ASPECTS COGNITIFS D'UN ENVIRONNEMENT HYPERMEDIA D'APPRENTISSAGE DE LA PROGRAMMATION

Yannick Brilhault, Marcel Duboué

LICIAP - Département d'informatique
Université de Pau et des pays de l'Adour
Avenue de l'université
64000 PAU - FRANCE

Résumé : *Une analyse didactique de l'apprentissage de la programmation impérative a mis en évidence des difficultés importantes au tout début de cet apprentissage. Ces difficultés proviennent entre autres d'une mauvaise représentation d'une configuration informatique, de confusions entre les différentes instructions ainsi que d'une mauvaise catégorisation de ces instructions par rapport aux concepts fondamentaux de l'informatique (répétitions, alternatives, entrée/sortie).*

Nous pensons que ces difficultés peuvent être considérablement atténuées par l'utilisation d'un environnement d'apprentissage plus adapté à des débutants que l'interface standard de Turbo-Pascal. C'est pourquoi nous proposons une interface couplée avec un hyperdocument qui propose un cadre structurant qui va aider l'apprenant à différencier les instructions et à les organiser par rapport aux concepts fondamentaux de l'informatique. Cette interface qui exige un effort de réflexion avant l'écriture de chaque instruction devient un agent actif du processus d'apprentissage. De même la structure de l'hyperdocument et la navigation autorisée sont prévues de manière à accentuer certains aspects structurels du domaine. En outre l'environnement d'apprentissage étant évolutif, il devient un outil d'expérimentation didactique.

1. INTRODUCTION

Les sources de connaissances pour une interface utilisateur ont été identifiées par (Rissland, 87), elles peuvent être groupées en trois catégories principales : les connaissances sur l'apprenant, sur la tâche et sur le système. Dans le domaine des Systèmes tuteurs intelligents, les travaux ont surtout porté sur la modélisation de la tâche et sur la modélisation de l'apprenant. Cependant des travaux récents intégrant dans l'interface utilisateur des connaissances qui portent sur le système :

- (Borne et Bourdeau, 91) dans l'apprentissage de la Programmation Orientée Objet, proposent d'aider le programmeur à rechercher une classe particulière dans une librairie de classes,

- (Mac Calla, Greer et Bhuiyan 92) guident le programmeur dans l'écriture des programmes Lisp en lui donnant de la connaissance sur la tâche et sur le système (i.e le programme à écrire et des concepts de base de Lisp),
- (Anderson et Anjaneyulu, 92) proposent une interface qui donne un support visuel aux débutants en Lisp,
- (Recker et Pirolli, 92) proposent un environnement hypertexte pour expliquer les notions de base de Lisp en liaison avec des exemples de fonctions.

De plus ces environnements de programmation ont pour objectif d'induire des comportements adaptés à l'aspect méthodologique propre au paradigme de programmation enseigné.

Notre travail s'inscrit dans une même approche à savoir proposer à l'apprenant un environnement de programmation impérative qui lui facilitera l'apprentissage des concepts fondamentaux de l'informatique par la pratique d'un langage de programmation.

Parmi les critères d'appréciation qui caractérisent une interface utilisateur figurent en bonne place les paramètres suivants:

- facile à apprendre,
- facile à utiliser.

Nous pensons comme Borne et Bourdeau qu'il est au moins aussi important de tenir compte du paramètre " facilitant l'apprentissage " dans un environnement éducatif. Afin d'illustrer notre propos, considérons l'interface standard de Turbo-Pascal qui autorise à se servir directement de la commande d'exécution, même si la compilation n'a pas été demandée. Le corpus à enseigner étant les concepts fondamentaux de l'informatique, il eût été préférable que l'apprenant ne puisse pas utiliser indifféremment la demande d'exécution ou la demande de compilation, afin de comprendre plus rapidement ces deux notions fondamentales. Dans ce contexte la facilité d'apprendre et la facilité d'utilisation de l'interface nous semblent moins importantes que l'aspect didactique qui n'aurait pas permis ce raccourci (du moins au début de l'apprentissage). Au delà de cet exemple, cela pose le problème de l'utilisation de la même interface au cours du processus d'apprentissage, et nous avons pour notre part décidé d'utiliser une interface adaptée à des débutants. Afin d'analyser le lien qui existe entre la définition de notre interface et le corpus à enseigner nous allons d'abord préciser le contexte d'apprentissage et le cheminement didactique qui a été le nôtre.

