L'informatique, objets d'enseignements enjeux épistémologiques, didactiques et de formation

Pierre-André Caron, Cédric Fluckiger, Philippe Marquet, Yvan Peter, Yann Secq

DIDAPRO 8 – DIDASTIC

L’informatique, objets d’enseignements enjeux épistémologiques, didactiques et de formation

5-7 février 2020 Université de Lille, France

https://www.didapro.org/8/

Actes du colloque

édités par
Pierre-André CARON, Cédric FLUCKIGER,
Philippe MARQUET, Yvan PETER, Yann SECQ

Le colloque DIDAPRO 8 est organisé par l’université de Lille,
les laboratoires CIREL et CRIStAL

Préambule

Depuis 1988, les colloques de didactique de l'informatique, puis le colloque Didapro – DidaSTIC explorent les thèmes autour de l'enseignement et l'apprentissage de l'informatique. La huitième édition du colloque a eu lieu à l'Université de Lille, en France, du 5 au 7 février 2020. Pour l'occasion, les thématiques principales suivantes ont servi de lignes directrices:

- Pratiques d'enseignement, activités d'apprentissage
- Ressources, dispositifs, scénarios et environnements pour l'enseignement de l'informatique
- Politiques publiques, curricula, institutions, formation des enseignants

Comité scientifique

Comité d'organisation

Pour citer les actes :
P.-A. Caron, C. Fluckiger, P. Marquet, Y. Peter, & Y. Secq (Éds), *L'informatique, objets d'enseignements enjeux épistémologiques, didactiques et de formation*. Actes du colloque DIDAPRO 8 – DIDASTIC, Lille, France, 5-7 février 2020

Table des matières

Dispositifs pédagogiques à l'école primaire

Approches didactiques et pédagogiques

L'informatique, objets d'enseignements – enjeux épistémologiques, didactiques et de formation. Éditorial des actes du colloque

Pierre-André Caron[1], Cédric Fluckiger[1], Philippe Marquet[2], Yvan Peter[2], Yann Secq[2]

[1] Théodile-CIREL (EA 4354), Université de Lille, France
[2] CRIStAL (UMR 9189), Université de Lille, France

Résumé Le colloque Didapro-Didastic 8 rend compte de l'évolution de l'enseignement de l'informatique, au moment où la France remet en place options et spécialités dans le secondaire et préconise des enseignements dès l'école primaire. D'autres pays connaissent des enseignements de plus longue date et le colloque, dans la continuité des précédents, permet de croiser tant les expériences que les recherches sur ces enseignements.

Mots-clé : didactique, informatique, numérique, enseignements scolaires

1 Évolutions curriculaires, évolutions de la recherche

Le colloque Didapro-Didastic 8 est organisé, début 2020, à un moment charnière pour l'enseignement de l'informatique. Depuis quelques années, notamment en France, de nouvelles options et spécialités ont été créées au niveau lycée et 2020 sera la première année d'un tout nouveau concours de recrutement d'enseignants (CAPES).

Ces enseignements posent un certain nombre de problèmes, à tous les niveaux :

— dès le primaire, où des enseignements liés aux usages ou à l'algorithmique et la robotique sont prescrits ;
— dans le secondaire, avec la construction de ressources pédagogiques, ou la nécessité d'une meilleure compréhension des obstacles didactiques pour les élèves ;
— ou dans le supérieur qui a en charge la formation des enseignants.

Il n'est guère surprenant que parallèlement à cette présence plus grande de l'informatique, entendue dans un sens large, dans l'enseignement, les colloques Didapro-Didastic témoignent d'une réaffirmation plus nette de préoccupations didactiques concernant l'informatique. Il ne s'agit plus désormais uniquement de décrire des activités instrumentées, mais bien de comprendre les processus d'enseignement/apprentissage spécifiques lorsqu'on enseigne (et qu'on apprend) *de l'informatique*.

Cette (re)centration du colloque sur de tels objets est le fruit d'une évolution progressive mais nette. La première édition du colloque de *didactique des progiciels*, en 2003, 7 ans après le dernier colloque de didactique de l'informatique de 1995, visait à « *présenter des contributions de chercheurs investis dans l'enseignement des outils bureautiques, de formateurs concernés par les problèmes didactiques rencontrés et de professionnels de la publication ou de chercheurs en typographie numérique* »[3]. Dans le contexte de l'époque, où les discours officiels accordaient une place prépondérante aux « *outils informatiques* » (voir Baron et Bruillard, 2001), les préoccupations liées à la programmation et à l'algorithmique régressaient (Rogalski, 2015), et les chercheurs se centraient davantage sur des questions d'usages et d'instrumentation. Le colloque Didapro était dans ces circonstances une manière d'affirmer que les outils ont malgré tout besoin d'être appris : « *sous une apparence simple et banale, les progiciels et même le traitement de texte (qui est en fait devenu un outil de production de documents multimédias) restent complexes à appréhender* » (*op. cit.*). Il s'agissait, de ce fait, de dépasser l'opposition binaire classique entre l'informatique outil d'enseignement ou objet d'enseignement.

À la fin des années 2010, où la préoccupation institutionnelle quasi exclusive concernait la question des "compétences" (qui trouvait son expression dans les certifications comme, en France, le B2i ou le C2i), le colloque a acté, en 2011, le retour de préoccupations plus nettement liées à la science informatique, par un changement de nom. Devenant Didapro-Didastic, le colloque a remis la question des contenus informatiques au centre de ses préoccupations, comme l'exprimait l'argumentaire du colloque de 2013 : « *Comment sont intégrées l'informatique et les STIC dans les curricula, quels sont leurs contenus et leurs objectifs de formation ?* »[4].

Le présent colloque se situe dans la continuité des rencontres précédentes, qui ont mis en question l'opposition historique entre d'un côté ce qui relève des « outils » et de l'autre ce qui relève des « objets » d'enseignement/apprentissage. Son intention première était de renforcer la confrontation des regards sur l'enseignement informatique entre praticiens, informaticiens et spécialistes de l'éducation (didacticiens, psychologues de l'apprentissage, sociologues des usages du numérique, linguistes, etc.), concernant tous les niveaux.

2 L'informatique en éducation, un objet labile

Le colloque porte en creux la question de ce qu'est, pour ceux qui étudient son enseignement, l'informatique. C'est ce que signalent les deux pluriels du titre du colloque : « L'informatique, objets d'enseignements… ». Le constat que l'informatique, en tant qu'objet à enseigner ou à apprendre, ne se laisse pas

3. Argumentaire du premier colloque Didapro, 2003, halshs.archives-ouvertes.fr/DIDAPRO
4. Argumentaire du 5e colloque Didapro-DidaSTIC, didastic.sciencesconf.org/resource/page/id/1

facilement appréhender, n'est guère nouveau. La tradition des travaux en didactique, particulièrement vives à la charnière des années 1980 et 1990, discutait frontalement de la nature ontologique de l'informatique, avec l'idée récurrente d'une pluralité interne (Mirabail, 1990 ; Lang, 1998 ; Bruillard, 2016). Reste que dans les pratiques effectives de classes, dans les dénominations des textes officiels ou dans les conceptions qu'en ont les enseignants et les élèves, les frontières sont perméables avec d'un côté technologie éducative, la robotique, la bureautique ou même certaines dimensions des mathématiques. Dans les petites classes, les frontières sont d'ailleurs encore plus floues : tous les participants au colloque ne seraient sans doute pas d'accord pour savoir si réaliser un collier de perle en suivant un « algorithme » de couleur ou de forme est, déjà, un pré-apprentissage de l'informatique... Identifier des activités et des objets (Fluckiger, 2019) que nous, chercheurs ou praticiens, identifions comme relevant de l'informatique n'est donc pas le moindre défi et ce colloque, sans chercher à apporter une réponse unique, permet une confrontation des points de vues.

À cette labilité des objets, s'ajoute une pluralité des regards disciplinaires. C'est en effet la grande originalité scientifique de la didactique que de prendre à bras le corps la question de la spécificité des savoirs en jeux, de « *prendre prioritairement toutes les questions par le fil du rapport au savoir* », (Lahire et Johsua, 1999, p. 35). Cela suppose une collaboration étroite et un dialogue constant entre les spécialistes de la discipline d'enseignement d'une part, ceux de l'enseignement/apprentissage d'autre part. Ce colloque témoigne de la complémentarité nécessaire de ces deux regards.

3 Un colloque varié... et des points aveugles

L'appel à communication du colloque proposait 3 axes principaux :
- Pratiques d'enseignement, activités d'apprentissage
- Ressources, dispositifs, scénarios et environnements pour l'enseignement de l'informatique
- Politiques publiques, curricula, institutions, formation des enseignants

Les propositions ont été nombreuses pour analyser des pratiques de classe, des activités d'enseignement et d'apprentissage, dans des contextes et à des niveaux variés. Cela confirme, qu'apprendre et enseigner l'informatique sont désormais des réalités, tant dans le monde scolaire, universitaire, de formation professionnelle que dans d'autres lieux, comme le périscolaire, des lieux d'apprentissage informel. Comme il était attendu, des éclairages disciplinaires complémentaires ont été apportés.

Pour autant, on peut relever que notre communauté de recherche ne s'est pas encore penchée avec autant d'attention sur tous les aspects de l'enseignement de l'informatique. L'appel à communication du colloque précisait que « les approches permettant de lier les phénomènes d'enseignement/apprentissage aux contextes sociaux et culturels seront appréciées, dans le sens où elles constituent encore un point relativement aveugle des travaux récents". Cependant, force est de constater que ce point n'aura que peu été éclairé par les propositions de

communication. Que presque aucune communication ne s'inscrive dans cet axe dit quelque chose de notre domaine de recherche. Il est bien sûr naturel, face à des enseignements dont on sait encore peu de choses, face à des difficultés qui n'ont pas toutes encore été décrites, cette question préoccupe prioritairement les chercheurs. Ce sera sans aucun doute un signe de maturité de notre domaine de recherche, lorsque, à l'image de ce qui se passe pour d'autres didactiques, nous attirerons davantage de chercheurs venus d'autres horizons théoriques et disciplinaires, qui étudieront les pratiques extrascolaires des élèves, les conditions sociales de ces enseignements, les effets de genre (qui font l'objet d'une conférence invitée au colloque), la manière dont concrètement se construisent les inégalités dans ce domaine de connaissance, etc.

On peut encore relever que dans la variété des approches théoriques convoquées par les communicants, peu s'inscrivent finalement dans des cadres théoriques relevant des didactiques. Pour analyser ce qui se passe dans les classes, les concepts majeurs construits par les didactiques (milieu, situation, jeu, savoir savant, discipline, champ conceptuel, posture, rapport à, pratique langagière...) sont encore peu mis à contribution. De même, les démarches classiques en didactique des sciences et technologies (analyse du savoir *a priori*, etc.) ne sont que peu mises en œuvre. On peut bien sûr penser que la nature intrinsèque des savoirs informatiques rendrait ces concepts et méthodes inopérants pour l'analyse de leur enseignement/apprentissage. On peut aussi penser qu'il y a là un capital, construit depuis une quarantaine d'année en didactique, qui pourrait, par un travail de construction théorique maîtrisé, apporter des éclairages nouveaux sur nos objets de recherche.

Ce colloque donne donc à voir un domaine de recherche en reconstitution, portant sur un objet en ré-émergence, qui répond à un besoin social majeur face au renouveau des enseignements et, partant, de formation des enseignants.

Références

1. Baron ,G.-L, Bruillard, É. : Une didactique de l'informatique ?, Revue française de Pédagogie, 135, 163-172 (2001).
2. Bruillard, É. : « Quelle informatique à repenser et à construire pour les élèves de l'école primaire ? » In F. Villemonteix, G.-L. Baron et J. Béziat, dir., L'école primaire et les technologies informatisées. Des enseignants face aux TICE, Lille, Presses Universitaires du Septentrion, p. 29-38 (2016).
3. Fluckiger, C. : Une approche didactique de l'informatique scolaire, Presses Universitaires de Rennes, Rennes (2019).
4. Lahire, B., Joshua, S. : Pour une didactique sociologique, Éducation et sociétés, 4(2), 29-56 (1999).
5. Lang, B. : L'Informatique : Science, Techniques et Outils. LexiPraxi 98, journée de réflexion sur le thème « Former des citoyens pour maîtriser la société de l'information ». Paris, Maison de l'Europe, 9 décembre 1998 (1998).

6. Mirabail, M. : La culture informatique. ASTER, 11, 11-28 (1990).
7. Rogalski J. « Psychologie de la programmation, didactique de l'informatique, déjà une histoire... » In G.-L. Baron, E. Bruillard & B. Drot-Delange (Éds.), Informatique en éducation : perspectives curriculaires et didactiques (pp. 279-305). Clermont-Ferrand : Presses Universitaires Blaise Pascal (2015).

Quelle actualité pour les questions abordées lors du premier colloque de didactique de l'informatique (1988) ?

Monique Grandbastien[1]

[1] LORIA, Université de Lorraine, Nancy, France
monique.grandbastien@loria.fr

Résumé. Ce document présente brièvement les sujets développés lors du premier colloque francophone de didactique de l'informatique (1988) et décrit plus particulièrement ceux relatifs à l'apprentissage des bases de la programmation par des publics débutants, notamment dans l'enseignement secondaire. Il les complète par l'analyse de quelques articles de publications internationales plus récentes (1995-2019). L'objectif est de questionner l'actualité de ces sujets dans le contexte actuel de la création des enseignements de SNT et NSI dans les lycées français pour la formation des maîtres et pour l'observation et la compréhension des difficultés rencontrées par les élèves.

Mots-clés : algorithmique, programmation, informatique au niveau scolaire, didactique de l'informatique.

1 Introduction

En 1988 avait lieu à Paris le premier colloque francophone de didactique de l'informatique. Il était porté principalement par des enseignants d'informatique à l'université et acteurs de formation et d'accompagnement des maîtres pour l'option informatique des lycées à partir de 1981. Il réunissait à la fois des universitaires ayant l'expérience de la formation d'étudiants et de futurs professionnels aux bases de l'informatique et des enseignants de terrain des premier et second degrés, soucieux d'échanger leurs expériences pédagogiques. Les questions de didactique abordées dans les communications de ce colloque sont donc à analyser dans ce contexte pédagogique et dans le contexte technologique de l'époque. J. Arsac, son président, avait intitulé sa communication "la didactique de l'informatique : un problème ouvert ?" ; il semble que trente ans après le problème soit encore plus ouvert.

