Robotique - Informatique : mêmes ébats, mêmes débats, mêmes combats ?

Conférence présentée au 4ème Colloque de Robotique Pédagogique, Liège, juillet 1993

Charles DUCHATEAU
CEFIS
Facultés Universitaires N-D. de la Paix
rue de Bruxelles, 61
B-5000 NAMUR
☎ +32 81 725060
FAX +32 81 725064
E-Mail cduchateau@cc.fundp.ac.be

1. Introduction

Ce n'est pas d'abord une comparaison d'un point de vue "pédagogique" entre l'univers de la programmation classique et celui de la robotique que je souhaite entamer ici, même si l'apprentissage et la didactique seront partout en filigrane dans mes propos. Je voudrais oser le mot "épistémologique" pour préciser le point de vue qui sera le mien, cette vision étant nourrie d'une lente découverte personnelle des traits essentiels de l'informatique d'une part et d'une expérience presque aussi longue des difficultés, pièges et embûches de son enseignement d'autre part. Je montrerai qu'*au fond* la programmation informatique classique (impérative ou procédurale) et la robotique procèdent d'une démarche conceptuelle commune et peuvent être analysées à travers une même grille théorique.

Par ailleurs, je montrerai que, du point de vue des activités cognitives permises, l'exercice de la robotique pédagogique peut être vu comme un précurseur facilitant l'appropriation de certains des savoir faire et modes de pensée de la programmation et comme une étape particulièrement adaptée et sémantiquement riche de la maîtrise de celle-ci. Ceci n'est pas un hasard puisque, rétrospectivement, toute l'approche et les recherches didactiques que j'ai menées ces dix dernières années en ce qui concerne l'apprentissage de l'algorithmique s'alimentent d'une vision qui place celle-ci fort près de ce qu'on appelle aujourd'hui la robotique pédagogique. En d'autres termes, j'ai tenté de faire de l'apprentissage de la programmation la découverte, l'exploration et la maîtrise d'un micro-monde particulier, à l'instar de Logo et des activités robotiques.

Enfin, informatique et robotique ont en commun cette potentialité de faire éclater les cadres étroitement disciplinaires, de faire passer au second plan l'acquisition de connaissances pour privilégier l'exercice d'une démarche méthodologique. Surtout, en objectivant la pensée organisatrice de l'apprenant à travers le reflet fidèle que constitue son interprétation par un dispositif exécutant, la robotique, comme la programmation, rendent possible la métaréflexion qui constitue l'un des buts essentiels de tout apprentissage[1] visant au développement de l'autonomie des apprenants[2].

2. Le concept de problème est commun à l'algorithmique et à la robotique

Qu'y a-t-il donc de commun entre le "problème" de "calculer une moyenne", celui de "l'écriture en toutes lettres d'un nombre", celui du "tracé d'un hexagone", ou encore celui du "tri de caisses de deux tailles différentes"[3] ?

On pourra commencer par rétorquer que dans ces énoncés, une partie importante est absente et qu'il faudrait peut-être ajouter qu'il s'agit d'un "calcul de moyenne" *en Pascal*, de "l'écriture en

1 "... le support à une démarche «épistémologique » qui permet, par une rétroaction soutenue et concrète, de nous faire réfléchir sur notre façon de planifier, de contrôler et d'exécuter une tâche complexe" (Nonnon 87)

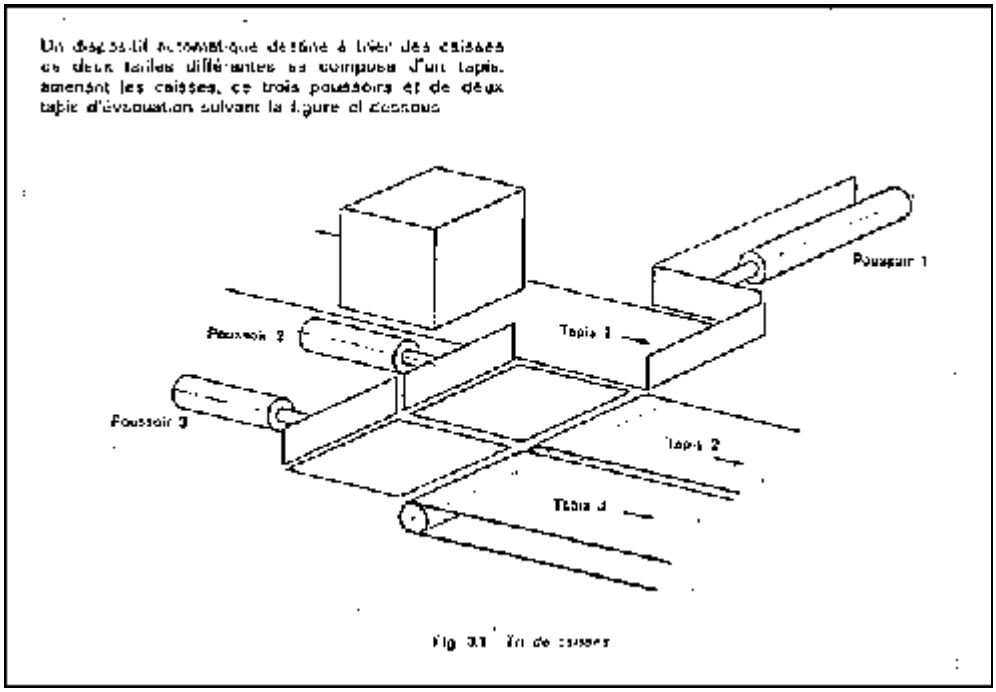
2 Je voudrais remercier Anne Gilbert pour sa lecture attentive de la première version de cette communication et pour la pertinence de ses remarques et suggestions.

3 Enoncé dans (Bossy 79), page 38, dont est tiré le schéma présenté ci-dessous.

toutes lettres d'un nombre" *en Basic*, du "tracé d'un hexagone" *en Logo* et du "tri de caisses de deux tailles différentes" *en Grafcet*.

Un point commun saute alors aux yeux : dans chaque cas on mentionne un langage "de programmation", à travers lequel le "problème" doit être traité. Mais pourquoi parle-t-on de "problème" pour des tâches aussi élémentaires que le calcul d'une moyenne ou le tri de deux types de caisses sur base de leur taille ?

C'est la considération de l'énoncé complet du "problème des caisses", comme d'ailleurs celui de n'importe quel problème de robotique qui va nous amener à l'essentiel :



Le "problème" n'est évidemment pas de "trier les caisses *manuellement*", mais *d'automatiser* ce tri, de commander le dispositif automatique de tri décrit ci-contre, autrement dit de ***faire trier*** ces caisses par le dispositif. Ainsi, très clairement, à chaque fois qu'un "problème" de robotique (ou de commande d'automate programmable) se présente, quel qu'il soit, il s'agit, non d'effectuer une tâche, mais de la ***faire faire par un dispositif exécutant*** ad-hoc.

