l'évaluation des projets eux-mêmes, qui rencontre les mêmes problématiques que l'évaluation des productions individuelles, aussi bien que pour l'évaluation des différents membres de l'équipe au sein du projet.

Nous n'avons eu, dans ce qui précède, qu'un bref aperçu des différentes méthodes d'évaluation automatique, sans être exhaustifs, et sans nous préoccuper outre mesure des contextes dans lesquels ces techniques sont appliquées. Celles-ci peuvent être combinées entre elles, scénarisées d'une infinité de façons ; une thèse ne suffirait pas à faire le tour de la diversité de leurs applications. Imaginez par exemple que l'on puisse, en fonction des réponses que vous avez données à tel ou tel exercice, sélectionner en temps réel un test qui soit parfaitement adapté à votre profil, vos connaissances et vos objectifs d'apprentissage. Tests adaptatifs, évaluation en deux étapes, il y a sans doute beaucoup à écrire encore sur la question …

Chapitre III :

Evaluation des organigrammes des apprenants en algorithmique

I. Introduction

Après avoir vu les limites des environnements d'apprentissage de l'algorithmique existants et leurs limites, nous proposons un système d'évaluation de l'algorithmique qui peut, très facilement, être adapté à l'évaluation de n'importe quel savoir-faire. Ce système est basé (inspiré de) sur la décomposition. Il garantit, en plus, l'évaluation de n'importe quelle solution de l'apprenant de manière directe ou différée.

Ce système repose sur la comparaison et l'appariement. Inspiré des concepts et des techniques d'appariement de modèles, il se concentre principalement sur les aspects structurels des solutions à apparier. A la fin, un prototype de ce système est présenté. Ce prototype va servir à la validation de notre approche.

II. Apprentissage par la pratique

La littérature met en lumière plusieurs stratégies d'apprentissage dont les plus analysées sont celles qui ont trait à l'apprentissage par la recherche (*learning by searching*), à l'apprentissage par la pratique (*learning by doing*), à l'apprentissage par l'utilisation des technologies de pointe (*learning by using*), à l'apprentissage par l'interaction (*learning by interacting*), à l'apprentissage par les externalités industrielles (*learningfromindustryspillovers*) et à l'apprentissage par les externalitésrégionales (*learningregion*) (Cohen 1990).

Dans le contexte de l'apprentissage de l'algorithmique et de la programmation, une matière qui requiert beaucoup de compétences, on trouve l'apprentissage par la pratique comme la forme la plus adéquate pour apprendre ce genre de compétence. Aristote a dit: « Il faut apprendre en faisant la chose, car si vous pensez que vous savez, vous n'avez pas la certitude jusqu'à ce que vous ayez essayé. »

Cet apprentissage par la pratique est le meilleur moyen d'apprendre la programmation puisque les apprenants ont besoin d'une meilleure compréhension de ce que l'algorithme signifie réellement. De plus, ils obtiennent une compréhension plus profonde de leurs algorithmes proposés surtout quand ils proposent une solution erronée et qu'ils reçoivent un feedback formatif (Labat, 2002).

Dans cette thèse, nous avons adopté l'approche *apprentissage par la pratique* et une stratégie qui vise à réutiliser des solutions communes et en essayant de trouver la solution la

plus similaire qui correspond à la solution proposée par l'apprenant à l'aide de mesure de similarité entre organigrammes. Dans la section suivante, nous expliquons la méthode d'évaluation proposée ainsi que l'outil proposé.

Qu'est-ce que l'organigramme et Pourquoi ?

Un organigramme est une représentation graphique normalisée, utilisée pour analyser ou décoder un problème. Les organigrammes sont utilisés pour :

- la Communication : Organigrammes sont une meilleure façon de communiquer la logique d'un système à tous les intéressés ou impliqués.
- Une analyse efficace : Avec l'aide de diagramme, un problème peut être analysé de manière plus efficace réduisant ainsi le coût et le gaspillage de temps.
- Une documentation adéquate : organigrammes du programme servent comme étant une bonne documentation du programme, qui est nécessaire à des fins diverses, ce qui rend les choses plus efficaces.
- Codage efficace : Les organigrammes agissent comme un guide ou un plan au cours de la phase d'analyse des systèmes et des programmes.
- Mise au point correcte : L'organigramme aide dans le processus de débogage.
- Maintenance du programme efficace : Le maintien du programme d'exploitation devient facile avec l'aide d'organigramme. Il aide le programmeur à mettre des efforts plus efficacement sur cette partie
- les symboles standards sont normalisés.
- Ils peuvent être compris par des personnes n'ayant aucune connaissance de la programmation.
- Ils peuvent servir à diviser le projet entier en sous-tâches. Ils serviront alors à évaluer les progrès accomplis.
- Ils décomposent les séquences d'opérations et aident ainsi à trouver les erreurs.

III. Modélisation d'une solution algorithmique

Pour simplifier des tâches complexes, il faut les décomposer en des tâches moins complexes et réitérer cette opération jusqu'à arriver à un niveau de décomposition comportant des opérations de base et/ou des opérations élémentaires. L'algorithme solution du problème sera alors une composition de ces dernières opérations (de base et élémentaires). Le nombre

d'étapes de décomposition dépend de la complexité du problème : plus ce dernier est complexe, plus le nombre d'étapes est important.

Cette méthode de raffinages successifs (dite également approche descendante) 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 résolution. L'algorithme est au dernier niveau de raffinage lorsqu'il ne comporte que des opérations de base, des opérations élémentaires et des structures de contrôle.

"*A deep understanding of programming, in particular the notions of successive decomposition as a mode of analysis and debugging of trial solutions, results in signicant educational bene ts in many domains of discourse, including those unrelated to computers and information technology per se.*'' (Seymou Papert, in 'Mindstorms')

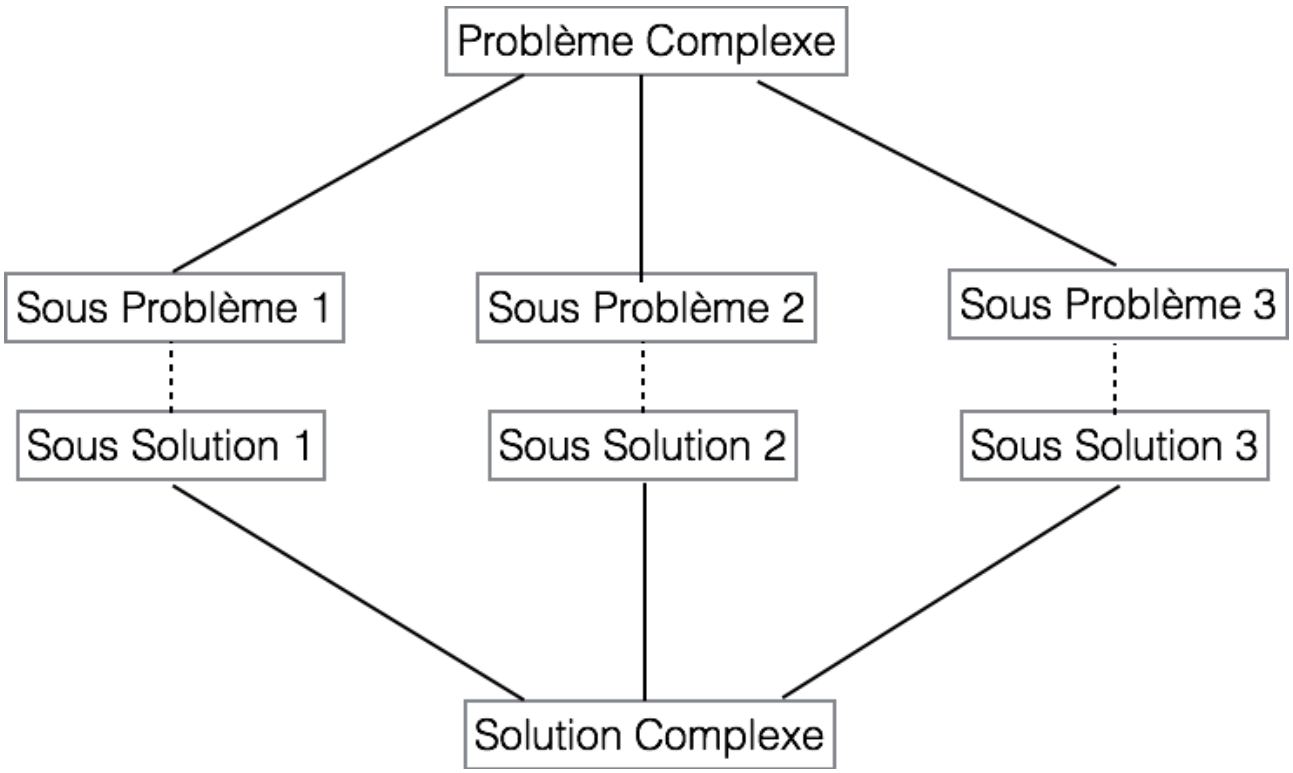


Figure N° 29: Décomposition d'un problème et La composition de sa solution

On définit une opération de base comme étant une opération connue en algorithmique telle que le tri d'un tableau. Une opération élémentaire, quant à elle, est une opération algorithmique simple (exemple : l'affectation).

Ainsi, au niveau 1, le problème est décomposé en un ensemble d'opérations de base, d'opérations élémentaires et d'opérations décomposables qui peuvent être liées par des structures de contrôle. Le nombre de niveaux de décomposition dépend de la complexité du problème à résoudre. En descendant dans les niveaux, seules les opérations décomposables sont décomposées, et cette décomposition s'arrête lorsqu'on arrive à un niveau constitué uniquement d'opérations de base et d'opérations élémentaires (figure 29).

Cette approche évite à l'apprenant de se noyer dans les détails dès le départ et diminue graduellement la complexité du problème abordé. En plus, l'apprenant peut librement exprimer sa solution, sans aucune influence ou restriction, ce qui favorise l'autonomie.

Notre objectif, par cette approche, est d'évaluer des solutions algorithmiques. Cependant, la retombée essentielle est l'apprentissage par l'apprenant de la décomposition. En effet, celle-ci est un passage obligé pour l'apprenant dans la formulation de sa solution.

1. Une méthode basée sur l'appariement structurel

L'appariement structurel détecte les correspondances en fonction de la structure des graphes.