En effet, en 2019, la communauté des enseignants d'informatique en France (et dans quelques autres pays) retrouve une situation similaire dans un contexte évidemment très différent. Un enseignement de Sciences Numériques et Technologies (SNT) est introduit au lycée dans toutes les classes de seconde, après 20 ans d'interruption. Des milliers de professeurs doivent être formés, des approches pédagogiques sont à inventer, à tester et à diffuser.

Dans ce document, notre objectif est de présenter brièvement les sujets développés lors de ce premier colloque et d'analyser plus particulièrement ce qui concerne

l'apprentissage des bases de la programmation par des publics débutants, notamment dans l'enseignement secondaire. Nous les complétons en interrogeant des publications internationales plus récentes (1995-2019), puis nous questionnons leur actualité dans le contexte actuel. : Y-a-t-il des éléments à reprendre dans ces travaux, notamment pour la formation des maîtres et l'observation et la compréhension des difficultés rencontrées par les élèves actuels ?

2 L'existant en 1988

A la fin des années 80, la programmation était enseignée dans la plupart des universités, parfois depuis plus de vingt ans, et dans certains établissements secondaires, à titre le plus souvent optionnel ou expérimental. Les enseignants avaient très tôt noté les difficultés rencontrées par leurs étudiants et proposé, au fur et à mesure que les fondements de la science informatique étaient explicités [Dijkstra, 1976], [Gries, 1981], [Wirth, 1976], des démarches et des outils pour y remédier. Certaines communications des congrès WCCE[1] de l'IFIP[2] à partir de 1970 et de nombreux ouvrages au niveau français consacrés à la construction de programmes en témoignent comme par exemple [Arsac, 1970], [Arsac, 1980], [Arsac, 1987] et [Ducrin, 1984]. Ces ouvrages sont illustrés par de nombreux exemples dont la résolution est détaillée pas à pas. Les thèmes peuvent faire sourire par rapport aux préoccupations des étudiants actuels, ils n'intègrent pas les approches de construction d'applications par réutilisation de composants, mais le souci du détail dans la description de plusieurs solutions et des raisonnements qui ont conduit à ces solutions mériteraient d'être repris. Les principales questions soulevées par ces premiers cours d'informatique dans l'enseignement supérieur concernaient la créativité à mettre en œuvre dans la résolution des problèmes, la rigueur de l'expression d'une solution et la méthode, organisation rigoureuse du processus conduisant de l'énoncé d'un problème à l'expression d'un programme pour le résoudre.

Au niveau de l'enseignement scolaire, une option informatique des lycées avait été créée en France en 1981 et avait donné lieu à des publications pour faire circuler les expériences entre les acteurs des différentes académies. Une note de service de la direction des lycées (DLC)[3] en 1988 rappelle les brochures éditées par la direction des Lycées et le CRDP[4] de Poitiers « Enseigner l'informatique » (1985), « L'option informatique, réalités et pratiques » (1986), « Pratiques et savoir-faire des élèves de l'option informatique » (1987), « Technologie des ordinateurs et enseignement de l'option informatique » (1987), ainsi que la revue *Informatiques* éditée par les mêmes acteurs. C'est donc dans ce contexte et avec cet existant en mémoire qu'il faut analyser les actes du colloque de 1988.

[1] WCCE : World Congress on Computers in Education
[2] IFIP : International Federation for Information Processing
[3] Direction des Lycées et Collèges, note 88-084, Ministère de l'Education nationale
[4] CRDP : Centre Régional de Documentation Pédagogique

3 Difficultés pédagogiques abordées dans les actes du colloque

Les actes [Baron et al, 1988] contiennent 18 communications dont les titres, fournis dans le tableau 1, montrent une grande diversité d'intérêts. Certaines communications traitent plutôt des objectifs et avantages espérés d'un enseignement d'informatique (3, 4, 13, 15), d'autres de mise en œuvre dans l'enseignement supérieur (11, 17) ou des contraintes propres à l'école primaire (10, 12), d'exemples d'outils logiciels ou de mise en œuvre de démarches particulières (5, 8, 9, 10, 12, 16, 18). Nous analysons principalement dans les paragraphes qui suivent celles qui traitent des questions générales de l'enseignement de la programmation aux débutants (1, 6, 7, 14).

Tableau 1. Sommaire des actes du colloque de 1988

(1) Arsac J.	La didactique de l'informatique : Un problème ouvert
(2) Neuville Y.	L'espace informatique francophone
(3) De Landsheere G.	Processus d'apprentissage et d'enseignement de l'informatique : Quelques questions
(4) Morin B. A.	Pygmalion dans la classe informatique
(5) Guéraud V., Peyrin J.-P.	Un jeu de rôle pour l'enseignement de la programmation
(6) Rogalski J.	Enseignement de méthodes de programmation dans l'initiation à l'informatique
(7) Pair C.	L'apprentissage de la programmation
(8) Deveaux D., Raphalen M., Revault J.	Spécification d'un interprêteur de mini-langage algorithmique pour l'initiation à la programmation
(9) Dufourd G., Dufourd J.-F.	Des univers et des outils variés pour commencer à programmer
(10) Parmentier C.	Didactique et programmation à l'école
(11) Deroite M., Le Charlier B.	Un système d'aide à l'enseignement d'une méthode de programmation
(12) Aigle M.	« Vous avez dit binaire ? »
(13) Cloutier J.-F.	Apport de différents paradigmes de programmation comme autant d'outils de pensée
(14) Hauglustaine –Charlier B.	Images pour apprendre à programmer
(15) Romainville M.	Une analyse critique de l'initiation à l'informatique : quels apprentissages et quels transferts ?
(16) Geoffroy C.	Une initiation à l'analyse descendante
(17) Delacharlerie A.	Apprentissage de la conception des bases de données : une méthodologie de la méthode
(18) Favre-Nicolin R.	Le tableur et l'option informatique : vers la programmation par objets

Une difficulté générale déjà mentionnée dans [Arsac, 1987] tient au fait que l'enseignement de l'informatique, et notamment de la programmation, en dehors du cadre professionnel, n'a pas d'abord pour objectif la transmission de connaissances et de techniques, mais plutôt le développement d'aptitudes intellectuelles, créativité, rigueur de pensée, clarté et précision de l'expression, sens de l'organisation. Pour atteindre ces buts et dans la continuité des sujets débattus dans les ouvrages déjà publiés, certaines communications proposent plutôt des principes illustrés par des exemples, mettant l'accent sur la créativité sans oublier la rigueur, d'autres des méthodes plus strictes dont il ne faudra pas rester prisonnier pour se montrer créatif face à des situations nouvelles. Nous présentons successivement la construction des algorithmes, les méthodes et la phase de programmation-exécution en 3.1, 3.2, 3.3. Un

autre sujet développé en 3.4 concerne les observations relatives aux étudiants. Les auteurs mentionnent que les cours font souvent l'hypothèse d'un public assez homogène, motivé et capable d'autonomie, ce qui est déjà loin d'être toujours le cas à l'Université. Mais pour des élèves plus jeunes, la prise en compte de leurs caractéristiques en termes d'états cognitifs et de prérequis s'impose.

Globalement, ces communications portent davantage sur des réflexions quant aux objectifs et des propositions de mise en œuvre que sur des retours d'expérience avec description des succès et des difficultés observés chez les élèves, ce qui n'est pas surprenant dans une phase de création d'enseignement.

3.1 Du problème à un algorithme

Il est habituel de décomposer l'activité désignée de façon trop générique par le terme « programmation » en plusieurs phases. La première consiste à passer de l'énoncé d'un problème en langue naturelle à la description d'un mode de résolution exprimé comme combinaisons de quelques actions simples. Cette étape est signalée comme difficile par beaucoup d'auteurs, notamment à cause de la distance entre l'énoncé initial d'un problème et un algorithme pour le résoudre.

Parmi les solutions envisagées pour assister l'apprenti programmeur à cette étape, plusieurs auteurs proposent des principes dont on peut s'inspirer alors que d'autres tentent des méthodes plus systématiques dont nous aurons un exemple au paragraphe suivant. Parmi les principes souvent évoqués, on trouve le recours à l'observation d'un humain en train de résoudre un problème analogue, avec plusieurs objections immédiates : il s'agit de faire faire le travail à une machine et le mode opératoire humain n'est pas toujours celui qu'il faut retenir, ensuite un mode opératoire humain peut être difficile à expliciter avec la rigueur et la précision requises par les descriptions d'algorithmes, enfin on cherche en général un algorithme pour le réutiliser plusieurs fois pour la résolution de plusieurs problèmes d'une même famille, ce qui crée une situation différente de la résolution d'un seul problème.

Après la difficulté relative à la découverte d'une stratégie de résolution, les auteurs pointent des difficultés dues aux constructions et aux notations des langages algorithmiques, notamment une qui vient heurter les habitudes prises par les élèves en mathématiques. Il s'agit de la notion de variable, avec, dans les notations mathématiques une association statique entre un nom de variable et une valeur et en informatique une association dynamique, un même nom de variable venant repérer plusieurs valeurs au cours de l'exécution du processus décrit. Cette difficulté a été travaillée en collaboration avec des didacticiens des mathématiques [Nguyen, 2005] dans les IREM[5]s et avec des spécialistes de psychologie cognitive pour la prise en compte des cadres cognitifs préexistant chez les élèves [Samurçay, 1985].

On note enfin l'absence de distinction dans cette phase entre la modélisation des données et celle de résolution du problème.

[5] Institut de Recherche sur l'Enseignement des Mathématiques, http://www.univ-irem.fr/

3.2 La question des méthodes

Avec la question des méthodes de programmation, nous avons un bon exemple de l'imbrication constante entre les progrès de la structuration de la science informatique et leurs conséquences sur l'évolution des enseignements. Faut-il des méthodes et pourquoi ? Un premier objectif tout à fait légitime consiste à assister les élèves dans leur activité de construction de programmes.
L'équipe de C. Pair à Nancy publie, sous le nom collectif Amédée Ducrin, en 1984, un ouvrage d'enseignement de la programmation en deux tomes intitulés respectivement « du problème à l'algorithme » et « de l'algorithme au programme BASIC – PASCAL – LSE ». Le premier tome est entièrement consacré à l'enseignement d'une méthode de construction d'algorithme, la méthode déductive MEDEE, qui guide le programmeur de l'énoncé d'un problème à un algorithme exprimé dans un pseudo-code et permettant sa résolution

Un article [Langer, 2010] du bulletin vert de l 'APMEP[6] publié en 2010 (donc avec des programmes de mathématiques antérieurs aux programmes actuels) propose d'utiliser la méthode MEDEE pour guider les élèves dans la construction d'un algorithme pour réduire des fractions au même dénominateur. Les traductions de cet algorithme sont ensuite proposées en Algobox, Python et Scratch. Il n'y a malheureusement pas de commentaires sur l'activité des élèves, mais a priori le professeur a souhaité proposer une méthode. Nous reproduisons en figure 1 la solution proposée qui nous permet d'illustrer sur cet exemple la méthode MEDEE :

- Partir du résultat final (c'est une méthode descendante) parce que le résultat est en général bien défini dans l'énoncé, donc ici écrire les valeurs calculées pour le numérateur NS et le dénominateur DS (on notera la présence d'un lexique précis à créer en colonne de gauche).
- Le définir dans le pseudo-code proposé en introduisant si nécessaire des intermédiaires et donc en décomposant le problème en autant de sous-
- problèmes, donc ici définir les deux résultats. D'abord NS à partir de N et du PGCD (N,D) dont le calcul nécessite l'introduction d'une fonction, à définir comme module séparé selon la même méthode, puis DS dont le calcul nécessite la valeur du PPCM (D1, D2), D1 et D2 étant les dénominateurs des fractions données.
- Définir de la même façon chaque résultat intermédiaire, donc ici N1, D1, N2, D2 qui sont les données du problème, puis les modules introduits, c'est à dire les fonctions PPCM et PGCD
- S'arrêter quand il n'y a plus de résultat à définir. L'algorithme est l'ensemble des définitions obtenues, à ordonner ensuite pour exécution.

Cependant, après plusieurs années d'expériences et de réflexions, C. Pair [Pair, 1988] se demandait dès 1988, comme d'autres auteurs, si l'on peut vraiment enseigner des méthodes et suggère de mettre simplement les élèves en situations d'en acquérir par leur pratique.