On est d'ailleurs en droit de se demander pourquoi souligner avec cette insistance le fait que le problème consiste à automatiser une tâche par la commande d'un robot qui va s'en charger, tant ce dernier est au coeur de l'énoncé, dans le cas de la robotique. Je pense en effet que, contrairement aux termes "algorithmique" ou "programmation" qui n'appellent guère d'images et ne contiennent pas d'éclairage particulier quant au sens qu'y prend le terme "problème", le mot "robotique" révèle le terme "robot" et fait clairement apparaître qu'un problème y est évidemment lié à la commande d'un dispositif exécutant.

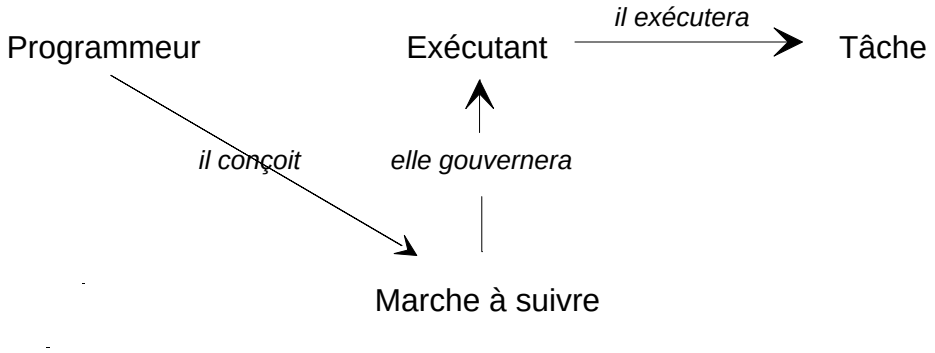
Cette perception d'un "***faire faire en différé***" et la prise en compte des contraintes du dispositif exécutant, tellement immédiate et naturelle, lorsqu'on parle de robotique[4] est par contre souvent occultée lorsqu'il s'agit de la programmation (au sens classique) et de son apprentissage. Ainsi, on trouve fréquemment dans les ouvrages pour débutant consacrés aux débuts de l'initiation à la programmation des énoncés comme "donner la date de demain, sachant quel jour nous sommes aujourd'hui", "écrire un mot à l'envers", "trier par ordre alphabétique une série de noms",...[5]. A chaque fois, la tâche évoquée ne donne lieu à un problème que parce qu'il s'agit de la faire faire par un dispositif exécutant : l'ordinateur équipé d'un langage de programmation.

4 "Au contraire de l'informatique, qui est difficile à appréhender dans son ensemble sans une connaissance minimale des concepts de programmation sous-jacents à son fonctionnement, le caractère anthropomorphique et opératoire des systèmes robotisés se prête assez bien, presque sans pré-requis, à une appréhension directe et globale." (Nonnon 87)

5 "La multiplication des exemples renforce le sentiment qu'on se trouve là face à une série de tâche ineptes et fastidieuses mais qui ne méritent certes pas de se voir qualifier de problèmes. Et l'injonction de l'enseignant demandant «d'analyser le problème » qui consiste à «déterminer le plus grand de trois nombres » ou (plus difficile !) de «vérifier si un texte est un palindrome » n'amène en général chez les apprenants, incapables de

Trop souvent, l'inventaire des contraintes et capacités de cet "exécutant-ordinateur" se contentait d'une description du langage de programmation associé en un mélange absolument inadéquat de sémantique et de syntaxe, sans faire percevoir les grands traits de l'exécutant sous-jacent, indépendants du langage (procédural) considéré. Si l'apprenti-programmeur finissait par se forger des représentations mentales des contraintes et capacités d'un dispositif exécutant, c'était en quelque sorte à travers les modes d'expression permis par les langages de programmation. Beaucoup continuent aujourd'hui encore à dire que "programmer", c'est "faire résoudre un problème par l'ordinateur", entretenant ainsi la confusion entre la *tâche* (et chacun sait qu'on ne parle pas de la "solution d'une tâche") et le *problème* de la déplier, de la morceler en constituants élémentaires qui doivent correspondre aux capacités d'un dispositif exécutant, constituants qui doivent être organisés en une marche à suivre traitable par cet exécutant.

En effet, il apparaît clairement que tant en ce qui concerne la robotique que la programmation classique, le schéma suivant[6], qui explicite le "faire faire en différé" mis en évidence, est au coeur du concept de problème dans ce contexte particulier :



Le problème, qu'il s'agisse de programmation impérative classique, d'activité Logo, de robotique, de commande d'automates programmables ou de conduite de machines-outils à commande numérique est dans ce fossé entre un "faire" et un "faire faire" : dans chaque cas, il s'agit

- face d'une part à une tâche (qu'il s'agira de bien définir et de préciser),
- face d'autre part à un exécutant aux capacités limitées mais connues (et "adapté" à la tâche spécifiée),
- de rédiger la *marche à suivre* qui permettra de faire faire cette tâche par cet exécutant.

On sait qu'il est parfois difficile de tracer une frontière entre simulation et robotique, entre expérimentation assistée par ordinateur et commande d'automate, entre utilisation d'un logiciel et programmation : je proposerais volontiers que ce critère du "faire faire en différé" soit l'un de ceux qui permettent de mesurer la part ou le degré de "programmation" présent dans des activités qui mettent en oeuvre l'ordinateur; jusqu'où une activité reste-t-elle un "faire avec...", dans quelle mesure devient-elle un "faire faire..." (Cf. (Pair 88)).

3. Le portrait nécessaire de l'exécutant

S'il est vrai que le schéma présenté ci-dessus est une manière d'unifier "robotique" et "informatique", il faut admettre qu'à la fois les tâches retenues et les dispositifs exécutants chargés de les mener à bien (quand ils disposeront du programme adéquat) sont divers et dissemblables. Que peut il y avoir de commun entre un tour à commande numérique, la tortue Logo, les "macros" permises par un tableur, un bras manipulateur Fischertechnik et l'ordinateur équipé d'un compilateur Pascal ? La réponse à cette question demande qu'on prolonge le travail d'unification entrepris. Cette question peut d'ailleurs être réécrite autrement : que doit comporter la description d'un exécutant pour que sa programmation soit possible[7] ?