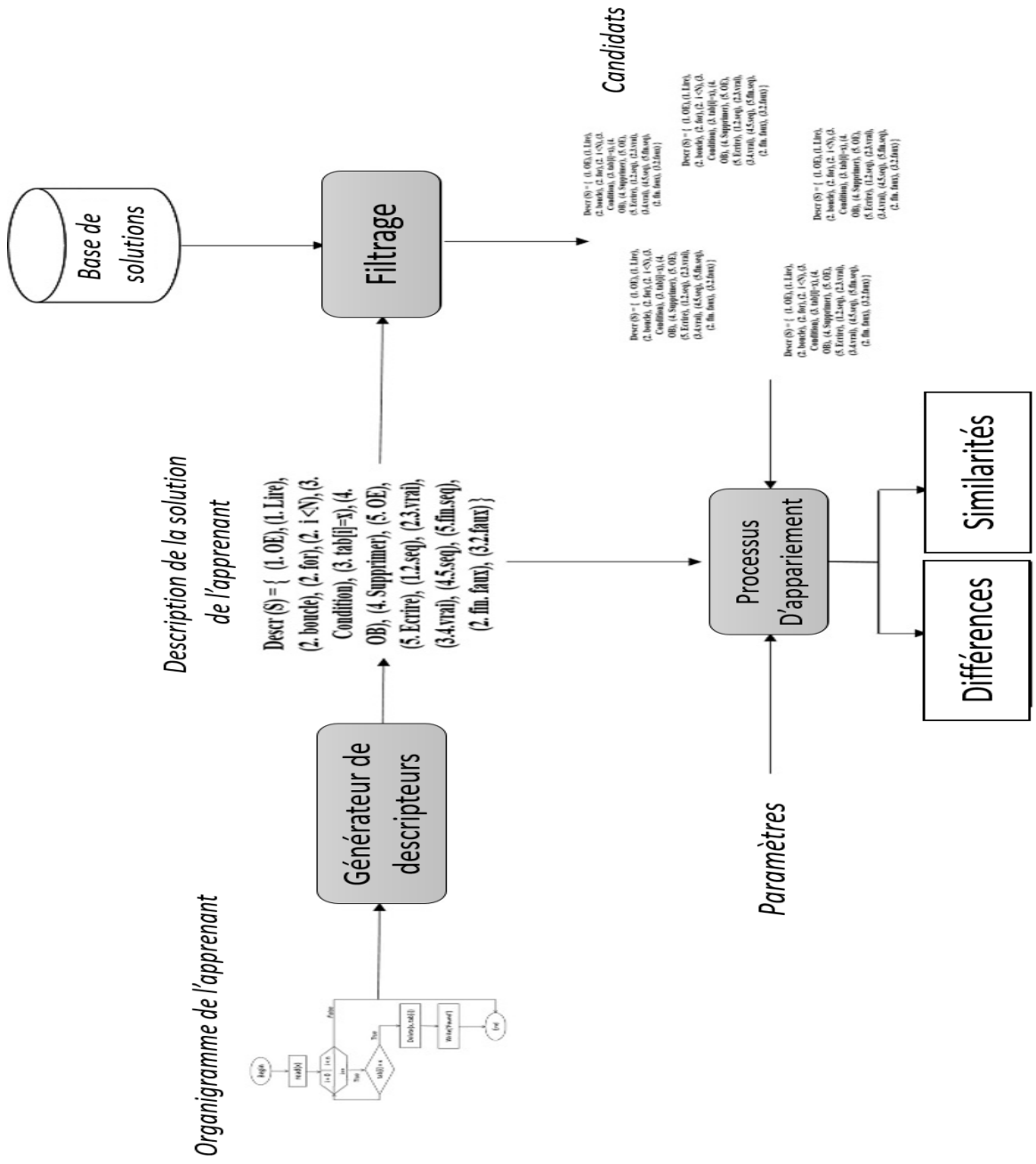
Anchor-PROMPT est une des premières méthodes d'appariement structurel utilisée pour aligner des ontologies du web sémantique, Noy et Musen (2001). Cette méthode prend en entrée un ensemble d'ancres (des correspondances exactes entre deux classes) et retourne un nouvel ensemble de correspondances entre classes. Cette méthode considère l'ontologie comme un graphe dans lequel les classes sont des nœuds du graphe et les propriétés des classes sont des arcs. Cette méthode analyse les chemins de même longueur entre deux ancres. Deux nœuds de deux chemins qui apparaissent dans la même position obtiennent un score non nul. Le score entre ces deux nœuds augmentera s'ils apparaissent à la même position dans deux autres chemins. Enfin, les correspondances obtenues en sortie sont les couples de classes ayant un score élevé. Cette méthode ne prend pas en compte l'étiquette des arcs (le nom des propriétés) entre les nœuds.

IV. Comment évaluer l'organigramme de l'apprenant ?

La figure ci-dessus représente une architecture fonctionnelle de la méthode proposée. Elle est composée principalement de trois composants essentiels : le générateur de descripteurs, le filtrage et le processus d'appariement (Aiouni et al. 2016).

IV.1. Généralités sur l'appariement

Avant de définir les différents problèmes et les techniques d'appariement classiques, il convient de préciser le sens du terme appariement (matching ou match). Le terme appariement prête à confusion car il peut être aussi bien un processus (ou une tâche) que le résultat de ce même processus. Les définitions de l'appariement varient sensiblement en fonction de ce facteur mais également en fonction de la nature des données à apparier, du domaine, du contexte d'application et de la présentation des résultats. Nous donnons ici quelques définitions que nous avons rencontrées au cours de nos lectures relatives au problème d'appariement et nous caractérisons le problème d'appariement de modèles.



L'appariement de formes ou de motifs (pattern matching) est la mise en correspondance de formes selon un ensemble prédéfini de règles ou de critères. L'appariement de formes se ramène à un problème de filtrage. Dans un programme informatique, le filtrage par motif (une autre traduction de pattern matching) est le fait de vérifier la présence des constituants d'un motif donné. À la différence de la reconnaissance de formes (pattern recognition), les motifs sont spécifiés rigidement et concernent conventionnellement des séquences ou des arbres. Le filtrage par motif est utilisé pour vérifier qu'un objet filtré a une structure désirée, pour trouver une structure appropriée, pour retrouver des parties alignées ou pour substituer les motifs reconnus par quelque chose d'autre.

Dans le cadre des ontologies, l'appariement (ontologymatching) est le processus de découverte des relations (ou correspondances) entre les entités de différentes ontologies (Euzenat&Shvaiko, 2007). Le terme alignement est employé plus communément dans le contexte des ontologies. L'alignement d'ontologies met en évidence les relations sémantiques de plusieurs ontologies à confronter (équivalence, subsomption, incompatibilité, etc.). L'expression des correspondances, appelée aussi alignement, peut par la suite être utilisée par exemple pour fusionner les ontologies, migrer des données entre ontologies ou traduire des requêtes formulées en fonction d'une ontologie vers une autre (DjoufakKengue et al., 2008).

L'appariement de schémas (schemamatching) consiste à trouver les correspondances sémantiques (i.e. appariements) entre deux schémas (Do et Rahm, 2007). L'appariement peut être considéré comme une opération ou un opérateur (match) qui prend deux schémas en entrée et produit un mapping entre les éléments des deux schémas correspondant sémantiquement les uns aux autres (Rahm et Bernstein, 2001).

Les problèmes d'appariement de graphes (graph matching) consistent à mettre en correspondance les sommets (noeuds) de deux graphes, l'objectif étant généralement de comparer les objets modélisés par les graphes.

Nous avons pu constater que la terminologie relative au problème d'appariement diffère entre les différents domaines et même au sein d'un même domaine. Quelles que soient la nature des modèles à analyser (ontologies, schémas, graphes) et leur représentation retenue en interne, la plupart des modèles en entrée peuvent être transcrits sous forme de graphes. Le but général du processus d'appariement est le même pour tous les types de modèles : l'identification et la qualification de correspondances entre les éléments.

Le résultat du processus d'appariement (match result), appelé également appariement, alignement ou mapping indique quels éléments des modèles en entrée correspondent les uns aux autres. Quelques variations d'interprétation existent entre les notions d'alignement et de mapping (Euzenat et Shvaiko, 2007). Un alignement est un ensemble de correspondances entre deux ou plusieurs (cas d'appariements multiples) modèles (par analogie aux alignements de séquence moléculaire). Une correspondance est une relation entre plusieurs éléments des modèles. Le terme mapping peut être considéré de manière équivalente à celui d'alignement. Néanmoins, dans le contexte des appariements de schémas, le terme alignement est très peu usité et celui de mapping est préféré. Cependant, un mapping peut aussi être vu comme une version orientée ou dirigée d'un alignement. La définition mathématique requerrait en principe que l'objet d'origine soit égal à son image (i.e. une relation d'équivalence). Un mapping est comme une collection de règles (mappingrules) toutes orientées dans la même direction, c'est-à-dire d'un modèle vers un autre, et telle que les éléments du modèle source apparaissent une fois au plus. Dans un mapping, une mappingrule associe un élément d'un modèle à un élément de l'autre modèle. Nous conservons la définition du mapping comme une version dirigée d'un alignement dans le reste du chapitre.

Le résultat final d'appariement peut associer un ou plusieurs éléments du premier modèle à un ou plusieurs éléments de l'autre modèle et réciproquement. Dans (Rahm et Bernstein, 2001) des relations de cardinalité entre les différents éléments appariés sont introduites pour qualifier les appariements produits. Elles correspondent à quatre cas de figure : 1:1 (un élément associé à un élément : couplage), 1:n (un élément associé à n éléments), n:1 (n éléments associés à un élément), ou m:n (m éléments associés à n éléments). Les trois derniers cas de figures correspondent à des appariements multiples (multivoques) d'éléments. De plus, un élément d'un modèle peut intervenir dans zéro, une ou plusieurs correspondances d'éléments de l'alignement produit entre deux modèles : la cardinalité est qualifiée de globale à cette échelle. En outre, à l'intérieur d'une correspondance individuelle, un ou plusieurs éléments d'un diagramme peuvent être appariés à un ou plusieurs éléments de l'autre modèle : la cardinalité de l'appariement est qualifiée ici de locale. Les cas de cardinalité globale qui concernent tous les éléments appariés sont orthogonaux aux cas des éléments appariés individuellement. C'est-à-dire qu'un résultat d'appariement peut avoir une cardinalité globale 1:n en combinaison avec des cardinalités locales d'appariement 1:1 ou 1:n.

