L'APPRENTISSAGE DE L'ITÉRATION DANS DEUX ENVIRONNEMENTS INFORMATIQUES

Vassilios Dagdilelis
Nicolas Balacheff
Bernard Capponi

Cet article présente une étude comparée de l'apprentissage de l'itération dans deux environnements informatiques différents : Pascal et Multiplan. Cette étude a été réalisée avec des élèves de troisième (fin de la scolarité obligatoire) d'un collège français. Nous avons pu dégager des analogies importantes entre les deux environnements à propos de l'itération. Mais l'hypothèse consistant à prévoir un transfert des connaissances d'un logiciel à l'autre ne s'est pas trouvée confirmée, ce qui confirme la complexité cognitive du concept d'itération, déjà mise en évidence dans d'autres recherches.

1. INTRODUCTION

enseigner l'informatique : une nécessité

La programmation informatique prend progressivement sa place dans le cursus scolaire. Dans le contexte de l'informatisation des métiers et, plus généralement, de la société, *l'alphabétisation informatique devient tout aussi nécessaire que l'initiation aux Mathématiques pour tous ceux dont ce ne sera jamais la profession* (Rogalski 1985). Une des principales composantes de l'alphabétisation en informatique est la programmation. Cet enseignement passe par l'appropriation de notions fondamentales d'algorithmique, par des élèves débutants, qui pose de nombreux problèmes.

l'itération : une structure fondamentale de la programmation...

Un certain nombre de recherches ont été entreprises, en France et à l'étranger, dans le but d'étudier les processus d'enseignement et d'apprentissage des savoirs concernés. La notion d'itération est au cœur même de ces recherches. Cet intérêt se justifie par le fait que l'itération constitue un élément fondamental de la programmation mais dont la mise en œuvre pose en revanche des problèmes conceptuels importants aux élèves débutants. Ainsi que le montrent certains travaux (Rouchier et als 1984) la mise en œuvre de l'itération dans un environnement informatique (sur un dispositif informatique donné) dépend de certaines caractéristiques de cet environnement : caractéristiques ergonomiques, disponibilité des divers types d'itération, etc. Certaines de ces difficultés semblent être liées plus directement au concept même d'itération, ou plus précisément au fait qu'il constitue un outil pour la réalisation d'un grand nombre de calculs caractérisés par un ou plusieurs invariants.

ASTER N°11. 1990. *Informatique, regards didactiques*, INRP, 29, rue d'Ulm. 75230, Paris Cedex 05.

... et de son apprentissage

Dans le présent travail nous présentons des résultats relatifs au problème de l'apprentissage de l'itération. Si, d'une manière générale, la mise en œuvre de l'itération correspond à l'utilisation d'un concept d'algorithmique dans le contexte d'un dispositif informatique, alors il semble raisonnable de faire l'hypothèse que l'introduction de l'itération dans un environnement donné facilitera son utilisation dans un autre environnement. Dans ce qui suit nous étudions certains aspects de cette question du transfert de connaissances sur l'itération. Il

nous étudions ici sa mise en œuvre dans deux environnements...

s'agit, plus précisément, de l'analyse de deux situations dans lesquelles des élèves avaient à résoudre un même problème itératif mais dans deux environnements de programmation différents : d'abord dans un environnement de programmation classique (MacPascal) puis dans l'environnement d'un tableur (Multiplan).

2. LE CADRE DE L'EXPÉRIMENTATION

Le problème que nous étudions est en fait issu des questions soulevées sur le terrain, à propos d'une initiation d'élève de Collège (classe de troisième) à l'informatique[1]. L'idée initiale était de réaliser un projet d'alphabétisation en s'appuyant sur un progiciel, Multiplan, qui est un tableur très répandu dans le monde du travail, mais dont les fonctionnalités sont, en général, sous-exploitées. En particulier, on peut constater que l'itération disponible sur Multiplan est souvent peu utilisée (voire absente des formations les plus classiques). En quelque sorte, on peut penser que Multiplan est un "écosystème" peu favorable à l'appréhension et l'utilisation de l'itération par les novices, bien qu'il s'agisse là d'un outil très performant dans bien des situations.

le tableur Multiplan

Partant de l'hypothèse que les langages de programmation "classiques" constituent un "lieu naturel" pour la mise en œuvre de concepts fondamentaux d'algorithmique, on peut penser que le langage Pascal offre un environnement bien adapté pour la présentation et la mise en œuvre de l'itération. Par ailleurs, l'utilisation de MacPascal (version interprétée de Pascal implémentée sur Macintosh) permet d'éviter les difficultés pratiques et conceptuelles liés à la compilation.
Ainsi nous avons abordé dans le cours de cette formation la notion d'itération successivement dans l'environnement Pascal, puis dans l'environnement Multiplan.

et le langage Pascal

(1) La formation décrite a eu lieu dans le cadre d'une option obligatoire et elle a été encadrée par l'équipe de Didactique des Mathématiques et de l'Informatique de l'Université Joseph Fourier à Grenoble, aux travaux de laquelle participait aussi l'enseignant.

Nous présentons dans le présent article l'analyse et la comparaison des résolutions d'un même problème proposé aux élèves dans les deux environnements. Notre hypothèse était que les élèves investiraient dans Multiplan les connaissances relatives à l'itération, déjà acquises dans le contexte de MacPascal.

les élèves observés travaillaient en binômes...

Les observations portent sur douze élèves de troisième qui travaillaient dans le cadre d'une formation initiale à l'utilisation des outils informatiques. Ils ont été regroupés en six binômes disposant chacun d'un micro-ordinateur Macintosh 512 K de Apple. Chaque binôme était observé par l'un des membres de notre équipe de recherche et enregistré. Cette disposition des élèves par binôme, peu fréquente dans l'enseignement, constitue un élément important dans les situations que nous avons conçues. Le travail en groupe permet en effet la confrontation d'opinions différentes et, par conséquent, il favorise des explicitations, voire la production d'explications. Il est donc susceptible de provoquer chez l'élève une véritable réorganisation de sa pensée et de contribuer à l'évolution de ses connaissances. Les productions (verbales ou écrites) au cours de ce travail commun constituent un corps de données expérimentales qui ont une grande valeur pour le chercheur. C'est sur un tel corpus que nous nous sommes appuyés pour réaliser les analyses présentées ici, notre travail a été guidé essentiellement par l'analyse a priori de la situation problème et l'analyse conceptuelle des structures algorithmiques en jeu. Nous en présentons ci-dessous les grandes lignes, une analyse complète est disponible dans (Capponi 1990).

... pour favoriser les explicitations des procédures

Les séances de formation comprenaient une activité de la classe toute entière avec le professeur, orientée essentiellement vers l'introduction de notions nouvelles, suivie de travaux pratiques réalisés sur le micro-ordinateur. Le professeur, après avoir fourni une tâche aux élèves intervenait à la demande de ceux-ci. Nous décrivons ici une séance mettant en œuvre l'itération en Pascal (présentée au paragraphe 6.1.) et deux séances dans Multiplan (présentées en 7.3.) .

