Utilisation des exemples et de la démonstration dans l'Apprentissage de l'Algorithmique

Faouzia Benabbou, Mostafa Hanoune

Utilisation des exemples et de la démonstration dans l'Apprentissage de l'Algorithmique

Faouzia BENABBOU, Université Hassan II, Faculté des sciences Ben M'Sik, Laboratoire Technologie d'Information et Modélisation, BP : 7955, 2200, Casablanca, hgfbenabbou@menara.ma
Mostafa HANOUNE, Université Hassan II, Faculté des sciences Ben M'Sik, Laboratoire Technologie d'Information et Modélisation, BP : 7955, 2200, Casablanca, m_hanoune@yahoo.fr

Résumé

Dans le présent papier, nous proposons un environnement informatique interactif à des fins d'apprentissage et d'autoévaluation de l'algorithmique. L'apprenant pourra non seulement acquérir des connaissances en algorithmique, mais aussi en faire l'expérience. L'utilisation des nouvelles technologies d'information n'est pas en elle-même l'objectif, néanmoins c'est un support sur lequel on peut se baser afin de proposer une nouvelle approche méthodologique dans l'enseignement. Notre approche est basée sur l'usage des exemples et de l'expérience. Elle met l'accent sur l'activité du sujet afin d atteindre les objectifs pédagogiques escomptés. Plusieurs scénarios sont possibles selon les pré-requis et le rythme de chaque apprenant. Plusieurs types d'exercices sont prévus pour sensibiliser l'apprenant et lui permettre de développer des modèles abstraits de données, de tâches et leurs séquences temporelles dans un algorithme. L'environnement met à la disposition de l'apprenant des évaluations et lui propose des recommandations qui varient selon le type d'erreurs. La gestion du profil de l'apprenant est une partie intégrante de cet environnement.

Mots clés : Apprentissage par l'exemple, Didactique de l'algorithmique, Autoévaluation, Profile d'apprenant, Environnement Informatique.

Summary

In this paper, we propose an interactive data-processing environment for algorithmic training and autoevaluations. Learner will acquire knowledge on algorithmic, but also experiment it. The use of new technologies of information is not the gaol in it self, nevertheless it's could be a rich support on which we can base a new methodological approach in teaching. Our approach is based on the use of the examples and their experimentation. We focus on the activity of the subject to achieve the

teaching goals discounted. Several scenarios are possible according to the knowledge and rate of learner. Several types of exercises are designed to sensitize learner, and enable him to develop easily abstract models of data, temporal tasks and their sequences in an algorithm. The environment proposes evaluations for learner and gives him recommendations which change according to the type of errors. The management of the profile of learning is an integral part of this environment.
Keywords: Training with Examples, Didactic algorithmic, Autoevaluation, Learner profiles, Data-processing Environnement.

1. Introduction

Le développement des Technologies d'Information et de Communication (TIC) a remis l'amélioration de la qualité de l'enseignement et de l'apprentissage à l'ordre du jour (Amerein, Proquin et al., 1998). La question qui se pose est de savoir comment tirer profit des facilités apportées par les TICs pour répondre aux interrogations du genre : « Comment enseigner ?» et «Comment apprendre ?». L'apport de la technologie joue un rôle très important comme le souligne Debray (2001) : «Si les technologies ne permettent pas d'accomplir le processus de transmission de connaissances dans son intégralité, elles en facilitent certains mécanismes» (p. 17-33).

Les TICs s'imposent de plus en plus comme support incontournable dans l'apprentissage des différents savoirs et comme moyen d'exécution pratique en même temps. Il est sûr que l'amélioration de la qualité des enseignements passe par le développement de nouvelles méthodologies didactiques et pédagogique, mais aussi par la remise en question des rôles respectifs de l'enseignant et l'étudiant. L'objectif de ce travail est de réaliser un support de cours interactif pour l'initiation à l'algorithmique pour débutants en informatique. Nous nous appuyons sur un environnement informatique pour mettre en place une pédagogie basée sur l'expérience et sur les exemples corrigés et commentés (Cyphe, Halbert et al., 1993) (Garner, 2001) (Kerjean, 2006). Le modèle conventionnel de l'instruction dans plusieurs domaines implique la présentation d'un principe, d'un concept, ou d'une règle, suivie de la pratique étendue sur des problèmes appliquant la règle. Dans le domaine de l'algèbre, Sweller et Cooper (1985) confirment que les étudiants apprennent mieux quand ils travaillent assez des « exemples » élaborés, plutôt que de résoudre des problèmes dès qu'ils se familiarisent avec les nouveaux concepts. Cette même méthode pédagogique a été utilisée pour l'enseignement de la physique par Ringenberg et Kurt (2006), mais ces derniers suggèrent que celle-ci n'est pas suffisante, dans la mesure où les apprenants peuvent ne pas déduire les modèles de réflexion, modèles que l'on désire leur transmettre à travers l'étude de ces exemples, et qu'en revanche il faut réfléchir sérieusement à des méthodes pour obliger les apprenants à étudier ces exemples correctement. Grosse et Renkl (2004) proposent d'inclure les

