Formation didactique des enseignants d'informatique : quels concepts utiliser ?

Sébastien Jolivet[1,2][0000−0003−3915−8465], Patrick Wang[3][0000−0003−3117−8189], and Eva Dechaux[0000−0001−8579−1398]1

[1] IUFE, Université de Genève, Suisse
[2] LDAR, Université de Paris, France
{sebastien.jolivet, eva.dechaux}@unige.ch
[3] Haute école pédagogique du canton de Vaud, Lausanne, Suisse
patrick.wang@hepl.ch

Résumé L'objet de cet article est d'illustrer la mobilisation, dans le cadre de la formation en didactique des enseignants d'informatique, de concepts développés en didactique des mathématiques. Nous avons retenu les notions de variable didactique et de registre de représentation sémiotique. Ceci nous situe dans le temps de la préparation d'une séquence d'enseignement et de la production ou l'adaptation de ressources.

Keywords: Didactique de l'informatique · Variable didactique · Registre de représentation sémiotique · Formation des enseignants.

1 Introduction

Depuis quarante ans, la place de l'informatique dans l'enseignement scolaire varie entre l'absence et la généralisation. Celà fait quelques années que l'enseignement de l'informatique au niveau secondaire (12 - 18 ans) est inscrit dans les curricula, aussi bien en Suisse Romande qu'en France. L'informatique peut être une discipline à part entière (lycée en France, secondaire 1 et 2 en Suisse) ou un objet d'enseignement intégré à d'autres disciplines (collège en France). Ces changements curriculaires induisent des besoins de formation d'enseignants pour les mettre en œuvre. Cette formation contient des apports transversaux et d'autres spécifiques au savoir enseigné, en particulier de la didactique de l'informatique.

Parallèlement, la littérature montre que la didactique de l'informatique est une question à la fois ancienne [1] et toujours d'actualité [6]. Les travaux qui interrogent l'enseignement de l'informatique selon un prisme didactique sont relativement peu nombreux (par exemple [8] sur l'algorithmique ou encore [2] sur l'enseignement de la programmation à la transition collège - lycée en France).

Il s'agit cependant de former des enseignants d'informatique dès maintenant en exploitant les travaux et résultats disponibles. Pour cela nous avons fait le choix d'exploiter des concepts développés dans les didactiques d'autres disciplines, en particulier les mathématiques, et d'interroger leur instanciation pour la didactique de l'informatique. Ainsi, cette contribution se veut pragmatique et

est essentiellement fondée sur une approche empirique construite lors de la formation d'enseignants d'informatique. Nous illustrons notre approche par deux concepts qui sont des moyens d'analyse, de modification, de sélection de ressources lors de la préparation d'une séquence : la notion de *variable didactique* et celle de *registre de représentation sémiotique*.

2 Les variables didactiques

La notion de *variable didactique* a été introduite par Brousseau dans le cadre de la théorie des situations didactiques [3]. Margolinas [7, p. 129] propose :

> "Une variable didactique est :
> — un élément de la situation sur lequel le maître peut agir,
> — qui provoque des changements qualitatifs dans les procédures de résolution des élèves,
> — qui permet d'expliquer les résultats de l'enseignement et d'agir sur eux,
> — et qui provoque une modification dans l'apprentissage."

Si la pertinence de cette notion a largement été établie dans la didactique de différentes disciplines, il nous semble que son exploitation en didactique de l'informatique est encore à construire. En particulier, il s'agit, à partir de l'identification de variables didactiques potentielles, de caractériser leurs effets sur l'apprentissage. Nous identifions ci-dessous plusieurs variables didactiques potentielles relatives à l'enseignement de l'algorithmique et de la programmation.

Nous proposons tout d'abord une première famille de variables didactiques, dans des situations d'apprentissage de la programmation, en lien avec le choix et le paramétrage de l'environnement de développement (IDE) proposé aux élèves [4].

Variables didactiques en lien avec l'IDE :

— l'activation / désactivation de l'auto-complétion et/ou de l'analyseur syntaxique. La désactivation de ces fonctions dans l'IDE va nécessiter de la part de l'élève l'apprentissage de la syntaxe et/ou la mise en œuvre d'une procédure de recherche de la fonction ou de sa syntaxe.
— (dés)activation du débogueur (ou utilisation obligatoire dans les consignes données, par opposition à l'utilisation de fonctions d'affichage de valeurs). Nous distinguons deux effets potentiels en termes d'apprentissages : meilleure compréhension de l'exécution d'un programme ; construire une représentation plus fine de la machine notionnelle qu'ils manipulent [4].

Nous proposons une deuxième famille de variables didactiques dans des situations de travail sur la programmation.

Variables didactiques en lien avec la limitation - l'interdiction d'un élément de la situation :

4. Nous n'interrogeons pas les contraintes institutionnelles et matérielles qui peuvent aussi avoir un effet sur le choix de l'IDE et les possibilités de le paramétrer.

— limiter nombre de lignes de code autorisées va permettre de rendre nécessaire l'utilisation d'une boucle plutôt que la réitération d'une même instruction. En choisissant d'introduire ou non cette limitation dans la situation l'enseignant dispose donc d'un moyen de contrôle des procédures pouvant être mises en jeu.
— limiter le nombre d'exécutions du programme va limiter la possibilité de mettre en place une stratégie du type essais-ajustements.
— limiter le nombre de variables autorisées peut permettre l'utilisation de fonctions ou de la récursivité.
— interdire la répétition de certaines lignes ou blocs de code dans un même script pour rendre nécessaire l'utilisation de fonctions.
— limiter le nombre d'instructions effectuées par l'algorithme permet à l'élève d'appréhender différentes stratégies de résolution d'un problème (gloutonnes, diviser pour mieux régner, etc).