3. UN POINT DE DÉPART

dans une perspective constructiviste...

Il y a plusieurs façons, bien différentes, d'aborder le problème de l'acquisition des connaissances. Nous nous sommes placés dans une perspective constructiviste : hypothèse d'un sujet qui construit ses propres connaissances en interaction avec son milieu. La théorie des *situations didactiques* (Brousseau 1987) permet, à travers la construction de situations adaptées, d'organiser la formation des élèves en donnant une place centrale à l'activité du sujet. Cette activité doit être orientée vers la résolution de problèmes qui sont "source et critère des savoirs" (Vergnaud 1981). Dans ce contexte nous avons voulu créer des situations problèmes telles que les savoirs en jeu

... les savoirs sont des outils pour résoudre des problèmes

(l'itération et sa mise en œuvre dans le contexte du dispositif donné) puissent apparaître comme des outils économiques et efficaces pour la résolution des problèmes. Il était donc important de créer ces conditions qui pourraient donner un sens aux nouveaux concepts.

4. L'ITÉRATION

4.1. Les éléments d'une structure itérative

Une construction itérative a la forme générale suivante :

```
A:
itérer
        S
        si B alors sortir
        T
encore
```

où B présente la condition d'arrêt de l'itération et S, T sont des ensembles d'instructions, éventuellement vides.
Si S est vide, alors la structure prend la forme classique :

tantque B faire T fintantque

Si, au contraire, T est vide, alors on retrouve la forme :

répéter S jusqu'à ce que B

Pour notre travail nous nous sommes appuyés sur l'analyse conceptuelle de l'itération faite par Mejias (1985).
Mejias distingue les parties suivantes dans une structure itérative :

une structure itérative comprend :
- des instructions à répéter
- une condition d'arrêt

- le corps de l'itération, ensemble des instructions qui doivent être répétées un certain nombre de fois.
- la condition d'arrêt qui sert à assurer que le programme "ne boucle pas indéfiniment". Cette condition porte sur des expressions qui sont mises à jour dans le corps même de l'itération.

Puisque l'itération s'intègre souvent dans un programme plus large, la structure itérative doit être articulée avec le reste du programme. Ainsi il y a :

-un problème d'initialisation des variables et

-un problème éventuel d'articulation après la fin de l'itération.
Mejias distingue aussi dans une structure itérative un avertisseur d'itération (par exemple tantque et répéter) et des délimiteurs du corps d'itération (par exemple **faire**...**fintantque** et **répéter**... **jusqu'à ce que**).
Cette analyse montre en fait la complexité d'une structure itérative; ces éléments sont pertinents non seulement d'un point de vue informatique mais aussi d'un point de vue didactique : ils constituent des "paramètres" didactiques, même si leur prise en compte reste parfois "transparente", implicite -

dans le langage Pascal la structure étudiée est :
- répéter
- instructions
- jusqu'à ce que condition

- surtout chez les programmeurs expérimentés -- parce que le choix de tel ou tel type d'itération (avec donc telle ou telle caractéristique) peut affecter les conduites de l'apprenant (Soloway et als 1984) et aussi le sens ou la complexité d'un problème. En s'appuyant sur les travaux déjà effectués dans ce domaine (Laborde et als 1985), parmi les différents types d'itération possibles dans le contexte du langage Pascal, nous avons retenu le type :

répéter... jusqu'à ce que condition

(**repeat** ... **until** en Pascal) qui, selon Mejias (Mejias 1985, Rouchier et als 1984) est plus proche des conceptions initiales des élèves.
La même problématique nous a conduits à restreindre encore la présentation de ce type d'itération à la forme suivante :

```
repeat
x:=x+1;  {le compteur}
...          { le corps de l'itération}
until x= N {N est un entier}
```

où la variable **compteur** est initialisée implicitement à zéro et est placée après l'avertisseur **repeat** dans le bloc de déclaration des variables.

4.2. Un problème-type d'itération

le problème étudié : programmer le calcul de la somme des 300 premiers entiers

Pour l'étude de l'itération dans diverses situations d'apprentissage les chercheurs utilisent souvent un problème-type dans lequel la tâche consiste à calculer la somme (ou parfois la moyenne) d'un ensemble d'entiers. En s'appuyant sur l'analyse de Soloway (Soloway et als 1984), nous avons choisi le problème suivant :

Trouver la somme des N "premiers" entiers.

Dans ce problème, ce qui est demandé n'est pas une formule algébrique qui "traduit", en quelque sorte, l'énoncé dans un langage symbolique :

$$\sum_{i=1}^{n} i$$

mais la réalisation effective du calcul.

la résolution "à la main" laisse dans l'ombre la nécessaire distinction entre...

Si ce problème peut être résolu "à la main" pour de petites valeurs de N, il est par contre difficile de le résoudre dès que N devient "assez" grand. Par ailleurs N peut être choisi de façon à ne pas permettre de remplacer l'itération par une procédure de réplication, c'est-à-dire les réécritures successives d'un même bloc d'instructions. La formule algébrique classique et la récursivité n'étant pas connues par les élèves de troisième, on

peut en déduire que l'itération apparaît comme un outil adapté pour la résolution de ce problème.

La résolution de ce problème en utilisant l'itération exige l'utilisation de deux variables : un compteur et un accumulateur. Les valeurs successives du compteur sont calculées par une instruction récurrente :

un compteur...

COMPTEUR:= COMPTEUR + 1

Ces valeurs sont ensuite ajoutées dans l'accumulateur par l'instruction :

... et un accumulateur...

SOMME:=SOMME+COMPTEUR

qui a, elle aussi, un caractère récurrent.

L'élève doit donc identifier les invariants qui permettent la réalisation des calculs demandés et les coordonner dans le corps d'une structure itérative adéquate (i.e. exécutable par un dispositif informatique). Il est probable que ces deux invariants ne se situent pas au même niveau de difficulté. Le compteur peut être considéré comme un générateur d'entiers et l'énumération d'entiers successifs est déjà connue des élèves grâce à d'autres activités. Par contre l'accumulateur SOMME est beaucoup plus spécifique du découpage des actions qui sont propres à la programmation. Si le calcul de la somme des 300 entiers était effectué avec du papier et un crayon, il est probable qu'existeraient des "sommes partielles" écrites au fur et à mesure que le calcul progresse, mais il ne serait pas nécessaire que ces sommes partielles soient désignées explicitement ; il suffirait de les avoir notées pour que l'opérateur (humain) puisse les utiliser ultérieurement.