erreurs dans les exemples. Dans cette étude on a constaté que proposer des solutions avec des erreurs peut stimuler l'exécution du transfert des connaissances surtout pour les « bons » étudiants.
L'environnement qu'on propose s'inspire de ces travaux et préconise trois phases dans le scénario d'apprentissage : lecture des concepts et étude des exemples, entraînement, et autoévaluation. L'apprenant commence par lire les concepts de base et traiter les exemples. L'entraînement permet à l'apprenant de mesurer son degré d'apprentissage des concepts abstraits et leur mise en pratique pour rédiger des solutions inspirées de modèles tirés (par analogie) des exemples étudiés. Dans cette phase, l'apprenant dispose aussi du soutien de l'environnement puisque les exercices comportent des aides progressives qui l'acheminent vers la solution proposée. L'autoévaluation reste un moyen pour confirmer l'évolution de l'apprenant dans son parcours pédagogique et passer d'une unité à une autre.

2. Principales Difficultés

L'algorithmique est une discipline longtemps utilisée de manière naïve comme le souligne Caignaert (1988), sans formalisme particulier. L'algorithmique permet d'organiser les idées du programmeur, de les représenter avec un formalisme qui peut être facilement traduit dans un langage compréhensible par l'ordinateur. Cette discipline est souvent source de problème pour l'enseignant ainsi que pour l'étudiant. L'enseignant parce qu'il doit trouver les méthodes adéquates pour faire assimiler des concepts assez abstraits à des étudiants qui ne sont qu'à leur phase d'initiation. Kaasböll (2002) confirme que le taux d'échec ou d'abandon aux cours d'initiation à la programmation en premier cycle universitaire varient de 25 à 80% de part le monde. Nous soutenons le principe qui dit « pour écrire un bon programme, il faut écrire un bon algorithme » puisque l'algorithmique est à la base de la programmation. Si les étudiants échouent dans la programmation c'est qu'ils n'écrivent pas de bons algorithmiques. Ce qui nous amène à la question suivante : « Quelles méthodes pédagogiques et avec quels outils peut-on améliorer l'apprentissage de l'algorithmique ?». Nous soutenons qu'un usage approprié des TICs avec des méthodes pédagogiques innovatrices comme l'apprentissage par l'exemple et l'expérience permettra d'atteindre nos objectifs pédagogiques dans ce domaine.

La rédaction d'un programme passe par plusieurs étapes (DuChâteau, 2002) : Allant de « *quoi faire ? », à « comment faire faire ? »*.

Nous avons regroupé ces étapes en 3 phases : Analyse, conception et réalisation. La relation entre la phase analyse et conception n'est pas univoque, puisqu'à un certain moment on peut choisir une stratégie de résolution, puis revenir et opter pour une autre. Dans la phase d'analyse l'étudiant n'a pas besoin de connaissances algorithmiques, puisque la solution est puisée depuis divers savoir

(mathématiques, chimique,…). La phase réalisation va servir à traduire ce modèle dans un langage de programmation, le tester et le valider.

La phase de conception est plus difficile : à ce niveau l'étudiant doit savoir traduire sa stratégie en modèle représentant une séquence de tâches à exécuter dans le temps. L'étudiant doit prendre conscience qu'il est entrain de concevoir une solution qu'il va soumettre à une machine virtuelle, qui n'a aucune intelligence, pour l'exécuter, et c'est là que se pose la plupart des obstacles. On relève trois types de difficultés et en même temps des objectifs à atteindre, dans l'apprentissage de l'algorithmique (Guibert, Guittet et al., 2005) (Du Boulay, 1989) :

a. Modélisation de la tâche : maîtrise du comportement de la tâche

b. Modélisation de l'exécution : élaboration d'un modèle mental du déroulement temporel de la tâche et son association à l'état de la machine

c. Modélisation des données : association d'une représentation abstraite des objets de la tâche