[6] APMEP Association des professeurs de mathématiques de l'enseignement public

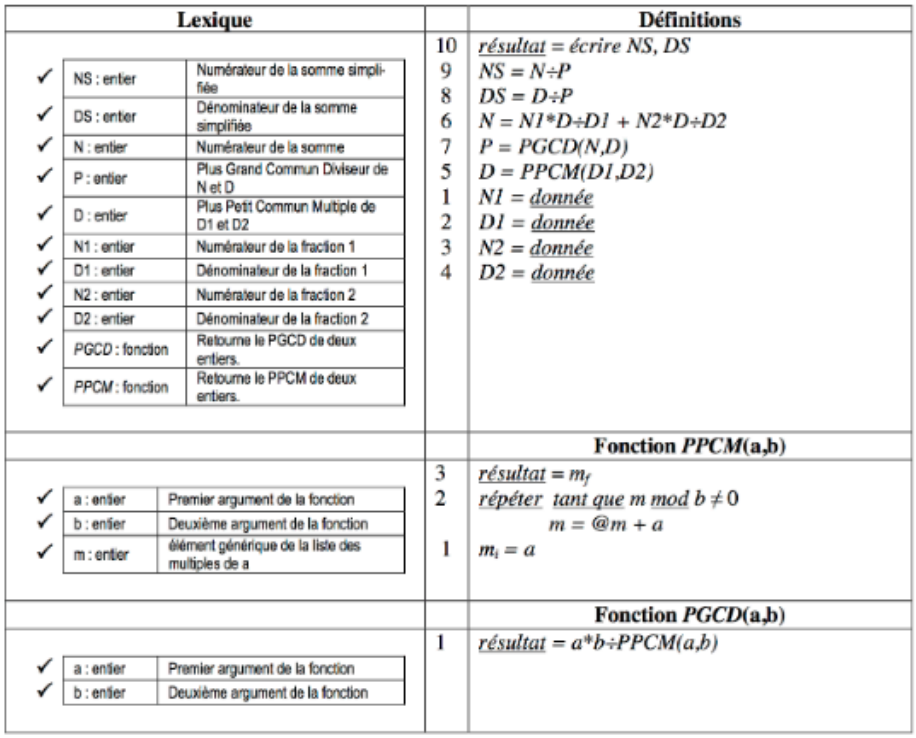


Fig. 1. Algorithme de réduction de deux fractions au même dénominateur : Copie de la solution proposée dans le bulletin vert APMEP n° 489

« Il n'y a pas un algorithme pour la fabrication d'algorithmes. Il y a des façons de faire » écrit J. Arsac, Il ne propose donc pas de méthode systématique, et pas de langage d'expression d'algorithme. On peut noter cependant qu'il utilise dans ses ouvrages un pseudo-code proche des langages de programmation impératifs de type PASCAL ou LSE. De façon plus générale, à cette période, la plupart des algorithmes étaient encore représentés graphiquement sous forme d'organigrammes. Les chercheurs reconnaissent aux organigrammes un pouvoir expressif certain, mais notent qu'ils ne sont d'aucune aide pour construire l'algorithme ainsi représenté, ils sont aussi un peu plus éloignés des langages de programmation, ce qui rend la phase de codage plus difficile.

Mais dans leurs différents articles de cette période, J. Arsac et C. Pair nous rappellent les problèmes rencontrés dans l'industrie du logiciel, attribués au manque de méthode pour la construction des applications, et l'arrivée de principes de génie logiciel pour assurer la construction de programmes plus sûrs dans des délais mieux maîtrisés. Il faut donc prendre de bonnes habitudes dès le début et apprendre à programmer avec méthode. Pour de futurs professionnels, cela ne fait aucun doute, pour la

formation générale, J. Rogalski (6) analyse les difficultés conceptuelles sous-jacentes à l'acquisition de méthodes chez les débutants.

3.3 Codage, vérification syntaxique et exécution

Plusieurs observations et propositions concernent la phase de codage, c'est à dire de traduction d'un algorithme exprimé avec un schéma ou un pseudo-code dans un langage de programmation. Certains auteurs du colloque (8), (16) voudraient l'éviter pour les premiers pas et proposent d'interpréter directement un langage algorithmique. Dans la logique de leur démarche, [Ducrin, 1984] proposent des schémas de traduction systématique des algorithmes MEDEE dans les langages utilisés à l'époque en France pour l'initiation à la programmation, c'est à dire BASIC, PASCAL, LSE.

En phase d'exécution, M. Romainville (15) montre comment des erreurs de logique interne se révèlent à ce niveau : variables non initialisées, nécessité de débogage, de test du comportement du programme.

Globalement cette phase d'exécution et de test est peu abordée, alors que c'est seulement avec l'exécution du programme que vont apparaître des imperfections au niveau de l'interface (quelle donnée faut-il entrer si on n'a pas pensé à faire afficher un message l'indiquant), des erreurs de logique dans le programme (pas de résultat affiché ou pas le résultat attendu). Enfin, les communications n'abordent pas non plus la lecture, la modification et l'exécution de code existant.

3.4 Quels prérequis et quelles représentations chez les élèves ?

Une quatrième préoccupation apparaît dans plusieurs communications et elle semble particulièrement pertinente dans le contexte d'hétérogénéité forte des classes actuelles de collège et de lycée. Elle constitue le fil conducteur du papier de C. Pair intitulé « l'apprentissage de la programmation » (7). L'introduction pose le problème : « La discipline à enseigner n'est pas alors le seul point de départ ; le professeur doit aussi être conscient du cadre préexistant chez l'élève. L'apprentissage a pour but de faire évoluer ce cadre pour le rapprocher plus ou moins de celui de l'expert, à travers des stades intermédiaires : la représentation que l'élève a de l'activité qu'il exerce va être confrontée à des expériences qui pourront la confirmer, l'enrichir, la remettre en question : il ne s'agit que du processus d'assimilation-accommodation de Piaget. » La suite du papier propose et illustre une décomposition de l'activité de construction de programme en stages successifs qui pourraient représenter autant d'états mentaux correspondants : Faire avec une machine, faire-faire, décrire un ensemble de calculs, décrire le programme, décrire de manière structurée, introduire les variables informatiques, etc.

B. Hauglustaine-Charlier (14) propose des animations vidéo comme support au cours donné par C. Duchâteau à de futurs enseignants. Il s'agit de concrétiser les métaphores utilisées dans le cours et de provoquer la construction d'images mentales chez les étudiants. Dans un autre ouvrage [Duchâteau, 90], cet auteur indique qu'il commence toujours ses cours en demandant à ses étudiants de dire ce que les mots « traiter des informations », qui interviennent dans la définition de l'informatique,

évoquent pour eux, avec l'objectif de relier les concepts nouveaux à ceux qu'ils ont déjà construits lors d'expériences antérieures. Un autre conseil pour amener son public au caractère formel des traitements informatiques est de travailler sur un continuum entre le facilement formalisable, par exemple calculer le prix à payer à la caisse du supermarché et le très difficilement formalisable, par exemple résumer un texte.

M. Romainville dans (15) parle aussi du besoin d'outils d'aide à la conceptualisation de certains processus sous-jacents à la programmation et suggère par exemple de fournir un modèle concret du fonctionnement d'une machine.

4 Plus récemment dans les revues internationales

Si des conférences internationales existent depuis 1970, les revues consacrées à l'enseignement de l'informatique apparaissent plutôt à la fin des années 1980, voire 1990. Pour les débutants en programmation, elles analysent essentiellement des cours d'un semestre d'introduction à la programmation en première année d'université, notamment aux USA. Les deux principales sociétés savantes américaines, ACM[7] et IEEE[8], ont constitué des groupes de travail sur l'enseignement secondaire, animé des conférences et proposé plusieurs versions d'un programme en informatique pour l'enseignement obligatoire (K12 curriculum). La revue ACM Transactions on Computing Education[9]créée en 2001 affiche comme objectif la publication de recherches sur l'enseignement de l'informatique à tous les niveaux de formation. Cependant, peu de communications traitent de l'enseignement scolaire, nous analysons ci-après des travaux traitant de formation des maîtres, d'élèves du niveau scolaire ou faisant une synthèse des difficultés des débutants avec l'objectif d'en transposer les conclusions vers l'enseignement secondaire.

4.1 Langage de programmation visuel ou textuel

La question de l'apprentissage des concepts de base de la programmation avec Scratch a été largement documentée. Par exemple [Armoni et al., 2015] publient une étude sur l'assimilation par des élèves de collège (middle school) d'un ensemble de concepts de programmation : initialisation de variables, boucles simples, boucles avec condition d'arrêt, envoi de messages, variables, parallélisme. De plus, l'utilisation de Scratch favoriserait la créativité, permettant ainsi d'atteindre l'un des objectifs de l'initiation à l'informatique chez les collégiens. On pourra consulter aussi [Kalelioglu et al., 2014] et les références du site de Scratch au MIT [Scratch].

Au delà de l'analyse du passage d'un langage visuel, Scratch pour les élèves de collège en France, au langage textuel Python proposé pour coder et exécuter leurs algorithmes à partir du lycée, d'autres modalités d'initiation à la construction de programmes méritent évidemment attention. C'est le cas du pilotage de robots ou autres

[7] ACM : Association for Computing Machinery
[8] IEEE Computer Society
[9] ACM Transactions on Computing Education : https://toce.acm.org/

objets tangibles et de l'utilisation de simulations numériques. Les robots sont proposés comme support possible pour atteindre les objectifs fixés aux cycles 2 et 3 de l'école primaire, le lecteur intéressé trouvera des références à ce sujet dans la bibliographie sélective commentée publiée par [Baron et al., 2016]. Une étude comparative, certes encore limitée, entre utilisation d'objet tangible ou de simulation numérique est proposée par [Fessard et al., 2019] et peut ouvrir la voie.

Enfin des modalités variées qui permettent d'aborder des sujets ludiques ou réels supposés plus stimulants ne doivent pas masquer la nécessaire unité des approches d'initiation à la programmation.

4.2 Contenu d'une formation pédagogique des enseignants

Les documents traitant de la méthodologie de l'enseignement de l'informatique sont assez rares, l'ouvrage « enseigner l'informatique » [Hartmann et al., 2012] traduit de l'allemand et rédigé par des enseignants suisses aborde de façon très pratique nombre de questions dans lesquelles les professeurs reconnaissent leur quotidien. Dans un article récent, [Yadav & Bergès, 2019] constatent qu'il y a partout dans le monde une demande très forte d'enseignants en informatique, des dizaines de milliers de professeurs à former, et font la proposition d'un test pour mesurer les connaissances pédagogiques nécessaires pour enseigner l'informatique. L'instrument se compose d'une liste de segments de travaux d'élèves (Scratch et Python) accompagnés de questions du genre « quelle explication donneriez-vous à ce stade du travail », « comment aideriez-vous cet élève ? ». Egalement à propos de pédagogie pour enseigner l'informatique au niveau scolaire, l'Académie des Sciences britannique (The Royal Society) a commandé une revue de littérature assez complète afin de synthétiser ce qui est connu à propos des pédagogies possibles pour enseigner l'informatique au niveau scolaire [Waite, 2017].

5 Actualité des questions et perspectives

Les difficultés des élèves dans leur activité de construction de programmes. Dans le programme de Sciences Numériques et Technologie pour la classe de seconde, la programmation apparaît comme notion transversale. On peut lire comme capacités attendues « Ecrire, exécuter et mettre au point un programme. » Il est rappelé ensuite que « Au collège (cycle 4) les élèves ont découvert et pratiqué les éléments fondamentaux d'algorithmique et de programmation. Le programme de seconde de mathématiques approfondit l'apprentissage de la programmation. » C'est donc au collège et au cours de mathématiques en seconde qu'il convient d'observer les difficultés des élèves dans leur activité de construction de programmes. Les compétences demandées en fin de collège sont illustrées par les sujets proposés à l'examen du Brevet des Collèges, on pourra consulter à ce propos l'étude réalisée par la commission inter-IREM informatique en octobre 2018 [Alayrangues et al., 2018].

La question de la construction d'un algorithme pour résoudre un problème donné est encore très ouverte, les images mentales que se forgent les élèves à propos des

programmes et de leur exécution sont importantes lors de leurs premiers contacts avec la programmation, elles devraient être mieux étudiées et les maîtres formés à en susciter. Elle pourrait être nourrie des observations faites par les professeurs de mathématiques et leurs collègues informaticiens à partir de certaines séquences du concours Castor[10] et déboucher sur une banque d'exercices ayant cet objectif, selon le niveau des élèves.

La place des méthodes est à redéfinir pour l'initiation. Pour les élèves qui choisissent le cours d'informatique en classe de première (NSI), le génie logiciel et ses phases de spécification, conception, architecture, test, maintenance doivent y avoir toute leur place tout au long des cours et des travaux pratiques.

La question des langages d'expression et du codage est loin d'être épuisée, elle peut être reprise à partir du passage de Scratch à Python, mais aussi plus directement et plus généralement en développant des outils pour évaluer les difficultés des élèves relativement aux concepts clé de la programmation.

La question d'habituer les élèves à lire du code existant dans le contexte du travail collaboratif ou de l'utilisation de bibliothèques de logiciels devrait aussi faire l'objet d'études et de recommandations. Commencer à faire programmer en modifiant du code existant est une approche souvent utilisée, cela suppose de savoir quand et comment « larguer les amarres » et proposer des exercices où tout est à construire, notamment le choix des données et de leur représentation

Enfin la question des prérequis et des représentations cognitives préexistant et évoluant chez les élèves ne semble pas avoir fait l'objet de beaucoup d'études et reste un champ important à défricher, en liaison avec les images mentales évoquées ci-dessus.

La formation des enseignants est évidemment un sujet transverse aux précédents et préalable à leur développement.

Pour l'enseignement primaire, il s'agit de sensibiliser à la pensée informatique en proposant des séquences réutilisables en classe. P. Tchounikine a mis en ligne un exemple possible de formation avec Scratch comme support [Tchounikine, 2017].

Au niveau du secondaire, plusieurs pays dont la France ont ouvert un grand chantier de formation continue dans des conditions difficiles, mais avec une implication remarquable des chercheurs, enseignants-chercheurs et professeurs du second degré et autres personnels de l'Education nationale, [Dowek et al., 2013] pour la spécialité ISN[11], puis [MOOC SNT, 2019], [DUI, 2019].

Pour les observations et recherches à entreprendre, [Hubwieser et al, 2015] ont coordonné deux numéros spéciaux rassemblant des expériences de développement de cours d'informatique au niveau secondaire dans différents pays afin d'offrir un panorama des initiatives récentes ou en cours. Dans leur introduction, ils abordent les questions de recherches ouvertes avec une impressionnante liste de 13 thèmes subdivisés chacun en plusieurs items, par exemple la qualification nécessaire pour les en-

[10] Castor Informatique France, http://castor-informatique.fr/

[11] ISN : Informatique et Sciences du Numérique, spécialité en terminales S et S-SI jusqu'en 2019

seignants et le cas du « tout formation continue », la motivation de l'élève et du professeur, les compétences qui doivent être mesurées et comment, la définition et l'évolution des objectifs et contenus de cours, le niveau de détail auquel il faut définir les contenus, la place du cours d'informatique dans les emplois du temps, etc. Dans cette longue liste, je retiens notamment le besoin d'instruments de mesure à long terme des compétences acquises aux divers niveaux de scolarité au travers d'activités individuelles mais aussi collectives.

Il faudrait aussi dépasser l'assimilation « didactique de l'informatique = didactique algorithmique-programmation » et mettre en évidence les difficultés d'enseignement posées par d'autres parties des programmes (représentation et gestion des données, internet et Web, réseaux, sécurité, etc.) ou celles plus générales liées à l'évaluation des compétences des élèves.

