Futurs enseignants de l'école primaire : connaissances des stratégies d'enseignement, curriculaires et disciplinaires pour l'enseignement de la programmation

Béatrice Drot-Delange, Gabriel Parriaux, Christophe Reffay

Futurs enseignants de l'école primaire : connaissances des stratégies d'enseignement curriculaires et disciplinaires pour l'enseignement de la programmation

BÉATRICE DROT-DELANGE
Université Clermont Auvergne, ACTé

GABRIEL PARRIAUX
HEP Vaud, université de Paris, EDA

CHRISTOPHE REFFAY
Université de Franche-Comté, ELLIADD

RÉSUMÉ : Dans un contexte de reconfiguration disciplinaire de l'informatique à l'école primaire, cette étude porte sur la formation des futurs enseignants et sur leurs représentations dans le domaine de la pensée informatique, en particulier de la programmation. Issues de trois écoles (deux françaises et une suisse) de formation des enseignants, 135 séances produites par 283 étudiants ont été analysées à travers le prisme des connaissances professionnelles des enseignants concernant les stratégies d'enseignement et les connaissances curriculaires et disciplinaires. Nos résultats montrent qu'en termes de stratégies d'enseignement, la progression des activités et les choix des modalités pédagogiques restent impensées. L'analyse des connaissances disciplinaires montre des confusions entre condition et événement et entre programme et donnée. Ces constats sont discutés en termes d'enjeux didactiques et de formation.

MOTS CLÉS : enseignant du primaire, didactique, formation des enseignants, formation informatique, connaissances, programmation

ABSTRACT: In the context of disciplinary reconfiguration of computer science in primary schools, this study focuses on the professional development of preservice teachers and on their representations in the field of computational thinking, particularly programming. Coming from three universities of teacher education (two French and a Swiss one), 135 sessions produced by 283 students were analyzed through the prism of teachers' professional knowledge of teaching strategies, curricula knowledge and computational thinking content knowledge. When looking at teaching strategies, our results show that the progression of activities and the choice of teaching methods are not completely thought through. The analysis of computational thinking content knowledge shows confusion between condition and event and between program representation and data representation. These findings are discussed in terms of didactics and professional development issues.

KEYWORDS: primary school teacher, didactics, teachers training, computer science education, knowledge, programming

Introduction

Depuis une dizaine d'années, on assiste à une (ré)introduction de l'informatique dans les curricula de nombreux pays, d'abord dans l'enseignement secondaire, puis dans l'enseignement primaire (Baron & Depover, 2019). Ce phénomène initie une reconfiguration disciplinaire importante (Drot-Delange, 2018), puisqu'il s'agit de rompre avec la période précédente durant laquelle les compétences liées aux usages de l'informatique étaient centrales. Il s'agit donc d'instaurer les conditions dans lesquelles les enseignants vont pouvoir s'approprier cette reconfiguration de l'informatique à l'école primaire, après quasiment 20 ans d'ancrage sur les usages.

On peut supposer que la formation a un rôle important à jouer dans cette appropriation. Or les recherches actuelles sur la formation des futurs enseignants du primaire dans le domaine de l'enseignement de l'informatique sont rares, ainsi que le montre la revue de littérature systématique menée par Mason et Rich (2019). À cela s'ajoute le fait que les recherches en didactique de l'informatique ont suivi les fluctuations de la présence ou de l'absence d'un tel enseignement en milieu scolaire (Rogalski, 2015) et que l'on manque de résultats récents sur l'apprentissage par les élèves du primaire en informatique.

Nous contribuons en tant que formateurs et chercheurs en didactique de l'informatique dans des institutions en charge de la formation initiale des futurs enseignants en France (Instituts nationaux supérieurs du professorat et de l'éducation [INSPÉ]) ou en Suisse (Hautes Écoles pédagogiques [HEP]), à l'élaboration et à l'évaluation de ces formations, au sein d'équipes de formateurs, dans un cadre temporel contraint par les maquettes des formations et les déclinaisons locales.

Notre recherche[1], exploratoire, souhaite contribuer au repérage des connaissances nécessaires à l'enseignement d'un contenu spécifique donné, que Kermen et Izquierdo-Aymerich (2017) rapprochent des *Pedagogical Content Knowledge* (PCK). Dans cet article, nous analysons les productions des futurs enseignants concernant la préparation de séances dans les domaines de l'algorithmique et de la programmation, bien que l'informatique scolaire couvre un champ bien plus large, celui de « l'ensemble des manifestations des technologies informatiques ou de la science informatique dans le champ scolaire » (Fluckiger, 2019, p. 17).

Le contexte curriculaire dans lequel s'inscrit cette recherche est le suivant : en France, l'informatique s'inscrit depuis la rentrée 2016 dans les programmes scolaires via, entre autres, la production d'algorithmes simples et l'initiation à la programmation. Les programmes scolaires intègrent la programmation dès le cycle 2[2], puis au cycle 3, en les associant à des activités de repérage, de déplacement ou d'activités géométriques. Cet apprentissage est considéré dans les instructions officielles comme un préalable à la compréhension et à la production d'algorithmes.

En Suisse romande, une réforme des plans d'études est en cours afin d'introduire un enseignement de l'informatique jusqu'alors inexistant. Le système confédéral permet aux

1 Cette recherche bénéficie d'un financement de l'Agence nationale de la Recherche dans le cadre de la convention ANR-18-CE38-0008 pour le projet IE-CARE (Informatique à l'école : conceptualisations, accompagnement, ressources).

2 En France, les cycles d'apprentissage 1, 2 et 3 correspondent respectivement aux classes d'âges 3-5 ans (maternelle), 6-8 ans (CP-CE1-CE2) et 9-11 ans (CM1, CM2 et 6e). En Suisse, les cycles d'apprentissage 1, 2 et 3 correspondent aux classes d'âges 4-8 ans, 9-12 ans et 12-15 ans respectivement. En termes de correspondance, les élèves des cycles 1, 2 et 3 en France sont d'un âge comparable à celui des élèves des cycles 1 et 2 en Suisse.

cantons d'exercer une relative autonomie sur leurs systèmes éducatifs et le canton de Vaud a lancé depuis deux ans un projet pilote dans le domaine de l'éducation numérique. Un plan d'études encore en cours de révision prévoit que les élèves acquièrent des connaissances relatives à la programmation, à la représentation des données et aux machines, ainsi qu'aux enjeux sociétaux de l'informatique. En termes d'attentes fondamentales liées au domaine de la programmation, il est attendu que l'élève « exécute et crée des algorithmes simples » aux cycles 1 et 2, puis « exécute et crée des algorithmes variés à l'aide d'un langage de programmation visuel » dans la seconde moitié du cycle 2 et au cycle 3.

Dans une première section, nous présenterons le cadre théorique concernant les modèles de connaissances des enseignants déclinés dans le domaine spécifique de l'enseignement de l'informatique, puis nos questions de recherche. Dans une deuxième section, nous expliciterons la méthode et la constitution du corpus, pour exposer les résultats dans la troisième. Enfin, nous ouvrirons la discussion sur les questions que l'analyse des productions étudiées renvoie à la formation des enseignants.

1. Cadre théorique

Pour analyser les préparations de séance dans le domaine de l'algorithmique et de la programmation par de futurs enseignants, nous mobilisons l'approche des connaissances professionnelles des enseignants.

1.1. Une modélisation des connaissances professionnelles des enseignants

Les modèles les plus connus concernant les connaissances professionnelles des enseignants renvoient aux travaux de Shulman (2007). Ce dernier propose de distinguer plusieurs types de connaissances concernant le contenu d'une matière : la connaissance disciplinaire du contenu, la connaissance pédagogique du contenu (*Pedagogical Content Knowledge*, PCK) et la connaissance du curriculum. De nombreux enrichissements ont été apportés au modèle originel, modèle qui connaît des déclinaisons spécifiques aux disciplines. Chesnais, Cross et Munier (2017) ou Jameau (2017) citent celui de Magnusson, Krajcik et Borko (1999), le plus utilisé pour l'enseignement des sciences.

Selon ces derniers, les PCK s'appuient sur cinq composantes. Deux sont reprises du modèle originel de Shulman, trois y sont ajoutées. Les composantes reprises du modèle de Shulman sont les suivantes. La première est celle des connaissances des enseignants sur la compréhension des sciences par les élèves, leurs difficultés et les prérequis. La seconde est celle des connaissances sur les stratégies d'enseignement des sciences, qu'elles soient spécifiques aux sciences en général ou spécifiques à un sujet ou concept d'une science. Dans ce dernier cas, elles incluent la connaissance des moyens de le représenter pour en faciliter l'apprentissage, la connaissance de ce qui rend facile ou difficile l'apprentissage d'une notion, ainsi que des activités qui vont aider les élèves à la comprendre.