Pour répondre à ces objectifs, il faut tout d'abord que les apprenants acquièrent les concepts théoriques (variable, condition, itération,..), et cela en étudiant une série d'exemples d'algorithmes pour chaque concept (Hilbert, Schworm et al., 2004). Les solutions doivent être soigneusement rédigées pour initier l'apprenant à une méthode d'analyse des problèmes (données d'entrées, données de sortie, les constantes, les variables, les traitements, ...). À travers les algorithmes exemples, on s'attend à ce que les apprenants comprennent les solutions, et puissent expliquer et identifier le but de chaque échantillon d'instruction utilisée dans chaque étape de l'algorithme. L'environnement doit fournir des aides progressives aux étudiants qui ont le plus de difficultés. Les étudiants résolvent des problèmes semblables, les répètent jusqu'à ce qu'ils soient résolus sans erreurs. Prenons l'exemple du calcul du factoriel d'un entier N :

Si l'apprenant comprend le concept de répétition de l'opération *fact=fact*i*, et ses retombées sur la variable *fact*, il peut aussi résoudre les problèmes du même type comme le calcul de N^m (N exposant m) ou encore N+(N-1)+...+1. Il s'agit d'aider les apprenants à construire une base de connaissance qu'ils peuvent réutiliser dans d'autres problèmes. Le rôle de l'expérimentation, est de montrer l'évolution des variables, de suivre le déroulement des instructions, mais surtout d'élaborer un modèle mental de l'exécution de l'algorithme par une machine virtuelle.

Dans ce contexte nous citons Allogène (Fournier et Wirz, 1992) qui est un système d'apprentissage par l'exemple. L'association du déroulement graphique de l'algorithme (à travers les variables) est intéressante. MELBA (Metaphor-based Environment to Learn the Basics of Algorithmics) (Guibert, Guittet et al., 2005) propose des métaphores pour expliciter le concept de variable, de type, de paramètre et de référence.

Une autre approche est proposée par Duchâteau (2002) : Cette méthode d'apprentissage des bases de la programmation était révolutionnaire; son principe de base est de fournir à un débutant une représentation simple et imagée de la manière dont cette boîte noire, que représente l'exécutant ordinateur, fonctionne et qu'il puisse s'appuyer sur cette « vision » mentale pour concevoir ses programmes. La Méthode CAR (Meurice, 1997) n'apporte fondamentalement rien de neuf par rapport à « Images pour programmer ». Les concepts, images et analogies sont généralement conservés, mais l'adoption d'un support interactif permet toutefois l'apport d'éléments supplémentaires tels que les animations, les auto-tests, etc.

L'environnement interactif que nous proposons, répond aux objectifs que nous avons cités au départ mais aussi à des contraintes liées à l'apprentissage lui même dans un environnement informatiques et surtout à l'absence du tuteur.

3. Contraintes d'un auto apprentissage

Un environnement informatique d'apprentissage doit combler la perte d'interaction humaine dans ce genre d'enseignement. Pour cela nous sommes intéressés spécialement à trois contraintes principales liées aux rôles attribués au tuteur et à l'apprenant.

3.1 Adaptation de l'apprentissage

Certes notre système d'apprentissage possède un scénario pédagogique principal, mais il offre une certaine souplesse vis-à-vis de l'exécution de ce scénario. Le processus d'apprentissage est adapté selon le pré-requis de l'apprenant moyennant une autoévaluation qu'il doit passer. Les apprenants qui préfèrent la découverte et l'expérimentation pourront exercer les différents exemples et les exécuter avec différentes valeurs. Le rôle du système est de guider l'apprenant, et l'aiguiller vers les concepts du cours qui ne sont pas acquises complètement pour renforcer son apprentissage.

3.2 Profil de l'apprenant

L'apprenant en s'auto-formant est confronté à lui-même, il doit pouvoir s'approprier progressivement son profil d'apprentissage. Il doit devenir le professeur qui analyse et interprète son processus d'apprentissage pour en découvrir progressivement toutes les implications. Par ce fait le système doit récolter et trier les informations liées au déroulement de l'apprentissage puis les mettre à la disposition de l'apprenant pour qu'il puisse les exploiter (Jean-Daubias, 2003).

En particulier, il est important que l'environnement permette à l'apprenant de prendre conscience de ses faiblesses lors des évaluations (Bourdet et Teutsch, 2000) en terme de connaissances. Dans l'environnement EasyAlgo nous nous sommes intéressés à trois types de profils :

- Connexion : Cet élément comporte la date des différentes connexions notamment de début et de fin de connexion.
- Entraînement : Cet élément comporte un identifiant de l'exercice, une date et son état : fait ou visité (vu mais pas réalisé).
- Autoévaluation : Cet élément comporte le résultat des évaluations avec les dates correspondantes.

