Approche cognitive et téléologique de la programmation

Yannick Brilhault, Marcel Duboué
Département d'informatique
Université de Pau et des pays de l'Adour
Avenue de l'université
64000 PAU - FRANCE

1 - INTRODUCTION ET TRAVAUX ANTÉRIEURS

L'enseignement post-baccalauréat de l'informatique concerne trois catégories de formations : les filières informatiques, les filières scientifiques, et les filières non scientifiques. Pour chacune de ces trois catégories la motivation de l'étudiant est très différente. Les futurs professionnels se sentent impliqués dans une discipline centrale pour leur future activité, ce qui n'est pas le cas pour les deux autres catégories. De plus la majorité des étudiants non scientifiques manifeste une crainte pour cette activité qui leur rappelle leurs échecs dans les matières réputées logiques. Nous centrons notre étude sur ces étudiants des filières d'économie, de gestion et d'administration, pour lesquels il est essentiel de trouver une approche spécifique qui s'adapte à leurs particularités.

Le but que nous nous fixons pour ces filières est de parvenir à rendre intelligible certains aspects de la réalité informatique en donnant une idée simplifiée mais pas trop réductrice de la plupart des applications informatiques auxquelles est aujourd'hui confronté tout citoyen. Notre attitude est donc très pragmatique, nous essayons au mieux de nous adapter à la réalité de nos étudiants (démarche téléologique) plutôt que de les contraindre à suivre une voie classique, qu'ils refusent.

Pour cela nous cherchons donc à donner de la matière à l'étudiant, afin qu'il se construise un système personnel de représentation et de traitement de l'informatique de gestion qui lui permette de s'adapter à de nouvelles situations, de les analyser et de les comprendre. cet objectif est clairement du domaine du cognitif car il ne suffit pas d'utiliser les représentations ou shémas que possède déjà l'étudiant, mais de les enrichir afin qu'ils englobent les réalités informatiques. C'est pourquoi une approche uniquement utilitaire,
l' informatique outil, nous semble inappropriée pour construire ou échafauder [Ausu 68] des représentations.

Afin de fixer les idées voici une liste non exhaustive des applications auxquelles nous faisons référence :
- gestion des abonnements pour une revue
- gestion de stock facturation dans une entreprise de vente par correspondance,
- réapprovisionnement automatique dans supermarché (code barre),
- gestion de bibliothèque, de vidéothèque,
- gestion d'une amicale d'anciens élèves,
- facturation aux abonnés des services de distribution d'eau, d'électricité, de télécommunication,
- système de réservation de compagnie aérienne ou ferroviaire, bornes interactives.
- utilisation de bases de données et de banques de données de serveurs télématiques,

...
De cette intelligence des applications informatiques nous attendons que les étudiants soient des interlocuteurs plus pertinents et plus avisés des professionnels de l'informatique, conscients des possibilités et des limites de l'informatisation, voire de ses dangers.

La question de la programmation

Si l'apprentissage des outils (sgbd, tableur, traitement de texte, ou logiciel intégré) dans le curriculum des étudiants de **filières économiques de gestion et d'administration** recueille un large consensus, la question de savoir si l'on doit leur apprendre la programmation(laquelle ?), reste par contre ouverte. On constate toutefois une diminution et parfois une disparition de l'apprentissage de la programmation dans ces filières. La plupart du temps, les enseignants y ont été conduits plus par des résultats décevants que par un choix pédagogique délibéré.

Malgré nos premières experiences, elles aussi négatives, nous avons continué à penser que les objectifs précédemment exposés peuvent et doivent être atteints par l'apprentissage d'un certain nombre de concepts essentiels pour la compréhension des mécanismes impliqués dans l'informatisation : la codification du réel, la problématique du faire faire, l'interaction homme-machine.