Les composantes ajoutées au modèle de Shulman sont les suivantes. La première composante, qui influence toutes les autres, englobe les connaissances et les représentations

concernant les buts et les objectifs de l'enseignement des sciences. Une deuxième composante est celle des connaissances sur les curricula en sciences incluant la connaissance des programmes et des matériels, des buts et objectifs des sciences. Cette composante inclut les connaissances de l'enseignant relatives aux contenus d'enseignement des autres matières et sa capacité à faire des liens avec les contenus de la matière enseignée à celles-ci. De même, il a des connaissances sur ce que les programmes prévoient concernant l'enseignement d'une notion à différents niveaux de scolarité. Une troisième composante est celle des connaissances sur l'évaluation, ce qu'il est important d'évaluer et les méthodes d'évaluation.

1.2. Les connaissances professionnelles pour enseigner l'informatique

Les recherches dans le domaine des PCK en informatique sont peu nombreuses et relativement récentes (Hubbard, 2018 ; Yadav & Berges, 2019) et concernent le plus souvent des enseignants de collèges ou lycées.

La revue de littérature systématique menée par Hubbard (2018) met en évidence deux tendances dans les modélisations des PCK en informatique. La première est celle où les travaux s'appuient largement sur les modèles existant dans d'autres disciplines pour proposer un cadre global. C'est le cas par exemple des travaux menés par Hubwieser *et al.* (2013), qui proposent une modélisation des connaissances professionnelles des enseignants d'informatique selon deux dimensions. La première est structurée selon trois étapes du processus d'enseignement (avant, pendant et après la leçon). Elle comprend la planification, l'interaction avec les élèves et l'évaluation. La seconde est structurée en cinq groupes, déclinés en 15 catégories, relatifs à la matière et au curriculum, aux méthodes d'enseignement et à l'usage de média, aux apprenants, aux enseignants et enfin au système éducatif. La grille ainsi constituée est mobilisée pour analyser les formations universitaires de futurs enseignants en informatique dans l'enseignement secondaire.

La seconde tendance est celle où les travaux contribuent à définir des PCK pour un sujet donné en informatique : par exemple l'usage pédagogique des métaphores dans l'enseignement, en étroite relation avec la nature métaphorique de l'informatique (Woollard, 2005) ou l'identification de la récurrence de certaines difficultés d'apprentissage chez les élèves, par exemple lors de la résolution de problèmes à l'aide d'une méthode de modélisation (Koppelman, 2008). Comme d'autres chercheurs (Delmas-Rigoutsos, 2018), Saeli (2012) identifie les idées fondamentales à enseigner en informatique, en se focalisant sur la programmation. Pour chacune d'entre elles, elle exploite la reformulation des PCK proposée par Howey et Grossman (1989), sous forme de questions. En déclinant ces questions pour une des idées fondamentales, par exemple les algorithmes, on obtient les formulations suivantes : que faut-il enseigner sur les algorithmes ? Pourquoi enseigner les algorithmes ? Quelles sont les difficultés d'apprentissage des algorithmes ? Comment enseigner les algorithmes ? Toutefois, les recherches visant à apporter des réponses à ces questions concernant l'enseignement primaire ne sont pas encore très nombreuses, du fait même de l'orientation donnée à cet enseignement pour ce niveau scolaire (Fluckiger, 2019).

Il existe en effet très peu de travaux concernant les connaissances que devraient développer les enseignants sur l'enseignement de l'informatique à l'école primaire (Mason

& Rich, 2019). Différentes pistes sont proposées, qui consistent le plus souvent à élaborer des curricula puis à décliner des plans de formation pour que les enseignants puissent mettre en œuvre ces curricula.

Le repérage des items de la pensée informatique (*computational thinking*) dans les matières déjà enseignées en primaire est vu par certains comme un point d'entrée stratégique pour la formation des enseignants, qui ne nécessiterait pas de les former à la science informatique ou à la programmation (Yadav *et al*., 2018). Angeli *et al*. (2016) proposent un curriculum tenant compte de la difficulté d'abstraction chez les élèves les plus jeunes. Ils déclinent les composantes de la pensée informatique – abstraction, généralisation, décomposition, pensée algorithmique, débogage – en compétences et en activités pour des élèves de 6 à 12 ans. À partir de ce curriculum, ils identifient les PCK nécessaires aux enseignants pour le mettre en œuvre.

Si ces travaux menés dans le domaine des PCK en informatique peuvent nous être utiles, ils ne correspondent ni au périmètre de notre étude, qui ne se situe pas au niveau global du système éducatif, ni au public concerné par notre recherche, à savoir des enseignants de l'école primaire, non spécialistes en informatique. Ces travaux ont également une visée prescriptive en termes de curricula, ce qui n'est pas l'objet de l'étude proposée ici.

1.3. Problématique et questions de recherche

Peu de recherches sont actuellement menées concernant la didactique de l'informatique à l'école primaire et les connaissances professionnelles nécessaires aux futurs enseignants dans ce champ. Or des formations sont dispensées, selon des modalités variables en fonction des contextes. Nous nous posons la question des connaissances développées durant ces formations par les futurs enseignants.

Dans cette étude exploratoire, nous nous focaliserons sur les connaissances curriculaires et sur les connaissances des stratégies d'enseignement issues des modélisations de Shulman (2007) et de Magnusson, Krajcik et Borko (1999). Celles-ci sont bien documentées dans la littérature (Baron & Drot-Delange, 2016), alors que nous ne disposons pas encore d'un corpus suffisant de connaissances sur les difficultés des élèves par exemple.

Les stratégies d'enseignement se dessinent dès la phase de préparation des séances, où nombre de décisions sont prises par l'enseignant (Jameau, 2017), s'appuyant sur les connaissances des curricula. Stratégies d'enseignement et connaissances des curricula constituent des enjeux des formations pour permettre aux futurs enseignants de se projeter dans des situations pédagogiques. Nous étudierons également les difficultés associées aux connaissances disciplinaires des contenus telles qu'elles se manifestent dans les productions des futurs enseignants.

Connaissances didactiques et disciplinaires sont liées d'après Sensevy et Amade-Escot (2007), les premières nécessitant les secondes, mais ne s'y réduisant pas. L'introduction récente d'un enseignement de l'informatique dans le second degré en France et l'absence possible de formation en informatique dans le cursus universitaire suivi par les futurs enseignants ne garantissent pas une connaissance suffisante des contenus disciplinaires. Hubbard (2018), dans sa revue de littérature concernant les PCK en informatique, montre de quelles manières les connaissances disciplinaires peuvent influer sur les connaissances didactiques. Cette distinction entre ces connaissances serait moins marquée chez les

enseignants expérimentés que chez les enseignants débutants. Un autre résultat souligné par Hubbard (2018) est que la confiance dans ses propres connaissances disciplinaires ne garantit pas la confiance dans ses connaissances didactiques correspondantes.

Nos questions de recherche sont donc les suivantes :

- Quelles sont les connaissances des stratégies d'enseignement de la programmation et de l'algorithmique développées par les futurs enseignants du primaire non spécialistes de la discipline durant leur formation ?
- Quelles sont leurs connaissances curriculaires s'agissant de l'informatique ?
- Quelles sont les difficultés liées aux connaissances disciplinaires mises en évidence dans les productions de ces futurs enseignants ?

2. Méthodologie

Chesnais, Cross et Munier (2017) s'interrogent sur la manière d'inférer les connaissances professionnelles des enseignants. Nous avons résolu d'analyser les séances ou séquences produites par les futurs enseignants de nos formations, comme révélatrices de leurs connaissances didactiques et disciplinaires.

2.1. Constitution du corpus des données

Notre recherche porte sur les productions réalisées par de futurs enseignants dans trois formations. Ces formations diffèrent selon leur caractère obligatoire ou optionnel, leur inscription dans un dispositif diplômant, certifiant ou expérimental, leur volume horaire et leurs modalités.

Au sein de l'INSPÉ de l'université de Franche-Comté, dans le master MEEF (Métiers de l'enseignement, de l'éducation et de la formation) pour le premier degré, l'informatique, en tant que science, est enseignée au sein d'un module obligatoire et présentiel intitulé « Numérique et Stage ». Dans ce module de 76 heures, 8 heures sont consacrées à la pensée informatique, dont 2 heures à Scratch en première année de master et 6 heures en deuxième année pour expérimenter des activités d'informatique débranchées, des activités mettant en œuvre des robots pédagogiques, et la programmation sur Scratch et Blockly. L'essentiel de la formation consacrée à la pensée informatique est concentré sur le dernier semestre au terme duquel les étudiants ont eu, en 2018-2019, à rendre (par groupes de deux ou trois) un devoir présentant une séquence d'enseignement sur la programmation.

Les données collectées pour cette formation sont les 87 devoirs déposés au cours de l'année universitaire 2018-2019 par 188 étudiants, pour la plupart professeurs stagiaires, pour répondre à la consigne suivante :

> Présenter une séquence d'enseignement, produite à l'aide d'un outil auteur (Canoprof, EduMoov…) permettant aux élèves un travail sur l'apprentissage du codage à l'école élémentaire. La séquence peut inclure des activités pouvant se faire avec ou sans machine (tablette ou ordinateur avec un logiciel [GéoTortue, Scratch…], robots…).