2. PROBLEMATIQUE DE L'ENSEIGNEMENT

Nous ne pouvons présenter notre recherche sans nous positionner par rapport à une question très actuelle sur l'enseignement des bases de l'informatique à une population de non scientifiques.

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 les concepts fondamentaux de la programmation reste par contre ouverte.

Néanmoins nous constatons une diminution et parfois une disparition de l'apprentissage de la programmation dans ces filières. Nous pensons pour notre part que cette tendance résulte plus d'un constat de relatif échec de cet apprentissage que d'un choix délibéré et que l'on ne peut faire l'économie de l'apprentissage des concepts fondamentaux de l'informatique par l'intermédiaire de la programmation. Nous avons montré dans (Brilhault, 91) que les étudiants de filières économie, gestion ou administration peuvent apprendre les concepts fondamentaux de la programmation, 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 un curriculum court (40 heures dont 30 pour la programmation) dont l'objectif est de mettre en place chez l'étudiant un Système de Représentation des données et des Traitements (voir (Hoc, 77) pour cette notion de SRT) sur lequel on pourra s'appuyer ultérieurement pour l'apprentissage des outils de l'informatique. Nous avons décrit ce curriculum et nous avons présenté son évaluation sur plusieurs promotions (3 promotions, ce qui représente 306 étudiants) dans (Brilhault, 92) et malgré les résultats très encourageants il reste une proportion importante d'étudiants en situation d'échec (aux alentours de 30 %).

D'autre part nous avons mis en évidence un barrage dès le début de l'apprentissage de la programmation : l'analyse de deux contrôles (un après 6 semaines et l'autre en fin de semestre), dans laquelle nous avons étudié le devenir des étudiants qui étaient en situation d'échec au premier contrôle, montre qu'ils sont pour plus de 90% dans la même situation au deuxième contrôle. Ce qui veut dire qu'un échec précoce est quasi-irrémédiable.

Afin de diminuer ce taux d'échec nous pensons comme (Johnson et Soloway, 86) que dans un premier temps les erreurs et confusions de bas-niveau doivent être éliminées à l'aide de programmes simples. Nous n'avons pas dans un premier temps de souci méthodologique, car comme le soulignent (Gueraud et Peyrin, 88), l'apprentissage des méthodes de la programmation doit faire face à des exigences contradictoires :

- soit l'exercice est trop simple pour justifier de la méthode,
- soit l'exercice est trop compliqué pour être traité par des débutants.

Nous situons l'utilisation de notre interface dans ce contexte d'apprentissage, dans la phase initiale de cet apprentissage au moment ou l'apprenant écrit ses premiers programmes simples sans méthode.

3. EXAMEN DE LA SITUATION D'APPRENTISSAGE

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 :

- 1 - confusion entre les instructions d'entrée et les instructions de sortie,
- 2 - 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,

- 3 - confusion entre les instructions de contrôle (IF et WHILE par exemple),
- 4 - ne rattachent pas les instructions de contrôle aux concepts originaux, (à la question : *citez les instructions conditionnelles que vous connaissez*. 67 % citent dans les réponses une ou deux instructions d'itération). Bien que cette confusion s'explique par le fait qu'il y a dans les instructions itératives une "partie conditionnelle" qui sert à arrêter l'itération, il n'en demeure pas moins qu'il y a une confusion dans les esprits et que les concepts ne sont pas maîtrisés).On notera que l'utilisation de mots anglais au demeurant peu clairs (while et for) ne facilite pas ce rattachement sémantique.
- 5 - ne font pas la distinction entre la partie déclaration et la partie exécutable du programme,
- 6 - ne comprennent pas exactement ce qu'est la compilation (pour 13 % d'entre eux la compilation est uniquement la vérification des erreurs),
- 7 - utilisent la commande run comme raccourci sans connaître la différence qu'il y a entre une compilation et une exécution,
- 8 - n'utilisent pas l'aide,
- 9 - n'ont pas une bonne représentation d'une configuration informatique,
- 10 - n'ont pas bien compris la notion de variable (en particulier le lien entre la déclaration de variable et la déclaration de type).