---

deviner en quoi ces travaux stupides peuvent donner lieu à une «analyse », qu'un malaise embarrassé. Même la question «comment fais-tu ?» est le plus souvent inutile ou insensée, tant l'exécution des besognes évoquées est immédiate et inconsciente." (Duchateau 92)

6 Ce schéma est à rapprocher à la modélisation tripolaire des situations d'interactions sujet-réel médiatisées par un instrument proposé par P. Rabardel. (Rabardel 89)

7 "Pour qu'un problème soit bien posé en programmation il faut non seulement que la tâche soit correctement précisée, mais encore que l'exécutant qui sera chargé de son exécution soit complètement décrit : sinon il est

Sans nier les différences entre ces divers "exécutants" (et sur lesquelles je reviendrai), une grille commune peut cependant être proposée; elle comportera :

3.1. La description de l'exécutant et de son "environnement de travail"

C'est ici qu'en quelque sorte on crée le *lien entre la tâche à automatiser et l'exécutant* qui en sera chargé. C'est cette description qui va *donner sens* au problème de programmation en montrant que l'exécutant décrit peut (moyennant la mise à sa disposition de la marche à suivre adéquate) venir à bout de la tâche.

Ainsi, s'agissant de la tortue d'écran Logo[8], la *tâche* étant de dessiner[9] on précisera que le triangle apparaissant à l'écran représente une "tortue" susceptible de se *déplacer* en laissant (ou non) une *trace colorée* de son trajet; que la tortue est susceptible de *tourner* sur place; etc. Bref on fera une description générale des possibilités de l'exécutant-tortue qui montre qu'il n'est pas déraisonnable de l'atteler à la tâche de dessiner.

Dans le cas de la programmation impérative classique, la description de l'exécutant-ordinateur classique (celui de la programmation "Pascal", "Basic", "Fortran", ...) cédera le pas à une description métaphorique de son "environnement de travail". Cette description est évidemment plus malaisée et j'y viendrai ci-dessous.

Notons qu'en général, s'agissant de robotique, il est souvent possible, et c'est là un avantage pédagogique appréciable, de *montrer* le dispositif-exécutant (de faire toucher, manipuler,...le robot, d'expliquer le rôle de ses composants, et même parfois de le construire) et de le faire agir[10]. De plus, les comportements possibles du robot lui-même ou son environnement de travail sont souvent naturellement liés à la tâche évoquée : le robot a parfois été explicitement construit pour réaliser une tâche spécifique (Cf. le dispositif de tri des caisses présenté ci-dessus).

Tout ceci pour signaler que la description ou la découverte de l'environnement de travail d'un robot est bien plus immédiate que lorsqu'il s'agit de dépeindre celui de l'ordinateur perçu comme exécutant (à travers la programmation impérative habituelle).

3.2. La liste des actions dont l'exécutant est capable et des instructions correspondantes

Les marches à suivre dont la conception (puis la rédaction) est au coeur de l'activité de programmation (au sens large) vont essentiellement comporter des ordres, des commandes, des instructions d'action pour le dispositif exécutant. Le coeur de ces instructions est le plus souvent un verbe (à l'impératif ou à l'infinitif), plus rarement un nom désignant une action, qui s'adresse à l'exécutant et correspond à une action dont il est capable.

Ainsi,

- "Avance poussoir 1" ou "Recul poussoir 3" pour le dispositif "trieur de caisses";
- "Descend", "Monte", "Repos",... pour la commande d'un ascenseur ou d'un monte charge[11];
- "LeveCrayon", "Avance 10", mais aussi, malheureusement, "Droite 30" ou même "Cache Tortue" pour l'exécutant tortue de Logo[12]

---

tout bonnement impossible d'écrire la marche à suivre qui doit forcément s'appuyer sur la connaissance des caractéristiques de celui-ci." (Duchâteau 92)

8 On aura compris que c'est évidemment à dessein que je ne retiens de LOGO que la partie du langage lié à la commande de la tortue, puisqu'il s'agit là d'une merveilleuse métaphore de l'univers de l'algorithmique tel que je l'envisage ici. Tout le reste de LOGO (traitement de listes, ...) échappe à mon propos (et échappe même en grande partie au paradigme impératif pour relever davantage du paradigme fonctionnel).

9 et le problème de "faire dessiner"

10 "...AV 10, c'est dans un contexte donné, la production d'un trait;... mais c'est aussi, dans une approche physique, envoyer des pulses aux moteurs des deux roues de la tortue. La nécessaire réponse à la question ci-dessus nécessite donc que l'on puisse «ouvrir les boîtes », qu'elles soient matérielles ou bien aussi logicielles." (Vivet 89)

11 Les dispositifs comme grues, monte-charge, ascenseur,... sont souvent rencontrés et présentés comme des micro-mondes à explorer et à modéliser (Cf. par exemple Gauthier 86)

12 On peut regretter que le langage, même en Logo, ne traduise pas toujours clairement une *instruction* (pourquoi pas "PivoteADroiteDe 30" plutôt que "Droite 30" -c'est tellement vrai que certaines implémentations prévoient TourneDroite-) ou même que l'instruction donnée s'adresse à "autre chose" que la tortue : pourquoi pas "CacheToi" plutôt que "Cache Tortue" qui s'adressant à "quelque chose" d'intermédiaire entre l'apprenti-programmeur et l'exécutant-tortue multiplie les interlocuteurs; dans ce contexte et au début des apprentissages, l'ordinateur et le "système" ont intérêt à se faire oublier.

- write('Donnez votre nom'), read(Nom),... pour l'exécutant-Pascal (Cf. plus bas).

Il est donc impératif de disposer, l'environnement de travail donnant sens à ce qui est mentionné, d'un lexique "Instruction → Action" qui décrive les ordres possibles à destination de l'exécutant et les actions provoquées et cela en insistant sur la sémantique des instructions plutôt que sur leur syntaxe.

3.3. La liste des conditions que l'exécutant est capable de tester

Chacun le sait, même si les formalismes sont divers et les modes de représentation variés, une "marche à suivre"[13] comporte à côté des instructions d'action, des instructions d'organisation de ces actions, ce que l'on appelle habituellement des structures de contrôle. Si le répertoire des instructions d'action dépend de manière cruciale de l'exécutant particulier commandé, ces structures organisatrices sont toujours les mêmes : séquence, alternative, répétition (sous diverses formes, y compris l'itération), branchement (et branchement conditionnel lorsqu'il se trouve combiné avec l'alternative), appel de procédure (simple ou récursif).

Les anciens styles de programmation, proches du langage machine et de l'assembleur (et de la programmation des calculatrices) mettaient en avant la séquence et le branchement; la programmation structurée préfère la séquence, l'alternative, les diverses formes de répétition et l'appel de procédure; une conception récursive (qui met assez naturellement sur le chemin d'une approche fonctionnelle) privilégie la séquence, l'alternative et l'appel récursif de procédure (ou souvent de fonction). Ces divers styles s'accommodent de modes de représentations adaptés : organigrammes, graphes de Nassi-Schneidermann, arbres programmatiques, grafcet,...

Mais, le plus important c'est l'existence, au sein de certaines de ces structures organisatrices, de *conditions* dont l'exécutant, le moment venu, doit être capable de décider si elles sont ou non vérifiées.