Nous avons montré dans [Bril 91] que les étudiants de filières économie, gestion ou administration peuvent apprendre un ensemble de concepts de la programmation pertinents pour nos objectifs, s'ils sont abordés selon leurs cadres de référence et leurs motivations et si l'enseignant n'a pas un souci de rigueur incompatible avec le niveau de pensée de l'étudiant. Nous avons défini pour cela un curriculum court (60-70 heures, voir annexe 1), nous avons présenté son évaluation sur plusieurs promotions dans [Bril 92] : les résultats sont très encourageants.
Nous tenons à souligner que l'objectif de cet apprentissage n'étant pas disciplinaire, cela permet de ne pas se focaliser sur des aspects méthodologiques et algorithmiques qui seraient des entraves à la réalisation de notre objectif.

2 ASPECTS TELEOLOGIQUES

Notre préoccupation essentielle a donc été d'introduire des significations à toutes les activités. Différents moyens ont été utilisés : "habiller" les problèmes posés, définition d'un curriculum centré sur la notion de fichier, réalisation d'une interface plus adaptée à des débutants que celle de Turbo-Pascal, réalisation de projets de gestion.

Habillage de la problématique informatique

L'expérience nous a amené à constater que les étudiants sont souvent pertubés par les aspects ésotériques ou même formels des activités proposées en programmation. Les exercices choisis leur semblent n'avoir aucun sens, soit parce qu'ils font référence à des réalités qu'ils ignorent, soit parce que ce sont des exercices d'école.

Afin d'illustrer notre propos, examinons un exercice très formateur que l'on trouve fréquemment dans les scénarii d'apprentissage sous cette forme (ou sous une forme voisine) :
Ecrire un programme qui calcule le maximum de 3 nombres.

Devant cet exercice, beaucoup d'étudiants ne voient pas le problème. Leur réaction est, soit " je rentre 3 nombres et l'ordinateur me dira quel est le plus grand !!! (ils ne voient pas ce qu'il faut faire), soit "je n'ai pas besoin d'un ordinateur pour me dire quel est le plus grand des nombres que j'ai entré (ils ne voient pas l'intérêt). La version "plus habillée", *Ecrire un programme qui affiche le gagnant et son nombre de voix parmi 3 candidats à une élection (les noms des candidats et les nombres de voix obtenues seront entrés au clavier), se révèle beaucoup mieux comprise*.

On peut même compliquer la difficulté informatique sans problème supplémentaire de compréhension avec cette formulation : *Ecrire un programme qui affiche par **ordre** décroissant les pourcentages de voix obtenues par **chacun** des 3 candidats à une élection (les noms des candidats et les nombres de voix obtenues seront entrés au clavier).*

Ces nouvelles formulations, plus satisfaisante à l'expérimentation, nous montrent la nécessité d'habiller les exercices canoniques de manière à ce que la finalité soit aisément compréhensible par l'apprenant. En outre, le sens contenu dans les exercices permet de cadrer la tâche à exécuter, et cela à un stade de son apprentissage ou l'apprenant manque sérieusement de repères.

Pour réaliser ces "habillages " nous avons besoin d'introduire très tôt les chaînes de caractères comme dans l'exemple ci-dessus, et surtout la notion de fichier.

Le fichier est au centre du curriculum

Le curriculum (annexe 1) a la particularité de présenter très rapidement les structures d'enregistrements et de fichiers (alors que généralement les tableaux sont présentés avant les fichiers), ce qui permet de traiter rapidement des problèmes de gestion significatifs . Cela nous permet de rester dans leur domaine d'étude au contact de préoccupations proche de leurs pratiques sociales de référence (métiers de cadres, gestionnaires ou d'administrateurs).

Au cours de nombreuses années d'enseignement dans ces filières économiques, nous avons constaté des améliorations quantitatives des résultats lorsque nous avons adopté une démarche qui partait du monde réel vers l'informatique [Bril 91]. Afin de préciser notre pensée, nous allons exposer notre manière de présenter les notions d'enregistrement et de fichier à ces étudiants. Nous prenons comme exemple la gestion d'une université dans une époque pré-informatique. Nous présentons (en nous appuyant sur la participation des étudiants) la démarche qui conduit, en fonction des renseignements qu'il faudra fournir et des traitements qu'il faudra effectuer, à définir un modèle de fiche et par conséquent cet ensemble de fiches que l'on appelle un fichier. Ensuite toujours sans aucune référence à l'informatique, nous montrons la nécessité des traitements fondamentaux que sont la création, les consultations et les mises à jour (constituées d'ajouts, de modifications et de suppressions). La reprise de ces notions dans un contexte de programmation apparaît alors comme naturelle et finalisée dans le monde réel. La finalité étant préalablement exposée, les notions informatiques correspondantes dans le langage de programmation apparaissent assez naturellement et posent moins de problèmes aux étudiants.
Enfin il est possible de s'appuyer sur certaines analogies pertinentes avec le fichier cartonné pour les aider à trouver les algorithmes de certains programmes.