3.3 Autoévaluation

L'autoévaluation permet à l'apprenant de connaître le niveau d'acquisition des connaissances, méthodes et sa capacité à résoudre un problème donné tout seul sans l'aide de l'environnement. C'est une opération qui s'avère compliquée dans le cas de l'algorithmique puisqu'un problème peut être résolu de différentes manières, et en utilisant des méthodes différentes. Une première solution est d'intégrer une sorte de compilateur algorithmique dans l'environnement, ainsi l'apprenant peut proposer une solution, la vérifier syntaxiquement parlant, puis la tester puis valider. L'apprenant transmet alors les résultats de l'exécution au système qui les compare avec les résultats attendus. Dans un apprentissage autonome, on ne peut envisager d'énumérer toutes les solutions possibles pour un problème donné, mais le résultat doit être le même, et peut être la base d'une évaluation. Dans notre approche, l'évaluation se fait de deux façons selon la phase d'apprentissage de l'apprenant. Dans une première étape on demande à l'apprenant de deviner une solution particulière, celle proposée par l'environnement, et cela de différentes manières :

a. En complétant l'algorithme : On fournit la solution mais avec des vides que l'apprenant doit compléter.
b. En fournissant l'algorithme et en demandant à l'apprenant de dire ce qu'il fait.
c. Une troisième alternative serait de donner l'algorithme mais en désordre et l'apprenant doit tout remettre en place.
d. Ou encore de donner la solution avec des erreurs à corriger

Le but étant d'obliger l'apprenant à passer plus de temps à réfléchir à la conception de l'algorithme plutôt que de passer directement vers la validation et test de l'algorithme, ce qui implique beaucoup de temps perdu à détecter les erreurs.

Dans une seconde étape l'apprenant pourra écrire son propre algorithme et le tester dans l'environnement.

4. Concepts et méthodes de représentation

Dans la plupart des support de cours que nous avons consultés, les méthodes les plus utilisées sont des langages naturels malheureusement non normalisés comme les **LDAs** (Langage de Description des Algorithmes), les graphes et les schémas (Organigrammes,…). Les diagrammes de Conway sont aussi utilisés, mais plus pour l'apprentissage de la syntaxe d'un langage de programmation qu'en algorithmique. Deux approches émanent de ces différentes méthodes :

a. l'une orientée vers le langage et la syntaxe : LDA

b. l'autre orientée vers la représentation schématique : Organigrammes.

Le formalisme par les LDA est important, puisque c'est un premier pas vers la programmation. La schématisation par organigramme (Saie et Barrand, 1987) offre la possibilité à l'étudiant de voir sa solution dans un repère spatio-temporel (visuel), ce qui lui permet d'assimiler rapidement sa mise en oeuvre et en facilite la vérification. Nous pensons que la combinaison des deux approches LDA et schématisation pourrait faciliter la synthèse de l'information et favoriser l'intégration des connaissances et leur transfert.

5. Scénario d'apprentissage

Le scénario de base que nous avons adopté est représenté dans la figure suivante (figure1). L'apprenant lit les concepts théoriques, étape dont on ne peut se passer, s'entraîne volontairement avec les exemples qui sont bien commentés, et dont la rédaction a été faite soigneusement pour l'initier à analyser correctement un problème. L'objectif des exercices est que l'apprenant soit capable de construire ses propres algorithmes, et qu'il prenne assez de recul pour compléter une autre solution qui n'est pas forcément celle à la quelle il a pensé au départ. Cela lui permettra de construire sa propre base de connaissance, d'atouts et d'astuces pour répondre à un problème quelconque.