La plupart des approches d'appariement sont restreintes à la cardinalité locale 1:1 car elles sélectionnent comme candidat d'appariement pour un élément d'un modèle, l'élément le

plus similaire dans l'autre modèle (Do et al., 2002). Les alignements produits ont donc en général une cardinalité globale 1:1 ou 1:n. Plus d'efforts et l'utilisation de critères plus sophistiqués sont nécessaires pour générer des appariements locaux et globaux de cardinalité n:1 et m:n. L'obtention d'alignements d'éléments de cardinalité globale m:n requiert par exemple de prendre en compte les structures agençant les éléments dans les modèles.

IV.1.1. Générateur de descripteurs

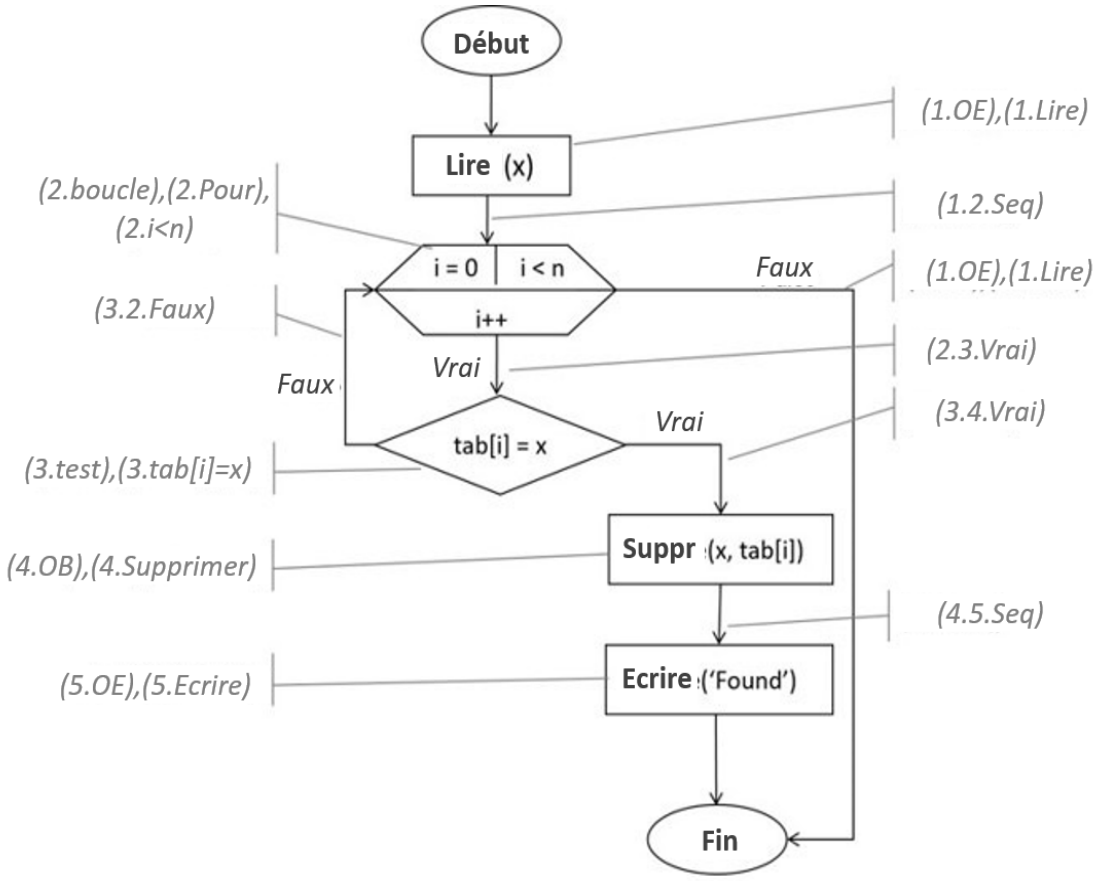
Pour automatiser la comparaison du processus d'apprentissage avec ceux de l'expert (plan de solutions), nous avons été inspirés par le travail de Sorlin (Sorlin et al., 2006) sur la mesure des graphiques multiples marqués. Cette approche nous a permis de proposer une méthode pour adapter des solutions algorithmiques.

De la structure organisationnelle de l'apprenant, une description de la solution est générée. Pour cela, nous avons attribué à chaque opération et chaque transition un ensemble d'étiquettes. L'ensemble du couple (Num_operation, étiquette) et triplets (Num_operarationS, Num_operationC, étiquette) sont des descripteurs et constituent la description de la solution. Le couple (Num_operation, étiquette) décrit les opérations de l'organigramme O et les triplets (Num_operationS, Num_operationC, étiquette) transition entre le fonctionnement (S et C sont pour la source et la cible).

Étant donné Lo un ensemble d'étiquettes d'opération et Lt un ensemble d'étiquettes de transition, un organigramme étiqueté est défini par une triplet S = <O, ro, rt> tel que :

- O est un ensemble fini d'opérations,
- ro⊆ O × Lo est une relation associant des étiquettes aux opérations (nœuds dans l'organigramme), à savoir ro est l'ensemble des couples (opi, l) de telle sorte que l'opération opi est marquée par l. Pour chaque nœud de l'organigramme, deux descripteurs sont affectés, l'un contient la nature du nœud, si elle est une opération, un test ou une boucle et le second contient l'étiquette. Par exemple, en figure 31, pour l'opération de lecture (x) deux descripteurs ont été assignés; (1.OE) mentionne le type de l'opération (opération élémentaire) et (1.read) pour l'étiquetage (nom de l'opération).
- • rt⊆ O × O × Lt est une relation associe les étiquettes aux arcs, à savoir rt est l'ensemble des triplets (opi, opj, l) tel que l'arc (opi, opj) est marqué par l. Étiquette en transition peut prendre deux valeurs seq pour la transition de séquencement ou Vrai / Faux quand il est un test de remplacement. La figure 31 résume le processus d'étiquetage d'un organigramme.

La description de la solution S est l'ensemble de toutes ses caractéristiques d'opération et de transitions Desc (S) = ro U rt.



Descr(S)={(1.OE),(1.Lire), (2.boucle),(2.Pour),(2.i<n), (1.2.Seq),
(1.OE),(1.Lire), (3.2.Faux), (3.test),(3.tab[i]=x), (4.OB),(4.Supprimer),
(2.3.Vrai), (3.4.Vrai), (5.OE),(5.Ecrire), (4.5.Seq)}

Figure N° 31: Descripteurs d'un organigramme

IV.2. Filtrage

L'étape filtrage se déroule avant le processus d'appariement afin d'optimiser le processus de recherche de l'organigramme le plus semblable. Elle consiste à chercher parmi les organigrammes prédéfinis ceux qui contiennent toutes les opérations de base critiques qui ont été prescrites par l'expert. Le filtrage permet de diminuer le nombre de solutions à mettre en correspondance. Par conséquent, on obtient un sous-ensemble de solutions provenant des prédéfinis qui contiennent des opérations critiques. Ce sous-ensemble de solutions sera présenté en tant que « candidats » où nous essayons de trouver la solution la plus proche de la solution de l'apprenant par le mécanisme d'appariement.

La validation d'une solution fournie par l'apprenant, se fait en comparant cette solution avec toutes les solutions de l'expert, pour le même exercice, référencées dans la base des plans.

IV.3. Processus de matching

Afin d'automatiser la comparaison de la démarche de l'apprenant avec celles de l'expert (plan de solutions) nous utilisons un appariement structurel que nous allons définir par la suite.

IV.3.1. Processus d'appariement

Le processus d'appariement peut être défini sous forme d'une « boite noire». Cette boite noire est une opération qui détermine l'alignement A' pour une paire de modèles m et m'. Quelques autres paramètres peuvent étendre la définition du processus : l'utilisation en entrée d'un alignement A qui est complété par le processus, des paramètres d'appariement (par exemple des poids et des seuils), des ressources externes utilisées par le processus d'appariement. Techniquement, le processus d'appariement peut être vu comme une fonction f qui, à partir d'une paire de modèles à apparier m et m', un alignement en entrée A, un ensemble de paramètres p et un ensemble de ressources externes r (par exemple des thésaurus), retourne un alignement A' entre ces modèles :

A' = f(m, m',A , p, r)

Le processus d'appariement peut être représenté schématiquement comme c'est le cas dans la figure suivante :

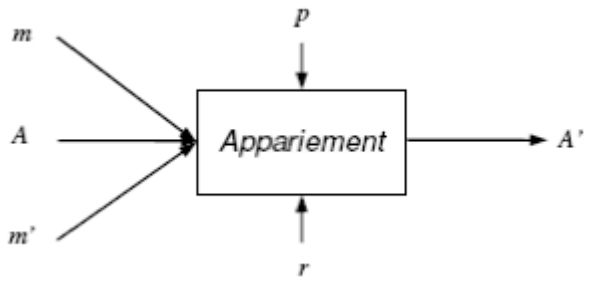


Figure N° 32: Processus d'appariement

Il peut être utile dans certains cas de considérer spécifiquement l'appariement de plus de deux modèles avec le même processus (Euzenat et Shvaiko, 2007). Ce processus est nommé dans ce cas appariement multiple (multiple matching). Le processus d'appariement multiple peut alors être vu comme une fonction f qui à partir d'un ensemble de modèles à apparier {m1, ...mn}, un alignement en entrée A, un ensemble de paramètres p et un ensemble de ressources, retourne un alignement A' entre ces modèles.

A' = f(m1, ...mn, A, p, r)

IV.3.2.Problèmes d'appariement et mesure de similarité

Nous venons de présenter le processus d'appariement sous forme d'une boite noire. En interne, l'appariement de plusieurs modèles et par conséquent de leurs constituants implique qu'ils soient comparés selon divers critères. Le problème de comparaison de plusieurs objets repose notamment sur l'évaluation de leurs similarités ou de leurs différences. La mesure de la similarité de deux objets consiste à identifier et à quantifier leurs points communs alors que la mesure de distance entre deux objets est l'identification et la quantification de leurs différences. Dans le contexte des graphes et donc des modèles assimilables à des graphes (c'est le cas pour les majorités des modèles manipulables dans les approches d'appariement de schémas ou d'ontologies), distance et similarité sont deux concepts duaux qui renvoient à un objectif commun. Le calcul de la distance ou de la similarité de deux structures de graphes permet habituellement de trouver le « meilleur » appariement des sommets d'un graphe, c'est-à-dire celui qui préserve le plus de caractéristiques des sommets et des arcs (Sorlin et al., 2007). Vu sous un autre angle, l'appariement est un problème d'optimisation où l'objectif est de trouver un alignement induisant la distance la plus faible entre les objets comparés.