Réalisation d'un projet informatique

La dynamique et la motivation induites par les projets d'applications est largement reconnue par la communauté éducative, l'assiduité et l'intérêt manifestés par les étudiants le confirment. On peut aussi noter la valeur qu'ils attribuent à ce travail par le fait qu'ils veulent récupérer leurs dossiers et leurs programmes après la soutenance. Comme pour le choix des exercices, le projet correspond aussi au domaine d'étude du futur gestionnaire ou administrateur.

De plus, ces projets sont l'occasion d'aborder une autre dimension des applications informatiques: le cycle de vie des logiciels depuis les phases d'analyse, jusqu'à l'exploitation et la maintenance.

Utilisation d'une interface plus adaptée à des débutants

Nous avons constaté en analysant des tests que les étudiants ne font pas le lien entre les concepts de contrôle et leurs réalisations dans les langages de programmation. Par exemple, à la question : *citez les instructions conditionnelles que vous connaissez,* les deux tiers des étudiants citent dans les réponses une ou deux instructions d'itération. Bien que cette confusion s'explique en partie par le fait qu'il y ait dans les instructions itératives une "partie conditionnelle" qui sert à arrêter l'itération, et aussi parce que les mots choisis par les concepteurs des langages de programmation ne sont pas univoques, il n'en demeure pas moins qu'il y a confusion dans leurs esprits et que les concepts et/ou les mots ne sont pas ou mal maîtrisés. Citons à ce propos un auteur anglophone *:* " *Such words have often come to have a very specialized meaning within computing, which is far removed from their every day connotation* " [Dubo 89], qui nous indique que cela pose aussi des problèmes aux débutants anglophones.

La problématique de l'informatique étant préalablement exposée, à savoir qu'elle est du domaine du "faire faire", nous pouvons analyser des modes d'emploi ou des notices d'utilisation de la vie courante (comme Duchateau [Duch 90]), pour mettre en évidence les concepts essentiels et incontournables que sont par exemple l'alternative simple ou l'alternative multiple. L'alternative multiple peut être abstraite de phrases comme " *Régler le bouton (TAPE SELECT) en fonction du type de bande utilisée* " et l'alternative simple de phrases du genre : " *Enfoncer complètement l'accélérateur pendant le lancement du moteur, si la température est inférieure à 0° C* ". Ainsi les instructions de contrôle apparaissent de manière contingente et sont plus facilement catégorisées.

Afin d'avoir une démarche intégrée qui favorisera l'assimilation, nous avons spécifié et réalisé une interface homme-machine en français cohérente avec notre approche [Bril 93].Cette interface a été concue de manière à être entièrement modifiable par l'enseignant didacticien. Elle repose sur une structure arborescente des concepts. Cette arborescence est définie par le didacticien . L'arborescence des concepts, les noms des concepts, la couleur dans laquelle est incorporé l'instruction, la ligne d'état, la ligne d'aide peuvent être modifiées de façon à tester différentes hypothèses didactiques. Cela permettra aussi de l'adapter en fonction du public (il est clair que pour des étudiants de filières économiques et de gestion nous n'avons pas besoin de leur apprendre trois instructions de répétition , une ou deux suffisent). **Cette interface est disponible sur simple demande** aux auteurs de cette communication.