Enfin nous allons disposer de beaucoup de données d'apprentissage (modulo autorisations et anonymisation éventuelle). Ces données vont permettre de répondre à certaines des questions posées précédemment à propos des difficultés rencontrées, de leur contexte d'observation et des remédiations efficaces. Il est possible de tracer les essais/erreurs des élèves, d'analyser ces traces et de renvoyer des diagnostics et explications personnalisés. Ces nouvelles modalités de recherche n'ont pas encore été beaucoup utilisées pour documenter l'enseignement de l'informatique au niveau secondaire. Pour les débuts dans le supérieur, de telles études existent. Elles ont par exemple permis d'analyser les erreurs des programmeurs novices, y compris le temps passé à les corriger, de les catégoriser et de proposer la notion de « sévérité » d'une erreur [McCall & Kölling, 2019] comme produit de la fréquence et de la difficulté.

En conclusion, les questions évoquées pour les débutants en 1988 restent d'actualité, même si elles doivent être revisitées dans le contexte actuel. Beaucoup d'autres doivent être abordées, notamment pour couvrir les différents aspects des programmes. Comme souligné en 1988, ce travail nécessite une étroite collaboration entre chercheurs et praticiens, collaboration qui doit être organisée dans des programmes et cadres spécifiques qu'il est urgent de mettre en place.

Références

1. Alayrangues S. et al., Une analyse des exercices d'algorithmique et de programmation du brevet 2017, prépublication, https://hal.archives-ouvertes.fr/hal-02077738, (2019)
2. Armoni M., Meerbaum-Salant O., Ben-Ari M., From Scratch to « Real » Programming, *ACM Trans. Comput. Educ.* 14, 4, Article 25, 15 p., (2015)
3. Arsac J., La Science informatique, Dunod, Paris, (1970)
4. Arsac J., Premières leçons de programmation, Cedic, Paris, (1980)
5. Arsac J., Des pédagogies pour l'option informatique des lycées, in Pratiques et Savoir-faire des élèves de l'option informatique des lycées, DLC, ministère Education Nationale, Paris, (1987)
6. Baron G.-L., Baudé J., Cornu P. eds., Actes du premier colloque francophone de didactique de l'informatique, publié par Enseignement public et Informatique (EPI), http://www.epi.asso.fr/association/dossiers/d07som.htm, (1988)

7. Baron G.-L., Drot-Delange B., Touloupaki S., L'éducation à l'informatique à l'école primaire : une bibliographie sélective commentée. Adjectif.net, [En ligne] http://www.adjectif.net/spip/spip.php?article382, (2016)
8. Dijkstra E. W., A discipline of programming, Prentice Hall, New York, (1976)
9. Dowek, G., Archambault, J. P., Cimelli, C., Wack B., Baccelli, E., Cohen, A., Eisenbeis, C., Viéville T., Informatique et sciences du numérique, Eyrolles, 342 p., (2013)
10. Duchâteau C., Images pour programmer, Apprendre les concepts de base. De Boeck-Wesmael, Bruxelles, (1990)
11. Ducrin A. (nom collectif), Programmation, Tomes 1 et 2, Dunod, Paris, (1984)
12. DUI : Diplôme Inter Universitaire Enseigner l'Informatique au lycée, https://sourcesup.renater.fr/www/diu-eil/, consulté déc. 2019, (2019)
13. Fessard G., Wang P., Renna I., Objet Tangible ou Simulation Numérique: Deux Situations Équivalentes pour l'Apprentissage de la Programmation? In Environnements Informatiques pour l'Apprentissage Humain, Juin 2019, Paris, France, hal.archives-ouvertes.fr/hal-02156174/file/EIAH_2019.pdf, (2019)
14. Gries D., The science of programming. Springer Verlag. Berlin, (1981)
15. Hartmann W., Näf, M., Reichert, R., Enseigner l'informatique, (édition originale en allemand parue en 2006), Springer, Berlin, (2012)
16. Hubwieser P.et al, How to implement rigorous computer science education in k-12 schools ? Some answers and many questions, *ACM Trans. Comput. Educ.* 15, 2, Article 5, 12 p., (2015)
17. Kalelioğlu F., Gülbahar Y., The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective, *Informatics in Education*, Vol. 13, No. 1, 33–50, Vilnius University pub., (2014)
18. Langer B., MEDEE : Une méthode pour construire des algorithmes, Le bulletin Vert, APMEP 489, pp. 403-412, consulté déc. 2019 à l'URL http://numerisation.univ-irem.fr/AAA/AAA10055/AAA10055.pdf, (2010)
19. McCall D., Kölling M., A new look at novice programmer errors, *ACM Trans. Comput. Educ.* 19, 4, Article 38, 30 p., (2019)
20. MOOC SNT, S'initier à l'enseignement en Sciences Numériques et Technologie, https://www.fun-mooc.fr/courses/course-v1:inria+41018+session01/about, (2019)
21. Nguyen, C. T., Etude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice. Thèse de doctorat. Université Joseph-Fourier-Grenoble I. http://tel.archives-ouvertes.fr/tel-00011500/, (2005)
22. Pair C., Je ne sais (toujours) pas enseigner la programmation, *Informatiques* 2, 5-14 (1988)
23. Samurçay R. Signification et fonctionnement du concept de variable informatique chez les élèves débutants, *Educational Studies in Mathematics*, 16, pp. 143-161, (1985)
24. Scratch, https://scratch.mit.edu/research, consulté déc. 2019
25. Tchounikine P., Initier les élèves à la pensée informatique et à la programmation avec Scratch, http://lig-membres.imag.fr/tchounikine/PenseeInformatiqueEcole.html, (2017)
26. Waite J., Pedagogy in teaching Computer Science in schools : A Literature Review, added to The Royal Society Computing Education Project Report, https://royalsociety.org/~/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf, (2017)
27. Wirth N., Algorithms + Data Structures = Programs, Prentice Hall, New York, (1976)
28. Yadav A., Berges M., Computer Science Pedagogical Content Knowledge: Characterizing Teacher Performance. *ACM Trans. Comput. Educ.* 19, 3, Article 29, 24 p., (2019)

L'informatique prescrite à l'école primaire. Analyse de programmes, ouvrages d'enseignement et discours institutionnels.

Isabelle Vandevelde et Cédric Fluckiger[1]

[1] Théodile-CIREL (EA 4354), Université de Lille, France

Abstract. Depuis la rentrée de septembre 2016 est désormais prévue une initiation à la programmation informatique dès le primaire. Face à ces nouvelles prescriptions, et en l'absence de formation spécifique pour enseigner l'informatique, les enseignants doivent s'adapter en s'appuyant notamment sur une série de documents (textes prescriptifs, instructions officielles, documents institutionnels, ouvrages scolaires disponibles). Ces documents constituent-ils une aide, apportent-ils des éléments de clarification quant à l'enseignement de l'informatique ? Quelle est la place prévue selon ces documents pour l'informatique scolaire ? Quelles thématiques sont proposées aux enseignants ? Nous proposons ici d'analyser ces documents du point de vue de la didactique de l'informatique, en tenant compte de contenus spécifiques à l'informatique que nous détaillons et justifions.

Keywords : informatique, numérique, école primaire, programmes, manuels

1 Introduction : contexte et hypothèses

Depuis la rentrée de septembre 2016, un enseignement de l'informatique est prévu au primaire et au collège, après avoir connu bien des vicissitudes (Baron et Drot-Delange, 2016). En outre, l'enseignement de l'informatique est inscrit dans le socle commun de connaissances, de compétences et de culture au sein du premier domaine, « les langages pour penser et communiquer » (MEN, 2015c).

Cet enseignement pose un certain nombre de problèmes, aux élèves comme aux enseignants. Les élèves sont en effet confrontés à des *usages* d'outils informatisés, y compris hors de l'école, développant ce qu'on peut appeler une *culture numérique* (Fluckiger, 2008), qui peut entrer en tension avec ces enseignements (Vandevelde, 2019), les élèves ne saisissant pas toujours le lien avec l'informatique (Drot-Delange, 2013).

Les enseignants, eux, en l'absence de formation spécifique (Fluckiger et Bart, 2012 ; Baron et Drot-Delange, 2016 ; Spach, 2017), doivent faire face à ces enseignements avec l'aide des documents prescriptifs et d'accompagnement.

Si l'on suit les propositions de Reuter (2004), adaptées au cas des contenus informatiques (Fluckiger et Bart, 2012 ; Fluckiger et Reuter, 2014), les matières scolaires s'actualisent dans différents espaces : espace scriptural des *prescriptions* et *recommandations*, espace des *pratiques*, espace des *représentations*, ainsi que dans certains

espaces *extrascolaires* (espaces de recherche, médiatique, etc.). C'est au premier de ces espaces que s'intéresse cette communication.

Face aux prescriptions nouvelles (du moins en France), les enseignants d'école primaire sont sommés d'enseigner des éléments d'informatique et donc de réguler des activités d'apprentissage de l'informatique sans toutefois être toujours à même d'identifier clairement les contenus et les enjeux de savoir (Spach, 2017). Se pose alors la question, en l'absence de formation spécifique pour enseigner l'informatique, du poids et du rôle des textes prescriptifs et d'accompagnement, des instructions officielles et programmes, documents institutionnels, jusqu'aux ouvrages et manuels disponibles. Ces documents constituent-ils une aide, apportent-ils des éléments de clarification, ou au contraire proposent-ils des injonctions inatteignables et contradictoires, comme c'était le cas pour les textes relatifs au Brevet Informatique et Internet (B2i, voir Fluckiger et Bart, 2012), en vigueur jusqu'alors ?

Pour répondre à ces questions, cette communication mettra en tension les attentes scolaires de l'enseignement de l'informatique au travers de trois corpus :

- les programmes d'enseignement de 2015 ;
- des ouvrages destinés à l'apprentissage de l'informatique, postérieurs à ces programmes et pouvant être utilisés comme manuels aux cycles 1, 2, 3 et 4 ;
- des textes et rapports institutionnels, contemporains à leur élaboration.

2 Cadre théorique : catégoriser les contenus de l'informatique

Les évolutions récentes de l'enseignement de l'informatique sont envisagées ici d'un point de vue didactique en informatique, articulant des apports en didactique des disciplines et ceux plus spécifiquement sur l'informatique scolaire, développés des chercheurs en informatique, didactique et science de l'éducation.

Dans la continuité des travaux en didactique sur ce qui constitue une discipline scolaire, nous envisageons ici les programmes, discours institutionnels et ouvrages comme relevant d'un *espace scriptural* constituant une partie de la configuration disciplinaire (Reuter, 2007/2013). Nous désignerons *ce qui s'enseigne* par le terme de *contenu* (Daunay, Fluckiger et Hassan, 2015), qui renvoie pour nous à deux caractéristiques majeures. Déjà, les *contenus* désignent tout ce qui s'enseigne, et peuvent donc renvoyer à la fois à des savoirs, des savoir-faire, des valeurs, des rapports à… (Delcambre, 2007/2013). Ensuite, les *contenus* font l'objet d'un double processus de construction : ils sont élaborés en amont, lors de processus transpositifs (Chevallard, 1985/1991) ou en référence à des pratiques sociales, ils font ensuite l'objet d'une appropriation subjective par les acteurs, qui les reconstruisent, leur donnent du sens, leur assignent des utilités et des finalités, etc. (Fluckiger, 2019a). Dans cette perspective théorique, nous analysons ici ce qui est défini dans l'espace scriptural afin d'être proposé aux élèves comme *contenu*. Les prescriptions, discours institutionnels et ouvrages peuvent alors être vus comme relevant de différentes étapes du processus transpositif au sein de ce que Chevallard (1985/1991) désignait par le terme de *noosphère*.

Soulignons encore que la désignation de ces contenus pose question : s'agit-il de contenus informatiques ou numériques ? Nous suivons sur ce point Baron et Boulc'h (2011) lorsqu'ils affirment que le terme numérique « a été de plus en plus utilisé comme un équivalent et souvent comme une euphémisation de ce qu'on reliait autrefois à l'informatique et aux logiciels », ce qui nous incitera à être attentifs au fait que les différents acteurs peuvent avoir des définitions et une vision différentes de ce qui relève de l'un ou de l'autre. Ainsi, apprendre à manipuler un clavier peut éventuellement sembler relever de l'informatique pour un enseignant... mais il n'est pas certain que ce soit le cas pour un informaticien, etc. Cette question de l'*identification* des contenus étant justement à débattre (Fluckiger, 2019a), nous préférons la laisser ouverte, précisément pour être attentif à ce qu'en disent les textes que nous étudions : la manière dont les contenus sont désignés sera pour nous une variable de l'analyse et non un *a priori* du chercheur.

Enfin, nous nous appuyons sur des travaux plus directement ancrés en informatique pour catégoriser les différents contenus d'enseignement relevant de l'informatique (ou du numérique), notamment la triple caractérisation de l'informatique par Mirabail (1990) : science, technologie et agent social et culturel de changement ; la proposition des quatre concepts clés de l'informatique par Dowek (2012) : machine, information, algorithmique et langage ; ou encore des attracteurs proposés par Bruillard (2009) : algorithme, matériels et réseaux, activités humaines.

L'analyse de cette littérature et la lecture des ouvrages nous a conduits à établir une liste de 3 catégories de contenus possibles que nous trouvons dans les textes prescriptifs, dans les ouvrages comme dans les programmes et discours institutionnels :

L'apprentissage du fonctionnement des technologies qui consiste à étudier le fonctionnement des machines, l'historique des technologies et les enjeux de leur développement. Il s'agit par exemple pour les enfants de « décrire l'architecture simple d'un dispositif informatique » (MEN, 2015b, p. 67), de montrer comment des périphériques sont reliés à l'unité centrale (Cohen et Marcialis, 2018) ou encore de découvrir l'importance de l'informatique et de la notion de boucle dans les chaines robotisées de montage de voitures (Croq et al., 2015b).