Un appariement de graphes est dit univoque (univalent) lorsque chaque sommet d'un graphe est associé avec au plus un sommet de l'autre graphe (cardinalité d'appariement 1:1). Un appariement est dit multivoque lorsque chaque sommet d'un graphe peut être associé à un

ensemble de sommets de l'autre graphe (Sorlin et al., 2007). Un appariement multivoque d'un sommet d'un graphe avec plusieurs autres sommets de l'autre graphe se nomme généralement éclatement de sommets (cardinalité d'appariement 1:n). Inversement, un appariement multivoque de plusieurs sommets associés à un seul sommet de l'autre graphe est une fusion de sommets (cardinalité d'appariement n:1).

Différents problèmes courants d'appariement de graphes existent. Les mieux connus sont les appariements de graphes univoques exacts lorsque toutes les caractéristiques des sommets et des arcs sont préservées dans l'appariement ; c'est le cas notamment dans l'isomorphisme de graphes et de sous-graphes qui permettent respectivement de vérifier si deux graphes sont structurellement identiques (relation d'équivalence) ou si un graphe est « inclus » dans un autre (relation d'inclusion). Les appariements peuvent être univoques tolérants aux erreurs lorsque toutes les caractéristiques des sommets et des arcs ne peuvent pas être préservées lors de l'appariement (les objets appariés ne sont pas équivalents en tout point) : la recherche du plus grand sous-graphe commun (la plus grande partie commune à deux graphes) ou le calcul de la distance d'édition de graphes (le plus petit nombre d'opérations à effectuer pour transformer un graphe en un autre) en sont deux exemples.

Dans les cas plus complexes où une mise en correspondance univoque (un sommet avec zéro ou un sommet) ne suffit pas, il est nécessaire de mettre en place une mesure de similarité multivoque tolérante aux erreurs. Cette mesure est nécessaire dans les problèmes concernant la comparaison d'objets décrits à différents niveaux de granularité et où la recherche d'appariements multivoques (éclatements et fusions) est obligatoire. C'est le cas notamment dans des problèmes d'appariement d'une image bruitée à un modèle schématique (imagerie du cerveau humain par exemple) ou dans des problèmes d'appariements d'objets sur/sous segmentés. Les problèmes de recherche de composants complexes ou patterns dans des modèles en génie logiciel nécessitent également ce type de mesure pour restructurer ensuite les modèles de manière automatique.

IV.3.3. Mesure de similarité générique pour l'appariement de graphe

Nous allons présenter une mesure de similarité générique dans le contexte de l'appariement de graphes. Cette mesure de similarité nous a intéressés du fait de sa généricité.

Sorlin et Solnon ont proposé une mesure générique de distance paramétrable en fonction du type de graphes à apparier pour les différents problèmes d'appariement de

graphes (Sorlin , 2006) (Sorlin et al., 2007). Cette mesure de similarité de deux objets est fonction de leurs caractéristiques communes sur l'ensemble de toutes leurs caractéristiques.

Elle est générique dans le sens où elle modélise les mesures de distance et de similarité de graphes existantes. De plus, elle présente l'avantage d'être paramétrée par des fonctions de similarité pouvant être exprimées en fonction des connaissances propres au domaine, à l'application considérée ainsi qu'au type d'appariement recherché (univoque et/ou multivoque). De manière générale, deux fonctions de similarité permettent respectivement de calculer le score de similarité de chacun des sommets et des arcs.

Notre objectif est d'évaluer des productions d'apprenants (solutions algorithmiques). Pour cela, la solution à valider est comparée avec celles de l'expert.

Ce besoin de mesurer la similarité a pour but de faciliter et d'automatiser la validation. Notre motivation pour l'appariement structurel vient de la représentation formelle que peut avoir un algorithme, qui est l'organigramme.

Un premier point important de la mesure de similarité que nous utilisons par rapport aux mesures existantes, est qu'elle n'est pas seulement quantitative (évaluant le degré de similarité de deux solutions) mais aussi qualitative (explicitant en quoi les solutions sont similaires et en quoi elles sont différentes).

Un second point important de notre mesure est qu'elle permet de définir l'importance relative des caractéristiques, les unes par rapport aux autres et par conséquent d'introduire la connaissance dans le calcul de similarité.

Afin d'automatiser la comparaison de la démarche de l'apprenant avec celles de l'expert (plan de solutions), et en s'inspirant des travaux de Sorlin (2006) sur la mesure des graphes multi-étiquetés, nous proposons une méthode pour l'appariement de solutions algorithmiques.

Cette méthode consiste à comparer la solution de l'apprenant avec chacune des solutions de l'expert afin de mesurer la similarité entre elles. Elle est constituée de deux étapes séquentielles.

A partir de l'organigramme de l'apprenant, une description de la solution est générée. Pour cela, on affecte à chaque opération ainsi qu'à chaque transition un ensemble d'étiquettes. L'ensemble des couples (Num_opération, étiquette) et des tuples (Num_opérationS, Num_opérationC, étiquette) sont des descripteurs et constituent la description de la solution.

A un organigramme, on attribue une définition formelle S = < O, ro,rt>. Étant donné Lo un ensemble fini d'étiquettes d'opérations et Lt un ensemble fini d'étiquettes de transitions, tels que :

- O est un ensemble fini d'opérations.
- ro $\subseteq$ O * Lo est la relation associant opérations et étiquettes, i.e. ro est l'ensemble des couples (oi, l) tels que l'opération oi est étiquetée par l.
- rt $\subseteq$ O * O * Lt est la relation associant transitions et étiquettes, i.e. rt est l'ensemble des tuples (oi, oj, l) tels que la transition (oi, oj) est étiquetée par l.

La description de la démarche S est l'ensemble de toutes ses caractéristiques d'opérations et de transitions, Descr (S)=ro U rt.

Comme le montre la figure 4.7, l'étiquetage de l'organigramme de l'apprenant se fait de la manière suivante.

Pour les structures de contrôle :

- Une boucle est décrite par trois étiquettes, la première indique que c'est une boucle, la seconde son type et la troisième son intervalle.
- Une condition est décrite par deux étiquettes, la première indique que c'est une condition, la deuxième la condition elle-même.
- Les opérations sont décrites par deux étiquettes, la première montre le type de l'opération (opération de base ou élémentaire), la deuxième le nom de l'opération.

Les transitions n'ont qu'une seule étiquette qui peut être : vrai, faux ou seq (dans le cas d'un simple séquencement).

De la manière suivante la description d'une solution est générée. C'est la réunion de tous les couples décrivant les opérations (structures de contrôle inclus), ainsi que les tuples décrivant les transitions, la figure 4.8 montre la description de la solution de la figure 4.7.

IV.3.3.1. Le calcul de la similarité entre solutions

Un appariement entre deux démarches S1 = < O1, rO1, rt1 >et S2 = < O2, rO2, rt2 >, est une relation :

- m $\subseteq$ O1*O2.

Un tel appariement associe à chaque opération d'une solution l'opération du même ordre de l'autre solution.

Pour mesurer la similarité entre deux démarches par rapport à l'appariement m, nous proposons d'adapter la formule de similarité de Tvesky (1977) généralisée par sorlin (2006) :

$$Sim1_m(S1,S2) = \frac{f(descr(S1) \cap descr(S2))}{f(descr(S1) \cup descr(S2))} \qquad (1)$$

La formule (1) calcule la similarité de deux solutions, en mettant en correspondance leurs descriptions.

La fonction f définit l'importance relative des descripteurs, les uns par rapport aux autres. Cette fonction formule (2) est souvent définie comme une somme pondérée :

$$f(F) = \sum_{(o,l)\in F} poids(o,l) + \sum_{(o1,o2,l)\in S} poids(o1,o2,l) \qquad (2)$$

L'attribution des poids aux différents descripteurs d'une solution est réalisée par l'expert, celui-ci tient compte de l'importance du descripteur, ainsi que de l'objectif de l'exercice résolu par l'apprenant.

Nous allons présenter un exemple de calcul de similarité entre une solution d'apprenant et une solution d'expert référencée :

La figure 33 montrent respectivement l'organigramme de l'expert et sa description et l'organigramme de l'apprenant et sa description. En mettant leurs descriptions en correspondance, suivant l'appariement m, on remarque dans la figure 32 les descripteurs communs des deux solutions en rouge.

Il est évident que la solution retenue sera celle présentant la plus grande similarité. Celle-ci doit impérativement être inférieure à un seuil d'acceptabilité indiqué par l'expert. La note attribuée dépend du degré de similarité maximal entre la solution à valider et les solutions de l'expert.

Lorsque le degré de similarité est inférieur au seuil, la solution est envoyée à l'expert pour évaluation. Si celui-ci la juge intéressante, la solution sera ajoutée au plan de solutions de l'exercice, ce qui assure l'évolutivité de la base des plans, dans le cas contraire la solution de l'apprenant est rejetée.

Cette évolutivité du plan de solutions garantit une évaluation quel que soit la solution proposée par l'apprenant. Ainsi, plus le temps passe plus l'évaluation différée diminuera au profit de l'évaluation directe.