Ce n'est pas en associant formellement un concept abstrait (par exemple la répétition) avec un mot étranger au demeurant étrange (le While que l'on traduit par Tant que ou encore le For que l'on traduit par Pour), pendant le cours magistral que l'on améliore suffisamment la situation. Même si nous adhérons à l'idée que la pédagogie a quelque chose à voir avec "l'Art de répéter", ces répétitions ont une limite. De plus une proportion non négligeable d'étudiants ne vient pas aux cours magistraux, c'est pourquoi nous avons décidé d'intervenir au niveau de l'interface utilisateur, dans une phase où les apprenants sont actifs.

4. ANALYSE DIDACTIQUE DE LA SITUATION D'APPRENTISSAGE

Le manque de compréhension des notions de compilation et d'exécution peut être aisément amélioré par la solution que nous préconisions dans l'introduction. Hormis ce point ainsi que le fait qu'ils ne se servent pas de l'aide, il nous semble que les autres erreurs relèvent de 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.

C'est pourquoi les objectifs de notre interface seront donc d'aider les apprenants :

A) - à assimiler une représentation opératoire du dispositif informatique (en particulier les conventions terminologiques relatives aux transferts de données entre l'ordinateur et la périphérie),

B) - à différencier les instructions car c'est une première étape vers la compréhension,

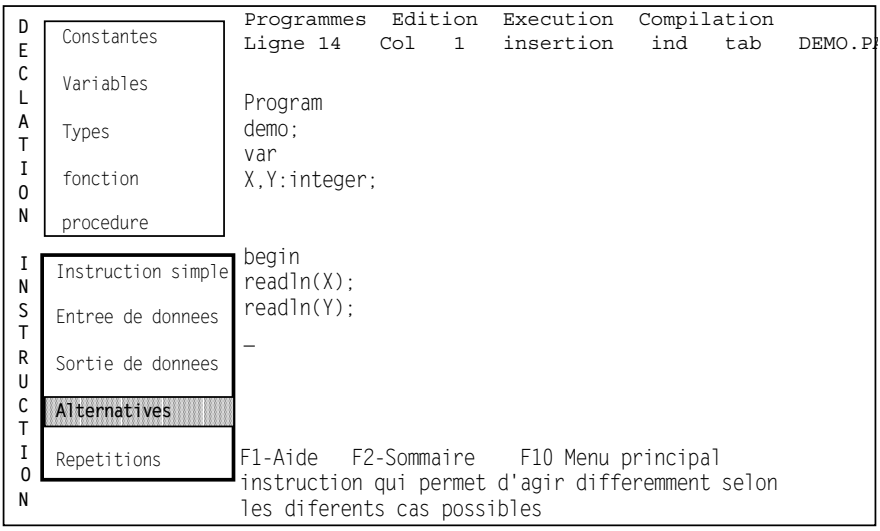
C) - à leur faire mieux comprendre les concepts fondamentaux qui sont utilisés au travers d'instructions d'un langage particulier (par exemple que le concept de répétition est un concept fondamental qui se traduit soit par une instruction WHILE soit par une instruction FOR soit par une instruction REPEAT).

5. REPONSE COGNITIVE

Examinons maintenant les processus cognitifs sur lesquels repose notre interface.