... c'est ce que révèlent les explications orales des élèves

D'autre part une description orale du calcul - en dehors du contexte informatique - a le plus souvent des formes en langue naturelle qui ne font aucune allusion à l'existence d'un accumulateur ou d'un découpage. Pour donner un exemple, voici comment Laurent, un élève, exprime ce qu'il faut faire :

prendre un nombre et puis ajouter le suivant et ainsi de suite...

la tâche du programmeur est de "décompacter" des expressions complexes en actions élémentaires

Notre hypothèse est que la "transparence" des algorithmes - comme l'algorithme exprimé par Laurent - qui ne sont pas exécutables par un dispositif informatique, peuvent se constituer en obstacles majeurs pour la construction d'un algorithme correct et exécutable. Le programmeur doit en effet transformer les expressions "concentrées" (i.e. tout dans une seule expression) en un ensemble d'actions élémentaires qui doivent s'articuler dans le corps d'une structure adéquate. C'est cette transformation qui constitue une tâche difficile pour les programmeurs débutants. Ce même élève, Laurent, l'exprime d'ailleurs clairement en disant :

on sait le dire mais on peut pas le faire...

Si donc on fait l'hypothèse générale que les élèves peuvent essayer de construire leur algorithme en procédant par une exécution mentale (Hoc 1979) ou par une stratégie de transfert de procédures qui sont valables dans d'autres environnements,

on s'aperçoit que la construction de l'accumulateur nécessite un effort intellectuel considérable. Elle exige une véritable construction cognitive.

5. L'ANTHROPOMORPHISME

Pour rendre compte des conceptions spontanées des élèves, Pea (Pea 1984) propose *l'hypothèse anthropomorphique*. Selon cette hypothèse la machine dispose, aux yeux des élèves, des compétences particulières analogues à celles d'un interlocuteur humain. Ainsi les élèves ont tendance à attribuer aux instructions d'un programme une sémantique beaucoup plus large que la sémantique réelle. Ils ne croient pas vraiment à l'existence d'une intelligence humaine "cachée" dans la machine. Mais souvent ils se comportent *comme s'il y en avait une*. Selon Pea les conceptions anthropomorphiques peuvent être la source des certaines erreurs des élèves. Dans un travail plus récent, Spohrer et als (1986) ont élaboré une taxonomie des "bugs" qui, à certains égards, est assez proche du modèle proposé par Pea.

l'élève attribue souvent à la machine une intelligence humaine

Nous nous sommes intéressés au modèle de Pea parce que nous pensons qu'il peut constituer un cadre explicatif de certaines erreurs des élèves (erreurs qui peuvent apparaître surtout au cours d'un enseignement d'introduction, d'alphabétisation). Par ailleurs, le travail dans deux environnements différents permet l'étude des rapports éventuels entre ces conceptions et la nature du dispositif utilisé : est-ce que, par exemple, un type d'environnement donné privilégie un type de conceptions anthropomorphiques plus qu'un autre ?

cela explique certaines de leurs erreurs

6. OBSERVABLES DANS MACPASCAL

6.1. Présentation de l'activité

dans l'environnement Pascal...

Après une séance d'introduction à l'itération, l'enseignant a proposé aux élèves le problème suivant :

> Calcule
> 1+2=
> 1+2+3=
> 1+2+3+4=
> 1+2+3+4+5=
>
> Ecris un programme qui permet à l'utilisateur de calculer ces sommes en choisissant lui-même combien de nombres il ajoute.
> Contrôle ton programme en recalculant les sommes ci-dessus.

Ensuite les élèves devaient calculer les sommes des entiers jusqu'à 100, puis 300, 500, 1000.

6.2. Une synthèse des observations

- Les étapes lors de la mise en œuvre de l'itération

... les premières tentatives des élèves utilisent peu l'itération

En ce qui concerne la mise en œuvre de l'itération, l'ensemble de nos observations confirme les résultats déjà connus : la syntaxe de l'itération n'a pas posé de problèmes. Mais, en revanche, les élèves n'utilisent pas spontanément l'itération comme un outil pour la résolution du problème posé. En général, c'est seulement après une longue période d'essais qu'ils la construisent.

ils ne dégagent aucun invariant...

Les premières tentatives des élèves se caractérisent par des constructions qui ne mettent en évidence aucun invariant. Ils recherchent plutôt la reproduction de résultats pour des petites valeurs des variables qui soient en accord avec les valeurs obtenues avec des calculs "à la main". Voici trois programmes caractéristiques de ces premières tentatives des élèves :

prog1	prog2	prog3
A:=3;	x:=x+1;	a:=1+2;
B:=4;	y:=x+x;	b:=a+3
C:=5;	z:=y+x;	c:=b+4;
D:=A+A;	writeln(z);	d:=c+5;

Si prog1 et prog3 semblent assez éloignés de la solution correcte puisqu'ils ne prennent en compte aucune invariant, par contre prog2 présente un grand intérêt : les élèves n'obtiennent pas les résultats voulus par des affectations de valeurs numériques mais par une manipulation de variables, ils travaillent déjà avec des structures "généralisables".

... mais procèdent de proche en proche

Ces programmes s'inscrivent dans une logique de "sommation par paires" qui correspond à l'expression en langue naturelle :

prendre un nombre et puis ajouter le suivant et ainsi de suite...

L'analyse des relations qui existent entre les variables peut conduire les élèves à l'abandon de cette stratégie et à l'adoption de procédures plus évoluées. Néanmoins il y a deux étapes à franchir:

- l'introduction d'une variable spéciale (l'accumulateur) et
- l'utilisation de l'itération.

Les élèves peuvent, parfois, rester bloqués jusqu'à la fin dans leur logique initiale. Par exemple, Corinne et Estelle ont poussé leur tentative jusqu'au bout : à la fin de la séance elles sont arrivées à un long programme qui reproduit quatre fois le bloc de prog2 . Il s'agit clairement d'une tentative de réplication qui, quoi qu'il en soit, ne peut pas aboutir, étant donné que la somme demandée était de 300 entiers.

L'adoption de l'itération (par quatre binômes sur six), n'a pas automatiquement conduit les élèves à l'identification simultanée des "bons" invariants. Ainsi certains élèves construisent des

l'adoption de l'itération commence par...

programmes avec un corps d'itération qui contient seulement un compteur. Voici par exemple un programme construit par Christophe et Richard :

```
prog4
begin
        repeat
                x:=x+1;
                writeln(x);
        until x=15;
end.
```

la construction d'un compteur

Ce type de programme n'est que relativement marginal ; d'ailleurs on a la nette impression qu'il a été construit par les élèves pour "voir". En revanche le programme suivant est assez typique :

```
prog5
begin
        repeat
                x:=x+1;
                a:=a+a;
                writeln(a)
        until x=4
end.
```

On remarque l'apparition d'une nouvelle variable (a) qui est un indice significatif de ce qu'il y a une prise de conscience de la nécessité d'une seconde variable qui doit jouer le rôle de l'accumulateur. Mais le passage de ce programme à la solution correcte n'est pas non plus automatique. Par exemple, Hervé et Pierre essayent longtemps de modifier leur programme (prog5). Le programme (prog 6) est encore une étape de leur travail :

```
prog6
begin
        a:=a+1:
                repeat
                        a:=a+1;
                        a:=a+b;
                        a:=a+c;
                until (a=300);
        writeln(a);
end.
```

la mise en place de l'accumulateur est une étape...