La liste, pour un exécutant, des conditions qu'il est capable de tester fait bien entendu partie du portrait de ce dernier. Ainsi, dans le cas du dispositif du tri de caisses, on trouvera dans la marche à suivre représentée ci-dessous dans le formalisme Grafcet[14] l'énoncé des conditions "petite caisse", "caisse devant poussoir 2",...

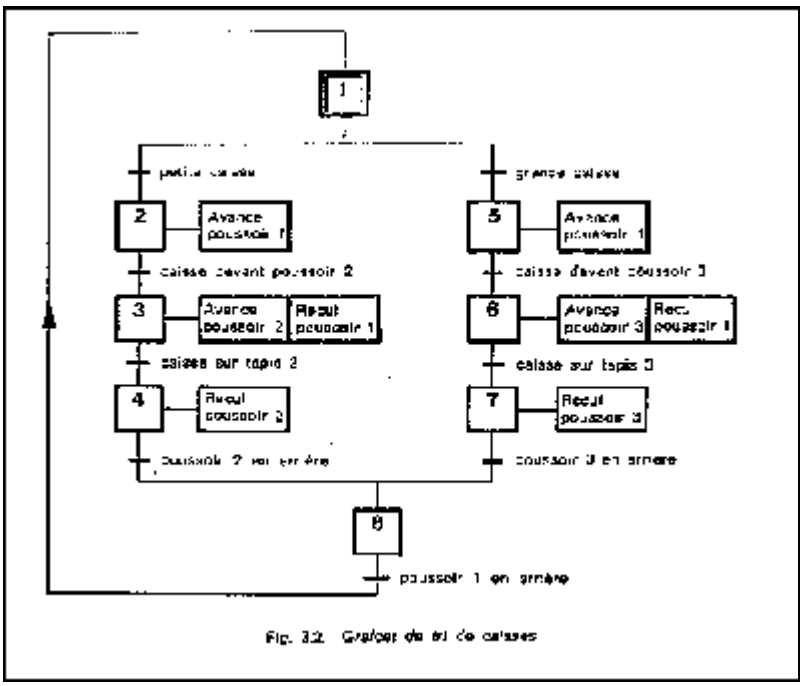


Fig. 3.2. Grafcet du tri de caisses

13 On peut également parler d'algorithme ou même de programme; mais le premier terme a des relents très "mathématiques" et sa connotation "absolue" passe complètement sous silence le fait qu'un "algorithme" n'est valable que pour un exécutant donné; le second met souvent en avant les caractéristiques syntaxiques d'un langage plutôt que le portrait de l'exécutant sous-jacent.

14 On aura compris que c'est seulement à titre d'exemple que la représentation GRAFCET est proposée ici; il existe bien d'autres modes de représentation des marches à suivre de commande de processus; l'important ce sont les concepts qui y sont, à chaque fois, présents.

Les structures organisatrices que constituent la séquence, l'alternative et le branchement apparaissent clairement dans le formalisme adopté.

- On verra ci-dessous qu'en programmation classique, les conditions seront toujours des comparaisons d'informations. En robotique, ce sont souvent les résultats transmis par les capteurs, les cellules photoélectriques. qui permettront l'expression des conditions.

Plutôt que de parler de conditions (qui sont ou non vérifiées), on peut également parler d'assertions (vraies ou fausses) ou même d'événement (pour sacrifier à une certaine terminologie orientée-objet).

Il est donc important, avant de débuter la programmation d'un dispositif exécutant donné, d'en dresser un portrait qui mette clairement en évidence ces trois éléments fondamentaux : l'environnement de travail, les actions élémentaires (et les instructions correspondantes) et les conditions testables (accompagnées éventuellement des structures de contrôle permises pour l'exécutant concerné).

Bien entendu, il n'est pas question, dans une saine perspective d'apprentissage, d'assommer les apprenants par des listes exhaustives et interminables, mais il est indispensable, lorsqu'une tâche à programmer est proposée, que les apprentis-programmeurs disposent d'un portrait adéquat du dispositif exécutant, ou que l'exploration menée leur permette de le découvrir et de le fixer.

4. L'exécutant-ordinateur : un "robot traiteur formaliste d'informations"

Si l'on fait le choix d'une présentation de l'apprentissage de la programmation qui s'inspire d'une approche "robotique", il est indispensable d'apporter une description de l'exécutant-ordinateur (= l'ordinateur, vu comme un robot à programmer) qui mette en avant les trois éléments évoqués ci-dessus. Il est évidemment hors de question d'en proposer ici un portait détaillé (Cf. Duchâteau 90 pour plus de détails). Brièvement :

4.1. Son environnement de travail

4.1.1. L'exécutant est un gestionnaire de tiroirs

Il dispose d'une série de "casiers" ou de "tiroirs" étiquetés. Un tel tiroir est caractérisé par :

- L'étiquette qui y est attachée : inamovible, choisie par le programmeur et non manipulable par l'exécutant. Cette étiquette constitue le nom du tiroir et servira dans le texte de la marche à suivre à désigner, soit le tiroir lui même, soit son contenu.
- Le contenu du tiroir : ce qui y transitera, c'est une information (nombre entier, nombre réel, chaîne de caractères) .
- Le type du tiroir qui détermine le genre d'information susceptible d'y prendre place (entier, réel, chaîne).

Il dispose également d'une énorme "malle à informations", dont il pourra, le moment venu et sur les injonctions de la marche à suivre le gouvernant, tirer n'importe quelle information de l'un quelconque des trois types cités.

4.1.2. L'exécutant peut "communiquer" avec l'extérieur

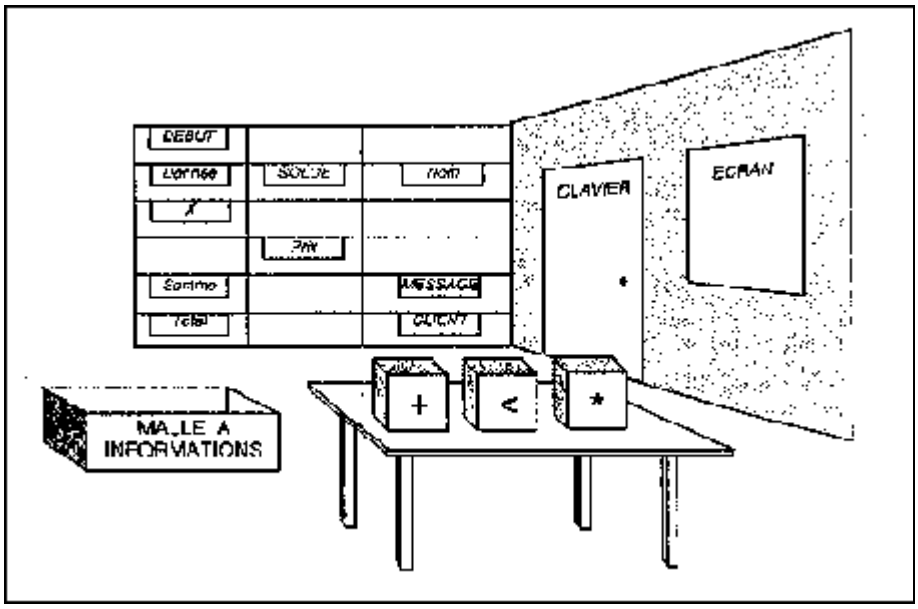
Son "local de travail" dispose en effet d'une porte-clavier à travers laquelle il peut recevoir des informations et d'une fenêtre-écran où il pourra en afficher.