Bien qu'il y ait pas toujours convergence à l'heure actuelle des théories cognitives sur les localisations, l'organisation et le fonctionnement de la mémoire à long terme (MLT) et de la mémoire à court terme (MCT), les mécanismes énoncés par (Ausubel, 1968) et repris par (Mayer, 81) dans le cadre de la programmation impérative sont toujours pertinents. On peut résumer la partie qui nous intéresse de la façon suivante : lorsque l'on présente à un apprenant une nouvelle quantité de connaissances à assimiler, cette assimilation a plus de chances de succès si les nouvelles connaissances peuvent être liées avec des connaissances, des représentations ou des schémas déjà présents en MLT. C'est pourquoi il est nécessaire que l'apprenant soit conscient qu'il utilise le concept de répétition lorsqu'il écrit une instruction WHILE. On note à ce propos que le terme d'itération nous semble mal adapté car il est inconnu de la plupart des étudiants. Notre interface va donc "forcer" le programmeur débutant à sélectionner le concept puis le sous-concept (fig 1) avant de pouvoir écrire une instruction. Nous pensons que cette petite gymnastique cérébrale (récompensée par l'écriture automatique d'une partie de l'instruction) l'aidera à associer les instructions aux concepts auxquels elles se rattachent. En accord avec les théories actuelles qui insistent sur la situation "active "de l'apprenant, cette interface qui exige un effort de réflexion avant l'écriture de chaque instruction, devient un agent actif du processus d'apprentissage.

Dans son célèbre article "The magical number Seven" (Miller, 56) relate un certain nombre d'expérimentations qui essayent toutes de mesurer le nombre de graduation discernables d'un stimulus unimodal. Par exemple combien de degrés d'intensité sonore différents est on capable de distinguer ? il constate qu'un "chiffre moyen" pour cette capacité de discernement est 7 (± 2 suivant les individus). Par contre si on fait varier plusieurs modalités (dans le même exemple on peut faire varier la localisation, la fréquence...) l'individu est alors capable d'augmenter sa capacité de discernement (elle passe de 7 à 150 si on fait varier 6 modalités). Nous nous servirons de cette propriété dans notre interface pour aider l'étudiant à différencier les instructions, en introduisant des couleurs différentes : les instructions d'entrée sont en bleu, les instructions de sortie sont en vert, les instructions de répétition en rouge et les instructions conditionnelles en jaune. L'apprenant dispose donc d'une modalité supplémentaire pour l'aider à différencier les instructions.