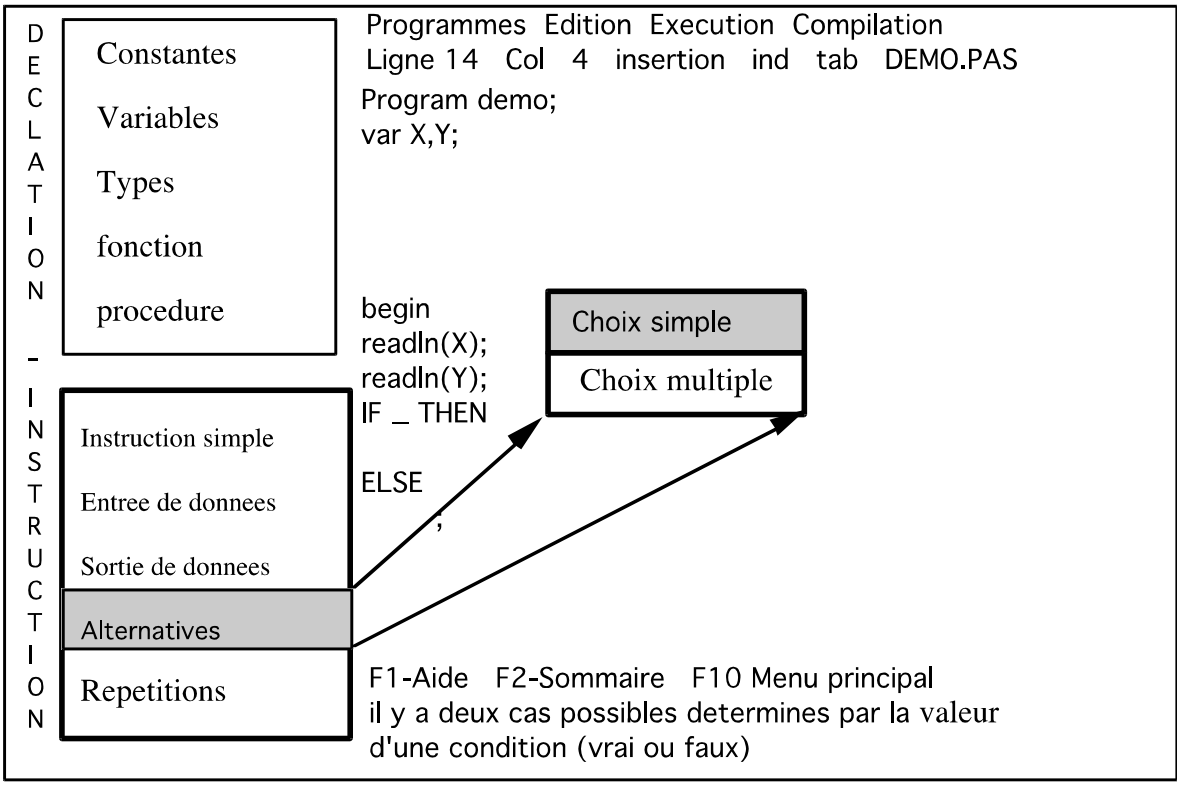

Figure 1 : Interface

La partie gauche de l'écran présente un menu des concepts dont nous souhaitons l'assimilation. On y distingue deux parties, liées aux déclarations et aux instructions. Lorsqu'il sélectionne un concept, l'apprenant se trouve devant une fenêtre qui est soit un sous-menu, soit un concept terminal dont la sélection provoque l'insertion automatique dans son programme de la structure correspondante (déclaration ou instruction). Il peut ainsi par un raffinement progressif de sa pensée, et dans sa langue maternelle accéder au bon concept, par exemple (cf figure 1) : instruction ---> alternative ---> alternative simple , dont la selection provoqque l'insertion du if then else dans son programme.

3 - ASPECTS COGNITIFS

Nous avons vu que se donner un objectif général de lisibilité ou d'intelligence (une éthymologie de intelligence est "inter legere", c'est à dire "lire entre", relier), nous a amené à formuler un objectif cognitif qui est de mettre en place chez l'apprenant **un Système de Représentation et de Traitement de l'informatique de gestion.** Hoc définit un S.R.T comme le produit de l'intériorisation d'un domaine d'action [Hoc 77]. Cet objectif nous amène à enseigner la programmation de manière inhabituelle : il faut privilégier le but plutôt que la méthode, la compréhension plutôt que la création de programmes, on ne peut pas tout dire, et souvent on doit dire de façon non puriste.