Au sein de l'INSPÉ de l'université Clermont-Auvergne, le module C2i2e[3] du master MEEF premier degré comprend 4 heures centrées sur la question de l'enseignement de l'informatique. La première séance de 2 heures vise, d'une part, à identifier dans les programmes officiels les mentions qui renvoient à l'informatique en tant que science et à présenter un lexique simplifié des notions informatiques. De plus, les modalités de cet enseignement en classe sont abordées : robotique pédagogique, informatique débranchée, micromonde. Les étudiants sont amenés à manipuler et expérimenter ces modalités via des activités « clés en main » proposées par les formateurs. Des robots (BeeBot, BlueBot et Thymio) sont mis à disposition des étudiants, et des séances types leur sont proposées qu'ils peuvent expérimenter durant la séance. Le prêt du matériel est possible pour ceux qui souhaitent mener des séances durant leur stage. La deuxième séance est consacrée à la programmation visuelle. Elle consiste en la création de scénarii pédagogiques avec le langage de programmation ScratchJr. Les professeurs stagiaires étudient les exemples de programmes fournis avec le logiciel. Ils doivent ensuite analyser des missions existantes (Perrot, 2016). Enfin, ils doivent élaborer des missions à destination de leurs élèves.

Les données recueillies pour cette formation sont les 12 comptes rendus réalisés par 22 étudiants pour la séance de programmation visuelle avec ScratchJr lors de l'année universitaire 2019-2020. Ce corpus ne concerne qu'un groupe d'étudiants, les autres groupes n'ayant pas participé à la recherche. La consigne donnée aux futurs enseignants était d'élaborer des missions ScratchJr. Il s'agissait ensuite d'inclure ces missions dans une séquence en informatique, intégrant ou non des activités d'informatique débranchée ou de robotique.

À la Haute École pédagogique du canton de Vaud, un dispositif transitoire est en place depuis deux ans pour apporter aux étudiants – futurs enseignants du primaire – une base de formation à l'enseignement de la science informatique. Conçu pour offrir aux étudiants une mise à niveau en lien avec de récents projets pilotes d'éducation numérique lancés dans le canton, ce dispositif est optionnel et 1/3 environ des étudiants y participent de manière volontaire. La formation est organisée sur un bloc de trois journées et demi ou 28 heures en présentiel, avec alternance de grands cours et d'ateliers. À la suite de cela, les étudiants préparent une séquence d'enseignement, seuls ou en groupe, et la présentent en fin de semestre lors d'une soutenance orale évaluée et notée, mais dont la valeur est relative, vu le caractère optionnel du dispositif. Ce dispositif intervient dans le dernier semestre de la troisième et dernière année de formation des étudiants. En termes de contenus disciplinaires, le dispositif n'est pas limité à la découverte de la programmation, mais vise à présenter un panorama beaucoup plus large de la science informatique.

Les données collectées pour cette formation sont les présentations réalisées par les étudiants ou groupes d'étudiants de l'année académique 2018-2019 pour la soutenance orale de leur examen de clôture du module. La consigne transmise aux étudiants consistait à leur demander de réaliser un scénario pédagogique, mis en œuvre en stage ou non, amenant les élèves à travailler une progression d'apprentissage du plan d'études en cours de construction dans le cadre des projets pilotes du canton et à l'évaluer. Le matériel pédagogique produit dans ce contexte devait être fourni, ainsi que des traces de l'expérimentation en classe. Une brève vidéo était explicitement demandée qui mettait en évidence ce qui avait été mis en œuvre ainsi que l'alignement curriculaire. Ceux qui ne mettaient pas en œuvre en classe exemplifiaient eux-mêmes directement en réalisant l'activité prévue pour les élèves et se

3 Certificat informatique et internet, niveau 2, Enseignant.

filmaient. Nous ne conservons pour cette recherche que les présentations portant sur des contenus disciplinaires en lien avec le domaine « algorithmique et programmation ».

Le tableau 1 récapitule les données collectées. Les futurs enseignants pouvaient choisir le cycle auquel s'adressait leur proposition de séances. Ils pouvaient, si cela leur était possible, en fonction de leur lieu de stage par exemple, tester dans la classe la séquence ou séance qu'ils avaient élaborée. Cependant, cela n'était pas une exigence de la formation.

Tableau 1 : constitution du corpus par étude de cas et cycle

	Corpus n° 1	Corpus n° 2	Corpus n° 3	**Total**
Nb. étudiants	188	24	71	**283**
Nb. productions analysées	**87**	**12**	**36**	**135**
Cycle 1	21	0	13 (+5)[a]	**39**
Cycle 2	37	7	15	**59**
Cycle 3	29	5	3	**37**

a. (+5) renvoient aux productions intermédiaires entre cycle 1 et cycle 2 du fait de l'équivalence opérée entre âge des élèves suisses et cycles scolaires français.

2.2. Méthode d'analyse du corpus

Lors d'une première étape, nous avons dépouillé chacune des productions des futurs étudiants en compilant les informations dans un tableau. Ce tableau décrit chaque production avec un numéro d'anonymat de l'auteur de la copie, le cycle visé, les objectifs annoncés de la séance, les modalités pédagogiques proposées, la référence aux programmes, les contenus et les tâches de la séance/séquence proposée. Les productions ont été réparties entre les chercheurs. Chaque chercheur pouvait compléter ce recueil de données par les questions soulevées selon lui par la copie.

Dans une seconde étape, nous avons analysé plus spécifiquement les productions selon nos trois questions de recherche. Concernant les connaissances des stratégies d'enseignement de la programmation et de l'algorithmique, ainsi que les connaissances curriculaires, nous avons procédé à une analyse de contenu des productions des futurs enseignants. Les contenus relevés dans les productions concernent les éléments suivants :

- la (ou les) modalité(s) pédagogique(s) choisie(s) par les futurs enseignants dans la séquence proposée. Nous avons exclu de cette analyse les modalités qui mobilisent des ressources propres à des matières autres que l'informatique ;
- les mentions aux programmes officiels ainsi que les objectifs des séances proposées ;
- le cycle pour laquelle l'activité est produite ;
- le lien effectué avec d'autres matières que l'informatique.

Les modalités pédagogiques sont celles de l'informatique débranchée, la robotique, les micromondes et la programmation visuelle (Baron & Drot-Delange, 2016). L'informatique débranchée, ou l'informatique sans ordinateur, propose des activités qui permettent de découvrir les concepts de l'informatique par des jeux impliquant la manipulation d'objets, de puzzles, des tours de magie (Drot-Delange, 2013). La robotique pédagogique vise à mobiliser des technologies informatiques dans l'observation, l'analyse, la modélisation ou

le contrôle de processus physiques dans une approche constructiviste de l'apprentissage (Depover, Karsenti & Komis, 2007). Les micromondes sont des environnements informatiques pédagogiques contrôlés par l'apprenant qui dispose d'un langage de communication avec la machine ; LOGO en est un exemple (Depover, Karsenti & Komis, 2007). La programmation visuelle repose sur le paradigme de la programmation orientée objets, dans lequel un objet est une entité caractérisée par des attributs et des comportements. Par exemple, dans le langage Scratch, le programme crée des « lutins », définit leur forme et leur comportement (avancer, sauter, etc.). Les objets ainsi créés sont dotés de scripts et se synchronisent par envoi de message. Il s'agit donc également d'une programmation par événements. Les scripts peuvent s'exécuter simultanément, permettant une initiation à la programmation concurrente et au parallélisme (Komis, Touloupaki & Baron, 2017).

Afin de mettre en évidence les difficultés liées aux connaissances disciplinaires, ce qui concerne la dernière question de recherche, nous avons relevé, après une première lecture du corpus des productions des futurs enseignants, deux thèmes prédominants : la confusion entre condition et événement et la notion de codage.

3. Résultats

3.1. Des connaissances de stratégies d'enseignement de la programmation et des connaissances curriculaires

L'analyse de notre corpus souligne la prédominance des modalités pédagogiques de l'informatique débranchée et de la robotique pédagogique. Elles représentent respectivement 38 % et 37 % des modalités proposées. Viennent ensuite la programmation visuelle (22 %) et les micromondes (3 %). Plusieurs modalités peuvent être proposées dans une même production.

En informatique débranchée, le jeu de l'enfant robot est abondamment mentionné mais aussi adapté. Il représente la moitié des activités proposées en informatique débranchée. En robotique pédagogique, viennent en tête les robots BlueBot (37 % des 80 citations), BeeBot (29 %) et Thymio (25 %). En programmation visuelle, ScratchJr représente environ deux tiers des citations, suivi par Scratch (13 citations sur 48).