4.1.3. L'exécutant dispose d'une table de travail portant un certain nombre d'outils

Ces outils vont fournir, pour peu que, le cas échéant, l'exécutant y entre des informations du type adéquat, une information nouvelle.

Ainsi il y a un outil SOMME, un outil DIFFERENCE, un outil NOMBRE AU HASARD,...

Parmi ces outils, il en est qui possèdent un statut particulier, ce sont ceux qui permettront au rédacteur des marches à suivre d'énoncer les *conditions* acceptables. Ce sont les outils qui fournissent le résultat de comparaisons d'informations : >, >, =, ...



4.2. Les instructions d'actions élémentaires

Elles sont seulement au nombre de trois :

- l'instruction de remplissage d'un tiroir (instruction d'affectation)
  Place telle information *dans* tel tiroir
- l'instruction de réception d'information de l'extérieur
  Va à la porte, attends l'information qu'on va t'y fournir et place la dans tel tiroir
- l'instruction d'affichage
  Affiche à la fenêtre telle(s) information(s)

Il reste à préciser ce que recouvrent dans deux de ces instructions les termes "*telle information*".

Une information pourra être écrite comme :

- une constante, comme dans
  Place 1 *dans* (le tiroir) Total, ou
  Affiche 'Bonjour'
- le nom d'un casier (c'est évidemment le contenu qui est alors désigné)
  Place (le contenu de) Somme *dans* (le tiroir) Total
  Affiche (le contenu de) Message
- le résultat fourni par un ou des outils (ce qu'on appellera une expression)
  Place (le contenu de) Somme + 1 *dans* (le tiroir) Somme
  Affiche (le contenu de) Somme - (le contenu de) Total

4.3. Les conditions

Ce sont les comparaisons d'informations permises par les outils cités, comme

Somme < Total
Nom = 'Dupont', ...

Des conditions plus élaborées pourront être construites grâce aux mots ET et OU.

4.4. Remarques

Le squelette auquel j'ai ci-dessus réduit la présentation a gommé bien des détails. Il me faut insister sur l'importance de quelques-uns de ceux-ci.

a. Je ne peux pas réellement montrer le robot-ordinateur ainsi décrit. Rien ne sert à ce propos d'ouvrir le capot de la machine ! Il faut donner à voir les notions présentées sous forme métaphorique. J'ai à cet effet réalisé et expérimenté une série de vidéos, porteuses d'images donnant corps aux métaphores employées. Ainsi, lorsqu'on *voit* que la "consultation" d'un casier ne le vide pas (c'est seulement une copie du contenu dont l'exécutant se saisit) mais

que par contre le remplissage d'un casier en chasse l'information contenue jusque là, bien des difficultés de manipulation relatives au concept de "variable" sont aplanies.

b. C'est toujours avec mauvaise conscience que nous nous éloignons des choix de vocabulaire et de notations consacrés par l'usage. Pourtant, il est peut-être préférable d'attendre que des images mentales pertinentes soient installées avant d'introduire les termes habituels. Ainsi, je préfère parler de "casier" ou de "tiroir" plutôt (*plus tôt*) que de "variable", de "remplissage" plutôt que "d'affectation", "d'étagère" plutôt que de "tableau",...[15]

   Il faut cependant signaler qu'une fois les images véhiculées par les périphrases et le vocabulaire installées, une expression plus concise sera adoptée. Mais elle se présentera non comme une contrainte extérieure mais plutôt comme une série de raccourcis ou de mnémoniques destinés à faciliter et fluidifier l'écriture.

c. Il est un trait important du robot-ordinateur sur lequel il faut insister, c'est son caractère "amnésique". Je ne peux lui parler qu'à l'impératif présent[16] sans jamais faire référence au passé : sa seule "mémoire", à tout instant de l'exécution, c'est le contenu *actuel* des tiroirs dont je l'aurai doté. Des termes comme "tantôt", des temps comme l'imparfait, sont à bannir.

d. Un reproche qu'on pourrait faire à cette peinture de l'exécutant-ordinateur, c'est son caractère très anthropomorphe. L'excès d'images et de métaphores est ici le remède même à ce reproche : je n'ai à ce jour rencontré aucun apprenant, même débutant, qui imagine un seul instant que c'est vraiment "comme cela que ça se passe dans l'ordinateur". Mais très vite tous comprennent qu'on peut "faire comme si...". Il est d'ailleurs impossible de concevoir un algorithme qui prenne en compte les capacités et contraintes d'un dispositif exécutant sans se projeter de quelque manière dans le monde de cet exécutant. Nous savons également qu'un certain nombre d'erreurs découleront de la tendance à prêter au dispositif des aptitudes, attitudes et jugements "humains".

J'ai insisté jusqu'ici sur ce qui faisait de la programmation et de la robotique des champs de pensée et d'activités semblables. Le moment est venu de mettre en évidence quelques différences essentielles

5. Les différences essentielles entre la robotique et la programmation classique

5.1. Différences conceptuelles

a. D'un point de vue strictement "technique", il faut remarquer que dans le cas de l'exécutant-ordinateur, les actions sont conceptuellement sans durée. Je ne veux pas dire par là que le temps d'exécution des actions demandées soit nul, mais plutôt que cette durée est sans importance pour l'écriture de la marche à suivre[17].

   Il y a, dans le cas de l'exécutant-ordinateur, un "avant" l'action commandée et un "après", pas de "pendant". Dans le cas des exécutants-robots, les actions ont souvent une durée, elles se déroulent. Plus symptomatique, lorsque le terme "jusqu'à ce que" est employé en programmation classique, il appelle toujours un "Répéter" : une action (ou un groupe d'actions) peut être répétée jusqu'à ce qu'une condition se trouve vérifiée, toute action est ponctuelle et ne peut, dans ce contexte, "durer" ou "se poursuivre" jusqu'à ce qu'une condition en interrompe le cours.