Cependant nous considérons ce type de programme (prog6) comme une étape importante : les élèves ont déjà l'idée d'une structure qui peut conduire à la solution. L'introduction d'une troisième variable est peut être un indice de la réapparition d'une stratégie d'addition "par paires" - mais cette fois-ci à l'intérieur de l'itération. D'ailleurs, le déplacement de l'instruction **writeln** en dehors du corps de l'itération, atteste de ce que les élèves considèrent déjà l'itération comme une partie de leur programme - et donc le problème de son articulation avec "le monde extérieur" se pose.

L'introduction d'un accumulateur constitue une étape importante dans le processus de résolution du problème. Pour donner un exemple, voici le protocole de Sandrine :

...on a A plus B...A plus C...A plus... A plus la somme...la variable qu'on avait calculé avant

Cet extrait montre que Sandrine a saisi le caractère invariant de l'accumulateur. Elle arrivera rapidement à un programme correct.

... importante mais difficile

Si la plupart des programmes cités ci-dessus comportent un compteur qui est correctement utilisé, en revanche l'utilisation correcte de l'accumulateur semble être beaucoup plus difficile pour les élèves. Cette observation a déjà été faite dans la plupart des travaux sur la programmation. Mais, dans notre expérience, il est possible que l'utilisation correcte du compteur ait été favorisée par son introduction par l'enseignant au cours précédent (introduisant justement de l'itération).

Dans le tableau qui suit (fig. 1) nous présentons les principales étapes au cours de la résolution pour chacun des binômes [2]. Les colonnes du tableau correspondent aux stratégies des élèves de la façon suivante :

I Reproduction de valeurs locales, sans itération et sans aucun invariant

II Structures "généralisables", calcul de valeurs par manipulation des variables, sans itération

III Itération avec compteur seul

IV Itération avec "sommation par paire"

V Itération avec compteur et accumulateur

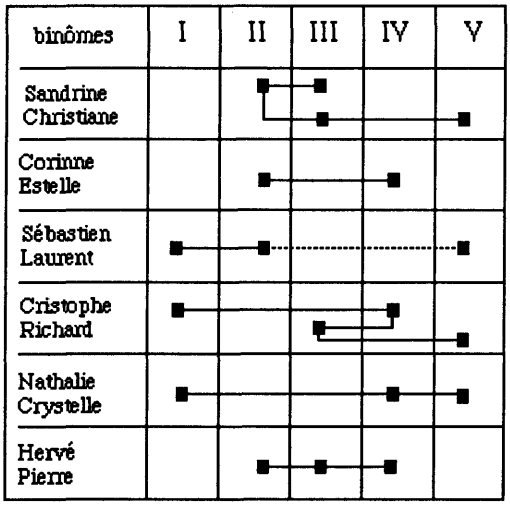

figure 1

(2) Le trait pointillé sur la ligne de Sébastien et Laurent signifie que nous ne possédons pas les traces intermédiaires.

- L'anthropomorphisme

Tout au long de cette activité nous avons détecté beaucoup d'erreurs qui ressemblent à celle que nous présentons ci-dessous du binôme Hervé et Pierre :

```
a:=a+b;
readln(a);
a:=a+b;
```

beaucoup d'erreurs sont liées à un modèle anthropomorphique

Il semble qu'ici la variable a peut contenir à la fois plusieurs valeurs: la valeur "lue" avec READLN et la somme a+b. Pour Hervé et Pierre une affectation ne détruit pas nécessairement une valeur antérieure dans la variable a. Il s'agit d'une erreur qui peut s'expliquer avec l'aide du modèle anthropomorphique : une seule variable peut jouer plusieurs rôles à la fois, elle peut contenir plusieurs valeurs simultanément et le dispositif, comme s'il s'était muni de capacités humaines, sait les distinguer. Nous avons nommé ce type d'erreur, erreur des variables polyvalentes (Soloway 1982 les nomme *mushed variables*).

une seule variable pouvait contenir plusieurs valeurs!

D'autres programmes sont plus difficiles à analyser. Le programme de Corinne et Estelle en est un exemple :

```
...
x:=x+1;
y:=x+x;
z:=y+x;
writeln(z);
...
```

Pour ces élèves les deux x ne sont pas identiques, chacun joue un rôle différent (sinon elles auraient mis 2x à la place de x+x). Cependant quand elles simulent l'exécution du programme sur papier, elles disent explicitement :

...x plus 1 ça fait 1...y égal x plus x...ça fait 2

certaines erreurs sont plus difficilement explicables

Il est difficile d'interpréter cette conduite : ces élèves comprennent l'effet des affectations mais en même temps elles attribuent un rôle particulier à la variable x. Ce type de programme a déjà été signalé par Soloway (op. cit.) qui a eu les mêmes difficultés que nous pour l'interpréter.

6.3. Conclusion

en résumé :

Notre expérience confirme des hypothèses déjà émises par d'autres (Mejias 1985) : les élèves débutants n'utilisent pas spontanément l'itération comme un outil pour résoudre le problème. D'ailleurs la découverte de ce que l'itération est la "bonne" structure ne constitue pas la seule étape importante que les élèves doivent franchir pour arriver à la solution. Il reste :

l'élève débutant n'utilise pas spontanément l'itération en Pascal...

- le découpage en actions élémentaires en fonction des capacités du dispositif;
- la "découverte" des variables qui jouent le rôle du compteur et de l'accumulateur, et celle des invariants que leurs interrelations doivent exprimer;

- une coordination éventuelle du corps de l'itération avec le test d'arrêt et le "monde extérieur" (i.e. le reste du programme).

Les stratégies des élèves peuvent être regroupées de la façon suivante :

Il développe d'autres stratégies : - sommation de valeurs numériques - sommation par paires

- recherche de sommations qui ne prennent en compte aucun invariant : ce sont en fait des valeurs numériques qui interviennent et non des variables. Mais ce genre de construction caractérise surtout les premières phases de la résolution, on pourrait donc les considérer comme éléments de phases exploratoires;
- des constructions qui probablement sont dûes à des conceptions du type sommation par paires : "prendre un nombre, ajouter le suivant et ainsi de suite...". Cette conception apparaît assez massivement. Il est cependant rare que cette stratégie soit suivie jusqu'à ses ultimes conséquences : la réplication n'est apparue que chez un seul binôme. La plupart des élèves préfèrent, par un procédé d'essais-erreurs, tenter des modifications de leurs "écritures" jusqu'à ce que les résultats obtenus soient satisfaisants.

Les conceptions que nous avons qualifiées d'anthropomorphiques apparaissent nettement. C'est surtout les conceptions du type "variables polyvalentes" qui nous semblent typiques dans l'expérience que nous avons menée.