La moitié des propositions prévoit deux modalités pédagogiques, un tiers une seule modalité. Lorsque les étudiants envisagent de mobiliser plusieurs modalités pédagogiques, la stratégie la plus courante est de débuter par des activités débranchées. La justification de cet enchaînement n'est pas toujours explicitée. Elle semble être perçue comme l'occasion de travailler les concepts avant leur mise en œuvre, comme l'atteste la citation suivante :

> « *nous avons envisagé de donner du sens aux apprentissages en travaillant avec de l'informatique débranchée* […]. *Le travail sur ordinateur s'effectuera une fois que les notions seront comprises* » (Be25_07)[4]

L'analyse du corpus révèle aussi d'autres stratégies. Ainsi environ une proposition sur 10 débute par des activités avec des robots. Elles peuvent être suivies d'activités débranchées

4 Les codes identifient une production de notre corpus.

(3 cas), d'activités débranchées puis à nouveau de robotique (5 cas), d'activités Scratch ou ScratchJr (3 cas), parfois suivies d'activités débranchées (1 cas). L'idée sous-jacente, parfois exprimée, est que les élèves pourront transférer les connaissances acquises d'un environnement à l'autre.

Les futurs enseignants n'évoquent que rarement les difficultés potentielles des élèves. Lorsqu'ils le font, ils semblent davantage se référer aux difficultés qu'ils ont pu eux-mêmes rencontrer. Ils évoquent par exemple le nombre élevé de personnages dans un programme ScratchJr ou l'utilisation du bloc de répétition, sans qu'il soit précisé en quoi cela pourrait constituer une difficulté. Parfois, la stratégie d'enseignement repose explicitement sur la connaissance des technologies employées :

> « *le travail en activité débranchée sera orienté vers des déplacements absolus, pour répondre ultérieurement à la problématique des instructions de l'application ScratchJr qui ne permet que des déplacements absolus* » (Be70_06).

Les connaissances curriculaires se manifestent lorsqu'on analyse la récurrence des connaissances et compétences visées par les futurs enseignants chez leurs élèves. On constate que plus des deux tiers des productions renvoient à l'objectif d'apprendre à (se) repérer et (se) déplacer dans l'espace, coder et décoder des déplacements. Cet objectif peut être décliné selon plusieurs approches pédagogiques : en utilisant des quadrillages, des représentations à l'écran, via le jeu de l'enfant robot ou un robot par exemple.

Concernant les liens faits avec les autres matières, deux cas de figure sont présents dans le corpus. Le premier est celui où les notions informatiques sont mobilisées dans une autre matière que l'informatique et servent l'apprentissage de notions dans cette autre matière. C'est le cas par exemple de la notion de codage qui peut être mobilisée en musique (Be25_01, Be25_15) ou la réalisation de figures géométriques (Be25_16, Be25_21).

Le second est celui où l'habillage de l'activité informatique renvoie à des activités menées dans d'autres matières, mais où l'apprentissage visé est *in fine* celui de notions informatiques. Les connaissances des élèves dans cette autre matière sont les prérequis de l'activité menée en informatique. C'est le cas par exemple lorsque les déplacements du robot ou d'un personnage ont pour contexte ou objet les couleurs en anglais, les personnages d'une histoire travaillée en lecture suivie (CF_01), une poésie (Be90_21), la carte de la Grèce antique (Be90_08), le projet d'école sur l'Afrique (CF_04), raconter une histoire dans le bon ordre (La_0108, La_0404), etc.

Les connaissances curriculaires peuvent aussi se traduire par les propositions d'activités selon les cycles. Ainsi, dans notre corpus, les activités débranchées sont plus souvent proposées en cycles 1 et 2 qu'en cycle 3. Le jeu de l'enfant robot a été proposé dès les années 1990 par Greff (1998) pour le développement d'une pensée algorithmique chez les très jeunes enfants (4-6 ans). Dans notre corpus, le jeu de l'enfant robot représente un tiers des propositions des activités débranchées pour tous les cycles. Alors que ScratchJr a été conçu pour des enfants de 5 à 7 ans, les futurs enseignants de notre étude le proposent beaucoup plus fréquemment en cycle 2 et 3 (une fois sur deux) qu'en cycle 1 (moins d'une fois sur dix). La constitution de notre corpus ne permet pas toujours de savoir si ces choix sont réfléchis et, s'ils le sont, sur quels critères.

3.2. Des connaissances disciplinaires

Les environnements choisis par les futurs enseignants peuvent relever de plusieurs paradigmes de programmation tels que la programmation séquentielle impérative (par exemple avec le robot Beebot), la programmation objet (Scratch, ScratchJr), la programmation événementielle (avec le robot Thymio). Un environnement peut mobiliser plusieurs paradigmes. De plus, dans une séquence pédagogique, un enseignant peut être amené à utiliser plusieurs environnements de programmation. Méconnaître ces paradigmes peut engendrer des confusions, visibles notamment dans les consignes données aux élèves. Une confusion émergente dans notre corpus est celle de l'amalgame entre « condition » et « événement ».

3.2.1 Une confusion entre « condition » et « événement »

La « condition » en informatique est une expression logique (ou booléenne) dont l'évaluation ne peut renvoyer que deux valeurs possibles : « Vrai » ou « Faux ». De telles expressions sont utilisées dans de nombreux langages (impératifs, fonctionnels, et bien sûr logiques). Dans les langages impératifs séquentiels souvent utilisés pour l'introduction à la programmation chez les (jeunes) enfants, l'expression booléenne est associée à l'instruction conditionnelle « Si <condition> Alors Instructions… » complétée parfois d'une clause « Sinon Instructions ». Chez les élèves un peu plus âgés on l'utilise aussi dans la structure « Tant que <condition> Faire Instructions… » Mais dans ces deux cas (« Si » et « Tant que »), la condition est une expression booléenne évaluée ponctuellement, une seule fois pour le « Si », ou éventuellement de façon répétée pour le « Tant que ». Le cours d'exécution du programme peut alors être modifié, mais en dehors de ce(s) moment(s) ponctuel(s) où la condition est évaluée, le programme suit le cours de la séquence. L'extrait 1 de notre corpus donne un exemple d'instruction conditionnelle correcte.

Extrait 1 : exemple d'instruction conditionnelle correcte

Recommence à la ligne de départ si on touche le mur.
On crée une nouvelle condition : si le lutin touche le mur du circuit alors il retourne au départ (que l'on vient de définir). Pour la condition « touche le mur » on va se servir de la couleur du contour du circuit (noire). Dans les blocs « capteurs » prend le bloc « couleur… touchée ? » en choisissant la couleur clique sur la couleur de ton mur de circuit (noire).

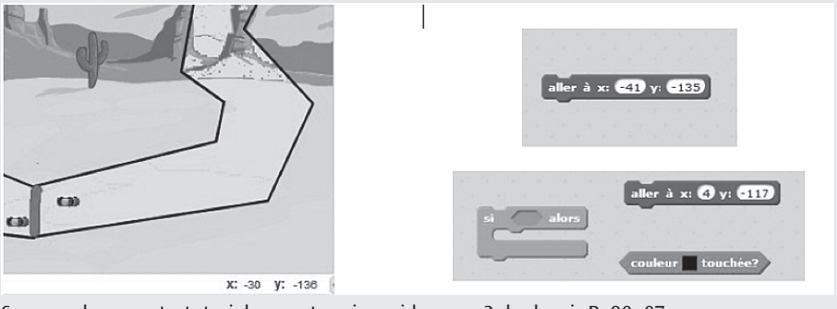


Fig. 1

Source : document « tutoriel_cree_ton_jeu_video », p. 3 du devoir Be90_07

L'événement en revanche, dans le paradigme de la programmation événementielle, est quelque chose qui se produit et qui est détectable par la machine (un capteur qui s'active, un message qui arrive, une collision qui se produit, un bruit qui est détecté, etc.) et qui permet de déclencher une réaction sous forme d'action ou d'une série d'actions. La différence fondamentale entre la condition (dans une instruction conditionnelle) et un événement, c'est que l'événement n'attend pas que le programme le teste, mais interrompt le programme en cours au moment où l'événement se produit, pour déclencher la réaction qui lui est associée. On devrait donc plutôt dire « Quand l'événement se produit la machine déclenche telle action » et non pas « Si l'événement se produit alors réaliser telle action ». Notons que, dans le paradigme événementiel, chaque fois que l'événement se produit, l'action qu'il entraîne se produira. L'extrait 2 donne un exemple d'événement correctement formulé, tiré de notre corpus.