   Dans le cas des robots, le mouvement d'un poussoir, la rotation d'un bras sont des actions "continues" : il y a un "pendant" qui fait qu'on s'approche peu à peu de la réalisation d'une condition qui va en provoquer l'interruption. Ainsi, dans l'exemple du dispositif trieur de

---

15 J'ai jusqu'il y a peu, pour des raisons de conformité avec l'usage et dans un souci de concision, remplacé l'instruction d'entrée ou de réception d'information ("Va à la porte ...") par le raccourci "Lis et place dans ...". Cela donne bonne conscience, puisque, classiquement, on parle d'instruction de "lecture" et que bien des langages la codent en "read ...". Malgré toutes les précautions et avertissements cette "lecture" perturbe inutilement les débutants qui y attachent je ne sais quelle image mentale non pertinente. Plutôt que de continuer pendant des années encore à étudier les difficultés d'apprentissage causées (en partie du moins) par des environnements langagiers (entre autres) inadéquats, ne pourrait on consacrer une partie de cette énergie à étudier et tester des descriptions et des "langages" qui s'affranchissent enfin de ces expressions impropres ou trompeuses, de vrais langage non "de programmation" mais "pour apprendre la programmation"

16 Le temps qui conviendrait en réalité pour ce "faire faire en différé" n'existe pas : il faudrait un "impératif futur" !

17 Je ne parle pas ici de l'évaluation de l'efficacité des algorithmes qui prend en compte le temps d'exécution en distinguant par exemple le temps nécessaire à une addition et celui d'une multiplication.

caisses, "Avance poussoir 1" est une action qui va se dérouler et qui ne sera interrompue que lorsque la caisse sera *arrivée* devant le poussoir 2 (Cf. le Grafcet présenté plus haut).[18]

b. Le parallélisme (et la programmation parallèle qui l'accompagne) est beaucoup plus aisément abordable par le biais de la robotique : des actions peuvent avoir lieu simultanément. Même si des architectures informatiques vectorielles ont fait depuis un certain temps leur apparition, leur maîtrise reste actuellement du domaine des spécialistes et ne sont pas prises en compte dans une perspective d'apprentissage.

c. La construction même du robot (ou son assemblage) font parfois partie de l'activité de robotique pédagogique. Cette dimension technologique est importante et apporte à l'univers de la robotique des dimensions plus techniques. Dans le cas de l'exécutant-ordinateur de la programmation classique, on peut sans doute affirmer qu'une description comme celle qui précède (au point 4.) a pour but de chasser les détails techniques qui ne sont d'aucune aide pour la conception des programmes. Rappelons bien que l'exécutant-ordinateur est non l'ordinateur-machine, mais plutôt l'ordinateur transfiguré par sa compréhension (apparente) des programmes respectueux de la syntaxe d'un langage.

5.2. Différences en ce qui concerne l'apprentissage et les activités cognitives permises

C'est surtout dans ce domaine que la robotique pédagogique va se démarquer de l'apprentissage de la programmation :

5.2.1. La motivation

Même si l'on admet que le *problème* central de la programmation réside dans le "faire faire en différé" et non dans la *tâche* à informatiser, il faut reconnaître que, surtout dans les débuts, les tâches proposées aux apprenants sont *insignifiantes*, et cela dans les deux sens du terme : ce sont des besognes sans importance et elles n'ont en elle-même guère de sens pour ceux à qui on les propose. Les programmes confectionnés ne seront d'ailleurs généralement jamais utilisés (sauf pour les tester).

Cet écueil est redoutable à l'heure ou apparaissent de plus en plus d'outils logiciels utilisables (et utiles), puissants et conviviaux. Il est bien difficile au sortir de l'emploi d'un logiciel de dessin à travers lequel les élèves auront réalisé un produit dont ils ont raison d'être fiers, de leur proposer l'écriture d'un programme de recherche du maximum de trois nombres ou de conjugaison des verbes réguliers du premier groupe.

Dès lors, si les concepts essentiels de l'algorithmique sont importants et si l'exercice de la pensée planificatrice à laquelle ils conduisent est source d'apprentissages essentiels, il importe de proposer des approches non classiques de la programmation. La robotique pédagogique (dans laquelle je classerais ici Logo) en est une. Comme je le signalais plus haut, le problème de commande d'un dispositif exécutant est immédiatement au coeur d'un énoncé de robotique : il n'y a pas comme en programmation classique cet écart entre la tâche et le problème : on est d'emblée confronté au "faire faire".

De plus, l'exploration des possibilités du dispositif exécutant (du robot), la compréhension de ses caractéristiques, en un mot les représentations mentales qu'on peut s'en forger sont sans doute plus immédiates que dans le cas du robot-ordinateur.[19] Comme je le signalais ci-dessus, l'assemblage du dispositif à programmer fait parfois partie de la démarche proposée : nul doute que dans ce cas d'une part la motivation soit plus aisée et d'autre part la connaissance "de l'intérieur" du dispositif à programmer soit plus immédiate.

5.2.2. La facilité de vérification des programmes conçus

Qu'il s'agisse des dessins réalisés à travers la commande de la tortue Logo ou du résultat de la programmation de feux de circulation, d'un ascenseur ou d'une écluse, le résultat est visible et vérifiable. Le plus souvent on *voit* le robot (le dispositif exécutant) agir sous l'emprise de la marche à suivre qu'on a conçue et qui lui a été fournie. On peut suivre l'entièreté du

---

18 On m'objectera peut-être qu'à un niveau plus profond de description, le mouvement continu du poussoir correspondra à l'avance pas à pas d'un moteur, ce pas étant "indivisible". Tout dépend évidemment du niveau de description du robot, pertinent pour l'objectif poursuivi. Dans le cas de l'exemple, "Avance poussoir 1" est bien considérée comme une action élémentaire qui n'est pas elle même décomposée.

19 "... l'information est utilisée de manière plus concrète en microrobotique (à travers des poussoirs, tensions de commande de moteurs, valeurs analogiques de photorésistances, etc...), et de manière plus abstraite en informatique..." (Vivet 89)

déroulement du processus. La correction et l'adaptation des programmes concernés sont donc plus aisées.

Dans le cas de la programmation de l'exécutant ordinateur, une fois l'exécution commencée l'apprenti-programmeur se mue en utilisateur du programme qu'il a conçu. Et, chacun le sait, *rien n'est alors visible du travail de l'exécutant-ordinateur sauf ce que le programme a prévu de lui faire afficher à l'écran*. Dans un programme de calcul de moyenne, par exemple, rien du processus de calcul lui même n'est visible (on ne voit pas le casier-compteur s'incrémenter, le casier accumulateur se mettre à jour, ...). La seule lucarne dont on dispose sur l'univers de l'exécutant, c'est l'écran où ne viendront s'afficher que les résultats du processus et non le processus lui-même.[20]

5.2.3. L'exécutant ordinateur est un robot abstrait et compliqué

Même si des portraits métaphoriques comme celui présenté plus haut facilitent la découverte et la prise en compte des contraintes liées à l'exécutant dans le processus de programmation, les tâches de traitement de l'information auxquelles celui-ci est adapté restent par nature abstraites. Par ailleurs, les environnements de programmation classiques ne permettent guère une exploration en mode direct des capacités du robot-ordinateur. L'univers Logo par exemple permet une découverte progressive des réactions de la tortue aux ordres fournis : la tortue est présente à l'écran et on la voit en direct exécuter les instructions élémentaires fournies. C'est seulement dans un stade ultérieur qu'elle disparaît pendant que l'apprenant (à travers l'éditeur) rédige les premiers programmes. Cette familiarisation, en direct, avec l'univers de l'exécutant auquel on destine les marches à suivre est évidemment primordiale.