7. MULTIPLAN

7.1. Introduction[3]

Multiplan est un tableur...

Multiplan est un tableur : il permet le traitement de grands ensembles de nombres. Les calculs sont réalisés automatiquement grâce à des formules définies par l'usager. Il peut, d'un certain point de vue, être considéré comme un langage de programmation. Cependant il se distingue nettement du langage Pascal par un certain nombre de caractéristiques.

- Les formules et les cellules

... c'est-à-dire un tableau comprenant des cellules...

Multiplan affiche un tableau constitué de cellules où l'utilisateur peut inscrire des nombres, des textes ou des formules. Le logiciel inscrit dans les cellules, après calcul, les résultats

(3) Pour la lecture de cette partie du travail nous supposons que le lecteur a déjà une certaine connaissance du fonctionnement d'un tableur et de Multiplan en particulier. Cette partie de notre travail a un caractère essentiellement exploratoire, car il y a encore très peu de travaux de recherche sur ce sujet. Nous donnons dans ce qui suit une brève description de certaines caractéristiques de Multiplan qui nous servirons comme appui pour la suite. Pour une analyse plus complète des caractéristiques du dispositif Macintosh/Multiplan, nous renvoyons le lecteur à (Capponi et als 1989).

déterminés par les formules. Ces résultats sont des nombres, des booléens, des textes ou des messages d'erreurs.

... référencées

Les formules sont des expressions construites à partir des règles formelles du calcul algébrique (priorités, parenthèses...) et des désignations des cellules. Il y a plusieurs types de désignation des cellules (références absolues, relatives, désignation par nom) qui donnent un caractère particulier à la notion de variable dans Multiplan. Le type retenu pour l'enseignement en question ici était celui des références relatives [4].

ce sont les références relatives qui seront utilisées ici

Les formules ont une signification algébrique mais ce caractère algébrique ne va pas de soi pour les utilisateurs débutants. La priorité donnée à l'affichage des résultats par le logiciel privilégie une interprétation des écritures comme description de calcul au détriment d'une interprétation en terme de relations entre les cellules (Capponi et als 1989).

une cellule peut contenir des formules...

Mais d'un point de vue informatique, les formules peuvent être considérées comme des instructions d'un langage de programmation impératif mettant en œuvre plusieurs notions fondamentales comme la variable, l'itération et la notion même de programme. Ainsi les cellules désignent à la fois le "lieu" d'écriture des instructions, le "lieu" d'affichage des résultats et les "variables".

... auquel cas elle contient aussi le résultat des calculs !

Par ailleurs l'exécution du calcul dans Multiplan suit un ordre déterminé, les formules doivent donc être considérées en tenant compte de leur position dans le tableau et, en ce qui concerne l'itération, l'instant où l'exécution a été interrompue.

le calcul dans un tableau a un aspect dynamique

Ces caractéristiques donnent au logiciel un aspect dynamique : les calculs sont exécutés à "l'avant de la scène" - le programmeur peut les "voir" - et chaque modification d'une expression entraîne automatiquement (au moins par défaut) un recalcul des résultats - de tous les résultats - sans que l'utilisateur relance une commande d'exécution.

- L'itération

la forme d'itération retenue dans Multiplan...

Elle présente, elle aussi, des particularités dans Multiplan, par rapport à l'itération en MacPascal.
Elle nécessite la mise en place :
- d'un calcul circulaire, c'est à dire d'un ensemble de cellules (éventuellement réduit à une) qui "s'appellent" mutuellement.
- du choix de la commande de calcul "itérer"
- d'un contrôle réalisé à partir d'une formule, test qui produit un booléen dans une cellule servant de cellule test (elle provoque l'arrêt quand le booléen est égal à VRAI).

... utilise la fonction NBITER ( )

La formule test retenue dans notre cas était "= NBITER( ) =a" qui affiche le booléen VRAI quand la fonction NBITER( ) donne un nombre d'itérations égal à l'entier a. NBITER( ) est une fonction qui pourrait d'ailleurs fonctionner comme un compteur "classique" à condition de gérer le message d'erreur, #NA, qu'elle

(4) Par exemple L(-2)C(+4) désigne la cellule placée deux lignes au dessus et 4 colonnes à droite, relativement à la cellule où est inscrite cette formule.

produit au moment de son édition, avant le déclenchement de l'itération.

... pour exprimer la condition d'arrêt

Dans Multiplan le corps de l'itération est constitué de toute la feuille de calcul. Chaque étape de l'itération réalise un recalcul de toutes les cellules de la feuille. La condition d'arrêt est ici l'ensemble constitué de la formule "=NBITER()=a" et des commandes nécessaires au choix de la cellule test qui contient cette formule. Mais il n'y a pas d'itération sans la présence de calcul circulaire dans la feuille.

Les actions nécessaires pour lancer et contrôler une itération sont assez complexes en raison du mélange de commandes à mettre en œuvre et de formules à éditer dans les cellules.

Nous avons déjà expliqué pourquoi nous avons voulu introduire l'itération dans Multiplan. Le choix de ce type particulier d'itération, malgré sa complexité, est justifié pour deux raisons :

a) il est très proche du type d'itération utilisé en Pascal, et,
b) malgré ses inconvénients, il est l'un des plus simples parmi ceux qui sont possibles dans Multiplan.

7.2. Eléments d'une analyse

le problème de la somme des 300 premiers entiers est posé aux élèves dans l'environnement Multiplan

La solution d'un problème du type de celui que nous avons proposé aux élèves passe nécessairement par un découpage en actions élémentaires - comme en Pascal. Notre analyse (cf. paragraphes 4, 5 et 6.2.1) reste valable dans ses grandes lignes. On doit cependant noter les particularités suivantes :

(i) L'existence d'une commande, directement accessible, "recopier vers le bas" rend très facile la réplication d'une formule et donc permet théoriquement l'adoption de stratégies différentes. Nous estimons cependant que cette solution est assez coûteuse pour deux raisons :

- puisque chaque colonne du tableur contient 255 lignes, pour calculer la somme des 300 nombres il faut utiliser au moins deux colonnes qui sont, en quelque sorte, "enchaînées"; ceci nécessite une gestion complexe des cellules;
- à chaque nouvelle valeur de N il faut repréciser le nombre de lignes nécessaires pour le calcul.

une solution coûteuse : la replication

Ainsi la possibilité de procéder par réplication exige une connaissance de Multiplan assez avancée et par conséquent nous ne pensons pas que des solutions de ce type puissent être proposées par des élèves novices.

(ii) L'interface particulière de Multiplan, notamment la possibilité de recalcul automatique pour toute modification d'un contenu de cellule, lui donne un aspect dynamique. On peut penser que cet environnement favorisera les stratégies de type essai-erreur - puisque elles coûtent relativement peu cher - surtout chez les débutants.

une hypothèse : l'environnement Multiplan favoriserait les stratégies par essai-erreur

(iii) La priorité accordée aux résultats centre les élèves sur ces résultats plutôt que sur les relations (formules) qui les justifient.