L'apprentissage de l'algorithmique, de la programmation, des langages, du binaire, d'éléments de cryptographie, etc. : par exemple, l'élève peut être amené à lire un message codé avec des flèches afin de tracer sur un damier le parcours d'un robot virtuel (Croq et al., 2015a), soit « coder et décoder pour prévoir, représenter et réaliser des déplacements dans des espaces familiers, sur un quadrillage, sur un écran » (MEN, 2015b, p. 84).

L'apprentissage de l'utilisation des outils informatisés qui vise à savoir se servir des outils informatiques, des logiciels, des moteurs de recherche, etc. Il peut s'agir de s'approprier un logiciel de traitement de texte via divers exercices (Cohen et Marcialis, 2018, p. 14), mais aussi d'apprendre à « écrire avec un clavier rapidement et efficacement » (MEN, 2015b, p. 111).

L'ensemble de ces considérations nous conduisent à formuler l'hypothèse que les enseignants sont soumis à des discours contradictoires, sur ce qui doit être enseigné, sur le fait qu'il s'agit d'informatique ou de numérique, mais aussi sur les compétences

des jeunes ou sur le fait de savoir si les outils informatiques doivent être vus comme une chance ou une source de risque (Fluckiger, 2011 ; 2016).

3 Constitution de corpus textuels

Dans le prolongement des propositions théoriques de Reuter pour analyser les disciplines scolaires (Reuter 2007/2013, 2014) et sa déclinaison pour les contenus informatiques (Bart et Fluckiger, 2012 ; Fluckiger, 2019a), nous avons considéré ici 3 segments de corpus :

- les programmes scolaires de 2015 pour les cycles 1-2-3-4 (2 à 14 ans). Ce qui représente respectivement 21, 86, 126 et 171 pages pour les cycles 1, 2, 3 et 4 ;
- un ensemble de textes institutionnels évoquant le numérique ou l'informatique dans l'éducation (rapport parlementaire ou d'organismes proches des décideurs ministériels). Un tel corpus ne pouvant être exhaustif faute de définition précise de ce qui constitue les discours de la noosphère, nous avons fait le choix de limiter notre analyse à 5 documents datés de 2014 à 2018 (liste en annexe) ;
- un ensemble de 10 ouvrages d'informatique, destinés aux élèves, (1 en cycle 1, 3 pour chacun des autres cycles). Nous n'avons retenu que des ouvrages postérieurs à 2015, qui renvoient explicitement à un âge, classe ou cycle, et disponibles en décembre 2019 sur le site du libraire Decitre.

Méthodologiquement, les ouvrages ont été découpés en unités élémentaires que nous avons classées en *exercice* ou en *page de texte*. Nous disposons donc d'un corpus de 1170 éléments : 730 *pages de texte* et 440 *exercices*.

4 Forme des ouvrages pour l'enseignement de l'informatique

Le premier constat concerne la forme très diversifiée de ces ouvrages : présence ou absence de chapitres, d'un sommaire, de corrigés d'exercices, etc. Au-delà de ce premier constat, nous distinguons ainsi trois formes principales, le *livre de référence*, le *cahier d'exercices*, l'*album de littérature de jeunesse*.

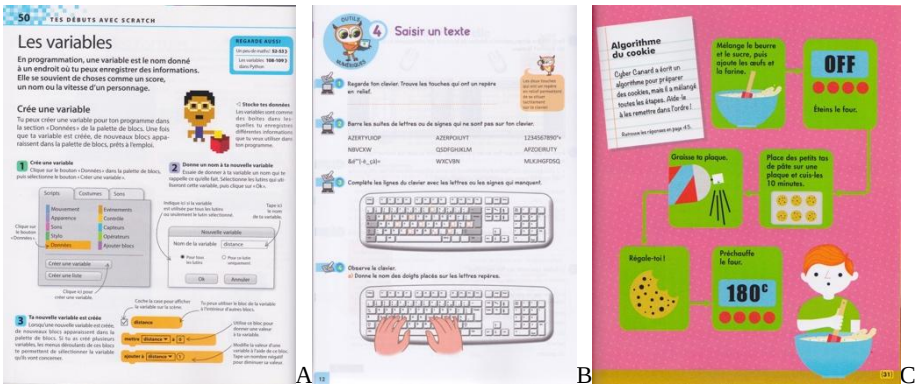
Les **livres de référence** (n = 4), proposent tous davantage de texte (n = 619) que d'exercices (n = 94). Il s'agit surtout d'expliquer aux élèves diverses notions relatives au codage en blocs, en JavaScript, en HTML, ou en Python, et d'expliquer l'utilisation des plateformes logicielles telles que Scratch ou Minecraft.

Les **cahiers d'exercices** (n = 4) proposent tous davantage d'exercices (n = 311) que de texte (n = 22). Ils s'apparentent à des feuillets d'exercices sur lesquels les élèves peuvent écrire dans des espaces dédiés. Les exercices proposés ont pour but de construire ou de consolider des savoirs en informatique.

Les **livres s'apparentant à des ouvrages de littérature de jeunesse** (n = 2), proposent tous davantage de texte (n = 89) que d'exercices (n = 35) mais dans un rapport moindre par rapport aux livres de référence : ils proposent seulement 2,5 fois plus de texte que d'exercices. Ces ouvrages proposent de découvrir l'informatique de manière déguisée : les notions sont abordées au travers d'une histoire ou au travers d'exemples issus de la vie quotidienne de l'enfant.

L'ensemble des ouvrages analysés compte en moyenne 73 pages de texte et 44 exercices. Nous pourrions ainsi croire que les ouvrages proposent davantage de textes que d'exercices. Néanmoins, ces valeurs sont à relativiser puisque l'analyse statistique montre un fort écart type à la fois pour les textes ($\sigma = 110$) et pour les exercices ($\sigma = 44$). Ceci signifie que le corpus d'ouvrages est particulièrement hétérogène, certains ouvrages étant beaucoup plus volumineux que d'autres de par leur nombre de pages : le plus petit ouvrage compte 46 pages et le plus volumineux en compte 338.

Figure 1. Aperçu d'un livre de référence (A), d'un cahier d'exercice (B) et d'un ouvrage de littérature de jeunesse (C).



A B C

5 Informatique et numérique : de quelques implicites

Dans le langage courant, les termes *numérique* et *informatique* sont souvent employés comme synonymes, sans réelle distinction. Cependant ces termes recouvrent deux dimensions non pas opposées, mais complémentaires. Baron et Drot-Delange proposent de les distinguer ainsi : « à l'informatique, l'information au sens mathématique ; au numérique, l'information au sens social » (2015, p. 20). L'information mathématique est mesurable et entre en jeu dans la transmission de messages sous forme de signaux, alors que l'information au sens social entre en jeu lors de l'interprétation et du sens donné aux messages (Shannon, 1948). Le numérique et l'informatique se différencieraient ainsi selon le caractère de l'information traitée : l'information sociale nécessitant une interprétation concerne le numérique alors que l'information au sens mathématique ne laissant pas de place à l'interprétation concerne l'informatique.

Les termes *informatique* et *numérique* désignent cependant des ensembles mal délimités, qui se recouvrent parfois très largement, et leur emploi est largement déterminé par des effets de modes (l'institution a longtemps abusé du syntagme « outil informatique », avant de parler de TIC, puis de numérique, sans que les réalités derrière ces termes ne soient toujours bien distinctes).

Cependant, nous pensons qu'il n'est pas inintéressant de faire le point sur les termes actuellement en vogue, pour lever d'éventuels implicites. Qu'en est-il donc dans nos corpus textuels ? Il apparait clairement que les textes institutionnels et les

programmes scolaires retenus utilisent bien plus largement le terme *numérique* (1831 fois pour les textes institutionnels et 217 fois pour les programmes scolaires) que le terme *informatique* (217 pour les textes institutionnels et 32 fois pour les programmes scolaires), mais à quelles idées renvoient ces deux termes dans notre corpus de texte ?

5.1 Le numérique

Dans ces textes, le numérique renvoie à un ensemble d'idées relativement hétérogènes. Le plus souvent à un ensemble d'outils relevant peu ou prou de la technologie éducative. Ce numérique étant « porteur de nombreuses opportunités » (Studer, 2018) et étant censé « pallie[r] les défaillances de notre système éducatif » (Institut Montaigne, 2016), ces discours institutionnels s'inscrivent dans les mythes du numérique en éducation largement démentis de longue date (Amadieu et Tricot, 2014), en affirmant à rebours des discours de recherche que le numérique (en général) aurait des effets, potentiellement transformatifs de l'école (Fluckiger, 2019b). Mais il fait également référence à un domaine (d'ailleurs nommé « domaine du numérique », MEN, 2018, « vie numérique », Studer, 2018, « monde numérique », CNNUM, 2014...), soit d'enseignements, soit relevant d'usages sociaux, pour lesquels il est nécessaire que les élèves développent des compétences car « le quotidien des enfants est déjà numérique » (Institut Montaigne, 2016). Ces deux acceptions du terme sont parfois marquées grammaticalement. Ainsi, dans plusieurs textes, numérique est utilisé à la fois comme substantif et comme adjectif, parfois dans la même phrase, comme « les enseignements portant à la fois spécifiquement sur le numérique ou utilisant des ressources et outils numériques » (MEN, 2018). Le terme peut enfin renvoyer à une supposée nouvelle forme de rapport au savoir remettant en cause la place même de l'éducation : « Le savoir change, ses modes de transmission et notre rapport à celui-ci également. Ce dernier échappe désormais au monopole des institutions académiques traditionnelles » (Institut Montaigne, 2016).

Le terme numérique est largement associé à une idée de changements brutaux et importants : « révolution numérique », « transformation numérique », « bouleversements numériques » sont des syntagmes qui reviennent avec constance sous la plume des acteurs institutionnels.

Dans toutes ces acceptions, ce n'est pas tant la nécessité d'une forme de pensée que l'on peut qualifier de pensée informatique (Wing, 2006) qui est donc en jeu. Il s'agit bien davantage d'utiliser des instruments (notamment pour enseigner) et d'apprendre aux élèves à les utiliser (cette idée étant exprimée généralement dans le vocable à la mode des compétences). Dans les programmes, le numérique est surtout perçu comme un outil parmi d'autres que les élèves doivent maîtriser afin de développer une série de compétences relatives au travail scolaire. Le socle commun (S4C) mentionne que l'utilisation de l'outil numérique doit notamment permettre à l'élève :

- d'organiser son travail personnel : l'outil numérique doit permettre à l'élève de réaliser des écrits pour s'entrainer, réviser et mémoriser ;
- de coopérer et réaliser des projets : l'outil numérique doit permettre à l'élève de s'organiser, d'échanger et de collaborer avec sa classe, son école, son établissement ;

- de rechercher et le traiter de l'information : l'outil numérique doit permettre à l'élève de produire, de recevoir et de diffuser de l'information ;
- d'échanger et de communiquer : l'outil numérique doit permettre à l'élève de créer, publier et transmettre des documents.

En d'autres termes, selon les programmes, l'enseignement des outils numériques vise à ce que l'élève puisse « utiliser de manière pertinente les technologies numériques pour faire des recherches, accéder à l'information, la hiérarchiser et produire soi-même des contenus » (MEN, 2015). Ce qui est important ici semble être l'utilisation de l'outil en tant que tel mais également le traitement de l'information. L'enseignement du numérique consisterait alors à apprendre aux élèves à utiliser les technologies informatisées et à réaliser un traitement social de l'information.

5.2 L'informatique pour comprendre le monde numérique ?

En réalité, dans aucun de nos corpus, les termes numériques et informatiques ne sont définis ou distingués l'un par rapport à l'autre, par exemple quant au degré de technicité de l'information traitée (Baron et Drot-Delange, 2015). Un seul ouvrage propose une distinction implicite, avec une première partie nommée « outils numériques » et une seconde « codage et programmation », mais sans mise en relation entre ces deux parties, mis à part l'utilisation de l'ordinateur, acquise dans la première partie, qui permet d'utiliser Scratch dans la seconde.

Dans les textes institutionnels, les deux termes se trouvent parfois associés, y compris en précisant qu'il faut les entendre au sens large : « Est venue s'y ajouter récemment la culture numérique et informatique au sens large, qui va de l'apprentissage de la citoyenneté numérique à celui des concepts de l'algorithmique et de la programmation » (Becchetti-Bizot, 2015).

Généralement se retrouve en filigrane l'idée que l'informatique serait une partie d'un ensemble plus vaste, le numérique. Ainsi, le CNNUM (2014) préconise « des enseignements numériques », qui se décomposent en « informatique » et « humanités numériques ». Il est significatif que lorsqu'il est question d'enseigner « le numérique », l'informatique ne soit ni le seul, ni même le principal ensemble de contenus. Ainsi, dans les propositions de Studer (2018) pour « l'enseignement du numérique à l'école », on trouve, en dernière proposition, « créer un CAPES et une agrégation d'informatique », mais toutes les propositions précédentes font référence à l'éducation aux médias, avec le renforcement de l'EMI et du rôle du CLEMI, mettant en avant les notions de « citoyenneté numérique » et de « fausse science ». Ce qu'il faut entendre alors par informatique est implicite, mais renvoie parfois à « une science » et le numérique à un phénomène de nature sociale : « un apprentissage de l'informatique en tant que science et un questionnement sur la place du numérique dans la société » (MEN, 2018).

Dans les discours comme dans les programmes, lorsqu'il est question d'informatique, il est fréquent qu'il ne soit en réalité question que de l'une de ses dimensions, la programmation ou l'algorithmique, souvent baptisée *codage*. C'est le cas aussi des ouvrages de notre corpus, puisque presque tous renvoient sur leur première de couverture à cette dimension : « à la découverte du codage », « je code »,

« deviens un programmeur », « cahier d'algorithmique et de programmation », « apprends à programmer », « premiers pas en programmation informatique », etc. Cela montre une forte tendance des ouvrages à associer, dans la continuité des textes prescriptifs et du Socle Commun (S4C), l'informatique à l'apprentissage d'un langage, du codage et de la programmation. Il ne faut cependant pas oublier que cela peut également être dû à nos critères de sélection.