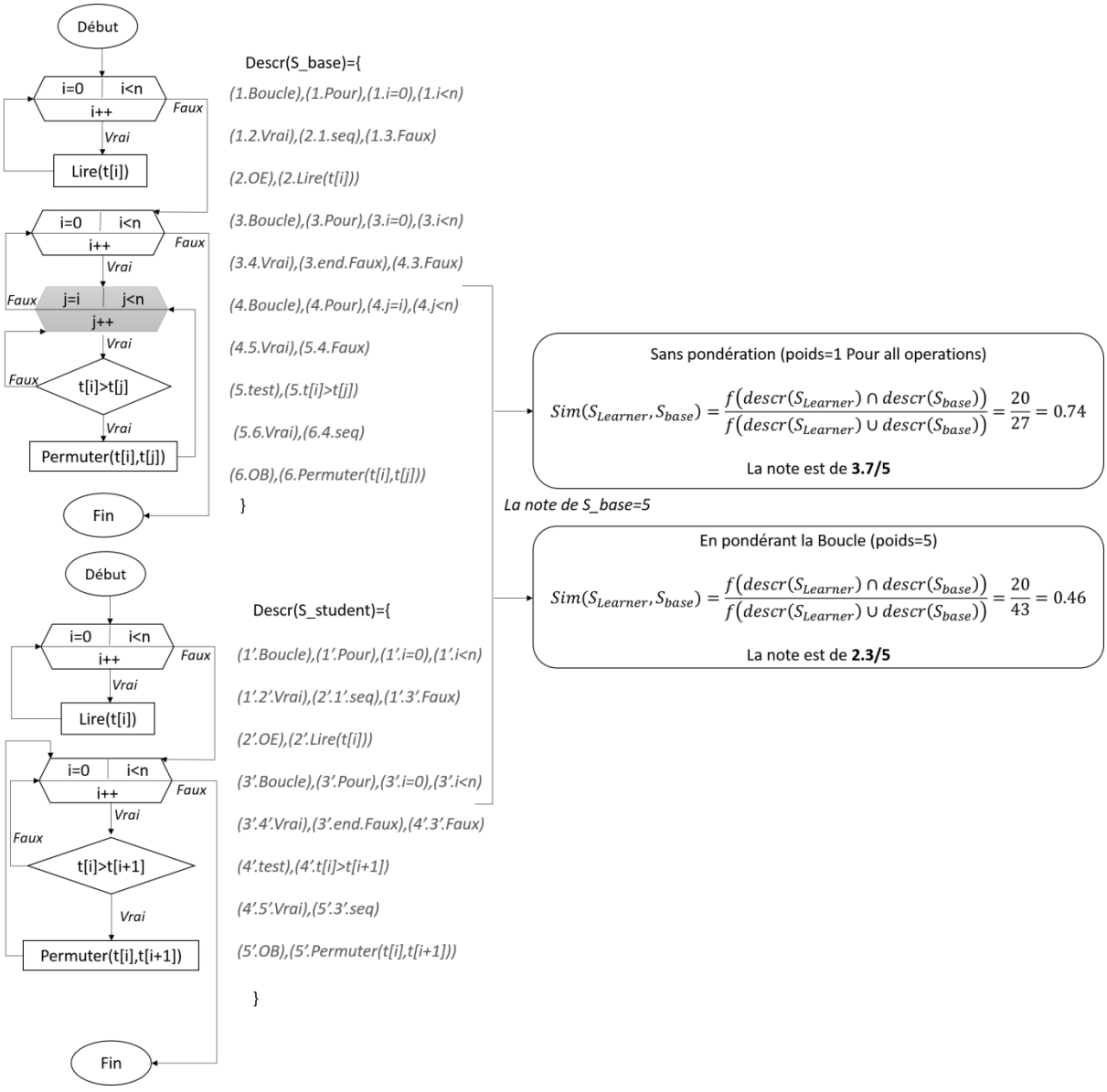


Figure N° 33: Processus de matching

V. Conclusion

L'approche proposée dans cette thèse a pour objectif d'évaluer les organigrammes des apprenants en algorithmique. Cette méthode est basée sur l'appariement structurel. L'évaluation exploite des aspects structurels et sémantiques intrinsèques à ces modèles. Leur comparaison, reposant sur des fonctions de similarité paramétrables, peut s'adapter aux modèles et au type d'appariement recherché.

Après avoir défini la méthode, nous l'avons implémentée et testée. Dans le chapitre suivant, nous allons présenter eAlgo, un outil d'évaluation automatique des organigrammes en algorithmique pour les apprenants novices ainsi que son expérimentation.

Chapitre IV :

eAlgo, Un outil d'évaluation des organigrammes en algorithmique pour les apprenants novices

I. Introduction

eAlgo est le résultat de l'implémentation de la méthode proposée. Ci-dessus une architecture fonctionnelle de ce dernier.

II. Architecture du système

Acteur principal du système, l'apprenant accède et bénéficie des fonctionnalités de l'environnement dans le but de faire une évaluation de ses compétences en algorithmique.

Cette architecture consiste à permettre aux différents acteurs (apprenant et enseignant) d'accéder aux différentes fonctionnalités offertes. Pour l'apprenant, l'apprentissage (s'auto-évaluer) et pour l'enseignant superviser l'apprentissage, ajouter, supprimer, modifier des exercices ainsi qu'analyser certaines solutions en attente, les analyser et éventuellement les incorporer à la base des solutions

- **Base des exercices**

Tous les exercices sont stockés dans cette base. Leurs mise à jour est faite par l'enseignant et elle est consultée par les apprenants. Elle contient les énoncés des exercices.

- **Base des solutions**

Les plans de solutions des exercices sont sauvegardés dans cette base. Les solutions des exercices sont stockées sous formes de descriptions pour faciliter le calcul de la similarité.

- **Base des solutions en attente d'être évaluées**

Contient les solutions des apprenants n'ayant satisfait aucune solution « expert ». Ces solutions sont soumises à l'expert pour évaluation. Elles resteront dans cette base jusqu'à l'intervention de l'expert.

- **Module processus d'appariement**

Avant d'évaluer une solution, il faut la reconnaitre. Ce module prend en charge la reconnaissance en appliquant la méthode AASA développée section en VI.3.1 . Si elle est reconnue, elle passe à l'évaluation, sinon elle sera mise dans la base des solutions en attente d'être évaluées.

Nous présentons ci-après le prototype qui concrétise notre approche. Il a pour rôle de valider l'approche d'évaluation proposée et l'appariement structurel réalisé.

II.1. Conception de l'environnement eAlgo

Le système eAlgo est basé sur l'approche développée pour l'évaluation des connaissances procédurales en algorithmique. Il permet aux étudiants de s'auto-évaluer et de tester leurs compétences en algorithmique.

Partant de nos objectifs, nous avons décidé de distinguer deux acteurs principaux dans notre système : l'apprenant et l'expert (enseignant).

Le rôle de l'expert est de gérer les bases de données, d'apporter des modifications concernant l'évolution des exercices et d'évaluer des solutions lorsque cela est nécessaire.

L'apprenant peut consulter les exercices existants pour en sélectionner un. Il peut éditer des exercices sous forme d'organigramme. Il est aussi invité à faire des autoévaluations pour se valider et consulter ses annotations sous forme de remarques ou résumés.

Les deux acteurs peuvent accéder aux différentes activités à travers l'environnement.

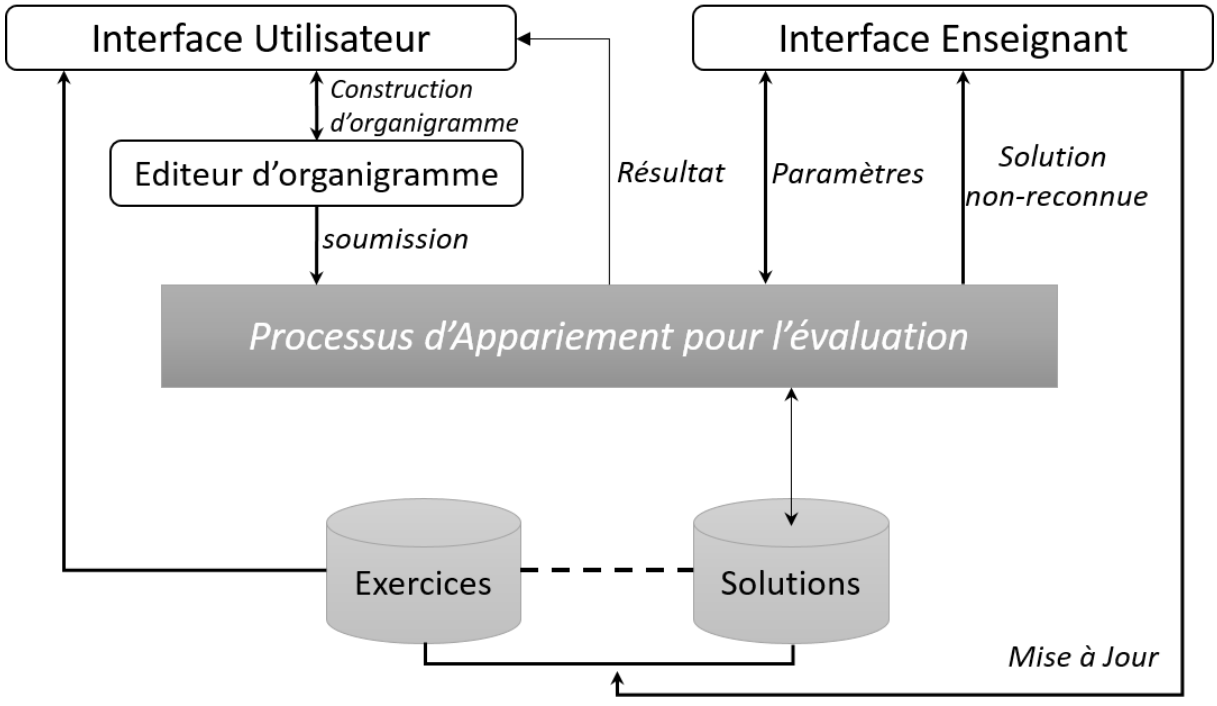


Figure N° 34: Architecture d'eAlgo

III. Expérimentation

Le but de l'étude est de mesurer l'efficacité du système eAlgo dans l'évaluation des organigrammes en le comparant avec les résultats de l'évaluation des experts humains. Pour cela, nous avons mené une étude de laboratoire. Cette section décrit les détails des résultats.

IV. Méthode

L'étude a impliqué 35 étudiants de première année Tronc commun Mathématique Informatique et cinq enseignants du module Algorithmique et Structures de données. Lors d'une séance de laboratoire, les étudiants ont été invités à produire des organigrammes pour résoudre un exercice sur eAlgo. L'énoncé de l'exercice était le suivant :

Ecrire un organigramme qui implémente la fonction mathématique suivante :

$F(x) = Ax+B \; ; \; X>0$

$F(x) = A \; ; \; X<0$

$F(x) = B \; ; \; X=0$

Le but de cet exercice est la capacité à utiliser les structures conditionnelles imbriquées.

Les organigrammes construits par les participants ont été notés instantanément par eALGO puis récupérés. Les organigrammes récupérés sont ensuite imprimés et évalués par les cinq enseignants d'informatique. Ces enseignants qui ont participé à cette expérience n'ont pas assisté au test avec les étudiants.

Les résultats des scores automatisés ont été comparés à ceux obtenus manuellement par les enseignants. Dans la section ci-dessous, nous décrivons les résultats obtenus.