(iv) Le système des messages d'erreur étant peu transparent pour l'utilisateur débutant, on peut attendre de difficultés

d'interprétation des messages du dispositif de la part des élèves; par conséquent leurs réactions risquent d'être peu cohérentes avec la signification de ces messages.

7.3. Présentation de l'activité

La consigne donnée aux élèves était de calculer la somme des trois cent premiers entiers.
Ils ont pu réaliser ce travail au cours de deux séances consécutives, à une semaine d'intervalle, un premier bilan de l'avancement du travail étant dressé au début de la deuxième séance.

7.4. Analyse des observations

- Les étapes de la construction

Les élèves n'ont pas abordé le problème de la même façon. Certains binômes ont commencé par la construction d'un "générateur d'entiers" du type "=LC+1" dont ils étaient convaincus qu'il leur serait utile pour la solution -d'ailleurs ils savaient déjà comment le construire (au cours précédent, l'activité proposée par l'enseignant étant justement la construction d'un "générateur d'entiers"). Par contre d'autres binômes se sont centrés très vite sur le problème de la construction des avertisseurs de l'itération et du test d'arrêt, en mettant dans une cellule une instruction du type =NBITER( ).

générateurs d'entiers

utilisation de NBITER( )

A partir de là il y a une encore plus grande diversification des stratégies des élèves. Très souvent d'ailleurs (beaucoup plus qu'en Pascal) les élèves procèdent par une recherche empirique : à partir d'une formule qui produit un certain nombre de résultats - pour les petites valeurs numériques - ils tentent par essais successifs d'adapter la formule sans pour autant prendre en compte sa signification algébrique. Ils construisent ainsi diverses formules "*pour voir*", comme ils le disent parfois. Ainsi Hervé et Pierre essayent des formules comme les suivantes :

des formules "pour voir"

=NBITER( )+1
=NBITER( )=LC+(LC+1)
=NBITER( )=LC+1

mais ces modifications sont apportées sans une analyse des contenus des cellules, ni d'ailleurs une analyse des messages d'erreur que le dispositif renvoyait pour chacune de ces expressions.

des erreurs importantes...

On a pu observer qu'au cours de la construction il y a eu une "régression" des conduites des élèves vers des procédures que l'on pourrait qualifier de moins élaborées. Plus particulièrement la formule "=LC+LC+1" semble avoir joué un rôle extrêmement important pour l'ensemble des élèves puisque tous les groupes sont à un moment ou à un autre passés par l'écriture de cette formule sous l'une ou l'autre des formes :

=LC+LC+1 ou = LC + (LC + 1)

Au cours de la séance de bilan organisée par l'enseignant, les élèves ont échangé leur point de vue sur cette formule. Ces échanges permettent d'interpréter ce phénomène :

Sébastien : (cette formule) *ajoute deux fois ce qu'il y avait dans la cellule et ajoute 1*

Pierre : *on peut mettre LC+1 entre parenthèses*

Sébastien : *ça ne sert à rien*

... vont être analysées avec l'enseignant

La proposition de Sébastien est issue d'une analyse de l'expression qui prend en compte sa signification algébrique. C'est-à-dire que LC+LC+1 est bien interprété comme 2*LC+1. En revanche Pierre veut indiquer, à l'aide des parenthèses de LC+(LC+1), deux temps dans le calcul. La proposition qu'il fera ensuite d'enlever le signe "+" confirme cette interprétation (cf. à ce sujet Capponi 1990).

Il est clair que certains élèves, comme Pierre, attribuent à cette formule un sens plus large que le sens imposé par Multiplan. On peut avancer l'interprétation suivante :

Dans la formule LC+(LC+1) qui devrait ajouter des nombres consécutifs :

LC est l'accumulateur

LC+1 est le compteur

+ (devant le parenthèse) signifie "verser le compteur dans l'accumulateur sans affecter le compteur";

( ) sont des séparateurs : pour que l'ordinateur puisse distinguer le compteur et l'accumulateur.

le cas de "LC + LC = 1"

Donc il s'agit d'une expression qui a, pour les élèves, un caractère dynamique : non seulement LC peut contenir plusieurs valeurs mais en plus il y a plusieurs actions à la fois et le temps est pris en compte (l'après et l'avant). Il est d'ailleurs évident que, pour au moins quelques élèves, la prise de conscience du fait que leur propre interprétation de la formule soit en contradiction avec les règles du fonctionnement du dispositif nécessite une longue élaboration intellectuelle - le dépassement de cette interprétation coûte assez cher en temps et par le nombre d'essais erronés. Il est donc légitime de faire l'hypothèse que la conception sous-jacente (la conception des expressions dynamiques, des expressions qui "contiennent tout à la fois") est assez profondément enracinée.

On peut aussi envisager d'autres interprétations : cette formule représenterait la somme de deux entiers consécutifs, LC et LC+1, dont la somme est mise dans la même cellule : cette interprétation est d'ailleurs plus proche des conceptions des élèves qui correspondent à une "sommation par paire". Néanmoins la formule ne produit pas le résultat voulu puisque, dès la première itération, LC ne contient plus l'entier suivant mais la somme des deux premiers.

Un deuxième élément des conduites des élèves qui nous permet de faire allusion à une résurgence des conceptions moins élaborées, est le suivant : les élèves ont fait très souvent appel directement aux compétences du dispositif. L'existence, pour ne donner qu'un exemple, d'une liste de fonctions préfabriquées

les élèves cherchent souvent dans le dispositif une fonction prédéfinie permettant de résoudre le problème

à l'intérieur même du logiciel, les a induit à chercher dans cette liste pour trouver une fonction pour le calcul de la somme d'entiers consécutifs. Cette recherche d'une fonction ou commande "qui existe quelque part" et qu'il suffit d'appliquer pour obtenir le résultat voulu, est apparue de façon significative et avec une certaine régularité. Nous devons remarquer que le problème dans ce cas n'est pas celui de l'existence d'une telle possibilité car en réalité elle existe : fonction "somme". Mais cette fonction peut s'appliquer uniquement à un certain ensemble de cellules qui contiennent déjà des résultats - elle n'est donc utilisable qu'avec une partie de la feuille qui contient déjà les nombres à ajouter. En revanche les tentatives des élèves montrent que ce qu'ils cherchent est plutôt une fonction qui pourrait accepter comme arguments deux valeurs numériques (disons 1 et 300) et qui fournirait comme le résultat la somme des entiers compris entre ces deux valeurs.

- L'utilisation de NBITER( )

la fonction NBITER( ) détournée...