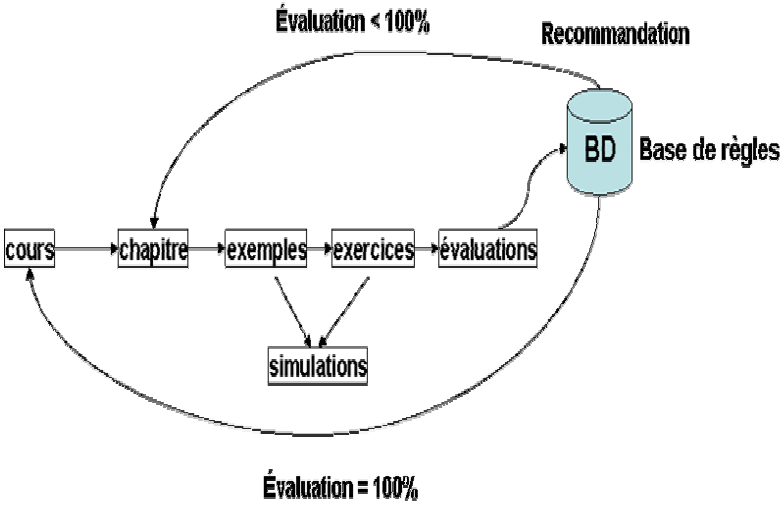

Figure 1 : Scénario pédagogique de base

Ce scénario n'est pas statique ; pour un apprenant qui possède déjà les concepts, EasyAlgo peut être un environnement d'entraînement et d'autoévaluation. Mais le passage d'un chapitre à un autre doit se faire forcément en passant une autoévaluation à 100% de réussite ou un pourcentage qui s'en approche.

La modélisation de l'environnement a été réalisée avec le langage UML (Rumbaugh, Jacobson et al., 1998) qui est un langage de développement objet reconnu et de plus en plus utilisé. L'environnement EasyAlgo propose à l'apprenant de faire un ensemble d'activités comme le montre la figure 2.

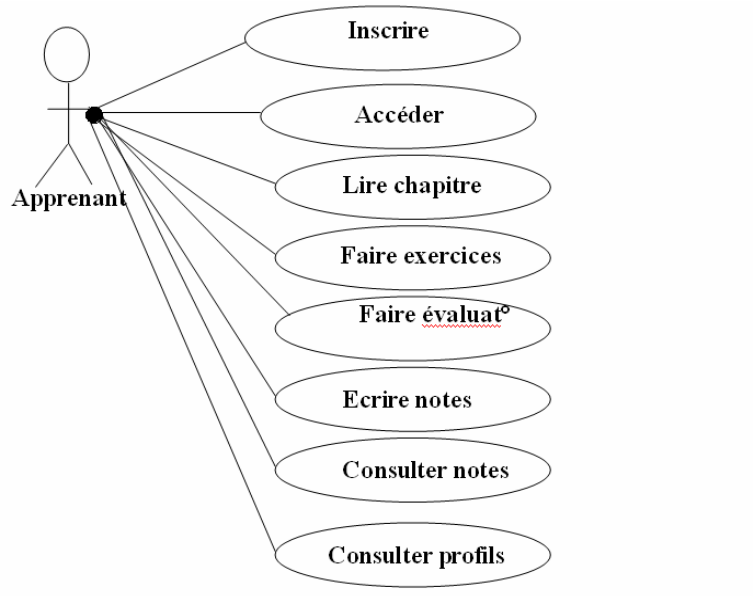

Figure 2 : DCU de EasyAlgo

L'apprenant peut consulter le cours à travers les différents chapitres qui le composent. Il peut faire des exercices. À chaque exercice on associe un type, des aides, une simulation et une ou plusieurs solutions. L'apprenant est invité à faire des autoévaluations pour valider le passage entre les unités. L'apprenant peut écrire des remarques ou synthèses sous forme de notes associés à chaque chapitre pour les utiliser par la suite. Les principales activités de l'apprenant :

a. Consultation du cours, exemples...
b. Entraînement avec des exercices
c. Autoévaluation
d. Consultation du profil : Connexion, entraînement, évaluation.
e. Edition, consultation d'annotations sous forme de remarques ou résumés.

Le rôle de l'administrateur qui est l'équipe pédagogique est de gérer la base de données, d'apporter des modifications concernant l'évolution des contenus pédagogiques pour leur fond comme pour leur forme dans l'environnement.

6. Mise en oeuvre

L'environnement d'apprentissage EasyAlgo est réalisé avec des langages et des outils open-source comme PHP et MySQL. L'environnement permet à l'apprenant d'élaborer ses connaissance à travers les exemples et les expériences auxquelles il doit participer activement. Manipuler les exemples, les tester avec différentes valeurs permettra à l'apprenant de construire un modèle mental de l'exécution de la solution proposée ce qui est un des objectifs primordiaux de l'apprentissage de l'algorithmique. La manière de représenter un algorithme est aussi un aspect important puisque certains apprenants préfèrent représenter leurs modèles avec des schémas alors que d'autres utilisent un LDA. Ainsi, pour certains exemples on propose sa représentation en LDA et sa schématisation sous forme d'un organigramme.