Extrait 2 : exemple d'événement correctement formulé

À partir de ce programme SCRATCH, aide le personnage à sortir du château en faisant disparaître la chauve-souris. Dans ce jeu, le joueur est enfermé dans un château et il veut s'échapper. Tu dois compléter le programme pour faire disparaître la chauve-souris lorsqu'on clique sur elle et ouvrir la porte du château. Plusieurs programmes sont déjà faits :
– Le programme du personnage qui explique la situation ;
– Les programmes qui font bouger la chauve-souris.
Le but du projet est de programmer la chauve-souris pour qu'elle disparaisse lorsqu'on clique sur elle. Sa disparition doit entraîner l'ouverture de la porte du château.
https://scratch.mit.edu/projects/154635923 (lien)
Puisqu'il existe un arrière-plan « Sortie » décrivant le château avec la porte ouverte et que le programme du personnage contient déjà l'événement suivant :

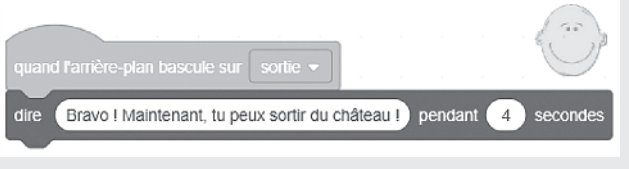


Fig. 2-a
Source : extrait de la copie Be70_10, séquence p. 9

Pour résoudre ce problème il faut ajouter dans le programme de la chauve-souris un événement : « Quand le personnage chauve-souris est cliqué par l'utilisateur » et lui associer la séquence des deux actions attendues : « cacher la chauve-souris » et « basculer sur l'arrière-plan sortie ». La figure 2-b indique comment cela pourrait être formulé dans le programme Scratch.

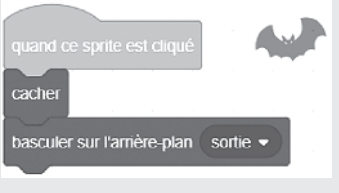
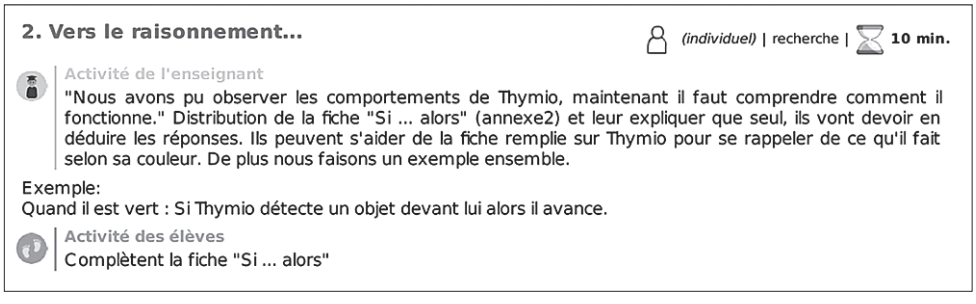


Fig. 2-b : solution proposée
Source : extrait de la copie Be70_10, séquence p. 9

Ce qui est en jeu dans la confusion entre instruction conditionnelle et événement est la connaissance des objets programmables, l'interprétation de leur comportement dans leur usage ou dans la phase de débogage pour corriger le programme, afin que le comportement de l'objet programmable corresponde aux attentes. Sans aller jusqu'à l'explicitation pour les élèves des différents paradigmes dans le curriculum à l'école élémentaire, il est important pour les enseignants d'en comprendre leur manifestation dans les situations auxquelles sont confrontés les élèves. Ici, il s'agit donc de comprendre que le moment où une action sera déclenchée dépend dans un cas d'une condition testée explicitement dans la séquence du programme tandis que dans l'autre il dépend d'un événement (ex : obstacle détecté par exemple par un capteur de proximité, ou un message reçu d'un autre lutin dans Scratch ou ScratchJr) qui peut survenir à tout instant.

Nous avons pu repérer cette confusion dans différentes séquences constituant notre corpus. Nous la retrouvons dans plusieurs séquences utilisant le robot Thymio, dont les comportements préprogrammés, repérés par une couleur, sont régis par une programmation événementielle. Plus précisément, de nombreuses fiches pédagogiques utilisant le robot Thymio, disponibles sur le web et (ré)utilisées par les futurs enseignants, proposent une formulation de la forme : « SI Thymio détecte…, ALORS Thymio fait… ». Autrement dit, la forme est « Si <événement> Alors <Action> ». Il nous semble que cette formulation puisse entraîner des conceptions erronées et qu'il serait préférable d'écrire : « Quand <événement>, réaliser <Action> » ou « Dès que <événement>, réaliser <Action> ». Cette dernière façon de formuler les choses montre mieux la temporalité du déclenchement de l'action en réponse à l'événement et non pas au moment où le programme testerait une condition. Les figures 3a et 3 b donnent un exemple de cette confusion avec Thymio dans notre corpus.



2. Vers le raisonnement…

(individuel) | recherche | **10 min.**

Activité de l'enseignant

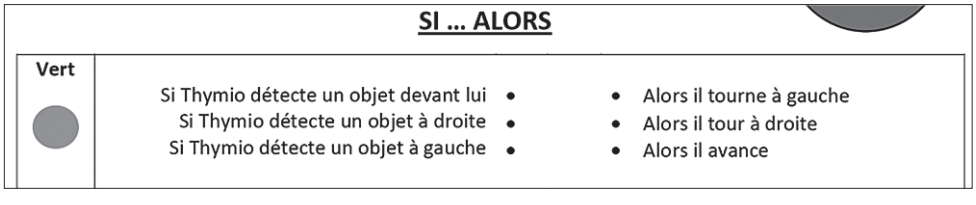
"Nous avons pu observer les comportements de Thymio, maintenant il faut comprendre comment il fonctionne." Distribution de la fiche "Si … alors" (annexe2) et leur expliquer que seul, ils vont devoir en déduire les réponses. Ils peuvent s'aider de la fiche remplie sur Thymio pour se rappeler de ce qu'il fait selon sa couleur. De plus nous faisons un exemple ensemble.

Exemple:
Quand il est vert : Si Thymio détecte un objet devant lui alors il avance.

Activité des élèves

Complètent la fiche "Si … alors"

Fig. 3a : consigne donnée dans la séquence extraite du corpus (Be70_08, p. 4)



SI … ALORS

Vert		
	Si Thymio détecte un objet devant lui	Alors il tourne à gauche
	Si Thymio détecte un objet à droite	Alors il tour à droite
	Si Thymio détecte un objet à gauche	Alors il avance

Fig. 3 b : extrait de la fiche « Si… alors » du corpus (Be70_08, annexe 2, p. 1)

Dans des séquences pédagogiques envisagées par nos étudiants, le robot Thymio n'est pas le seul support technologique qui révèle la confusion entre l'instruction conditionnelle et un événement déclenchant une action. Nous trouvons également cette difficulté dans

l'environnement de programmation ScratchJr. En effet, lorsqu'un événement déclenche l'action d'un personnage, les futurs enseignants l'assimilent parfois à une condition. Notons que dans ScratchJr, il n'existe pas de moyen d'exprimer l'instruction conditionnelle « Si <condition> Alors Instruction ».

Une autre confusion repérée lors de l'analyse du corpus est celle de la distinction entre donnée et programme.

3.2.2. Notion de « codage » : représentation d'une donnée ou d'un programme

L'ensemble des informations sont représentées de manière binaire sous forme de 0 et de 1 dans les ordinateurs, pour des contraintes liées à l'électronique des transistors qui les composent. Au plus bas niveau de la machine, les données sur lesquelles un programme va travailler sont donc représentées de la même manière que le programme lui-même : sous forme de 0 et de 1. Et pourtant, lorsque l'on aborde l'enseignement de l'informatique, il semble crucial de permettre aux apprenants de distinguer « donnée » et « programme » qui sont conceptuellement deux éléments tout à fait distincts.

Dowek (2011) définit l'informatique par quatre concepts : machine, algorithme, langage et information. Il semble donc essentiel d'amener les apprenants à les distinguer entre eux, la programmation étant incluse dans les langages et les données dans l'information. Une donnée est statique, elle peut être lue ou écrite par un programme à un emplacement mémoire de la machine, mais ne « fait » rien en elle-même, tandis qu'un programme est dynamique : il est constitué d'une suite d'instructions qui sont exécutées par la machine et produisent un effet concret.

Au-delà, la question de la représentation d'un programme revêt un intérêt particulier dans le domaine de l'apprentissage de la programmation à l'école primaire. Elle est en lien avec la question du contrôle d'un objet. Ce contrôle est l'une des « pratiques computationnelles » qui permet d'amener de jeunes élèves à construire la notion de « programme » en tant que « spécification du comportement futur d'un objet » [traduction libre], selon la définition de Blackwell (2002). Cette notion est importante en didactique de l'informatique et a été l'objet de travaux dès son origine (Rogalski & Samurçay, 1990).