La fonction NBITER( ) a été souvent utilisée comme un "compteur" ou comme un "accumulateur", en voici quelques exemples :

=NBITER(1+2+3+4+5)=300

ou

=LC(+1)+300+NBITER( )=300

ou même

=NBITER( )=299
=LC+1
=L(-2)C+1

... pour l'utiliser à la fois comme moyen de calcul et comme test d'arrêt

D'une manière générale NBITER( ) intervient assez fréquemment dans la construction de formules comme une expression qui permettrait de combiner à la fois le calcul de la somme et le test d'arrêt. Il est probable que le nom même NBITER peut inciter les débutants de lui attribuer une signification plus "large" que sa signification réelle.

- Les invariants

peu de programmes sans invariant...

Nous n'avons observé que très peu de programmes sans invariants comme celui de Hervé et Pierre :

=NBITER(1+2+3+4+5)=300

... ou réduits à un simple compteur

Les programmes qui ne contiennent qu'un simple compteur (=LC+1) sont aussi assez rares.
Par contre les programmes qui produisent des nombres "en paire" sont très fréquents. Ils ne se placent cependant pas tous au même niveau. Ainsi les programmes suivants fonctionnent comme des "doubles compteurs" sans aucune accumulation :

mult1	mult2
=LC+1	=L(+1)C+1
=LC+1+1	=L(-1)C+1

Par contre d'autres programmes comme les deux suivants prennent en compte la nécessité d'accumuler les valeurs :

=LC+1
=L(-1)C+L(-1)C+1

ou

=LC+1
=L(-1)C
=L(-2)C+L(-1)C

des programmes qui produisent des nombres "en paires"...

Ce type de programme est, en général, apparu, assez tard. Ils sont peut-être la conséquence du débat organisé par l'enseignant au début de la deuxième séance autour la signification des formules.

Notons que la conception des nombres "en paire" est très persistante et nous paraît très proche des programmes, déjà observés en MacPascal, qui produisent la somme de deux entiers consécutifs. Ces formules présentent un intérêt particulier puisque deux binômes sur les quatre qui ont résolu le problème sont passés par ce stade. Elles montrent la prise de conscience, par les élèves, du fait que les contraintes du dispositif exigent l'utilisation de plusieurs cellules pour ajouter des nombres consécutifs. C'est un progrès important dans la résolution du problème, vu l'importance que les élèves attribuent initialement aux formules du type =LC+(LC+1). Néanmoins cette prise de conscience n'aboutit pas automatiquement à la solution correcte. En effet, deux binômes ont échoué sans qu'on puisse savoir si ce sont des problèmes de temps ou un obstacle très important qui en soit la cause.

... sont fréquents et très persistants

Le tableau qui suit (fig. 2) donne un résumé des principales étapes par lesquelles sont passés les élèves pendant la première séance. Pendant la deuxième, les procédures utilisées par les élèves sont nettement améliorées (Capponi et als 1989). Nous avons tendance à attribuer ce phénomène à la phase de bilan (au début de la deuxième séance).

Les colonnes du tableau correspondent aux stratégies des élèves de la façon suivante :

I Construction d'un générateur d'entiers "=LC+1"

II Construction d'une cellule pour le contrôle de l'itération "=NBITER( )=a"

III Formule "=LC+LC+1"

IV Appel à des compétences de la machine

V Formules faisant intervenir la fonction NBITER( ) dans le calcul de la somme

VI Construction d'une cellule contenant l'entier suivant le dernier nombre produit par le générateur d'entiers.

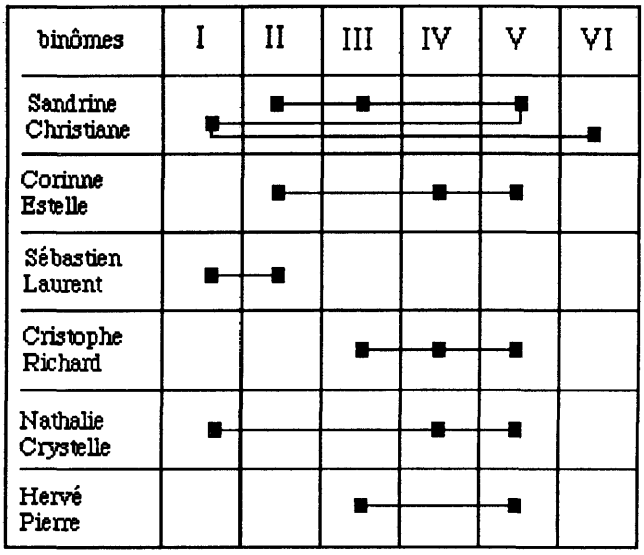

figure 2

- L'anthropomorphisme

Le type d'erreur qui caractérise les productions des élèves est "l'instruction-procédure" en combinaison avec les "variables polyvalentes". Nous avons déjà mentionné et commenté la formule :

=LC+(LC+1)

qui a été interprétée par les élèves comme une procédure qui incrémente simultanément le compteur et ajoute sa nouvelle valeur dans l'accumulateur.
Ajoutons que nous avons pu observer d'autres manifestations anthropomorphiques comme celle de Christophe et Richard, pour citer un exemple, qui interprètent les booléens VRAI-FAUX, fournis par la formule de la cellule test, comme un commentaire sur la pertinence de leurs actions.

7.5. Conclusion

Comme nous l'avons déjà souligné, cette partie de notre recherche avait essentiellement un caractère exploratoire. Néanmoins nous estimons que l'on peut établir les résultats suivants :

- les expressions dans Multiplan peuvent être étudiées à la fois comme des expressions algébriques - mais qui ont leurs propres règles de syntaxe par rapport à la syntaxe algébrique classique - et comme des commandes ou des instructions d'un langage de programmation. Pour la résolution du problème proposé, les deux aspects doivent être pris en compte à la fois. Or cette nécessité de prendre en compte plusieurs registres à la

fois peut être considérée comme assez complexe, pour le débutant, et par conséquent peut être une source de difficultés pour les élèves.

dans Multiplan l'aspect dynamique favorise la recherche empirique...

- Ce que nous avons appelé l'aspect dynamique du logiciel en combinaison avec les particularités de son interface (messages d'erreurs, existence de fonctions facilement accessibles, facilités d'entrées sorties) peut favoriser :
  - la recherche empirique, c'est-à-dire une recherche des "bonnes formules" à l'aide des résultats obtenus, par adaptation d'expressions déjà écrites dans le but d'obtenir des résultats déjà calculés à la main; la priorité que Multiplan donne aux résultats peut être à l'origine de ces recherches empiriques qui se font au détriment d'une analyse des relations entre les cellules et leur interprétation ;

... et l'émergence de certains types d'anthropomorphismes

  - l'apparition ou l'émergence d'un certain type d'anthropomorphisme, celui qui est probablement lié à l'aspect dynamique du logiciel : la persistance de formules du type =LC+LC+1 atteste la présence d'une telle conception. Nous remarquons aussi l'existence de recherche de fonctions préfabriquées, donc l'exploration des compétences du dispositif, comme un indice qui va probablement dans cette direction.