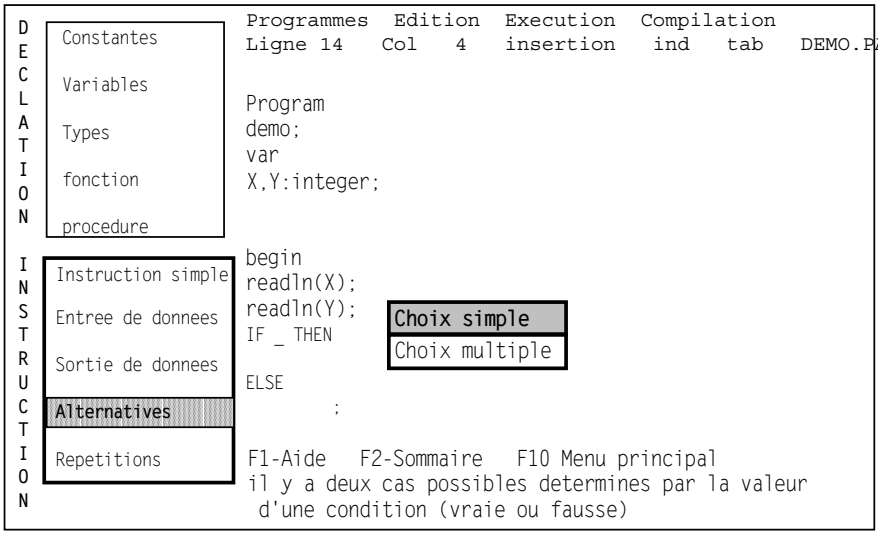


Figure 1.a et 1.b

L'hypothèse d'une propriété "structurante" que peut avoir un hyperdocument sur un apprenant a été avancée par de nombreux auteurs : (Quentin et Depover, 1991), (Michau, 1991), (Leclercq, 1991) et (Dufresne, 1991). Nous adhérons à cette idée, c'est pourquoi la structuration de l'hyperdocument repose sur un arbre hiérarchique des notions, dans lequel la navigation est majoritairement hiérarchisée. **Nous pensons que la période d'apprentissage qui nous intéresse est particulièrement propice à l'utilisation de cette propriété des hyperdocuments.** En effet le novice se trouve confronté entre autres avec des difficultés de structuration : structure d'une configuration informatique, structure syntaxique d'un programme, d'une instruction, les différents types de données, les différents types d'instructions. Citons ce que disent (Deveau, et al, 88) de cette période : "les étudiants sont assaillis pendant une courte période par une quantité d'informations fondamentales et de techniques

nouvelles et il leur est demandé en plus de réfléchir et de comprendre ". On peut aussi noter que ce manque de repères est ressenti par l'apprenant lui même: dans un questionnaire ouvert qui demandait aux étudiants de qualifier l'apprentissage de l'informatique, l'adjectif le plus cité était le mot déroutant.

6. DESCRIPTION ET FONCTIONNEMENT DE L'INTERFACE

Cette interface remplit deux fonctions :

- elle induit un mode de travail structurant,
- elle donne accès à un hyperdocument qui doit aider l'apprenant à organiser ses connaissances du domaine.

6.1 Mode de travail

Nous reconnaissons en partie droite (cf figure 1) l'interface standard de Turbo Pascal francisée car il nous semble souhaitable d'éliminer la difficulté linguistique ("une difficulté à la fois") à un stade de l'apprentissage où l'apprenant a déjà beaucoup de difficultés .

Une autre particularité pour diminuer les confusions de types -6- et -7-, est que l'exécution n'est acceptée que si une compilation a eu lieu sans modification postérieure du texte.

La partie gauche de l'écran (cf figure 1) présente un menu des concepts dont nous souhaitons l'assimilation. On y distingue deux parties, liées aux déclarations et aux instructions : l'une seulement de ces parties est clairement visible à un instant donné (vidéo intense) en fonction de la position du curseur d'édition avant ou après le "begin" du programme. Nous pensons ainsi diminuer les erreurs de type 5. 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 $\rightarrow$ alternative $\rightarrow$ alternative simple , dont la sélection provoque l'insertion du " if then else" dans son programme.

Afin de pallier au très faible taux de consultation de l'aide dans l'interface standard, une information sur le choix en cours est automatiquement produite en bas de la partie droite de l'écran et l'hyperdocument est accessible de manière contextuelle par la touche F1, ou bien on accède à la page du sommaire par la touche F2 (cf figure 2).

Pour les raisons évoquées plus haut, nous avons choisi de générer les instructions à l'aide de quatre couleurs : E/S = bleu/vert, le rouge pour les répétitions et le jaune pour les conditionnelles. Nous pensons qu'il n'est pas nécessaire d'en mettre davantage pour tester la pertinence de nos hypothèses. d'autre part la conception de l'interface nous autorise à modifier les attributs de couleurs si cela est nécessaire d'un point de vue ergonomico-didactique.