3 Les registres de représentation sémiotique

Dans ses travaux sur les *registres de représentation sémiotique*, Duval [5] précise qu'un concept n'est accessible qu'au travers de ces différentes représentations sémiotiques, un concept pouvant admettre des représentations dans différents registres sémiotiques. Il propose ensuite que la compréhension d'un concept nécessite de le rencontrer dans différents registres et d'être capable de les articuler. Duval caractérise trois types d'activités relatifs à ces registres : la formation, le traitement et la conversion. L'articulation entre les différents registres nécessite en particulier de mettre en œuvre des activités de conversion.

Pour la formation d'enseignants d'informatique, la notion de registre de représentation sémiotique est un moyen d'analyser un corpus de ressources avec une attention à porter aux questions suivantes : "quels sont les différents registres de représentation sémiotiques proposés aux élèves pour une notion donnée ?" ; "quelles sont les activités proposées autour de ces différents registres ? En particulier, est-ce qu'il y a bien des activités de conversion qui sont proposées ?".

Nous illustrons maintenant l'utilisation des registres dans le cadre du travail sur l'algorithmique. Concept au cœur de l'apprentissage de l'informatique, l'algorithme admet plusieurs registres de représentation sémiotique. Le langage naturel, le pseudo-code et les logigrammes sont trois d'entre eux. Chacun de ces registres permet de mettre en évidence ou non certaines propriétés de l'algorithme et permet de décrire, plus ou moins facilement, certaines structures.

Par exemple, le langage naturel présente un fort pouvoir d'expressivité mais possède le défaut d'être parfois ambiguë. Le pseudo-code tente de pallier ce défaut, mais introduit parfois un formalisme symbolique (e.g. avec le symbole `:=` pour l'affectation) et/ou structurel (e.g. retours à la ligne et indentations) qui peut complexifier sa compréhension ou son utilisation. Enfin, les logigrammes permettent de visualiser schématiquement les séquences d'instructions ainsi que les branchements conditionnels mais utilise également un formalisme rigide et même standardisé. Le logigramme ne permet par ailleurs pas de représenter

de manière distinctes toutes les structures. En effet, il n'est pas possible de différencier en logigramme la boucle *Tant que* de la boucle *Pour*, ce qui entraîne souvent une difficulté pour les élèves de passer d'une représentation à l'autre.

Il est intéressant de noter que l'activité de conversion allant du pseudo-code ou du logigramme vers le code est usuellement proposée aux apprenants (passage algorithme vers programme). Par contre certaines activités de conversion telles que $code \rightarrow logigramme$ le sont nettement moins souvent. Pourtant, la lecture et la compréhension d'un programme sont sans doute des compétences importantes travaillées dans une telle situation. En effet, ces compétences peuvent ensuite être mises en application lorsqu'il s'agit de compléter ou déboguer un programme préexistant. De même la conversion $codelangage1 \rightarrow codelangage2$ ne nous semble pas très fréquente. La question de l'intérêt d'une telle activité est ouverte.

4 Conclusion et perspectives

La démarche ébauchée dans cette contribution est d'identifier, parmi les nombreux cadres théoriques et concepts développés depuis plusieurs décennies maintenant dans les didactiques de diverses disciplines, ceux qui peuvent être exploités avec pertinence pour la formation d'enseignants d'informatique, dans le cadre d'enseignements de didactique de l'informatique. Nous identifions au moins deux prolongements nécessaires à cette première approche : 1) Les exemples présentés reposent sur une approche essentiellement empirique. Une évaluation des effets sur l'apprentissage doit d'être menée pour confirmer ou infirmer ces premières observations. Cela nécessite la mise en place de véritable expérimentations. 2) La solidité de la démarche et la portée des résultats obtenus doit s'appuyer sur le développement de la didactique de l'informatique comme champ disciplinaire.

Références

1. Arsac, J. : La didactique de l'informatique : un problème ouvert ? In : Colloque francophone sur la didactique de l'informatique. pp. 9–18. Association EPI (1988)
2. Branthôme, M. : Apprentissage de la programmation informatique à la transition collège-lycée. Revue STICEF **28**(3) (2021)
3. Brousseau, G. : Théorie des situations didactiques : Didactique des mathématiques 1970-1990. La pensée sauvage Grenoble (1998)
4. Du Boulay, B. : Some Difficulties of Learning to Program. Journal of Educational Computing Research **2**(1), 57–73 (Feb 1986)
5. Duval, R. : Registres de réprésentation sémiotique et fonctionnement cognitif de la pensée. Annales de didactique et de sciences cognitives **5**, 37–65 (1993)
6. Fluckiger, C. : Une Approche Didactique de l'informatique Scolaire. Presses universitaires de Rennes (2019)
7. Margolinas, C. : Eléments pour l'analyse du rôle du maître : les phases de conclusion. Recherches en didactique des mathématiques **12**(1), 113–158 (1992)
8. Modeste, S. : Prendre en compte l'épistemologie de l'algorithme : Quels apports d'un modele de conceptions ? Quelle transposition didactique ? Recherches en didactique des mathématiques **40**(3), 363–404 (2020), publisher : La Pensée Sauvage