5.3 Enseigner informatique et numérique : quels enjeux, quelles finalités ?

Enfin, tous ces textes portent une vision des enjeux et finalités d'un tel enseignement. Pour le CNNUM (2014), « c'est une réponse à l'attente sociale d'une politique de l'égalité : permettre à tous les élèves d'avoir une « clé » comprendre le monde numérique, participer à la vie sociale et se préparer à de nouveaux mondes professionnels ». Or, précise-t-il, la recherche est métamorphosée par les outils issus de l'informatique mais aussi par la « pensée informatique ». Il s'agit donc de « donner à nos enfants les clefs de leur vie numérique » (Studer, 2018), « d'enseigner la pensée informatique pour mieux comprendre le monde numérique qui nous entoure et être pleinement un citoyen actif dans la société » (CNNUM, 2014).

Le lien est, symétriquement, plusieurs fois fait entre les « compétences dans le domaine du numérique » (MEN, 2018) et les usages extrascolaires du numérique par les élèves. Ce texte précise que l'acquisition de ces connaissances « passe souvent par des expériences concrètes que les élèves peuvent vivre et poursuivre, dans un cadre scolaire ou hors temps scolaire ». L'idée que « les enfants français évoluent donc déjà dans un univers très largement numérique » (Institut Montaigne, 2016) est un constat à l'origine de la plupart des argumentations de ces discours.

De quelle manière les ouvrages mettent-ils en œuvre cette volonté de « réinvestir certains apprentissages informels, acquis en dehors de l'école » (Becchetti-Bizot, 2017) ? Ce qui est courant dans les ouvrages, c'est de lier l'algorithmique non pas directement à des activités numériques des élèves, mais plutôt à des opérations de la vie courante (organiser un gouter, construire une échelle en bois en reproduisant plusieurs fois la même opération, marcher en dessinant un carré, trouver un trésor sur une carte, etc.). Tout se passe comme si les opérations algorithmiques et la pensée informatique occupaient plus de place dans les activités non branchées que dans l'usage d'outils numériques. Il est en revanche rare que les opérations algorithmiques soient étayées par des exemples issus de l'usage des technologies.

6 Quels types de contenus informatiques ?

Nous avons précédemment identifié trois catégories de contenus informatiques (apprentissage du fonctionnement des technologies, apprentissage de l'algorithmique, apprentissage de l'utilisation des outils informatisés). Sous quelles modalités se retrouvent ces trois types de contenus dans les programmes d'enseignement et dans les ouvrages scolaires ?

6.1 Évolution au cours des cycles

Lorsqu'on s'intéresse aux programmes d'enseignement des cycles 1-2-3-4, et notamment aux « attendus de fin de cycle [et aux] connaissances et compétences associées » permettant le développement d'une culture informatique (soit 53 éléments) croisés avec les trois types de contenus informatiques identifiés précédemment, nous remarquons que c'est surtout le contenu relatif à l'apprentissage de l'utilisation des technologies qui est présent pour les cycles 1 à 3 et le contenu relatif à l'apprentissage de l'algorithmique qui est présent pour le cycle 4.

Nous remarquons également une véritable évolution au cours des cycles. Premièrement sur l'apprentissage de l'utilisation des outils informatisés qui commence au cycle 1 (découverte de divers outils), s'affine au cycle 2 (dactylographie), continu au cycle 3 (utilisation de logiciels usuels), et termine au cycle 4 par (utilisation des outils en réseaux). Deuxièmement sur l'apprentissage de l'algorithmique : absent au cycle 1, cet apprentissage démarre dès le cycle 2 (codage de déplacements) puis se perfectionne au cycle 3 (instauration de notions, spécifiques) pour aboutir au cycle 4 (application des concepts étudiés).

Qu'en est-il des types de contenus informatiques au sein des ouvrages scolaires ? Une logique similaire aux programmes d'enseignement se dégage-t-elle ?

Nous commençons par croiser les quatre cycles d'enseignement et les trois types de contenus informatiques identifiés.

Table 4. Tableau croisé : les types de contenus en fonction des cycles.

Cycle	algorithmique	utilisation	technologie	total
Cycle 1	84	0	1	85
Cycle 2	130	0	33	163
Cycle 3	452	299	39	790
Cycle 4	29	100	3	132
Total	695	399	76	1170

Nous observons ainsi que la moitié des éléments des ouvrages portent sur l'algorithmique, un tiers sur les questions d'utilisation des outils informatisés. L'informatique en tant que technologie est très sous-représentée dans les ouvrages analysés. Ce qui correspond à peu près à l'analyse réalisée sur les programmes d'enseignement : sans distinction du cycle, les programmes scolaires portent surtout sur les questions d'utilisation de l'informatique et sur l'algorithmique. La corrélation entre le cycle et le thème de contenu majoritaire est importante ($chi2 = 278.08$, $dl = 6$, $p < 0.001$). Ainsi, parmi les ouvrages analysés, ceux à destination des cycles 1 et 2 se concentrent sur les questions d'algorithmique, alors que la question de l'utilisation n'apparaît qu'à partir du cycle 3. De même, les questions liées à la dimension technologique de l'informatique se concentrent sur les cycles 2 et 3.

Ces résultats diffèrent de ceux établis lors de l'analyse des programmes d'enseignement, dans lesquels l'apprentissage de l'utilisation des outils informatisés est prévu dès le cycle 1, les questions d'algorithmique n'apparaissent qu'à partir du

cycle 2 et les questions liées à la dimension technologique de l'informatique se concentre sur les cycles 3 et 4. Ces différences peuvent notamment s'expliquer par le fait que les programmes, s'ils listent les compétences à acquérir à l'issu de chacun des cycles, ne donnent pas d'information quant à la durée et à l'ampleur d'une compétence par rapport à l'autre. Dans notre analyse, chaque compétence du programme scolaire a été comptabilisée comme un élément unique, mais il est possible que la traduction dans les livres de cette compétence ne représente pas un élément (soit une seule page ou un seul exercice) mais plusieurs.

6.2 Types de contenus et types d'exercices

Nous nous sommes ensuite demandé s'il existe une corrélation entre le type de contenus et les modalités, texte ou exercice, au sein des ouvrages scolaires. On constate que les ouvrages sont surtout expositifs : ils proposent davantage d'exercice que de texte, d'*expliquer* l'informatique que d'inviter les enfants à *pratiquer* l'informatique.

Table 5. Tableau croisé : les modalités texte ou exercice en fonction du type de contenu.

Type de contenu	texte	exercice	total
algorithmique	502	193	695
utilisation	180	219	399
technologie	48	28	76
Total	730	440	1170

Ceci peut sembler en contradiction avec les résultats de recherche qui indiquent que l'informatique doit être pratiquée pour être apprise (Fluckiger, 2019a). Cependant, il faut noter quelques ouvrages présentent très peu de texte et beaucoup d'exercices : en réalité, seuls 6 ouvrages ont plus de texte que d'exercices. L'informatique se pratique donc (les élèves sont amenés à faire de l'informatique) tout comme elle s'explique (les élèves sont amenés à se documenter sur l'informatique).

Ensuite, il existe une corrélation entre la modalité texte ou exercice et le type de contenu informatique abordé ($chi2 = 73.16$, $dl = 2$, $p < 0.001$). Les questions liées à l'utilisation des technologies passent plus par des exercices que par des textes, l'algorithmique passe surtout par les textes. Ce résultat est contre-intuitif, on aurait pu supposer que les ouvrages contiendraient des exercices liés aux questions d'algorithmique et de programmation et que les questions d'utilisations seraient plus abordées dans les textes.

7 Autonomie et ancillarité de l'enseignement de l'informatique ?

Il n'est pas rare, pour les disciplines scolaires, d'être prises dans une tension entre autonomie et ancillarité. Comme le rappelait Astolfi (2010), « aucune discipline n'a le monopole d'un contenu d'enseignement (puisqu'on n'apprend pas à lire qu'en français, ni à calculer qu'en mathématique), mais chaque contenu a besoin d'une discipline de référence qui prenne en charge sa structuration » (p. 217). Le pôle instrumental de l'informatique ayant dominé pendant plusieurs années les instructions officielles, les contenus informatiques ont de ce fait été conçus uniquement comme devant être au service des autres disciplines (Fluckiger et Bart, 2012), comme le disaient explicitement les programmes d'enseignement de l'école primaire de 2007, qui parlaient bien « des outils au service des diverses disciplines ». Il s'agissait là d'une rupture assez nette avec les arguments classiques pour l'enseignement de l'informatique à l'école, qui affirmaient tout à la fois :

- l'irréductibilité de l'informatique aux autres sciences, comme Arsac qui affirmait, en 1980, que « l'informatique a sa place à l'école au milieu des autres disciplines scientifiques [...] en raison de sa spécificité, de l'originalité de ses méthodes, et de l'extraordinaire enrichissement de la pensée scientifique qui en est résulté » ;
- sa proximité avec d'autres matières, lui donnant un statut de discipline carrefour, comme lorsque Lang affirme : « on y rencontre des problèmes de logique, des questions strictement mathématiques, des problématiques apparentées à la physique la plus théorique... On peut donc y trouver matière à discuter de nombreux concepts qui sont aussi pertinents dans d'autres domaines, et donc à éventuellement réduire la dichotomie qui est souvent perçue entre les sciences et les humanités » (1998).

Comment cette question de l'autonomie ou de l'ancillarité de l'informatique est-elle déclinée dans les textes de notre corpus ? Dans les programmes d'enseignement actuels, l'informatique n'est pas identifiée comme une matière scolaire à part entière : l'informatique est transdisciplinaire, inscrite dans chacune des matières des programmes d'enseignement et les trois types de contenus informatiques précédemment identifiés sont repérables à la fois dans des matières littéraires, scientifiques et artistiques. Ces catégories de contenus identifiées ne sont pas exclusives. En outre, l'informatique est très largement utilisée pour diversifier les apprentissages et les supports de cours. Il s'agit alors d'apprendre une autre discipline au travers de l'informatique. Cela se voit nettement dans les « exemples de situations, d'activités et de ressources pour l'élève » proposés dans les programmes d'enseignement. Cela se retrouve également dans notre corpus d'ouvrages lorsque les enfants doivent commencer par réaliser des calculs mathématiques avant de pouvoir créer des constructions dans Minecraft par exemple.

Conclusion

L'analyse des programmes, des discours institutionnels et des ouvrages fait apparaître plusieurs points saillants. L'informatique est conçue et présentée aux élèves comme une partie d'un ensemble plus vaste et moins délimité, dénommé « le numérique ». L'objectif affiché de l'enseignement de l'informatique est de donner des éléments de culture scientifique et technique pour comprendre le monde numérique entourant les élèves. Cependant, plusieurs éléments laissent penser que cet objectif ne peut être que partiellement atteint. Premièrement parce que les contenus référés directement à la dimension technologique sont minorés dans les programmes d'enseignement comme dans les ouvrages scolaires. Ce n'est que très partiellement que les élèves trouveront à l'école des éléments pour comprendre la diversité et les modes de fonctionnement du monde numérique. Deuxièmement, parce que même lorsque sont abordés les langages informatiques et l'algorithmique, ils sont moins souvent reliés à des questions et problèmes dans le champ du numérique qu'à d'autres domaines de la vie. On est alors en droit de se demander si les enfants peuvent prendre conscience que ces éléments d'algorithmique sont bien au fondement des outils numériques qu'ils manipulent quotidiennement.

Plus largement, nous retrouvons pour les contenus informatique un fonctionnement déjà mis en évidence pour d'autres contenus scolaires ou certaines évaluations comme celles du PISA qui reposent sur une contextualisation des savoirs dans des situations dites « proches de la vie réelle » (Bart et Fluckiger, 2015) mais dont les découpages opérés doivent beaucoup aux cadres disciplinaires. Ainsi, supposer qu'un enfant reconnaitra un algorithme dans la construction d'une échelle en bois c'est supposer qu'il peut prendre conscience qu'il s'agit là d'une construction disciplinaire, spécifiquement informatique, d'une activité banale, alors même que les savoirs de base nécessaires à cette conscience sont précisément l'objet de son apprentissage. Il est bien plus probable que les élèves ne voient là qu'un contenu disciplinaire sans plus de rapport avec la réalité que le théorème de Pythagore n'en a avec la mesure des distances hors du cadre des mathématiques scolaires.

En d'autres termes, l'informatique scolaire semble courir le risque de revivre les mêmes tensions entre savoirs scolaires et savoirs quotidiens que ce qu'ont vécu d'autres disciplines, fondées sur les mêmes illusions (Johsua et Dupin, 1993).

Références

1. Amadieu F., Tricot A. : Apprendre avec le numérique. Mythes et réalités. Retz, Paris (2014)
2. Arsac, J. : Informatique et enseignement général, Annexe 1 de L'éducation et l'informatisation de la société. Rapport au Président de la République sous la direction de Jean-Claude Simon, La documentation française, p. 152-165 (1981).
3. Astolfi, J.-P. : La saveur des savoirs. Disciplines et plaisir d'apprendre. ESF, Lyon (2008).
4. Baron, G.-L, Drot-Delange, B. : L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. Revue française de pédagogie 195 http:// rfp.revues.org/5032 (2016).