6.1 Scénario d'apprentissage

L'apprentissage débute par des notions de base sur les ordinateurs et le codage (figure 3). Le passage d'une unité à l'autre se fait par validation d'un test avec un taux de réussite égal à 100%. L'apprenant peut refaire le test plusieurs fois s'il le désire. Le taux de réussite est un paramètre qu'on peut modifier pour rendre l'apprentissage plus souple.

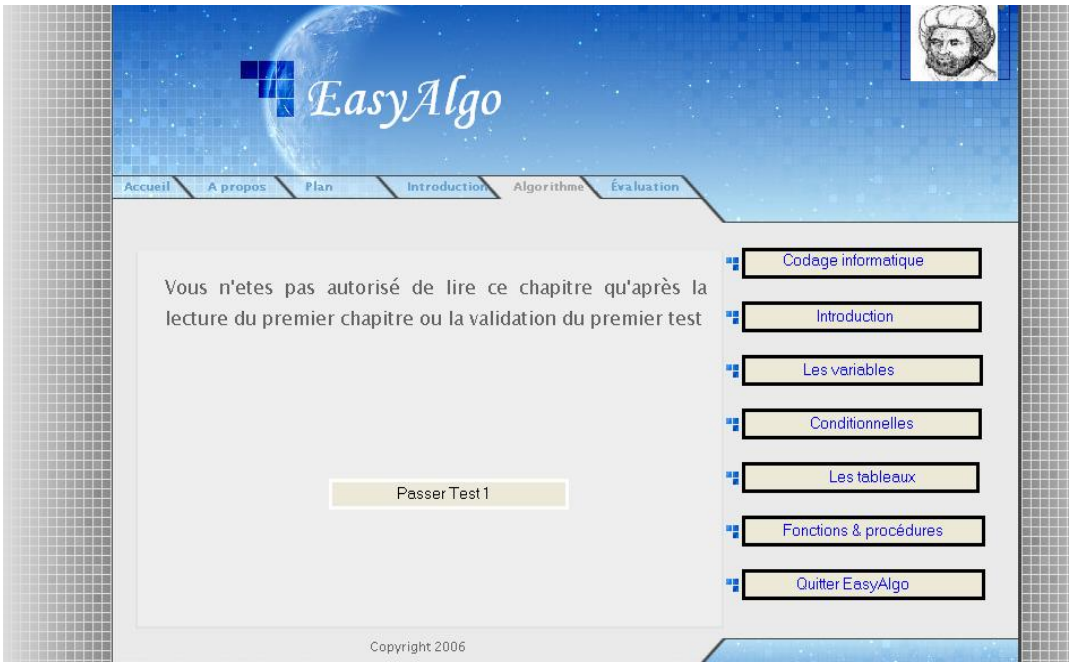

Figure 3 : Validation du scénario par test

6.2 Entraînement

L'environnement propose à l'apprenant de renforcer son savoir-faire à l'aide d'entraînement au moyen d'exercices de types différents :

a) À Aide : on fournit des indices sous forme d'aides pour orienter l'apprenant vers une solution donnée (figure 4. a). Si la première aide (qui peut porter sur la compréhension du problème lui même) n'est pas suffisante, une deuxième est proposée (figure 4. b)

b) À trou : L'algorithme solution est fournit mais avec des vides à remplir par l'apprenant

c) À séquence : L'algorithme solution est fournit mais dans une fausse séquence, l'apprenant doit séquencer les instructions de l'algorithme. Ce qui permet à l'apprenant de d'acquérir une approche pour réorganiser en séquence les instructions.

Il faut remarquer que dans ce cas l'apprenant est orienté vers une solution particulière, mais l'environnement lui propose de voir d'autres solutions pour enrichir ses techniques de résolution.

Figure 4.a : Exemple d'exercice avec des aides

Figure 4.b : Exemple d'exercice avec des aides

L'entraînement se termine par une autocorrection : l'apprenant essaye de répondre à l'exercice, et s'il n'y arrive pas la solution est dévoilée progressivement (avec suggestion d'autres solutions possibles). En comparant son travail avec la solution complète, l'apprenant essaye de localiser et de corriger ses erreurs. Découverte des erreurs peuvent augmenter la réflexion, et pousser l'apprenant à faire plus d'efforts pour trouver les bonnes explications.

Pour l'apprenant la phase d'entraînement peut être aussi une phase d'autoévaluation où il peut disposer de l'aide de l'environnement. Cette phase a aussi pour objectif de transmettre à l'apprenant des méthodes pour répondre à un problème donné (définir les données d'entrée... de sortie, définir les traitements,...).

Lorsque l'étape de réalisation de l'algorithme se termine, l'apprenant peut simuler l'algorithme solution proposé et le tester avec différentes valeurs. La figure 5 représente un exercice sur la conversion d'un entier en nombre d'heures, minutes et secondes.