La notion de « contrôle » réfère à la manière dont des élèves peuvent apprendre à planifier le comportement futur d'un objet, dont ils peuvent représenter leurs plans extérieurement et travailler avec, c'est-à-dire les lire, les analyser, anticiper leurs effets, les partager, les modifier, les simplifier. Kalaš, Blaho et Moravcik (2018) distinguent deux dimensions dans le contrôle d'un objet, la première portant sur la modalité du contrôle, la seconde sur sa représentation. Les modalités de contrôle comptent quatre niveaux : la manipulation directe, la manipulation indirecte, le contrôle direct et le contrôle computationnel. La représentation connaît cinq niveaux : pas de représentation, comme trace interne, comme trace externe, comme plan interne ou encore comme plan externe. Lorsqu'un élève déplace à la main une BeeBot sur un quadrillage en interprétant un programme écrit à l'aide de cartes représentant les déplacements, le contrôle de la BeeBot est analysé comme une manipulation directe et sa représentation (les cartes) comme un plan externe. Lorsque l'élève programme la BeeBot directement à l'aide des touches du robot, le contrôle est alors un contrôle direct, sans représentation.

La représentation d'un programme constitue donc un support à l'appropriation de la notion même de « programme » par les apprenants. Une attention aiguë aux diverses moda-

lités possibles de cette représentation semble nécessaire afin d'assurer un apprentissage qui suive une progression de la charge cognitive pour les élèves (Kalaš, Blaho & Moravcik, 2018).

Plusieurs exercices de programmation proposés dans notre corpus portent sur le fait de déplacer un objet sur une grille à l'aide d'instructions de déplacement. La représentation de ces programmes (cf. figure 4) peut être sujette à caution : lorsque l'on représente de manière symbolique une suite d'instructions de manière linéaire, par exemple sous forme de flèches, et que l'on en parle comme d'un processus (dynamique) et non comme d'une simple information (statique) à communiquer, cela fait assez peu de doute qu'il s'agit d'un programme en tant que séquence d'instructions qui sont exécutées, avec une valeur opérative. Mais par contre, lorsque l'on représente le programme par la trace du chemin parcouru sur la grille, visuellement cela s'apparente beaucoup plus à un dessin. Cette trace transmet une information statique concernant les cases sur lesquelles un personnage a transité, une mémoire d'un déplacement que l'on pourrait transcrire par une phrase du type « la case a été traversée dans le sens vertical ». Cela s'apparente beaucoup plus à la représentation d'une série de données qu'à l'exécution d'une séquence d'instructions. Ce contexte des activités de déplacement sur grille semble donc propice à la confusion entre représentation d'une donnée et représentation d'un programme.

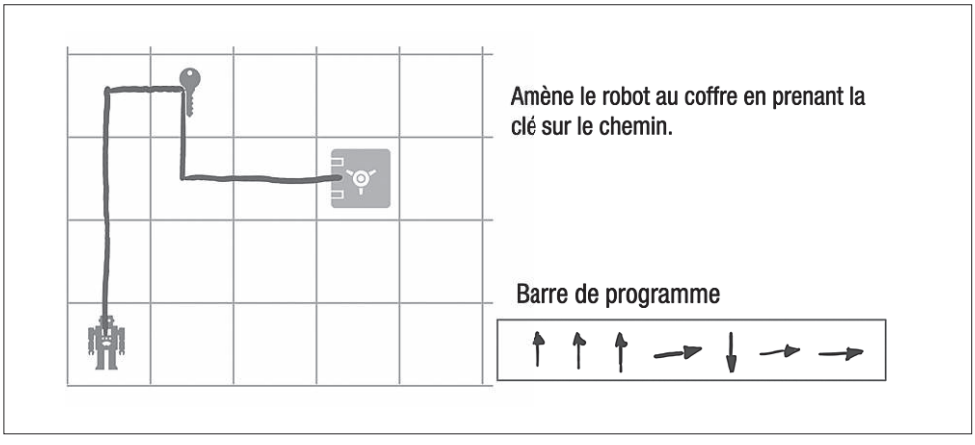


Fig. 4 : exemple de représentation d'un déplacement sur grille sous forme de chemin et sous forme d'instructions fléchées

Plusieurs activités tirées de notre corpus laissent penser à une relative confusion entre les notions de représentation de programme et de donnée.

Dans l'exemple Be25_07, l'activité consiste à déplacer un navire sur une grille pour qu'il retrouve un coffre-fort. L'activité de l'enseignant est décrite comme suit : « Il relance les échanges et attend des élèves qu'ils précisent "nous devons lui donner des informations/instructions" ». Plus loin, deux consignes aux élèves sont explicitées sous cette forme : « Vous devez écrire sur votre cahier de brouillon une succession d'informations pour que le navire retrouve le coffre ». « Chaque information doit commencer par un verbe. […] Sur ce modèle, vous allez continuer à écrire des instructions pour que le navire retrouve le trésor ».

On note ici l'utilisation alternative du terme « d'information » ou « d'instruction » pour décrire ce qui va être transmis au navire pour lui permettre d'arriver au trésor. Cela peut

laisser penser à une relative confusion entre la représentation d'une donnée statique et celle d'une instruction dynamique.

Plus loin, dans la même production, un exercice similaire est organisé sous forme de chasse au trésor sur une grille. La consigne de l'enseignant est formulée ainsi : « Sur la fiche, vous devez transcrire le chemin qui est tracé sous forme de code, comme nous venons de le voir. Lorsque vous avez fini, vous vérifiez votre code pour voir s'il est juste ».

Dans ce cas, le code semble être considéré comme une donnée (écrite selon certaines conventions) permettant de représenter une donnée graphique : le chemin. Il semble que l'aspect dynamique de l'exécution d'un programme soit évacué dans cette représentation. La succession de flèches qui représente le chemin n'est pas considérée comme une suite d'actions à exécuter pour réaliser un déplacement (et atteindre le trésor), mais simplement comme une transcription symbolique du parcours ou chemin.

Dans un autre exemple tiré de notre corpus (La_0501), les étudiants proposent une activité intitulée « Scratch paravent » consistant en une adaptation visiblement originale d'une activité présentée en formation appelée « Pixel paravent ». « Pixel paravent » porte sur la représentation de l'information : deux élèves doivent se communiquer des informations relatives à la composition d'une image pixellisée dans une grille aux cases soit coloriées, soit vides, en définissant un langage compréhensible pour tous les deux. L'information est codée selon le langage choisi par l'émetteur, puis transmise au récepteur qui va décoder l'information et reconstituer l'image.

Dans cette version adaptée par les étudiants, deux objectifs sont définis comme suit : « L'élève est capable de communiquer un itinéraire clair dans un langage de son choix ; l'élève est capable d'utiliser les blocs d'instructions pour communiquer un itinéraire clair. » À nouveau, il nous semble repérer derrière ces formulations une potentielle équivoque entre la représentation d'un chemin sous sa forme graphique dessinée et la formalisation d'un déplacement sous forme d'instructions données à un personnage se déplaçant sur un repère.

En référence au modèle de Kalaš, Blaho et Moravcik, il est intéressant de voir que l'on passe par plusieurs phases différentes : l'élève émetteur du message commence par imaginer et dessiner le chemin de son personnage sur la grille, ce qui correspond à une représentation sous forme de plan interne. Puis il va transcrire ce chemin sous forme d'un message dont les composantes semblent relativement libres (expression impérative écrite, dessin, succession de flèches, etc.), ce qui constitue une trace externe. Le récepteur va utiliser le message reçu comme un plan externe pour reconstituer le déplacement du personnage sur la grille et marquera au fur et à mesure son chemin sur celle-ci, ce qui correspond à une trace interne du message.

Nous pourrions mentionner encore d'autres exemples dans notre corpus qui présentent une ambiguïté entre données et programmes. Bien que ces potentielles confusions ne nous semblent pas forcément trop problématiques au niveau des activités proposées, elles dénotent tout de même une difficulté conceptuelle qui pourrait avoir des répercussions plus loin si elle n'est pas abordée. Il nous semble donc que dans les premières phases de l'enseignement de la programmation, pour un meilleur ancrage des concepts, il serait important de porter plus d'attention à la distinction entre donnée et programme.

L'enjeu ici est bien celui de la connaissance construite par l'apprenant, qui peut être assez différente de celle que l'enseignant pense viser. Sur cet aspect, nous pouvons émettre l'hypothèse que les expressions « coder un programme » et « coder une information » peuvent engendrer ou entretenir cette ambiguïté. Nous pouvons également nous questionner sur

l'opportunité d'utiliser le terme de « codage » ou le verbe « coder » comme un synonyme de « programmation » ou de « programmer ».

4. Discussion

La démarche retenue ici mérite d'être questionnée sur les choix méthodologiques. Les connaissances professionnelles peuvent-elles être inférées en dehors de l'action ? Deux approches des connaissances pédagogiques de contenu se confrontent (Fagnant & Demonty, 2019). La première est de considérer que les connaissances peuvent être appréhendées de manière statique. On pourrait donc évaluer ces connaissances par des tests indépendamment des situations vécues en classe. La seconde est de considérer que ces connaissances sont nécessairement situées, dans l'action, et qu'elles ne peuvent être appréhendées qu'en situation d'interaction entre l'enseignant et les élèves.