V. Résultats

Cette étude a été conçue pour déterminer si les notes d'eALGO correspondent bien à celles des enseignants, validant ainsi sa méthode de notation. Dans l'expérience, les organigrammes d'étudiants ont été notés par eALGO et les enseignants travaillant avec une solution d'échantillon illustrée dans la figure 35.

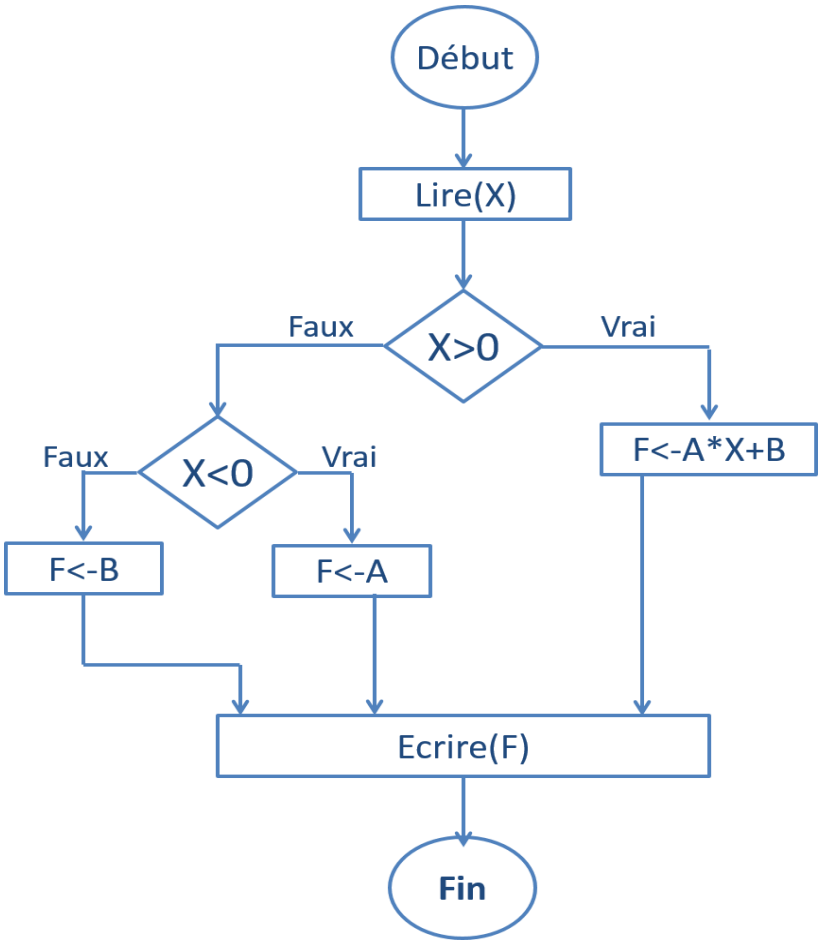


Figure N° 35: Solution type utilisée dans l'expérimentation

Les statistiques descriptives indiquées dans le tableau 3 montrent une forte homogénéité dans l'attribution des notes par eALGO ainsi que les enseignants. Les notes d'eAlgo montrent un bon accord avec la moyenne (MeALGO = 2.22, MAVG = 1,98) et la plupart des enseignants (MT1 = 2.21, MT3 = 2, MT4 = 2.57). Quand nous regardons la corrélation entre les deux ensembles de notes dans le tableau 2, le coefficient de corrélation de Pearson est de 0,67 (ce qui est significatif au niveau de 0,001). La corrélation de spearman peut être utilisée pour voir comment les notes classent les élèves par rapport aux notes des enseignants et on obtient 0,74 (ce qui est significatif au niveau de 0,001). Ces résultats montrent une excellente correspondance entre les deux ensembles de notes à la fois pour la comparaison directe avec les notes des enseignants humains et à l'ordre de classement des réponses des élèves.

Selon le tableau 4, nous constatons que toutes les corrélations sont positives ce qui signifie que toutes les notes varient dans le même sens. Nous avons constaté que eALGO est en corrélation positive avec tous les enseignants et surtout avec T4 et la moyenne de l'ensemble des enseignants.

Tableau N° 3: Statistiques Descriptives

	T1	**T2**	**T3**	**T4**	**T5**	**Moyenne score**	**eAlgo**
Min.	1.00	0.50	0.75	1.00	0.50	1.15	0.70
1st Quartile	2.00	0.87	1.75	2.00	1.25	1.65	1.64
Median	2.25	1.00	2.00	2.50	1.50	1.95	2.47
Moyenne	2.21	1.34	2.00	2.57	1.78	1.98	2.22
3rd Quartile	2.50	1.75	2.25	3.00	2.50	2.27	2.88
Max.	3.25	4.00	3.25	3.75	3.50	3.40	3.26
SD	0.53	0.73	0.59	0.75	0.85	0.49	0.77
Mode	2	1	2	3	1.5	2.25	3.25
Variance	0.29	0.53	0.34	0.57	0.72	0.24	0.60

Les statistiques descriptives du tableau 1 montrent une forte homogénéité dans l'attribution des scores par eAlgo ainsi que les enseignants.

Tableau N° 4:Corrélations

	eAlgo					
	T1	T2	T3	T4	T5	Moyenne.
r(33)	0.51***	0.32 *	0.47**	0.65***	0.44**	0.67***
P	0.53 ***	0.40 **	0.48**	0.70***	0.49**	0.75***

*Significativité * >0.05 ; **<0.05 ; ***<0.001*

Quand nous regardons la corrélation entre les deux ensembles de marques (tableau 2), le coefficient de corrélation de Pearson est de 0,67 (ce qui est significatif au niveau de 0,001). rho la statistique de Spearman peut être utilisé pour voir comment le marqueur automatique classe les élèves par rapport aux marqueurs humains et on obtient 0,74 (ce qui est significatif au niveau de 0,001). Ces résultats montrent une excellente correspondance entre les deux

ensembles de marques à la fois pour la comparaison directe avec les marques humaines et à l'ordre de classement des réponses des élèves.

Selon le tableau 2, nous constatons que toutes les corrélations sont positives ce qui signifie que toutes les marques varient dans le même sens. Nous avons constaté que eAlgo en corrélation positive avec tous les enseignants et surtout avec T4 et la moyenne sur l'ensemble des enseignants.

Pour mettre en évidence la distribution des scores, la figure 36 résume les différents scores de chaque enseignant avec eAlgo.

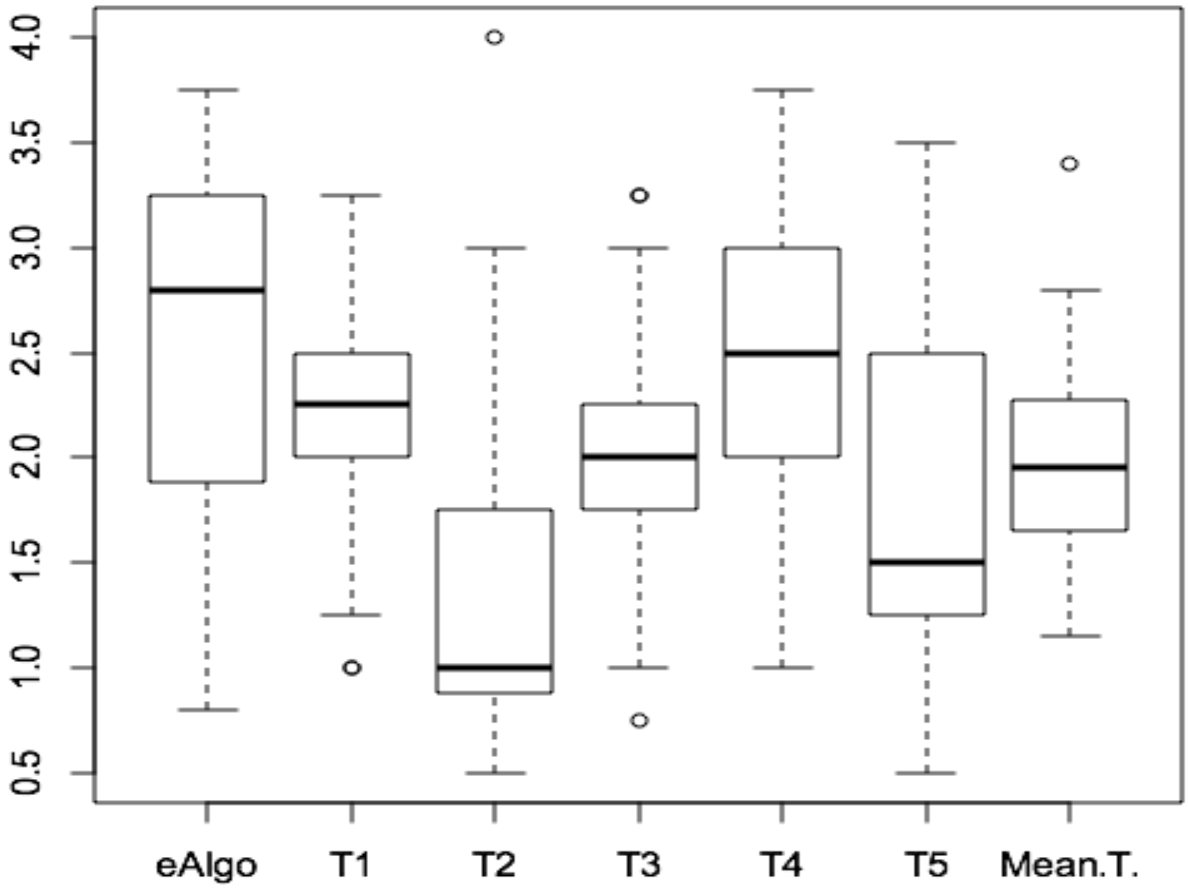


Figure N° 36: Distribution des notes des enseignants et d'eALGO

VI. DISCUSSION

Le but de cette étude était d'examiner la relation entre l'évaluation d'eAlgo et l'évaluation des enseignants d'algorithmique pour la notion d'organigramme. L'hypothèse de recherche qu'eAlgo a adoptée est : est-ce que l'appariement structurel peut être utilisé pour évaluer les organigrammes afin d'améliorer l'apprentissage de l'algorithmique. Pour répondre à cette question nous avons conduit une expérimentation pour comparer les résultats d'eAlgo avec des évaluations de plusieurs enseignants.