5. Bart, D., Fluckiger, C. : Évaluation, fabrication des contenus et disciplines d'enseignement, in B. Daunay, C. Fluckiger & R. Hassan (Eds.), Les Contenus d'enseignement et d'apprentissage. Approches didactiques, Bordeaux, Presses Universitaires de Bordeaux, 91-102, 2015.
6. Bruillard, É. : Place de l'informatique dans l'enseignement secondaire, réflexions introductives », *in* G.-L. BARON, É. BRUILLARD et L.-O. POCHON (Eds.), Informatique et progiciels en éducation et en formation. INRP, Lyon, 21-27 (2009).
7. Chevallard, Y. : La transposition didactique. Du savoir savant au savoir enseigné. La Pensée Sauvage, Grenoble (1985/1991).
8. Daunay, B., Fluckiger, C., Hassan, R. : Les Contenus d'enseignement et d'apprentissage. Approches didactiques. Presses Universitaires de Bordeaux, Bordeaux (2015).
9. Delcambre, I. : Contenus d'enseignement et d'apprentissage, *in* Y. REUTER (Ed.), Dictionnaire des concepts fondamentaux des didactiques, De Boeck, Bruxelles, 43-48, (2007/2013).
10. Dowek, Les quatre concepts de l'informatique », *in* G.-L. Baron, É. Bruillard & V. Komis (dir.), Actes du quatrième colloque international DIDAPRO 4 - Dida&Stic, New Technologies Editions, Athènes, 21-29 (2012).
11. Drot-Delange, B. : Enseigner l'informatique débranchée : analyse didactique d'activités. Colloque AREF, aout 2013, 1-13 (2013).
12. Fluckiger, C. : L'école à l'épreuve de la culture numérique des élèves. Revue Française de Pédagogie, 163, 51-61 (2008).
13. Fluckiger, C. : La didactique de l'informatique et les constructions sociales de la figure des jeunes utilisateurs. Recherches en Didactiques, 11, 67-84 (2011).
14. Fluckiger, C. : Culture numérique, culture scolaire : homogénéités, continuités et ruptures. Diversité, 185, 64-70 (2016).
15. Fluckiger, C. : Une approche didactique de l'informatique scolaire, Presses Universitaires de Rennes, Rennes (2019a).
16. Fluckiger, C. : Numérique en formation : des mythes aux approches critiques. Education permanente, 219, 17-30 (2019b).
17. Fluckiger, C., Bart, D. : L'introduction du B2i à l'école primaire : évaluer des compétences hors d'une discipline d'enseignement ?, Questions Vives, 7(17), http://questionsvives.revues.org/1006 (2012).
18. Fluckiger C., Reuter, Y. : Les contenus "informatiques" et leur(s) reconstruction(s) par des élèves de CM2. Étude didactique. Recherches en Education, 18, 64-78 (2014). http://www.recherches-en-education.net/IMG/pdf/REE-no18.pdf
19. Johsua, S., Dupin, J.-J. : Introduction à la didactique des sciences et des mathématiques. PUF, Paris (1993).
20. Lang, B. : L'Informatique : Science, Techniques et Outils. LexiPraxi 98, journée de réflexion sur le thème « Former des citoyens pour maîtriser la société de l'information ». Paris, Maison de l'Europe, 9 décembre 1998 (1998).
21. Mirabail, M. : La culture informatique. ASTER, 11, 11-28 (1990).
22. Reuter, Y. : Discipline scolaire. In Reuter, Y., Cohen-Azria, C., Daunay, B., Delcambre, I. & Lahanier-reuter, D. (Eds), Dictionnaire des concepts fondamentaux des didactiques, p. 85-89. De Boeck, Bruxelles (2007/2013).
23. Reuter, Y. : Analyser la discipline : quelques propositions. Actes du 9e colloque de l'AIRDF, Québec, 26-28 aout (2004).
24. Shannon, C. :A Mathematical Theory of Communication. The Bell System Technical Journal, 3, 379-423 (1948).

25. Spach, M. : Activités robotiques à l'école primaire et apprentissage de concepts informatiques. Quelle place du scénario pédagogique ? Les limites du co-apprentissage. Thèse de doctorat en Sciences de l'Education, Paris Descartes (2017).
26. Vandevelde, I. : Culture numérique et apprentissages scolaires de l'informatique. Journées jeunes chercheurs du Gis2IF, 22 novembre 2019, Saint-Denis (2019).
27. Wing, J. M. « Computational thinking ». Communications of the ACM, 3(49), 33–35 (2006).

Corpus d'ouvrages

Cycle 1 :
1. Liukas, L. (2016). Hello Ruby ! À la découverte du codage ! Grenoble : Glénat jeunesse.

Cycle 2 :
2. Croq, A., & al. (2015a). J'apprends à programmer tout seul ! Paris : Bordas.
3. Croq, A., & al. (2015b). Mon cahier pour apprendre à programme. Paris : Bordas.
4. Lyons, H., & Tweedale, E. (2017). Mon atelier code. Paris : Fleurus.

Cycle 3 :
5. Cohen, A., & Marcialis, J. (2018). Comprendre les outils numériques et programmer. Paris : Hatier.
6. Morgan, N. (2017). Javascript pour les kids. Paris : Eyrolles.
7. Vorderman, C. & al. (2017). À vos marques, prêts ? Codez ! Paris : Larousse.

Cycle 4 :
8. Anguenot, G. & al. (2016). Cahier d'algorithmique et de programmation. Paris : Delagrave.
9. Bassette, T. (2017). 1, 2, 3, je code avec Scratch. Paris : Larousse.
10. Plumel, D. (2017). 1, 2, 3, je construis avec Minecraft. Paris : Larousse.

Corpus de textes institutionnels

1. Becchetti-Bizot : (2017) : Repenser la forme scolaire à l'heure du numérique. Vers de nouvelles manières d'apprendre et d'enseigner. Rapport à monsieur le ministre de l'éducation nationale. https://cache.media.education.gouv.fr/file/2017/55/1/IGEN-Rapport-2017-056-Repenser-forme-scolaire-numerique-nouvelles-manieres-apprendre-enseigner_849551.pdf
2. CNNum (2014) : Jules Ferry 3.0, Bâtir une école créative et juste dans un monde numérique. https://cnnumerique.fr/files/2017-10/Rapport_CNNum_Education_oct14.pdf
3. MEN, 2018 : Le numérique au sercice de l'édcole de la confiance. http://cache.media.education.gouv.fr/file/08_-_Aout/36/1/DP-LUDOVIA_987361.pdf
4. Institut Montaigne (2016) : Le numérique pour réussir dès l'école primaire https://www.institutmontaigne.org/ressources/pdfs/publications/institut_montaigne_le_numerique_pour_reussir_des_l_ecole_primaire.pdf
5. Studer (2018) : rapport parlementaire Studer, 2018, http://www.assemblee-nationale.fr/15/rap-info/i1296.asp

Corpus de programmes

1. MEN (2015a) : Programme d'enseignement de l'école maternelle.
2. MEN (2015b) : Programmes pour les cycles 2, 3, 4.
3. MEN (2015c) : Socle Commun de Connaissances, de Compétences et de Culture.

Variables, grandeurs et types

Yannis Delmas-Rigoutsos[0000−0002−9274−3316]

Laboratoire TECHNÉ, Université de Poitiers, EA 6316, France
yannis.delmas@univ-poitiers.fr

Résumé En France, les enseignements du collège abordent en informatique, en mathématiques et en sciences, les notions de variable, d'indéterminée et de grandeur. Plusieurs travaux [17,8,21] montrent les difficultés conceptuelles associées à ces notions, y compris chez les enseignants. En nous appuyant sur l'histoire et l'épistémologie de ces disciplines, nous clarifions ces notions. Nous formulons ensuite des recommandations curriculaires [1] en nous appuyant sur les concepts informatiques de substitution, de correspondance preuve–programme [9] et d'indirection [24,6,25]. Nous plaidons notamment pour l'exploitation de la substitution et du lien fondamental entre type computationnel et nature de grandeur.

Keywords: Didactique de l'informatique [cs.CY]· Épistémologie de l'informatique [cs.GL]· Didactique des mathématiques [math.HO]

1 Contexte et problématique

1.1 Besoin initial des apprenants

En France, le cycle 4 [2] introduit de nombreuses nouvelles notions. Parmi celles-ci la variable et d'autres notions proches, en mathématiques et en sciences : *indéterminée* et *inconnue* en calcul littéral, *variable* et *constante* en sciences.

Nous verrons que ces notions sont historiquement associées (sec. 1.2). Pour autant, leur épistémologie et leurs usages pratiques diffèrent. Plusieurs travaux antérieurs ont souligné la confusion qui pouvait en résulter pour les élèves et même pour certains enseignants, cf. [17,8,21] et leurs références. Les notions de variable et de grandeur étant des notions fondamentales, au sens de [9], elles sont omniprésentes en informatique et en sciences, y compris à un niveau élémentaire. Il serait donc pertinent de travailler leur conceptualisation afin d'alimenter des didactiques disciplinaires, ne serait-ce que pour réduire cette confusion.

Cet article a pour objectif d'analyser épistémologiquement la notion de variable informatique et les notions proches afin de proposer des pistes didactiques. Nous centrerons cette étude sur le cycle 4 (collège), mais quelques recommandations peuvent être pertinentes pour les cycles précédents. Après avoir rappelé le cadre curriculaire actuel et présenté quelques-uns des principaux jalons historiques les concernant, nous analyserons ces notions au regard des concepts informatiques de substitution (sec. 2), de correspondance preuve–programme (sec. 3.1) et d'indirection (sec. 3.2).

1. Définitions et recommandations sont récapitulées en fin d'article, sec. 4.
2. Cycle 4 : trois dernières années du collège (secondaire inférieur), 12–15 ans.

1.2 Éléments d'histoire : variables et notions associées

Les variables, grandeurs et notions associées sont extrêmement banales en sciences et en techniques, ce qui rend plus opaques leurs difficultés conceptuelles et par conséquent didactiques. Afin de les faire mieux apparaître et surtout d'expliciter leurs interrelations, rappelons quelques éléments de leur histoire.

Mesurage [3] **et grandeurs** Parlant de la science moderne, on cite souvent Galilée affirmant que la connaissance de l'Univers est écrite en « langue mathématique » [11, cit.]. Pour autant, le mouvement de mathématisation du monde qui nous entoure est beaucoup plus ancien et remonte aux débuts de l'écriture : les premières traces écrites ne transcrivent pas des phrases, mais des registres économiques, des nombres, accompagnés ou non de pictogrammes [7]. On trouve là déjà une première forme numérique : les entiers, et une première forme d'unités : les pictogrammes d'ovin, de bovin, etc.

Les mesurages sont d'abord des comptages : Pythagore définit le mesurage d'une grandeur comme la comparaison à une unité, ou à une de ses divisions [11], à l'image de l'arpenteur qui établit le côté d'une parcelle en reportant une longueur étalon. La découverte des irrationnels conduit à repenser le mesurage comme une comparaison de rapports de grandeurs. Ainsi, Euclide [11, cit.] axiomatise la *grandeur* à partir de trois opérations fondamentales : la comparaison de deux grandeurs de même nature, la comparaison de rapports de grandeurs, la multiplication d'une grandeur par un entier.

La situation reste, pour l'essentiel, inchangée jusqu'à la fin du 16^{e} siècle, quand Stevin affirme avec force l'unicité de nature des nombres [11, cit]. Prolongeant cette réflexion, Descartes identifie les nombres aux rapports de grandeurs, définit la multiplication scalaire et explique que le choix d'une unité permet d'appliquer le calcul des nombres à celui des grandeurs, sans pour autant assimiler les deux [11].

Aujourd'hui, ces conceptions sont toujours vivaces : les « rods and clocks » d'Einstein sont toujours une abstraction de l'arpentage antique. Pour autant, la métrologie contemporaine intègre d'autres chaînes de mesurage, plus complexes [20], telles que le mesurage d'une température par thermocouple.

Calculs et types La notion de *fonction* nait au 14^{e} siècle, en particulier chez Nicole d'Oresme, comme « qualité » ou grandeur d'un objet, associée à sa représentation graphique comme courbe [11, cit.]. Ceci installe durablement la longueur comme modèle universel de référence et jette les bases de la future analyse. À cette époque, fonctions et grandeurs ne sont pas conceptuellement séparées : une *variable* est simplement une grandeur qui peut varier.

La perspective évolue avec l'apparition des formalismes. Jusqu'à la fin du 16^{e} s., grandeurs et quantités sont désignées par des périphrases [4] : chose, [chose]

3. Hors citation, nous distinguons le mesurage et sa valeur, la mesure [20,3].

4. Diophante d'Alexandrie, au 3^{e} s. EC, utilise déjà une notation symbolique pour désigner une inconnue, mais il s'agit alors seulement d'une abréviation.

inconnue, plus petite grandeur, la première, la deuxième, etc. Les choses changent quand Viète crée le calcul littéral en désignant les grandeurs d'un problème par des lettres, complétées par un type (p. ex. « plano » pour une aire) [21]. Descartes systématise ensuite l'usage des lettres, sans type, et assigne le début de l'alphabet aux grandeurs connues et la fin aux inconnues [11]. Au milieu du 18^{e} s., Euler ajoute la conception de certaines fonctions comme des « expressions » [11, cit.]. Une variable est alors une grandeur qui apparait dans une expression.

L'algèbre contemporaine, qui prend son essor au 19^{e} s., inverse la perspective : les *indéterminées* sont d'abord des entités formelles, qui peuvent s'instancier, ensuite, dans des grandeurs effectives. Les équations elles-mêmes deviennent des objets mathématiques. Le mouvement se prolonge par la formalisation des mathématiques et l'introduction du concept de langage formel, à la fin du 19^{e} siècle et au début du 20^{e} : les lettres des variables deviennent de simples écritures formelles, dont la propriété fondamentale est de pouvoir être substituées par des termes. Les travaux de Dedekind [11] posent un jalon en construisant les nombres eux-mêmes à partir des entiers. C'est une première forme de raisonnement computationnel (au sens de : représentation de données), qui est ensuite considérablement développé par les recherches sur les fondements des mathématiques, et en particulier sur l'*Entscheidungsproblem*, notamment par Turing (machine universelle) et Church (λ-calcul). Ceux-ci aboutissent à la création de l'ordinateur puis de la science informatique. Ce terreau produit également une idée fondamentale qui fait explicitement le lien entre preuves logiques et programmes informatiques : la correspondance de Curry–Howard [13]. L'élimination des coupures d'un côté correspond à l'exécution de l'autre. L'objet central permettant de faire le lien entre les deux est le concept de *type*.

Les langages de programmation se multiplient au cours de l'histoire de l'informatique. Les *variables informatiques* sont une structure fondamentale de la plupart d'entre eux. Elles associent généralement un identificateur (éventuellement muet), un type de donnée (éventuellement contraint) et une valeur (mutable ou non, éventuellement indéfinie). Si l'identificateur rappelle les variables des mathématiques ou de la physique, il n'y a pas d'association fonctionnelle comme en mathématiques ni statistique comme en sciences.