- Nous avons aussi remarqué l'existence et la mise en œuvre de procédures du type d'addition "en paire" (réparties en une, deux ou trois cellules), phénomène que nous interprétons comme une tentative de transfert des procédures d'autres domaines dans un environnement de programmation où elles ne sont plus valides.

8. DISCUSSION GÉNÉRALE

Nos analyses à propos des deux activités étudiées, nous permettent de tirer un certain nombre de conclusions relativement aux questions que nous nous sommes posées.

le réinvestissement de Pascal vers Multiplan est faible...

D'une certaine façon le réinvestissement des connaissances acquises en Pascal a été très faible dans Multiplan. Nous avons tout d'abord remarqué qu'aucun élève n'a essayé de reprendre les procédures ou les méthodes déjà utilisées en Pascal dans le nouvel environnement; le problème était "oublié", en quelque sorte, par les élèves. Ce fait oblige à s'interroger sur la signification que les élèves ont attribué à ces activités. Malgré l'importance que nous leurs avons attribué et le statut que l'enseignant leur a donné - pendant les diverses phases de bilan dans les deux contextes - nous pouvons estimer que la connaissance acquise par les élèves était relativement "faible". Il est probable que les deux activités ont eu des effets différents de ceux attendus, qu'ils n'ont pas été considérés comme des savoirs mais plutôt comme une sorte de savoir faire dans les deux contextes. Ceci montre que la complexité cognitive de l'itération - même sous des formes "simples" - est peut être plus grande encore que ne l'on avait cru. Si ce qui a été "retenu" par les élèves n'avait

... ce qui montre la complexité cognitive de l'itération

qu'une valeur "locale", ceci nous laisse relativement désarmés face à la question de l'acquisition de connaissances de ce type - surtout quand il s'agit de formations de courte durée.

Si l'on veut tirer de conclusions plus précises des deux expériences on peut faire les remarques suivantes.

en conclusion...

- Dans les deux contextes il y a des conceptions, quant à la construction du corps de l'itération, qui sont très fréquentes et persistantes. Nous faisons allusion surtout à des conceptions du type "nombre en paire" que nous avons interprété comme des tentatives de transfert de procédures valables - mais dans d'autres contextes. Leur apparition dans les deux contextes nous font penser que l'on peut envisager l'hypothèse qu'elles sont plutôt intrinsèques au problème proposé et indépendantes de l'environnement utilisé.

des conceptions intrinsèques au problème posé

- De même pour la conception de l'anthropomorphisme on peut conclure que son apparition, sous diverses formes, peut être indépendante du contexte. Ceci est en fait l'hypothèse de Pea. Nous sommes cependant enclins à penser que ce sont les formes d'anthropomorphisme qui sont différentes dans les contextes différents. Si dans Pascal nous avons détecté surtout des conceptions du type "variables polyvalentes", en revanche dans Multiplan les deux types que nous avons mentionné, variables polyvalentes et "instructions concentrées", sont apparues massivement. Nous avons attribué cette différence à l'aspect dynamique du logiciel. Il est donc probable que dans un environnement (autre que Multiplan), qui possède cette caractéristique, l'apparition de cette conception sera également observée - et de façon significative.

des formes d'anthropomorphisme différentes

- Les caractéristiques particulières de Multiplan ont probablement joué sur les stratégies adoptées par les élèves. En fait, dans Multiplan nous avons observé beaucoup plus de démarches que nous avons qualifiées "d'empiriques". La facilité des entrées-sorties, l'affichage prioritaire des résultats et le fait que le résultat est obtenu automatiquement dès qu'il y a une modification d'une cellule, semblent favoriser cette démarche. Par ailleurs le fait que les divers messages du logiciel sont difficilement interprétables par des débutants, a aussi contribué à l'adoption de recherches empiriques - un message incompréhensible ne peut constituer un feed-back pertinent. Quoi qu'il en soit, cet élément nous semble très important quant aux choix que l'enseignant doit faire : si l'activité proposée à des élèves débutants peut aboutir sans une analyse des éléments qui interviennent, l'enseignant risque d'obtenir davantage de résultats à partir d'essais-erreurs que fondés sur une analyse des relations entre les cellules.

l'influence de caractéristiques particulières de Multiplan

Vassilios DAGDILELIS
Nicolas BALACHEFF
Bernard CAPPONI
Équipe de didactique des mathématiques et de l'informatique
Laboratoire LSD2, IMAG
Université Joseph Fourier, Grenoble

RÉFÉRENCES BIBLIOGRAPHIQUES

BROUSSEAU G. (1986) *Théorisation des phénomènes d'enseignement des mathématiques*, Thèse d'état, Université Bordeaux I.

CAPPONI B., (à paraître), *Calcul algébrique et programmation dans un tableur, le cas de Multiplan*. Université de Grenoble 1.

CAPPONI B., BALACHEFF N. (1989) "Tableur et calcul algébrique", *Educational Studies in Mathematics* 20, pp. 179-210.

HOC J. M. (1979) "Le problème de la planification dans la construction d'un programme informatique", *Le Travail humain*, 42, n° 2, pp. 245-260.

LABORDE C., BALACHEFF N., MEJIAS B.(1985) "Genèse du concept de l'itération: une approche expérimentale", *Enfance* 2-3, pp. 223-239.

MEJIAS B. (1985) *Difficultés conceptuelles dans l'écriture d'algorithmes itératifs chez les élèves du collège*, Thèse de Doctorat, Université de Grenoble I.

PEA R.D. (1984) "Language-independent conceptual "bugs" in novice programming", *Journal of educational computing*, special issue on "novice programming", pp. 1-12.

ROGALSKI J. (1985) "Alphabétisation informatique, problèmes conceptuels et didactique", *Bulletin de l'association des professeurs de mathématiques de l'enseignement public* , 347.

ROUCHIER A., SAMURÇAY R., ROGALSKI J., VERGNAUD G. et coll (1984) *Concepts informatiques et programmation. Une première analyse en classe du seconde des lycées*, CEPL, IREM d'Orléans.

SOLOWAY E., EHRLICH K. (1984) "Empirical Studies of Programming Knowledge", *IEEE Transactions of Software Engineering*, SE-10 (5), pp. 595-609.

SOLOWAY E., EHRLICH K., Greenspan J. (1982) "What do novices know about programming", *Directions in Human-Computer Interactions*, Shneiderman B. and Badre A. Eds, 1982, Ablex Publishing Co.

SPOHRER J.C. et SOLOWAY E. (1986) "Novice mistakes : are the folk wisdoms correct ?", *Communications of the ACM*, July 1986, vol 29, N° 7, pp. 624-632.

VERGNAUD G. (1981) "Quelques orientations théoriques et méthodologiques des recherches françaises en didactique des mathématiques", *Recherches en Didactique des Mathématiques*, vol 2, La Pensée Sauvage, Grenoble, pp. 215-232.