Un plan de recherche corrélationnelle a été utilisé pour répondre à la question, la corrélation entre la performance eAlgo et enseignants évaluateurs a été examinée. Les résultats des analyses de corrélation, en utilisant letest de coefficient de corrélation non paramétriques de Spearman, ont indiqué que des corrélations étaient statistiquement significatives entre les notes d'eAlgo et la moyenne des notes des enseignants ($rs = .75$, $p < .001$).En outre, les résultats ont montré une corrélation positive entre les évaluateurs et eAlgo testé individuellement.

Cette étude a démontré qu'eAlgo évalue l'algorithme des élèves indépendamment de tout critère idiosyncrasique contrairement aux enseignants humains chez qui la notation est généralement subjective et influencée par de nombreux paramètres (comme effet de halo, etc.). L'évaluation paramétrique d'eAlgo permet aussi une adaptation selon l'évaluation voulue par chaque enseignant. Ce qui nous permet d'avoir aussi une bonne corrélation entre eAlgo et les enseignants un par un (tableau 2).

Pour examiner plus en détail la logique de notation de tous les enseignants, ainsi que le système eAlgo, nous avons utilisé l'Analyse par Composantes Principale pour mettre en évidence les similitudes et les différences entre la méthode automatique adaptée par eAlgo et les notations des enseignants et de découvrir et résumer l'inter corrélations entre les correcteurs humaines et eAlgo.

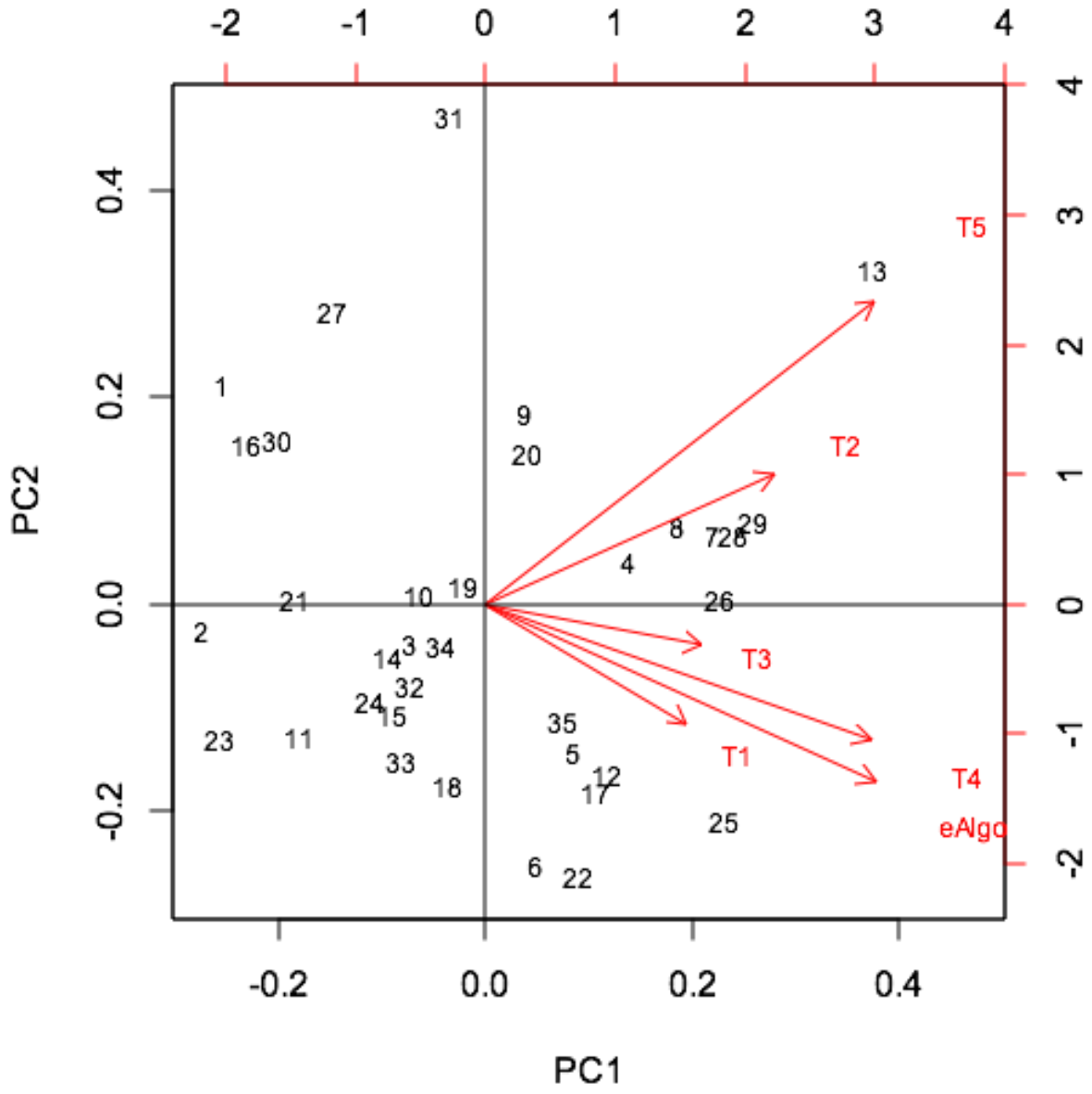


Figure N° 37: Analyse par Composante Principale entre eALGO et les enseignants

La figure 37 montre que les enseignants et eAlgo sont orientés dans la même direction ce qui signifie que les enseignants et eAlgo ont évalué les mêmes objets de la même manière.

La première composante principale est fortement corrélée avec les six variables d'origine. La première composante principale augmente avec le correcteur automatique et les enseignants humains. Cela montre que les évaluations des enseignants convergent. Si l'on augmente, alors le reste augmente également leurs scores. Cette composante peut être considérée comme la moyenne des six évaluateurs. En fait, nous pourrions affirmer que toutes les corrélations de la première composante principale sont comprises entre 0.37 et 0.47 ce qui

signifie que tous les correcteurs corrèlent le plus fortement avec ce composant principal qui est la moyenne des scores moyens.

La deuxième composante principale est fortement corrélée avec l'enseignant 5 qui attribue des scores faibles (médiane = 1.5). Ce composant peut être considéré comme une mesure de la sévérité ou d'indulgence dans l'attribution des notes. Elle augmente avec l'augmentation de la sévérité de la note et diminue avec la diminution de l'indulgence des correcteurs dans l'attribution des notes. Selon l'interprétation de chaque composante principale, eAlgocomparé aux enseignants humains pourrait être considéré comme un correcteur indulgent dans l'attribution des notes.

VII. Conclusion

Dans cette recherche, nous avons développé un outil de notation automatisé appelé eALGO qui repose sur une méthode d'appariement et offre aux apprenants débutants un environnement de pratique et une correction instantanée de leurs solutions. En résumé, l'analyse des résultats de l'enquête discutés dans cette thèse semble suggérer que le classement automatisé en utilisant le graphique correspondant peut réduire en toute sécurité la charge des étudiants titrant en particulier dans le cas d'un grand nombre de missions comme dans MOOCs cours Sandeen (2013).

Au cours de cette étude, eALGO a montré son efficacité dans l'attribution des scores par rapport aux enseignants humains qui sont toujours influencés par de nombreux faits. Un objectif à long terme de la notation automatisée est d'être en mesure de générer une rétroaction formative précise et de voir ainsi si eALGO a la capacité de donner une rétroaction formative aux étudiants et combien cette rétroaction peut aider les apprenants à progresser et à développer leurs compétences en programmation.

CONCLUSIONS ET PERSPECTIVES

La programmation est un savoir-faire. Or, l'acquisition des savoir-faire en algorithmique et en programmation est un aspect un peu négligé car probablement trop complexe. Dans l'apprentissage de la programmation, nous devrons être plus exigeants : nous voulons que les apprenants novices apprennent comment procéder. Et, comme cette visée est nouvelle pour eux, les apprenants résistent. Mais là réside sans doute l'originalité de l'enseignement de la programmation et donc ce qu'elle peut apporter à la formation générale comme une nouvelle forme de pensée.

Dans cette thèse, nous avons traité une problématique intrinsèque à l'apprentissage de l'algorithmique et de la programmation, l'évaluation des organigrammes produits par les apprenants. Car les apprenants ont besoin d'un outil automatique qui les aide à apprendre la conception des organigrammes.

Après avoir précisé le contexte et les objectifs de la thèse, nous avons présenté un état de l'art sur les outils et les systèmes existants qui traitent la problématique de l'apprentissage de la programmation à l'aide des organigrammes. Nous avons essayé de présenter une méthode d'évaluation automatique des organigrammes élaborés par les apprenants. Cette méthode d'évaluation repose sur la comparaison entre deux organigrammes (l'organigramme de l'apprenant et un organigramme solution) au moyen d'un algorithme d'appariement structurel.

Pour évaluer notre proposition, un outil baptisé eAlgo a été développé et testé. Au cours de cette étude, eAlgo a montré son efficacité d'attribuer des scores par rapport aux enseignants humains qui sont toujours influencés par des paramètres subjectifs.

Les expérimentations de multi corrections des organigrammes des apprenants ont montré qu'eAlgo attribue des notes qui se rapprochent de la moyenne des enseignants. C'est le cas le plus favorable lors de la correction des réponses des apprenants par eAlgo car la moyenne des notes des enseignants représente la note qui représente le mieux la réponse.

Un objectif à long terme de la notation automatisée est d'être en mesure de générer une rétroaction formative précise.

REFERENCES BIBLIOGRAPHIQUE

Aiouni R., Bey A., Bensebaa T. (2016), An automated assessment tool of flowchart programs in Introductory Programming Course using graph matching, Journal of e-Learning and Knowledge.

AIOUNI ,Rym and BENSEBAA, Tahar and BENSALEM, Hana (2010) *Approched'évaluation de l'algorithmiquedans un EIAH*. In: LEAFA 2010 The First International Conference on e-Learning For All, 3-5 June 2010 ,Hammamet, Tunisia.

AIOUNI, Rym and BENSEBAA, Tahar, Algorithmic Evaluation Method of Learning Environment, 5th International Forum on Engineering Education (IFEE 2010), Sharjah, United Arab Emirates, November 2010.

Aiouni R., Bey A., Bensebaa T. (2016), eALGO, An Assessment tool of Flowchart Programs for Novices. *À apparaitredans International Journal of Learning and Innovation (IJIL) Inderscience.*

Arai, M. and Yamazaki, T. (2006) 'Design of a learning support system to aid novice programmers in obtaining the capability of tracing', 6th International Conference on Advanced Learning Technologies, Washington, DC, USA, IEE Computer Society, pp.396–398.