Nous subodorons qu'après cette activité de programmation que l'on pourrait qualifier d'imprégnation, les étudiants auront construit une représentation opératoire pour comprendre les applications citées dans l'introduction. Il auront aussi une représentation de l'ordinateur qui permet de comprendre les implications qui résultent par exemple de la différence de nature entre la mémoire centrale et la mémoire de stockage, et cela nous semble aussi fondamental.

Dans un deuxième temps nous pouvons formuler un autre objectif qui est aussi du domaine cognitif : les étudiants doivent avoir assimilé le schéma (au sens de frame [minsky]) de l'étude et de la réalisation d'une application de gestion (enquête, modélisation du problème par des fichiers d'enregistrements, traitements de création et mise à jour sur les fichiers, et quelques traitements de gestion qui utilisent ces fichiers). Nous pensons qu'après une application vue en cours, une application réalisée en TD (par ex. une gestion de stock facturation à et une autre réalisée en projet (par ex. une gestion de bibliothèque de prêt), et quelques-unes commentées, l'étudiant aura suffisamment de matière pour assimiler ce schéma.

La définition du curriculum est déterminée par un objectif cognitif

En effet c'est la poursuite de cet objectif, qui nous a amenés à enseigner les fichiers avant les tableaux (alors que généralement les tableaux sont enseignés en premier). Cela a eu des conséquences décisives sur la réussite des étudiants car les traitements séquentiels sur les tableaux sont plus difficiles à mettre en oeuvre que les traitements séquentiels sur les fichiers. (la condition d'arrêt de l'itération est plus délicate, il faut désigner les éléments, incrémenter l'indice, ...)

D'autre part ce déterminisme nous guide dans le choix des instructions et des notions du langage à aborder. Par exemple une seule instruction d'itération suffit. Ce déterminisme s'est révélé être particulièrement économe en ce qui concerne le choix des notions et des concepts à enseigner.

Aide aux processus cognitifs de différentiation, de structuration et de catégorisation

Examinons une liste non exhaustive d'erreurs ou de confusions que nous avons rencontrées pendant les séances de TD et/ou dans les réponses aux questions de cours :
- confusion entre les instructions d'entrée et les instructions de sortie,
- même si les E/S (clavier/écran) sont comprises, les étudiants oublient souvent les E/S sur fichier (test : citez les instructions de sortie que vous connaissez : moins de 15 % citent les sorties sur fichier). Cela est probablement dû à une mauvaise représentation d'une configuration informatique,
- confusion entre les instructions de contrôle (IF et WHILE par exemple),
- ne rattachent pas les instructions de contrôle aux concepts originaux, ne font pas la distinction entre la partie déclaration et la partie exécutable du programme,
- n'ont pas une bonne représentation d'une configuration informatique, (très souvent ils ne savent pas la différence de nature ni la localisation de la mémoire centrale et de la mémoire auxiliaire),
- n'ont pas bien compris la notion de variable (en particulier le lien entre la déclaration de variable et la déclaration de type).

L'analyse de ces erreurs semble mettre en évidence trois causes majeures qui entravent l'apprentissage :
- la structure d'une configuration informatique n'est pas assimilée,
- la différenciation entre les différentes instructions n'est pas faite,
- les instructions ne sont pas rattachées au concept qui les généralise.

Cette analyse nous **a fait prendre conscience de la mobilisation importante, dans cet enseignement, des processus cognitifs de différentiation, de structuration et de catégorisation** : structure d'une configuration informatique, structure syntaxique d'un programme, d'une instruction, les différents types de données, les différents types d'instructions (dont les structures de contrôle). Notons à ce propos la réflexion de Piaget sur la formation de l'intelligence : " Pas de genèse sans structure, pas de structure sans genèse " [Piag 79]. Les défaillances relatives à ces

aspects structurels ont des conséquences très lourdes dans la compréhension et la résolution des tâches : par exemple comment appréhender un traitement utilisant un fichier si l'apprenant a une représentation qui inclut la mémoire auxiliaire dans l'unité centrale ?