Figure 5 : Exemple de simulation

6.3 Autoévaluation

Dans l'exemple de l'évaluation de la figure 6, l'apprenant doit compléter les parties vides de l'algorithme (le but étant de calculer 1/1!+1/2 !+1/3 !+….1/N !) et soumettre ses réponses au système.

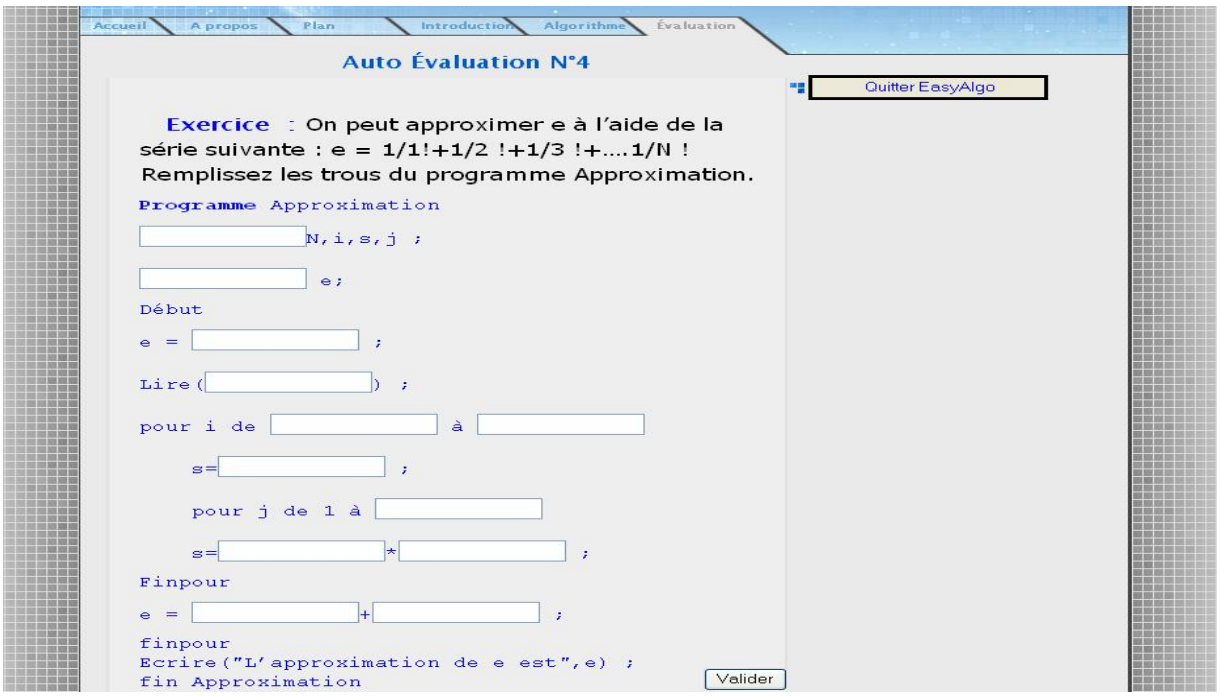

Figure 6 : Exemple d'Evaluation avec des trous

Dans les premiers chapitres notre soucis est que l'apprenant arrive à construire des briques de l'algorithme proposé, mais en seconde phase l'apprenant doit rédiger un algorithme lui même. Nous avons donc prévu d'intégrer un compilateur d'algorithmes qui permettra à l'apprenant de vérifier ses solutions et de les corriger, mais on lui proposera aussi l'une des meilleures solutions pour faire une comparaison avec son approche.

Le système analyse les valeurs proposées par l'apprenant, et propose des recommandations à l'apprenant selon le type d'erreurs. Dans le cas où il n'y a pas d'erreur, le système suppose que l'exercice est validé. Sinon, il y a deux possibilités : l'exercice a été réalisé partiellement ou pas du tout. Dans le premier cas nous donnons un pourcentage de réussite de l'exercice, sinon nous indiquons les recommandations nécessaires. Les recommandations sont les actions d'un ensemble de condition par exemple : « Si l'apprenant commet une erreur dans la partie X, du chapitre Y alors afficher recommandation XY ». Les recommandations peuvent être le renvoi vers la partie du cours concernée ou l'incitation à retraiter un ensemble d'exemples.