Areias, C. and Mendes, A. (2007) 'A tool to help students develop programming skills', International Conference on Computer Systems and Technologies, Ruse, Bulgaria, ACM, pp.1–7.

Atanasova, G. and Hristova, P. (2003) 'Flow chart interpreter: an environment for software animation representation', International Conference On Computer Systems and Technologies, Ruse, Bulgaria, ACM, pp.453–458.

Bagert, D.J. and Calloni, B.A. (1999) 'Teaching programming concepts using an icon-based software design tool', IREEE Transactions on Education, Vol. 42, No. 4, pp.365–370.

Brown, M. and Sedgewick, R. (1984) 'A system for algorithm animation', 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 84), Minneapolis, MN – USA, ACM, pp.177–187.

Calloni, B. and Bagert, D. (1997) 'Iconic programming proves effective for teaching the first year programming sequence', ACM SIGSCE '97 CA, USA, pp.262–266.

Calloni, B.A. and Bagert, D.J. (1994) 'Iconic programming in BACCII vs. textual programming: which is a better learning environment?', ACM, SIGSCE Phoenix AZ, pp.188–192.

Carlisle, M., Wilson, T., Humphries, J. and Hadfield, S. (2004) 'RAPTOR: introducing programming to non-majors with flowcharts', Journal of Computing Sciences in Colleges, Consortium for Computing Sciences in Colleges, Washington DC, USA, Vol. 19, No. 4, pp.52–61.

Carlisle, M., Wilson, T., Humphries, J. and Hadfield, S. (2005) 'RAPTOR: a visual programming environment for teaching algorithmic problem solving, ACM SIGCSE Bulletin, CM, New York, NY, USA, Vol. 37, No. 1, pp.176–180.

Chen, S. and Morris, S.(2005) 'Iconic programming for flowcharts, Java, Turing, ETC', Conference on Innovation and Teaching Computer Science Education (ITiCSE), Caparica, Portugal, ACM, pp.104–107.

Chun, S-J. and Ryoo, J. (2010) 'Development and application of a web-based programming learning system with LED display kits', Proceedings of the 41st ACM Technical Symposium on Computer Science Education, pp.310–314, Milwaukee, WI, March, ACM.

Cilliers, C., Calitz, A. and Greyling, J. (2005) 'The effect of integrating an iconic programming notation into CS1', Proc. ACM ITiCSE, pp.108–112.

Cohen, W. et D. A. Levinthal. 1990. "Absorptive Capacity : A New Perspectiveon Learning and Innovation". Administrative Science Quarterly, 35: 128-152.

Crews, T. (2001) Using a Flowchart Simulator in a Introductory Programming Course, pp.307–312, Western Kentucky University, Bowling Green, KY, USA.

Flowchart-based programming environments for improving comprehension 55 Crews, T. (2009) Visual Logic, PGS Systems, ACM, Bowling, Green, KY, USA [online]

Crews, T. and Murphy, C. (2004) Programming Right From Start With Visual Basic, Prentice Hall, Upper Saddle River, NJ, USA, ISBN 0131085797.

Crews, T. and Ziegler, U. (1998) 'The flowchart interpreter for introductory programming courses', Proceedings of FIE '98 Conference, Tempe, Arizona, USA, pp.307–312.

Domagala, M. (2006) 'Dev Flowcharter, SourseForge', ACM SIGCSE Bull, Vol. 32, No. 2, pp.120–127.

Fatesoft (2009) Code Visual to Flowchart, FateSoft, Eden Prairie, NM, USA [online] http://www.fatesoft.com/s2f/ (accessed 4 May 2014).

Glezou, K. and Grigoriadou, M. (2007) A Novel Didactical Approach of The Decision Structure for Novice Programmers, EuroLogo, Bratislava, Slovak Republic, Comenious University (accessed 4 May 2014).

Goldenson, D.R. (1989) 'The impact of structure editing on introductory computer science education: the results so far', SIGCSE Bulletin, September , Vol. 21, No. 3, pp.26–29.

Greyling, J., Cillers, C. and Calitz, A. (2006) 'B# the development and assessment of an iconic programming tool for novice programmers', 7th International Conference on Information Technology Based Higher Education and Training, IEEE. Ultimo, NSW, Australia, pp.376–375.

Hall, M.S. (2007) 'Raptor: nifty tools', J. Comput. Sci. Coll., Vol. 23, No. 1, pp.110–111.

Hooshyar, D., Ahmad, R.B., Nasir, M.H.N.M. and Mun, W.C. (2014) 'Flowchart-based approach to aid novice programmers: a novel framework', International Conference on Computer and Information Sciences (ICCOINS), DOI: 10.1109/ICCOINS.2014.6868826.

Hooshyar, D., Alrashdan, M.T. and Mikhak, M. (2013) 'Flowchart-based programming environments aimed at novices', Int. J. In ovative Ideas (IJI), January–March, Vol. 13, No. 1, pp.52–62.

Khwaja, A.A. and Urban, J.E. (1993) 'Syntax-directed editing environments: issues and features', Proc. 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice, Indianapolis, USA, pp.230–237.

Kunimune, H., Yokoyama, K., Takizawa, T. and Fuwa, Y. (2008) 'Development and evaluation of a web-based asynchronous discussion system on e-learning materials', Int. J. Advanced Intelligence Paradigms, Vol. 1, No. 2, pp.163–177.

Lane, H.C. and van Lehn, K. (2004) 'A dialogue-based tutoring system for beginning programming', Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS), AAAI Press, pp.449–454, Miami Beach, FL.

Lavonen, J., Meisalo, V. and Lattu, M.(2001) 'Problem solving with an icon oriented programming tool: a case study in technology education', The Journal of Technology Education, Virginia Tech, Blacksburg, VA, USA, Vol. 12, No. 2, pp.21–34.

Lavonen, J., Meisalo, V., Lattu, M. and Sutinen, E. (2002) 'Concretising the programming task: a case study in a secondary school', Computers and Education, Elsevier Science, Vol. 40, No. 2, pp.115–135.

Levitin, A. (2005) 'Analyze that: puzzles and analysis of algorithms', Proceedings of the 36th ACM Technical Symposium on Computer Science Education (SIGCSE'05), St. Louis, MO, USA, February, pp.171–175.

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',

Proc. 5th Int. Conf. Computer Systems and Technologies, ACM, New York, NY, USA, pp.1–6.

Mendes, A.J., Gomes, A., Esteves, M., Marcelino, M.J., Bravo, C. and Redondo, M.A. (2005) 'Using simulation and collaboration in cs1 and cs2', SIGCSE Bull, Vol. 37, No. 3, pp.193–197.

Mukherjee, A. and Garain, U. (2008) 'A review of methods for automatic understanding of natural language mathematical problems', Artificial Intelligence Review, Vol. 29, No. 2, pp.93–122.

Mukherjee, A., Garain, U. and Biswas, A. (2013) 'Automatic text-to-diagram conversion: a novel teaching aid for the blind people', J. on Educational Technology & Society (ETS), Vol. 17, No. 3, pp.40–53.

Nikunja Swain, P.E. (2013) 'RAPTOR – a vehicle to enhance logical thinking', 120th ASEE Annual Conference & Exposition, American Society for Engineering Education, 23–26 June.

Price, B., Baecker, R. and Small, I. (1998) 'An introduction to software visualization', in Stasko, J., Domingue, J., Brown, M.H. and Price, B.A. (Eds.): Software Visualization, Chapter 1, pp.3–27, MIT Press, Cambridge, MA.

Roy, G.G., Kelso, J. and Standing, C. (1998) 'Towards a visual programming environment for software development', Software Engineering: Education and Practice, Vol. 3, No. 5, pp.381–388.

Santos, A., Gomes, A. and Mendes, A.J. (2010) 'Integrating new technologies and existing tools to promote programming learning algorithms', Journal of Algorithms, Vol. 3, No. 9, pp.183–196.

Scott, A. (2010) Using Flowcharts, Code and Animation for Improved Comprehension and Ability in Novice Programming, University of Glamorgan, UK.

Scott, A., Watkins, M. and Mcphee, D. (2007) 'A step back from coding – an online environment and pedagogy for novice programmers', Proc. 11th Java in the Internet Curriculum Conf., The Higher Education Academy, London Metropolitan University, UK, pp.35–41.

Scott, A., Watkins, M. and McPhee, D. (2008a) 'E-learning for novice programmers – a dynamic visualisation and problem solving tool', 3rd Int. Conf. Information and Communication Technologies: From Theory to Applications, ICTTA, 7–11 April, Damascus, Syria, pp.1–6.

Scott, A., Watkins, M. and McPhee, D. (2008b) Progranimate – A Web Enabled Algorithmic Problem Solving Application, CSREA EEE, pp.498–508.

Sharma, R., Bedi, P. and Banati, H. (2013) 'Stigmergic agent-based adaptive content sequencing in an e-learning environment', Int. J. Advanced Intelligence Paradigms, Vol. 5, Nos. 1/2, pp.59–82.

Shneiderman, B., Mayer, R., McKay, D. and Heller, P. (1977) 'Experimental investigations of the utility of detailed flowcharts in programming', Communications of the ACM, Vol. 20, No. 6, pp.373–381.

Shuaib Ahmed, S., Purna Chandra Rao, B. and Jayakumar, T. (2013) 'A framework for multidimensional learning using multilabel ranking', Int. J. Advanced Intelligence Paradigms, Vol. 5, No. 4, pp.299–318.

Watts, T. (2004) 'The SFC editor a graphical tool for algorithm development', Journal for Computing Sciences in Colleges, Consortium for Computing Sciences in Colleges, Chicago, IL, USA, Vol. 20, No. 2, pp.73–85.

Xinogalos, S. (2013) 'Using flowchart-based programming environments for simplifying programming and software engineering processes', Proceedings of 4th IEEE EDUCON Conference, Berlin, Germany, 13–15 March, IEEE Press, pp.1313–1322.

Xinogalos, S. and Satratzemi, M. (2004) 'Introducing novices to programming: a review of teaching approaches and educational tools', Proceedings of the 2nd International Conference on Education and Information Systems, Technologies and Applications (EISTA 2004), Orlando, Florida, USA, 21–25 July, Vol. 2, pp.60–65.

Ziegler, U. and Crews, T. (1999) 'An integrated program development tool for teaching and learning