Nous avons opté pour ce qui pourrait être un entre-deux de ces approches. Ni évaluation statique de connaissances, ni observation directe de séances menées en classe, nous avons cherché à repérer les connaissances dans une situation où les futurs enseignants devaient se projeter dans la réalisation d'une séance avec des élèves et la préparer. Parfois, certains d'entre eux ont pu réaliser véritablement cette séance ou séquence. Nous avons estimé que l'évaluation de ces préparations de séances pouvait nous permettre de révéler, si ce n'est des connaissances, au moins des représentations sur ce qu'était pour eux l'enseignement de la programmation et de l'algorithmique à l'école primaire.

Les résultats obtenus, avec la mise en évidence de difficultés récurrentes rencontrées par les futurs enseignants, nous confortent dans le bien-fondé de ce choix. De même, cette récurrence nous invite à considérer que le biais qui pourrait résulter de la double posture de chercheur et de formateur est ici limité. Il l'est également par le fait que si les chercheurs contribuent à la formation, celle-ci est aussi élaborée par des collectifs de formateurs, et ne relève donc pas de la seule décision des chercheurs. Enfin, les corpus ont été analysés de manière collégiale par les chercheurs.

Dans cette recherche exploratoire, nous n'avons pas traité l'ensemble des connaissances professionnelles des futurs enseignants dans le domaine de la programmation et de l'algorithmique mais avons retenu trois types de connaissances.

Selon les modélisations des connaissances professionnelles des enseignants auxquelles on se réfère, les connaissances curriculaires peuvent être au même niveau que les PCK (par exemple le modèle de Shulman) ou être partie intégrante des PCK (par exemple le modèle de Magnusson Krajick et Borko, 1999). En informatique, des recherches tendent à élaborer un curriculum puis à élaborer les PCK nécessaires aux enseignants pour le mettre en œuvre. Cette démarche s'explique par la diversité de la situation des pays concernant l'enseignement de l'informatique à l'école primaire. Dans cette recherche, nous nous sommes focalisés sur les connaissances des stratégies d'enseignement, celles-ci étant bien documentées dans le champ de l'enseignement de la programmation et de l'algorithmique. Nous nous sommes également intéressés aux connaissances curriculaires et disciplinaires, en considérant que les stratégies d'enseignement en dépendaient.

En matière de stratégies d'enseignement et de connaissances curriculaires, les productions des futurs enseignants analysées révèlent des choix parfois directement inspirés de

la formation, probablement selon le principe d'isomorphie entre situation de formation et situation d'enseignement (Schneeberger & Triquet, 2001). La stratégie majoritairement adoptée, informatique débranchée d'abord, reprend une pratique courante chez les enseignants chevronnés. Il conviendrait de commencer l'apprentissage par des activités débranchées. La recherche menée auprès de collégiens par Grover, Jackiw et Lundh (2019) va dans le même sens. Les chercheurs ont élaboré un micromonde qui permet aux élèves de travailler les concepts de variable, d'expression, de boucle et d'abstraction avant de programmer. Les résultats montrent un gain dans l'apprentissage de ces concepts.

Cependant les travaux de Waite, Curzon, Marsh, Sentance et Hadwen-Bennett (2018) montrent les limites des activités débranchées pour la compréhension des liens entre algorithme et programme. Des confusions naissent selon eux de l'usage des jouets programmables et des activités débranchées, qui pourraient favoriser l'amalgame entre algorithme et programme, amener à une forme d'identité entre instruction de programme et représentation d'un déplacement. Notre propre étude a souligné les difficultés posées par la confusion entre représentation d'un programme et représentation d'une donnée. Le double usage du terme « codage » entretient sans doute cette confusion en servant depuis longtemps la représentation d'une donnée, il est maintenant aussi employé dans son origine anglo-saxonne (*coding*) pour l'écriture d'un programme.

Ces confusions interrogent également le curriculum vertical. On peut questionner ce que signifie programmer à l'école primaire et la progression possible de cet apprentissage. Kalaš et ses collègues proposent une modélisation de la progression cognitive des élèves pour les amener au stade où, selon eux, on peut véritablement parler de programmation (contrôle computationnel et représentation externe).

On a pu constater dans les productions que les activités et moyens proposés par les futurs enseignants dépassent le cadre d'usage prescrit par leurs concepteurs (par exemple ScratchJr pour les 5-7 ans, le jeu de l'enfant robot pour les 4-6 ans, etc.). Un des enjeux de la formation pourrait être de donner aux futurs enseignants les outils conceptuels permettant de conscientiser et d'objectiver leurs choix.

En ce qui concerne les connaissances disciplinaires, nous avons souligné les difficultés engendrées par l'usage d'environnements multipliant les paradigmes de programmation, sans nécessairement que les futurs enseignants n'en aient conscience. Nous avons ainsi relevé la confusion entre instruction conditionnelle et événement.

Les recherches en didactique de l'informatique menées dans les années 1980 montraient les difficultés pour les apprenants de la structure conditionnelle en elle-même (Rogalski, 1987). Si les environnements tels que ScratchJr permettent aux élèves d'écrire des programmes syntaxiquement corrects, ils ne proposent pas toutes les structures importantes de la programmation, notamment l'instruction conditionnelle (Komis, Touloupaki & Baron, 2017). Rogalski souligne le rôle des prérequis jugé primordial dans cet apprentissage (égalité, inégalité, opérateurs logiques).

Concernant la confusion relevée entre donnée et programme dans les consignes produites par les futurs enseignants, la lecture des activités proposées aux élèves en termes de contrôle de l'objet, plus spécifiquement selon les deux dimensions du contrôle lui-même et de sa représentation, nous semble heuristique pour rendre intelligibles les phénomènes à l'œuvre dans une situation pédagogique.

Nos résultats montrent enfin les difficultés posées par la mobilisation de plusieurs paradigmes (*paradigm shift*) dans les formations. Ces difficultés ont fait l'objet de recherches

auprès d'élèves de l'enseignement secondaire (Saeli *et al.*, 2011), mais restent à explorer au niveau des élèves de primaire ainsi qu'au niveau des enseignants.

Conclusion

Notre étude porte sur l'analyse des connaissances mises en œuvre par de futurs enseignants, dans l'enseignement de la programmation et de l'algorithmique à l'école primaire, durant leur formation. Les modèles des PCK dans le domaine de l'informatique ne prennent que très peu, voire pas, en considération la spécificité de l'enseignement primaire, à savoir des enseignants non spécialistes de l'informatique. Nous avons donc retenu les modèles de Shulman (2007) et de Magnusson, Krajcik et Borko (1999) que nous considérons comme complémentaires.

Nous nous sommes intéressés plus particulièrement aux connaissances des stratégies d'enseignement, aux connaissances curriculaires et aux connaissances disciplinaires concernant la programmation et l'algorithmique. Nous avons analysé 135 préparations de séance/séquence réalisées par les futurs enseignants de trois institutions de formation. Toutes ces séances ont des objectifs de connaissance directement liés à la programmation et à l'algorithmique même si certaines d'entre elles les partagent avec d'autres sujets de l'informatique scolaire.

Nos résultats montrent l'existence de tendances dans ces productions. Concernant les stratégies d'enseignement, une stratégie récurrente est celle de privilégier les activités débranchées d'abord. La progression des activités et le choix des modalités pédagogiques ne semblent pas toujours étayés par une réflexion sur les enjeux cognitifs et les difficultés potentielles pour les élèves. Concernant les connaissances disciplinaires, nous avons relevé des récurrences dans la confusion entre donnée et programme et entre condition et événement.

Nos perspectives de recherche sont d'approfondir l'étude des connaissances professionnelles des futurs enseignants d'une part du point de vue de leur modélisation dans le contexte de l'enseignement primaire et d'autre part concernant les orientations de l'enseignement de l'informatique à l'école. Cette question de l'orientation de l'enseignement pose celle des articulations entre les travaux en didactique de l'informatique pour les classes de l'école primaire et les formations des futurs enseignants. Comme le souligne Béziat (2019), il n'y a pas encore, pour l'informatique, de culture scolaire, de représentations partagées, sur la manière d'enseigner l'informatique ni sur les objectifs d'un tel enseignement. Or ces questions relèvent de l'orientation de l'enseignement des sciences du modèle de Magnusson, Krajcik et Borko (1999), orientation qui influence toutes les autres dimensions des PCK selon ces auteurs. C'est un enjeu fondamental pour l'enseignement de l'informatique à l'école primaire, qui fait l'objet de nombreux débats (Baron & Drot-Delange, 2016 ; Bruillard, 2016 ; Fluckiger, 2019).

Béatrice Drot-Delange
beatrice.drot-delange@uca.fr

Gabriel Parriaux
gabriel.parriaux@hepl.ch

Christophe Reffay
christophe.reffay@univ-fcomte.fr

Bibliographie

ANGELI C., VOOGT J., FLUCK A., WEBB M., COX M., MALYN-SMITH J. & ZAGAMI J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. *Educational Technology & Society*, vol. 19, nº 3, p. 47-57.