Les tâches pour lesquelles les robots sont construits sont aisément compréhensibles et repérables. Le robot *agit* (et réagit) et ces actions laissent des traces moins fugaces qu'un ensemble de caractères sur un écran. On a côte à côte en quelque sorte, le modèle du dispositif et le dispositif lui-même[21].

5.2.4. La robotique étend considérablement le champ d'applications possibles

Si les robots, du point de vue mis en avant ici, sont avant tout des dispositifs exécutants, ce sont, contrairement à l'ordinateur, des exécutants *en mouvement* qui, souvent, peuvent agir dans un espace, qui manipulent, qui "bougent",... Ils sont aussi dotés "d'organes" qui renseignent sur l'état de leur environnement : cellules, capteurs, ... C'est dire que le champ des tâches traitables ne se limite pas, comme dans le cas de l'ordinateur, aux seuls traitements (formalistes) d'informations. Par ailleurs les activités de robotique peuvent "donner corps" à ce concept d'information qui reste souvent une abstraction en programmation classique[22].

Enfin, il me semble que la commande et la programmation des robots garde un caractère ludique que la programmation classique est plus longue à manifester.

6. La robotique, étape de l'apprentissage de la programmation[23]

Deux difficultés essentielles sont au coeur de l'apprentissage de la programmation : celle d'abord liée à la difficulté de conception d'une marche à suivre et, particulièrement, à l'utilisation convenable des structures de contrôle (et la pré-vision mentale de l'exécution provoquée); celle aussi liée aux traits spécifiques de l'exécutant ordinateur gestionnaire formaliste d'informations et particulièrement à l'emploi des variables, avec la différence entre "étiquette" (ou "dénomination") et "contenu", et surtout la difficulté liée à l'instruction d'affectation (qui est au coeur du modèle d'ordinateur Von Neuman).

L'apprentissage de la programmation devra affronter (successivement ou simultanément) ces deux écueils. Une approche possible, qui privilégie dans la programmation l'aspect "rédaction

20 C'est tellement vrai qu'on a du inventer des programmes spéciaux, les "debugger", pour permettre de suivre pas à pas, de manière plus précise, le déroulement de l'exécution et pas seulement les affichages produits.

21 "... dans la robotique pédagogique ainsi conçue, le réel et le modèle se trouvent côte à côte, sur la même table..." (Vivet 89)

22 "La robotique pédagogique apporte incontestablement une extension du champ accessible à la modélisation, et un éclairage supplémentaire sur l'information et son traitement." (Lagrange 89)

23 "La programmation directe d'objets techniques manipulables devrait favoriser l'acquisition des primitives du langage et, par une plus grande facilité à définir et planifier l'ensemble des opérations complexes de cet environnement concret, permettre à l'élève un accès plus rapide à une programmation descendante et structurée." (Nonnon 87)

d'une marche à suivre", consiste à consacrer, avant la découverte des caractéristiques et contraintes de l'exécutant-ordinateur, un certain temps à la programmation de "robots" (malheureusement) imaginaires ou abstraits, décrits en terme d'environnement, d'actions primitives et de conditions "testables".

Beaucoup d'approches de l'algorithmique font ainsi de la programmation de ces "robots" un préalable ou une étape importante : Karel le robot (Pattis 81), Tobor le robot (Danloy 90), le "robot peleur de pommes de terre", le "robot au compte-gouttes" (Duchâteau 90), ...

Bien entendu, toute la facette "technologique" que la robotique pédagogique permet de rencontrer est absente de cette démarche : ce qui compte ici ce n'est pas de construire ou de découvrir de "vrais" robots, c'est d'apprendre à intégrer les contraintes d'un exécutant, qui pour un temps reste plus simple à imaginer et plus maîtrisable que l'exécutant-ordinateur de la programmation classique.[24] On isole en quelque sorte la seule difficulté de conception et de rédaction d'une marche à suivre et de l'emploi adéquat des structures de contrôle en minimisant les efforts à faire pour intégrer les contraintes venant de l'exécutant. Dans beaucoup de cas, les pseudo-robots à contrôler sont liés ou adaptés à des tâches de tracé de graphiques qui partagent avec celles de l'environnement Logo une grande facilité de les appréhender et de corriger les programmes correspondants.

Si l'on veut bien à l'issue de cette étape proposer une description de l'exécutant-ordinateur vu comme un robot, dans des termes comparables à ceux employés pour les "robots" déjà maîtrisés, on minimise le saut du passage à la programmation classique. L'idéal est d'ailleurs de disposer d'un environnement, comme il en existe un certain nombre, qui permette de visualiser, de manière métaphorique, l'activité du robot-ordinateur soumis aux premiers programmes qu'on lui destine, exactement comme cela se fait naturellement dans le cas de la robotique[25]. Malheureusement, ces environnements, le plus souvent ne permettent pas de découvrir le comportement de l'exécutant en mode "direct" : on ne peut donner des instructions et juger immédiatement de leur effet; on ne peut tester que de petits programmes, par le détours de leur écriture à travers un éditeur de texte.

Si l'on veut que la programmation classique devienne véritablement un micro-monde, il faudrait sans doute élaborer des environnements d'*exploration* des capacités de l'exécutant-ordinateur avant de proposer des environnements de *programmation* de ce même exécutant.

7. Tout cela, pour faire apprendre quoi ?

Je serais tenté de répondre par une boutade : "pour faire apprendre *rien*, donc *l'essentiel*", mais ceci demande probablement un mot d'explication.

Nous avons souvent au sein de l'école la tentation, face à des activités nouvelles, d'essayer de classer celles-ci à l'intérieur de l'une ou l'autre discipline traditionnelle. On a parfois, à tort, enfermé l'algorithmique dans les mathématiques; le même danger existe peut-être de classer la robotique pédagogique dans la physique ou le fourre-tout d'une certaine "initiation technologique". On perçoit pourtant que l'une comme l'autre échappent à ces classifications rudimentaires :

a. L'aspect "acquisition de connaissances" y est bien peu important. Les concepts y sont peu nombreux. Ces activités se prêtent fort mal (ou pas du tout) à de longs développements théoriques ou au drill : il est impossible de faire de la programmation ou de la robotique en se contentant d'écouter un enseignant en parler ou en répétant sans fin des exercices tous identiques ! Les mots qui se sont imposés sont clairs : on parle *d'activités* de robotique pédagogique, pas de *cours*[26]. Autrement dit, la programmation et la robotique sont de merveilleuses occasions de promouvoir le véritable EAO : des **E**lèves **A**cteurs grâce à l'**O**rdinateur.