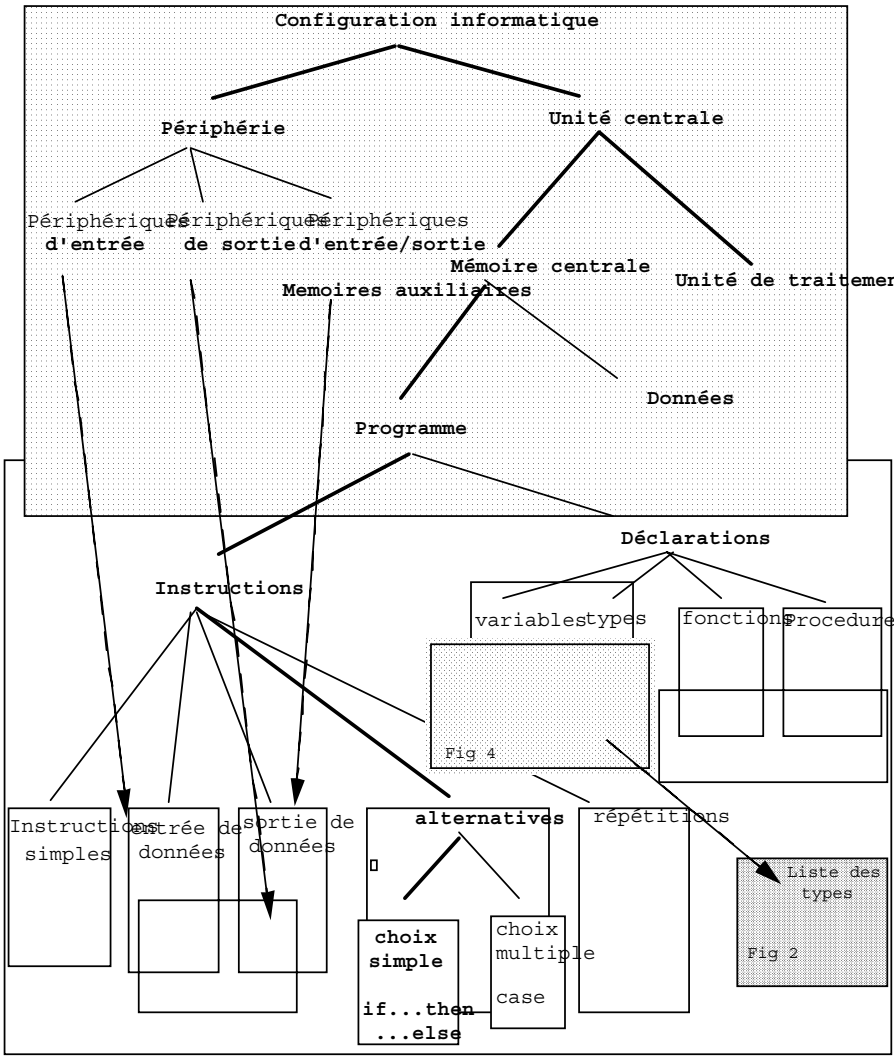


Figure 2

6.2 Accès à l'hyperdocument

On peut dresser la liste des objectifs que nous assignons à l'hyperdocument :

- renforcer par une navigation hiérarchisée et par des pages ad hoc (figure 3) la structuration des connaissances de l'apprenant.
- pallier à la très faible utilisation de l'aide en mettant à la disposition des apprenants des pages d'information sémantiquement riches.

Pour le premier point, on peut vérifier que l'hypergraphe de la figure 2 est cohérent avec le mode de fonctionnement de l'interface. Cet hypergraphe contient tous les concepts fondamentaux du domaine nécessaires à l'apprentissage.