BARON G. L. & DEPOVER C. (2019). Impact du numérique sur les curricula et les programmes d'étude. In G-L. Baron & C. Depover, *Les effets du numérique sur l'éducation. Regards sur une saga contemporaine,* Villeneuve d'Ascq *:* Presses universitaires du Septentrion, p. 37-55.

BARON G.-L. & DROT-DELANGE B. (2016). L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. *Revue française de pédagogie*, nº 195, p. 51-62.

BÉZIAT J. (2019). À l'école primaire, robotique éducative en milieu ordinaire. *Spirale. Revue de recherches en éducation*, vol. 63, nº 1, p. 91-109.

BLACKWELL A. F. (2002). What is programming? *14th Workshop of the Psychology of Programing Interest Group*, p. 204-218.

BRUILLARD É. (2016). Quelle informatique à repenser et à construire pour les élèves de l'école primaire ? In F. Villemonteix, G.-L. Baron & J. Béziat (éd.), *L'école primaire et les technologies informatisées. Des enseignants face aux TICE*, Villeneuve d'Ascq : Presses universitaires du Septentrion, p. 29-37.

CHESNAIS A., CROSS D. & MUNIER V. (2017). Étudier les effets de formations sur les pratiques : réflexions sur les liens entre connaissances et pratiques. *Recherches en didactique des sciences et des technologies (RDST)*, nº 15, p. 97-130.

DELMAS-RIGOUTSOS Y. (2018). Proposition de structuration historique des concepts de la pensée informatique fondamentale. In G. Parriaux, J.-P. Pellet, G.-L. Baron, É. Bruillard & V. Komis, *De 0 à 1 ou l'heure de l'informatique à l'école,* Berne : Peter Lang, p. 31-59.

DEPOVER C., KARSENTI T. & KOMIS V. (2007). *Enseigner avec les technologies : favoriser les apprentissages, développer des compétences*. Québec : Presses de l'université du Québec.

DOWEK G. (2011). Les quatre concepts de l'informatique. In G.-L. Baron, É. Bruillard & V. Komis (éd.), *Sciences et technologies de l'information et de la communication en milieu éducatif : analyse de pratiques et enjeux didactiques*, Athènes : New Technologies Editions, p. 21-29.

DROT-DELANGE B. (2013). Enseigner l'informatique débranchée : analyse didactique d'activités. *Communication présentée au colloque Actualité de la recherche en éducation et en formation (AREF, 2013), Montpellier, 27-30 août 2013*. En ligne : <https://halshs.archives-ouvertes.fr/sic_00955208>.

DROT-DELANGE B. (2018). Reconfiguration de l'enseignement de l'informatique à l'école primaire : quelle conscience disciplinaire chez les professeurs des écoles stagiaires ? *Recherches en didactiques*, vol. 25, nº 1, p. 27-40.

FAGNANT A. & DEMONTY I. (2019). L'évaluation : une question centrale à propos des connaissances pédagogiques de contenu. *Revue française de linguistique appliquée*, vol. 24, nº 1, p. 37-52.

FLUCKIGER C. (2019). *Une approche didactique de l'informatique scolaire*. Rennes : Presses universitaires de Rennes.

GREFF É. (1998). Le « jeu de l'enfant-robot » : Une démarche et une réflexion en vue du développement de la pensée algorithmique chez les très jeunes enfants. *Revue Sciences et techniques éducatives*, vol. 5, nº 1, p. 47-61.

GROVER S., JACKIW N. & LUNDH P. (2019). Concepts before Coding: Non-Programming Interactives to Advance Learning of Introductory Programming Concepts in Middle School. *Computer Science Education*, nº 29, p. 106-135.

HOWEY K. R. & GROSSMAN P. L. (1989). A Study in Contrast: Sources of Pedagogical Content Knowledge for Secondary English. *Journal of Teacher Education*, vol. 40, nº 5, p. 24-31.

HUBBARD A. (2018). Pedagogical content knowledge in computing education: A review of the research literature. *Computer Science Education*, vol. 28, nº 2, p. 117-135.

HUBWIESER P., BERGES M., MAGENHRIM J., SCHAPER N., BRÖKER K., MARGARITIS M., SCHUBERT S. & OHRNDORF L. (2013). Pedagogical content knowledge for computer science in German teacher education curricula. *Proceedings of the 8th workshop in primary and secondary computing education*, p. 95-103.

JAMEAU A. (2017). Connaissances professionnelles et travail documentaire des enseignants : une étude de cas en chimie au lycée. *Recherches en didactique des sciences et des technologies* (*RDST)*, nº 15, p. 33-58.

KALAS I., BLAHO A. & MORAVCIK M. (2018). Exploring Control in Early Computing Education. In S. Pozdniakov & V. Dagiené (éd.), *Informatics in Schools. Fundamentals of Computer Science and Software Engineering*, ISSEP 2018, Saint-Petersbourg, New York : Springer, p. 3-16.

KERMEN I. & IZQUIERDO-AYMERICH M. (2017). Connaissances professionnelles didactiques pour l'enseignement des sciences et des technologies. *Recherches en didactique des sciences et des technologies (RDST)*, nº 15, p. 9-32.

KOMIS V., TOULOUPAKI S. & BARON G.-L. (2017). Une analyse cognitive et didactique du langage de programmation ScratchJr. In J. Henry, A. Nguyen & E. Vandeput (éd.), *L'informatique et le numérique dans la classe : qui, quoi, comment*, Namur : Presses universitaires de Namur, p. 109-122.

KOPPELMAN H. (2008). Pedagogical Content Knowledge and Educational Cases in Computer Science: An Exploration. *InSITE 2008 : Informing Science + IT Education Conference*. En ligne : <https://doi.org/10.28945/3228>.

MAGNUSSON S., KRAJCIK J. & BORKO H. (1999). Nature, Sources, and Development of Pedagogical Content Knowledge for Science Teaching. In J. Gess-Newsome & N. G. Lederman (éd.), *Examining Pedagogical Content Knowledge: The Construct and its Implications for Science Education*, Amsterdam : Springer, p. 95-132.

MASON S. L. & RICH P. J. (2019). Preparing Elementary School Teachers to Teach Computing, Coding, and Computational Thinking. *Contemporary Issues in Technology and Teacher Education*, vol. 19, nº 4, p. 790-824.

PERROT H. (2016). *Missions Scratch*. En ligne :<https://www.reseau-canope.fr/atelier-yvelines/spip.php?article1161> (consulté le 16 mars 2020).

ROGALSKI J. (1987). Acquisition et didactique des structures conditionnelles en programmation informatique. *Psychologie française*, nº 4, p. 275-280.

ROGALSKI J. (2015). Psychologie de la programmation, didactique de l'informatique. Déjà une histoire... In G.-L. Baron, É. Bruillard & B. Drot-Delange, *Informatique en éducation : perspectives curriculaires et didactiques*, Clermont-Ferrand : Presses universitaires Blaise Pascal, p. 279-306.

ROGALSKI J. & SAMURÇAY R. (1990). Acquisition of programming knowledge and skills. In J.-M. Hoc, T.R.G. Green, R. Samurçay & D.J. Gilmore (éd.), *Psychology of programming*, Elsevier, p. 157-174. En ligne : <https://doi.org/10.1016/B978-0-12-350772-3.50015-X>.

SAELI M. (2012). *Teaching Programming for Secondary School: A Pedagogical Content Knowledge Based Approach* [Phdthesis]. Eindhoven University.

SAELI M., PERRENET J., JOCHEMS W. M. & ZWANEVELD B. (2011). Teaching programming in Secondary school: A pedagogical content knowledge perspective. *Informatics in education. An International Journal*, vol. 10, nº 1, p. 73-88.

SCHNEEBERGER P. & TRIQUET É. (2001). Didactique et formation des enseignants : des recherches en didactique des sciences à la formation des enseignants : quels liens, quelles interactions? *Aster*, nº 32, p. 3-13.

SENSEVY G. & AMADE-ESCOT C. (2007). Une présentation de «Those who understand: Knowledge Growth in Teaching». *Éducation et didactique*, vol. 1, nº 1, p. 95-96.

SHULMAN L.-S. (2007). Ceux qui comprennent. Le développement de la connaissance dans l'enseignement. *Éducation et didactique*, vol. 1, n°1, p. 97-114.

WAITE J. L., CURZON P., MARSH W., SENTENCE S. & HADWEN-BENNETT A. (2018). Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal of Computer Science Education in Schools*, vol. 2, n° 1, p. 14-40.

WOOLLARD J. (2005). The implications of the pedagogic metaphor for teacher education in computing. *Technology, Pedagogy and Education*, vol. 2, n° 14, p. 189-204.

YADAV A. & BERGES M. (2019). Computer Science Pedagogical Content Knowledge: Characterizing Teacher Performance. *ACM Transactions on Computing Education*, vol. 19, n° 3, p. 1-24.

YADAV A., KRIST C., GOOD J. & CAELI E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*, vol. 28, n° 4, p. 371-400.