Conclusion

Notre réflexion pédagogique propose un environnement d'apprentissage et d'autoévaluation dans le but d'initier les étudiants dans le domaine de l'algorithmique. Nous avons mis en oeuvre une méthode basée sur les exemples et l'expérience. Le scénario pédagogique permet de cerner les différents problèmes rencontrés et d'inculquer dans la mémoire de l'apprenant un modèle abstrait

de l'exécution de ses algorithmes sur un ordinateur. L'objectif étant de développer les capacités d'abstraction et de logique chez les apprenants. Nous sommes entrain de tester l'environnement avec un groupe d'étudiants de première année, ce qui nous donnera la possibilité de tirer les conclusions nécessaires par rapport à l'avantage d'utilisation d'un tel outil et permettra de justifier l'approche et la méthodologie que nous avons adoptées. Le recueil des informations sur les profils des apprenants nous permettra de constituer une base de connaissance sur leurs activités et leur styles d'apprentissages et donc de faciliter le processus de re-ingénierie de EasyAlgo pour l'adapter au mieux aux profils des apprenants.

Références

Amerein, S. B., Proquin, M., Renaud, C., & Trigano, P. (1998). De la réciprocité éducative dans le cadre d'une nouvelle pédagogie de l'enseignement supérieur : un didacticiel au service de l'informatique fondamentale. *Acte de colloque NTICF'98*. Rouen : 18-20 Novembre.

Bourdet, J.-F., Teutsch, P. (2000). Définition d'un profil d'apprenant en situation d'autoévaluation. *Revue Alsic. , 3(1),* 125-136.

Caignaert, C. (1988). Étude de l'évolution des méthodes d'apprentissage et de programmation. *Bulletin de L'EPI, 50*, 52-60.

Cyphe, A., Halbert, D. C., Kurlander D., Lieberman, H., Maulsby, D., Myers, B. A., & Turransky, A. (1993). Watch What I Do: Programming by Demonstration. *The MIT Press*, 49-66.

Debray, R., (2001). Malaise dans la transmission. *Les Cahiers de Médiologie*, *11*, 17-33.

Du Boulay, B. (1989). Some Difficulties of Learning to Program. S*tudying the Novice Programmer*, Lawrence Erlbaum Associtates, 283-299.

DuChâteau C. (2002). Images pour programmer. *Publication de CeFIS*, Namur, Facultés Universitaires Notre Dame de la Paix.

Fournier, J.-P., & Wirz, J. (1992). ALLOGENE : Un environnement d'apprentissage de l'algorithmique. *Actes de la 3ème rencontre francophone sur la didactique de l'informatique EPI, AFDI, IUKB*, 101-113.

Garner, S. (2001). *A Tool to Support the Use of Part-Complete Solutions in the Learning of Programming. Proceeding de conférence* Informing Science IS'2001, 222-228.

Grosse, C., S., & Renkl A., A. (2004). Learning from worked examples: what happens if errors are included? *Proceedings de conférence EARLI SIGs*, Genève: 20-22 Septembre, 356-364.

Guibert, N., Guittet, L., & Girard, P. (2005). Initiation à la Programmation « par l'exemple » : concepts, environnement, et étude d'utilité. *Acte de colloque EIAH'05*, Montpellier : 25-27 Mai, 461-466.

Hilbert, T. S., Schworm, S., & Renkel, A. (2004). Learning from worked-out examples: The transition from instructional explanations to self-explanation prompts. *Proceedingsde conférence EARLI SIGs*, Genève: 20-22 Septembre, 184-192.

Jean-Daubias, S. (2003). Exploitation de profils d'apprenants. *Acte de la conférence* EIAH'03, Strasbourg : 15-17 avril, 535-538.

Kerjean, A. (2006), *L'apprentissage par l'expérience*. Paris: ESF. (coll. Formation Permanente).

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

Meurice de Domale R. (1997). *Robotique virtuelle : un environnement pour un apprentissage dynamique de l'algorithmique*. Rapport de CeFIS Namur, Facultés Universitaires Notre Dame de la Paix.

Ringenberg, M., Kurt, V. (2006). Scaffolding Problem Solving with Annotated Worked-Out Examples to Promote Deep Learning. K. Ashley & M. Ikeda (Eds.), *Intelligent tutoring systems*: 8th international conference, ITS'2006, IOS Press.

Rumbaugh, J., Jacobson, I., & Booch, G. (1998). *The Unified Modelling Language Reference Manual*. Addison-Wesley.

Saie, P., & Barrand, F. (1987). Micado : Méthode d'Initiation à la Compréhension Décrits par des Organigrammes. *Bulletin de l'EPI, (46)*. 178-183.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2(1),* 59-89.