Outre l'interface de la figure 1 qui va favoriser ces processus de différentiation catégorisation nous faisons référence le plus souvent possible au schéma d'une configuration informatique dans laquelle on distingue mémoire principale et mémoire secondaire, ainsi qu'a des classifications comme par exemple le synoptique des types pascal.

3 CONCLUSION

Malgré le faible volume horaire imparti à l'informatique dans les filières d'économie de gestion et d'administration il est possible en 70 heures de présenter un sous-ensemble de concepts de la programmation pertinent pour notre objectif d'ordre cognitif. On pourra s'appuyer sur les représentations mises en place pour, dans un deuxième temps (30 heures), montrer les limites d'une approche par les fichiers, et donc d'aborder avec de la matière l'approche par les base de données (avec la présentation et l'utilisation d'un SGBD). Cela donnera aussi l'occasion d'illustrer la notion de couche logicielle, en utilisant un SGBD pour refaire des parties d'application qui ont été programmées l'année précédente.

BIBLIOGRAPHIE

[AUS 68] AUSUBEL D.P., *Educational psychology :A cognitive view*. Holt, Rinehart and Winston,New York,1968.

[BRI 91] BRILHAULT Y, «Taking into account the student model to define curriculum for apprenticeship of programmation »*Proceedings of the Sixth International PEG Conference on Knowledge Based Environments for teaching and Learning, RAPALLO*.

[BRI 92] BRILHAULT Y, «Proposition de Curriculum pour les filières d'économie, gestion et d'administration.»*Actes du 3ème Colloque Francophone sur la didactique de l'informatique SION*.

[BRI 93] BRILHAULT Y, DUBOUE M, « Aspects cognitifs d'une interface pour l'apprentissage de la programmation » Environnements Interactifs d'Apprentissage avec Ordinateur. 3 èmes journées E.I.A.O de CACHAN .Paris, EYROLLES.

[DUB 89] DU BOULAY B, « Difficulties of learning to program » in *Studying the novice programmer*.Hilsdale : Lawrence Erlbaum 1989.

[DUC 90] DUCHATEAU C, Images pour programmer. De Boeck Université, Bruxelles.

[HOC 77] HOC J.M, « Role of mental representation in learning a programminglanguage». *International Journal of Man-Machine Studies,* 9, 87-105.

[PIA 79] PIAGET J, Epistémologie génétique, Paris, PUF,1979, p. 153.

ANNEXE 1: Curriculum

PARTIE 1 *: (10 H cours + 8 H TD)*

Concepts pertinents de la programmation pour nos objectifs :
CODAGE : variables, types simples (entier, réel, caractère, chaîne, booléens),
ENTRÉES/SORTIES,
CALCUL ET AFFECTATION
CONTRÔLE : séquentialité, alternative, répétition,
TECHNIQUE DE BASE : compteur, accumulateur, Dialogue homme-machine.
à l'aide d'exemples et d'exercices simples qui ont du sens.

PARTIE 2 *(8 H Cours + 8H TD)*

Présentation des enregistrements et des fichiers. Le renforcement des concepts de la première partie est effectué à l'aide d'exercices simples de manipulation de fichiers : création, consultation, mise à jour, traitements simples.

PARTIE 3 *(7 H cours + 14 H TD)*

Analyse et Conception des Systèmes d'information.
Les procédures.
En Travaux dirigés, on assistera les étudiants dans l'étude et la réalisation d'une petite application de gestion, par exemple la gestion des adhérents et des livres d'une bibliothèque de prêt. Deux séances de T.D sont réservées à l'étude préliminaire dans laquelle l'enseignant joue le rôle du responsable de la bibliothèque qui n'a aucune connaissance en informatique et qui devant les limites de son organisation actuelle envisage une informatisation.
Les tableaux.

Il faut ajouter à ce curriculum 13 H de cours d'informatique générale que nous ne détaillerons pas ici. Ces heures se trouvent disséminées surtout dans la partie 1 et concernent entre autre la structure et le fonctionnement des ordinateurs, le codage de l'information, les langages, les logiciels systèmes et d'application.