---

24 Même s'il évoque ici la "vraie" robotique pédagogique, on peut certainement dans ce contexte reprendre la remarque de Pierre Nonnon : "Cette activité de planification ou de programmation concrète devrait (lui) permettre une transition plus aisée vers une pensée plus structurée, et faciliter ainsi une démarche plus abstraite comme l'apprentissage de la programmation structurée." (Nonnon 87)

25 L'environnement pour l'apprentissage de la programmation "Images pour programmer", outre les vidéos et leurs documents d'accompagnement et l'ouvrage écrit, comporte aussi un petit logiciel (START) qui permet de suivre, dans un environnement schématique correspondant à celui employé lors de la description de l'exécutant, le déroulement de l'exécution des premiers petits programmes conçus par les apprenants.

26 "L'accent sera placé sur la créativité plutôt que sur la répétition et la mémorisation" (Velasco 89)

b. Je n'insisterai pas sur l'exercice de la pensée *organisatrice* et *anticipative* permis par la programmation et la robotique : *pré-voir* (imaginer, voir à l'avance) est au coeur de ce "faire faire *en différé*". Je voudrais souligner cependant que ce sont des activités où la pensée créatrice prend forme (se formalise) à travers l'exécution par un dispositif externe d'une organisation qui a été conçue[27]. On a face à soi un dispositif qui *agit*, comme on l'a *pensé* et voulu, erreurs de conception comprises : il y a quelque chose de fascinant dans ce reflet matérialisé de sa propre pensée. Face à un robot à commander ou à l'exécutant-ordinateur à gouverner, c'est toujours avant tout à sa propre pensée qu'on est affronté.
c. Je viens de souligner que la programmation (classique ou celle des robots) est une merveilleuse occasion de métaréflexion. J'ajouterai également qu'on y rencontre des *problèmes* et non des *exercices* et que les *erreurs* y sont des *tremplins* et non des *fautes*...
d. Mon propos a pratiquement passé sous silence les aspects proprement technologiques liés en robotique à la conception des couches profondes de contrôle des dispositifs : conception et assemblage du "robot", interfaçage, ...Ces facettes peuvent aider à réconcilier formation générale et technologie et changer le statut assigné à tout l'univers *technique*.
e. Il resterait à évoquer l'interdisciplinarité "naturelle" de ces activités, l'approche inductive, les aller et retour "théorie-pratique" et "conjecture-vérification", ... Je préfère témoigner du plaisir et de la fascination que je continue à éprouver dans l'exercice de cette pensée algorithmique... Et si robotique et informatique permettait à ce plaisir de retrouver le chemin de l'école, ce ne serait déjà pas si mal...

8. Conclusions : des ébats et des débats, mais où sont les combats ?

J'espère avoir montré l'unité profonde de cet univers de l'algorithmique qui s'incarne dans l'automatisation des tâches et qui est au coeur de l'informatique : qu'il s'agisse, pour rester dans le domaine de la micro-informatique et de ses outils logiciels, de l'écriture de fichiers de commandes, de la rédaction de macroinstructions pour un tableur ou un progiciel de traitement de texte, qu'il s'agisse. encore, à travers la médiation d'un ordinateur, de programmer un ascenseur ou un bras articulé. La pensée algorithmique, celle du faire faire en différé, est centrale pour apprivoiser et maîtriser les environnements informatisés.

Mais ces activités sont difficilement classables et nouvellement arrivées au sein des écoles; elles nécessitent un matériel parfois onéreux (et vite dépassé); elles ont aussi (chez les observateurs non avertis ou de mauvaise foi) des relents ludiques. Pourtant, elles constituent une des manières exemplaires de changer au sein de l'école les relations au savoir et ses modes d'appropriation; elles sont un puissant levier pour modifier les pratiques pédagogiques et faire passer l'enseignant d'un rôle de haut-parleur à celui de gestionnaire d'environnements pour apprendre, de complice et de co-explorateur.

L'école est en crise : l'informatique et la robotique (comme bien d'autres technologies) ne sont assurément pas une panacée; elles peuvent cependant contribuer à entamer les remparts que l'école à souvent dressé pour s'isoler de la vraie vie, celle où les ordinateurs ont envahi nos bureaux et les robots nos usines...

9. Bibliographie

BLUME C., JAKOB W., FAVARO J. (1988)
PasRo et PasRo/C. Le Pascal et le langage C appliqués à la robotique
Masson, Paris, 1988.

BOSSY J-C., BRARD P., FAUGERE P., MERLAUD C.
Le GRAFCET, sa pratique et ses applications
Educalivre, Paris, 1979.

DANLOY B. (1990)
Syllabus du cours "Sytématique de la programmation"
Louvain-La-Neuve, 1990.

27 "Il s'agit en fait de créer des situations didactiques propices à produire chez les élèves des idées et à leur fournir une occasion de les explorer" (Velasco 89)

DUCHATEAU C. (1983)
Programmer. Pour une découverte des méthodes de la programmation.
Wesmael-Charlier, 1983.

DUCHATEAU C. (1990)
Images pour programmer.
De Boeck, Bruxelles, 1990.

DUCHATEAU C.
De «FAIRE» à «FAIRE FAIRE » : au coeur de la programmation. Quelques réflexions didactiques.
Publications du CeFIS, n° 5.30, Namur, 1992.

GAUTHIER A.
A propos de micro-mondes Logo
Publications du CUFIAP, Besançon, 1986

LAGRANGE J-B. (1989)
Robotique pédagogique dans une formation à l'informatique
Actes du 1er congrès francophone de robotique pédagogique, pp 104-113.

NONNON P, (1987)
La robotique pédagogique
Le BUS, mai 1987, pp 16-19.

PAIR C. (1988)
L'apprentissage de la programmation
Actes du Premier Colloque Francophone sur la Didactique de l'Informatique, pp 77-85
Editions EPI, Paris, 1988

PATTIS R. (1981)
Karel the Robot
John Wiley & Sons, New York, 1981

RABARDEL P. (1989)
Analyse de l'activité cognitive et modélisation des situations pour l'évaluation et la conception de robots pédagogiques
Actes du 1er congrès francophone de robotique pédagogique, pp 49-64.

VELASCO SANCHEZ R.(1989)
Un robot pédagogique pour l'apprentissage de concepts informatiques
Actes du 1er congrès francophone de robotique pédagogique, pp 115-128.

VIVET M., (1989)
Robotique pédagogique, soit... mais pour enseigner quoi ?
Actes du 1er congrès francophone de robotique pédagogique, pp 7-16.