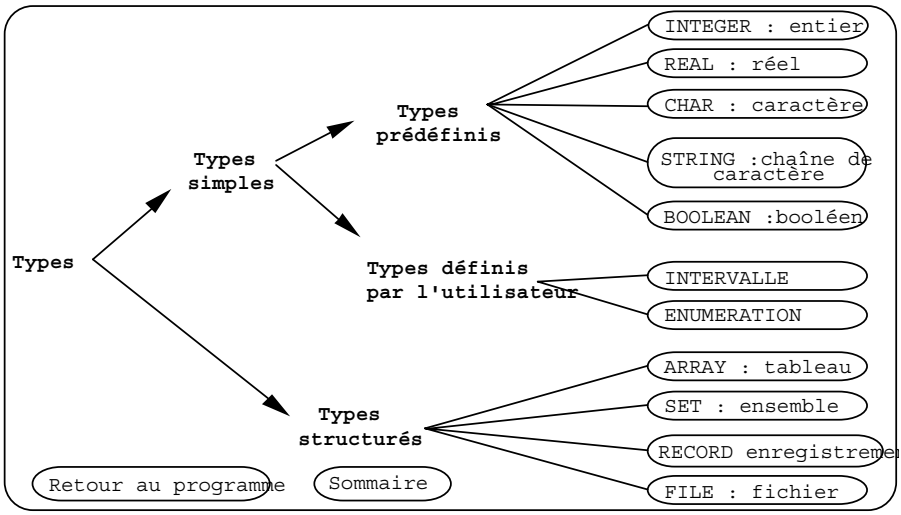


Figure 3

Malgré le choix volontairement hiérarchisé de la navigation il existe quand même des liens associatifs, nous avons indiqué en pointillé sur la figure 2 un exemple de ce type de liens.

La figure 4 est un exemple de page que nous qualifions de sémantiquement riche. Ce sont des pages qui regroupent de manière synthétique plusieurs notions fondamentales et qui explicitent le lien entre ces différentes notions ou concepts. Afin d'encourager l'utilisation de cet hyperdocument, nous proposons d'en faire plusieurs démonstrations pendant les cours magistraux (à l'aide d'un matériel de rétroprojection). Au cours de ces démonstrations, nous commenterons des pages de l'hyperdocument dans lesquelles certaines notions qui posent problème sont mise en scène.

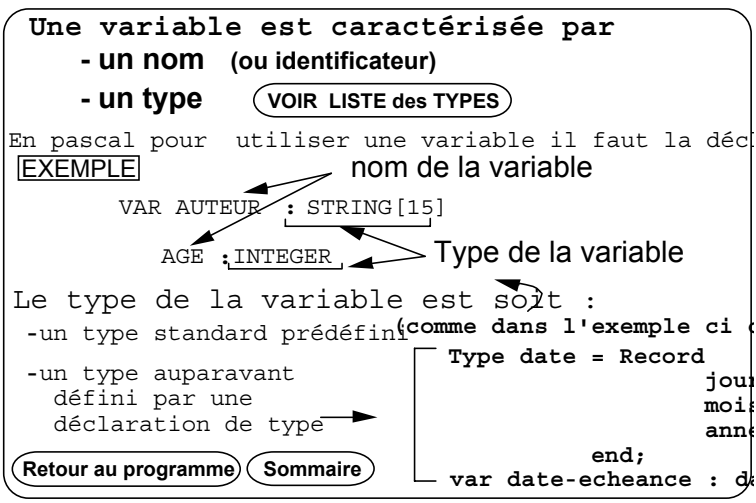


Figure 4

7. CONCEPTION DE L'INTERFACE

Cette interface a été conçue 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 (sous-arbre programme de la figure 2). 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 les 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).

D'autre part il est intéressant d'un point de vue didactique de prévoir une interface qui peut évoluer au fur et à mesure de l'augmentation des connaissances de l'apprenant car cela introduit une diversité propice au maintien de l'attention. Néanmoins nous avons gardé une cohérence avec celle de Turbo-Pascal de façon à ne pas introduire de "dépaysement" inutile dans leur environnement d'apprentissage.

Cette adaptabilité de l'interface ainsi que la flexibilité intrinsèque des hyperdocuments nous fournit un outil d'expérimentation facile à maintenir et à adapter qui nous sera très utile pour tester différentes hypothèses didactiques.

8. CONCLUSION

La mise en évidence de difficultés importantes au tout début de l'apprentissage de la programmation impérative, justifie la conception d'environnement d'apprentissage plus adapté à des débutants que l'interface standard de Turbo-Pascal et cela même si l'interface n'est utilisée que durant une période très limitée. Au delà de cette application, nous pensons que les processus cognitifs sur lesquels nous nous appuyons peuvent être également pris en compte dans d'autres situations d'apprentissage lorsque l'apprenant se trouve face à des difficultés de différenciation et de catégorisation.

BIBLIOGRAPHIE

Anderson (J.R.), Anjaneyulu (K.S.R.), 1992- "The Advantages of Data Flow Diagrams for Beginning Programming". In *proceedings of the International conference on Intelligent Tutoring Systems*. C.Frasson.,Gauthier G., G.I. McCalla (eds.) Spinger-Verlag, 585-592.

Ausubel, (D.P.), 1968.- *Educational psychology :A cognitive view*. Holt, Rinehart and Winston, New York, 1968.

Bourdeau (J.) Boorne (I.), 1991-"Knowledge-Based User Interfaces to facilite learning". In *proceedings of the Sixth International PEG Conference* on Knowledge Based Environments for teaching and Learning, RAPALLO.

Brilhault (Y.), 1991-"Taking into account the student model to define curriculum for apprenticeship of programmation". In *proceedings of the Sixth International PEG Conference* on Knowledge Based Environments for teaching and Learning, RAPALLO.

Brilhault (Y.), 1992 -"Proposition de curriculum pour les filières d'économie, de gestion et d'administration". *Actes du 3ème Colloque Francophone sur la didactique de l'informatique*, SION.

Bhuiyan (S.), Greer (J.I.), McCalla (G.I.), 1992 - "Learning recursion through the use of a Mental Model-Based Programming Environment". In *proceedings of the International conference on Intelligent Tutoring Systems*. C.Frasson.,Gauthier G., G.I. McCalla (eds.) Spinger-Verlag, 50-57.

Deveaux (D.), Raphalen (M.) Revault (J.), 1988- "Spécification d'un interpréteur de mini-langage algorithmique pour l'initiation à la programmation". *Actes du 1er Colloque Francophone sur la didactique de l'informatique*, Paris, 87-101.

Dufresne (A.) 1991 - "Ergonomie cognitive, hypermedias et apprentissages". *Actes des 1ères journées scientifiques Hypermédias et Apprentissages.* Paris. 121-131.

Guéraud (V.), Peyrin (J.M.), 1988- "Un jeu de rôles pour l'enseignement de la programmation". *Actes du 1er Colloque Francophone sur la didactique de l'informatique*, Paris, 48-59.

Hoc (J.M.), 1977- "Role of mental representation in learning a programming language". *International Journal of Man-Machine Studies*, 9, 87-105.

Johnson (W.L), Soloway (E.), 1986.- *Intention-Based Diagnosis of novice programming errors.* Morgan Kaufmann Publishers, Inc., Los Altos, California.

Leclercq (D.), 1991 -"Hypermédias et apprentissage : Vers un compromis". *Actes des 1ères journées scientifiques Hypermédias et Apprentissages.* Paris. 19-35.

Mayer (R.E.), 1981- "The psychology of how Novices learn computer programming". *ACM, Computing surveys*, Vol. 13, No. 1, March 1981. 121-141.

Michau (F.), 1991- "Accès par hyperdocument à une banque d'exercices sur un logiciel de simulation en automatique". *Actes des 1ères journées scientifiques Hypermédias et Apprentissages.* Paris. 111-118.

Miller (G.A.), 1956- "The magical number Seven, plus or minus two". *The psychological Review*, Vol. 63, No 2, Mars 56.

Quentin (J.J.), Depover (C.), 1991- "Pour une approche signifiante de l'apprentissage à partir d'une base de données multimédia". *Actes des 1ères journées scientifiques Hypermédias et Apprentissages*. Paris. 121-131.

Recker (M.M.), Pirolli (P.), 1992- "Student strategies for learning Programming from a computational environment". In *proceedings of the International conference on Intelligent Tutoring Systems*. C.Frasson., Gauthier, G., G.I. McCalla (eds.) Spinger-Verlag, 50-57.

Rissland (E.), 1987.- "Ingredients of intelligent user interfaces". in Baecker, R.M & BUXTON W.(eds) *Readings in Human-Computer Interaction: a Multidiciplinary Approach*, NY: Morgan Kaufmann, 703-708.