1.3 Le curriculum du cycle 4 français actuel [16]

L'informatique fait l'objet d'un enseignement structuré à partir du cycle 4. Celui-ci introduit la « notion de variable informatique », notamment de « variables d'entrée et de sortie ». Le programme acte la proximité avec les notions de variable et de fonction mathématiques, mais en indiquant que les différences sont de forme et non de nature.

Les grandeurs physiques et géométriques les plus usuelles sont introduites à partir du cycle 2, mais pas la notion de grandeur. Chaque type de grandeur étudié est pratiqué de façon opératoire, en effectuant des comparaisons, des mesurages et des calculs. Épistémologiquement, le choix est fait d'appuyer la connaissance des nombres sur celle des quantités et des grandeurs, et non l'inverse.

Le calcul littéral apparait au cycle 4 et constitue une rupture épistémologique majeure, notamment par son recours à l'abstraction. En mathématiques, il permet d'exprimer les fonctions sous forme de termes formels, introduisant ainsi les variables pour l'algèbre et l'analyse. En sciences, il permet d'introduire les relations formelles entre grandeurs ainsi que les grandeurs-quotients. Pour autant, le lien n'est pas explicitement prévu avec l'informatique : l'enseignement présente les « différents statuts de la lettre (indéterminée, variable, inconnue, paramètre) », mais seulement d'une façon opératoire, par familiarisation.

2 De la variable à la substitution

Nous nous proposons maintenant d'éclairer la question de l'enseignement des variables et notions associées en explorant leur conceptualisation.

2.1 Indéterminée, substitution et calcul littéral

En logique, un terme est une suite de symboles qui se combinent selon des règles précises. Les termes élémentaires sont appelés atomes. L'algèbre des termes repose sur une opération fondamentale : la *substitution*. Par définition, la substitution d'un terme y à un atome a dans x, $x[a \rightsquigarrow y]$, est obtenue en remplaçant toutes les occurrences de a dans x par la suite y, au besoin en ajoutant des groupements comme des parenthèses.

En mathématiques, l'utilisation d'atomes abstraits permet d'introduire les notions de polynôme formel et de fraction rationnelle. Un atome libre utilisé uniquement aux fins d'être substitué est appelé une *indéterminée*. Le plus souvent il s'agit d'une majuscule de la fin de l'alphabet latin : X, $Y\ldots$ On utilise aussi beaucoup le mot « variable », mais de façon impropre puisqu'il n'y a pas là de référence à une grandeur.

En algèbre, cette notion de substitution, extrêmement féconde, est omniprésente, même si souvent implicite. Par exemple, on peut se représenter comme une substitution la multiplication par un scalaire dans un espace vectoriel [5], ou encore le produit de deux espaces vectoriels, $E \otimes F$, comme l'espace vectoriel engendré par les substitutions de $f \in F$ à l'unité dans $e \in E$ ([6]).

Qu'en est-il en pédagogie ? Les programmes évitent la confusion conceptuelle entre indéterminée, au sens d'écriture substituable, et variable, au sens d'écriture désignant une grandeur, mais la réintroduisent en effaçant ces deux notions devant la « lettre », le calcul sur les fractions rationnelles étant appelé « calcul littéral ». D'un point de vue informatique, ceux-ci mettent en avant l'identificateur, plutôt que les notions sous-jacentes. Ceci pose des difficultés pour certains élèves qui rentrent plus difficilement dans l'abstraction et ont du mal à se

5. Par exemple comme la substitution à l'unité des coordonnées dans une base quelconque. P. ex. en dimension 2 : si $v = (6\,\mathrm{u},\ 12\,\mathrm{u})$, la substitution de $3\,\mathrm{u}$ à u donne : $v[\mathrm{u} \rightsquigarrow 3\,\mathrm{u}] = (6.3\,\mathrm{u},\ 12.3\,\mathrm{u}) = (18\,\mathrm{u},\ 36\,\mathrm{u}) = 3v$.

6. En réalité à ses coordonnées dans une base. D'autres applications sont courantes, telles que le polynôme caractéristique d'une matrice carrée M : $\det(\mathbf{1}[\mathrm{u} \rightsquigarrow X] - M)$.

représenter un calcul sur des noms, sans qu'ils soient des noms *de quelque chose*, cf. [8] et ses références. De la même façon, les collégiens doivent effectuer des résolutions d'équations sur des inconnues qui ne sont pas interprétées comme des grandeurs ; là encore, certains peinent à leur donner du sens [8].

2.2 Épistémologie des grandeurs

Or, en sciences, les variables sont des dénominations de grandeurs. Pour montrer l'intérêt pour la didactique de l'informatique de s'attacher au concept de grandeur, décrivons donc rapidement celui-ci.

Pour le Bureau international des poids et mesures, une grandeur est une « propriété d'un phénomène, d'un corps ou d'une substance, que l'on peut exprimer quantitativement sous forme d'un nombre et d'une référence » [3]. La notion antique a été conceptualisée par plusieurs auteurs au cours du 20^{e} s., notamment en termes d'espaces vectoriels [21]. Un autre courant épistémologique, dit opérationnaliste, veut qu'une grandeur soit seulement ce qu'on mesure avec les instruments appropriés [7]. De fait, il est habituel de ne guère faire de différence dans le langage courant entre une grandeur, sa valeur et sa ou ses mesures — on confond ainsi souvent la surface d'une figure, l'aire de cette surface et la mesure de cette aire [21]. C'est la conception qui prévalait dans l'enseignement primaire jusqu'aux années 1970 [12]. Cette approche, occultant les grandeurs, ne donne pas d'accès conceptuel à leur manipulation directe, seulement indirecte par l'intermédiaire des mesures. L'approche didactique actuellement conseillée, en revanche, qui « [aborde] les grandeurs en dehors du nombre, à partir d'activités de comparaison directe, indirecte, avant l'introduction de l'étalon arbitraire est ainsi une proposition qui a suivi le mouvement de la réforme des mathématiques modernes et des activités d'éveil en sciences » [17]. Ceci est, bien sûr, essentiel pour comprendre la nature conventionnelle de l'étalon [19], mais aussi permet d'éviter d'identifier une grandeur à un nombre : le nombre et l'unité, ensemble, *représentent* une grandeur. La dimension statistique de la mesure n'apparait pas au collège, qui nous intéresse ici, mais au lycée, et seulement à partir de la fin des années 1960 [23,17]. Est-ce trop récent, ou insuffisant ? « Pour une part importante des professeurs des écoles stagiaires, une grandeur correspond à quelque chose de flou, de mal déterminé, de peu précis. [...] La comparaison des grandeurs directement, et non la comparaison de leurs mesures, n'est pas envisagée, or c'est elle qui permet de définir la grandeur physique. [...] Malgré leur formation universitaire en sciences au moins jusqu'au niveau L3 [licence], les futurs enseignants de mathématiques et de sciences physiques éprouvent, comme

7. Cf. [20,17]. Ce courant se développe suite aux réflexions de Mach, formalisées par Bridgman. Pour Ullmo (1969), la grandeur « ne préexiste pas à sa mesure, comme une intuition sommaire l'a longtemps fait croire ». La Mécanique quantique entérine effectivement de façon structurelle le fait que tout mesurage perturbe le système observé et donc le fait qu'une grandeur ne puisse être séparée de son mesurage. Pour autant, plusieurs recherches récentes [2] décrivent cette théorie comme étant d'abord une forme de statistiques. Celle-ci ne serait donc qu'un formalisation du mesurage et, par conséquent, impropre à trancher seule la question de l'opérationnalisme.

les enseignants du premier degré, beaucoup de difficultés à définir le concept de « grandeur » [17]. L'approche euclidienne dépasse les simples représentations instrumentales, mais est-elle suffisante pour installer une représentation *conceptuelle* ? Ces observations suggèrent que non.

Élargissons notre vision des grandeurs grâce à la science contemporaine. De Galilée à Newton, la physique classique conçoit le mesurage comme l'observation d'un état du monde, absolu et extérieur, nonobstant la compréhension de la relativité de l'observation. Boltzmann opère un premier changement de perspective en raisonnant sur les grandeurs intensives (température, pression) ; il montre que celles-ci se comprennent moyennant l'existence un nombre seulement fini de degrés de liberté dans une quantité de matière donnée. Ce ne sont plus seulement les mesures qui ont une valeur statistique, mais certaines grandeurs elles-mêmes. Popper amène un second changement de perspective avec la méthode hypothético-déductive ; il montre que l'observation ne peut pas se comprendre seulement comme vérification, mais comme *épreuve* d'une théorie. Les grandeurs ne sont plus alors simplement des observables mesurables, ce sont des variables expérimentales, dont certaines, considérées comme indépendantes, sont contrôlées et les autres, dépendantes, sont observées. Cette approche, largement adoptée par les méthodologies scientifiques contemporaines, adosse le concept de grandeur à celui de *variable expérimentale*. À un niveau scolaire, il serait tout à fait possible d'aborder, de façon simplifiée ce concept, par exemple en envisageant des paramètres ou *variables de modèle*, très proches des *variables d'entrée et de sortie* du programme d'informatique.

2.3 Substitution et épistémologie de la variable informatique

Revenons maintenant à l'informatique. Le concept de variable y est particulièrement vaste. Limitons-nous à quelques aspects fondamentaux [9] :

2) Flux d'information : notion de variable d'entrée, de sortie, de paramètre ;
6) Modularité : la variable peut constituer une forme d'indirection ;
7) Codage : la valeur de la variable est codée en mémoire ;
8) Typage : la variable a un type de données (éventuellement mutable)
9) ... qui définit les opérations qu'elle autorise.

En termes de flux d'information, une variable est un moyen de passer une valeur à l'aide d'un identificateur. Elle fournit deux opérations : la conservation d'une valeur, la substitution de sa valeur à son identificateur. Cette dernière constitue un premier lien entre variables mathématiques et informatiques. Notons que la substitution est également essentielle en physique : Einstein a abouti à sa théorie générale de la relativité en partant du constat que les lois physiques sont invariantes par substitution d'un système de référence à un autre et, en particulier, ne dépendent pas des unités [18,22]. Pour lui, la substitution devrait être un concept central de la physique [8].

8. Dans l'autre grand domaine de la physique fondamentale, la Mécanique quantique, la substitution joue également un rôle historique essentiel : l'opération dite de quantification est la substitution d'opérateurs hermitiens aux grandeurs numériques.

Cette perspective épistémologique, compatible avec l'approche euclidienne, peut-elle jeter quelque lumière sur les enseignements du cycle 4 ?

Prenons l'exemple de l'expression « 200 mL de lait » dans une recette de pâtisserie. Celle-ci peut se comprendre de deux façons : soit comme « (200 mL) de lait », une capacité appliquée à une substance, soit comme « 200 (millilitre·s de lait) », un scalaire, ou une quantité partitive, associée à une unité. Notons qu'il s'agit là d'un changement d'unité, même si le mot « millilitre·s » apparait toujours : dans d'autres cultures ou d'autres temps, on peut employer différentes unités de capacité selon la substance mesurée [9]. Si l'on n'a pas de verre doseur et qu'on utilise une balance, on peut également substituer des grammes de lait. Ceci est un premier jeu de substitution sur les unités. Deuxième type de substitution, déjà en œuvre dans le primaire, (une version simple de) la covariance qu'évoquait Einstein : utiliser une autre unité commensurable, litre, galon, pinte ou pot à yaourt. Enfin, un troisième type de substitution est possible, dès le primaire : si nous avons plus de convives, nous pouvons multiplier les quantités d'ingrédients *du côté des unités*. C'est ce que nous faisons quand nous prenons comme référence deux pots à yaourt plutôt qu'un seul dans la recette du gâteau au yaourt.

Cette dernière opération, complètement intégrée par la plupart des adultes, ne va pas de soi. Elle n'est possible que parce qu'il s'agit d'une grandeur de rapport (au sens des statistiques). Elle a du sens pour une durée, par exemple, mais pas pour une date ni pour une température. Les unités qui correspondent à des grandeurs de rapport se comportent (jusqu'à un certain point) comme des atomes substituables du calcul formel.

Les travaux classiques de Dehaene ont montré que nous disposons d'un « sens du nombre », d'une représentation des dénombrements. L'histoire suggère également que le dénombrement, c'est-à-dire le comptage de choses, précède le nombre abstrait —abstrait *stricto sensu*, c'est-à-dire privé de certaines de ses propriétés. Mathématiquement, même si c'est utile de l'oublier une fois une certaine aisance acquise, il y a toujours une unité présente. Quand nous écrivons « $5 + 8 = 13$ », nous utilisons, en réalité, « $5\,\mathrm{u} + 8\,\mathrm{u} = 13\,\mathrm{u}$ ». En cycle 2 [15], l'élève comprend que « 5 jetons + 8 jetons font 13 jetons » dépasse la matérialité des jetons : l'opération s'applique aussi bien à 5 et 8 boites. Ici aussi, la substitution pourrait être exploitée de façon plus explicite : ces boites peuvent être des dizaines et donc « 5 dizaines + 8 dizaines font 13 dizaines » [10]). Cette substitution, n'est pas un simple truc : c'est une opération fondamentale pour certaines grandeurs (de rapport), opération complexe, qui mérite en soi un apprentissage [11]. Le collégien pourrait ensuite s'appuyer sur cette compréhension pour mieux appréhender ses notions de grandeurs et de variables.

9. Selon Pressiat [21], Lebesgue remarquait déjà que « Toute question qui conduit à une multiplication est un problème de changement d'unité, ou d'objet [...] ».

10. Ce cas précis de substitution est utilisé par plusieurs méthodes d'enseignement de la numération, mais sans que l'opération de substitution soit toujours explicitée, ni exploitée dans sa généralité.

11. L'étudiant pourra retrouver cette complexité au premier cycle universitaire. Une multiplication peut être cardinale, ordinale, scalaire, vectorielle...

3 Pédagogie de l'abstraction

3.1 Nature de grandeur et type computationnel

Nous avons défendu l'intérêt didactique des jeux de substitution. Intéressons-nous maintenant à leurs limites.

Substitution et nature de grandeurs Considérons l'équation « $6\,\mathrm{m} \times 12\,\mathrm{m} = 72\,\mathrm{m}^2$ ». Imaginons de très grands carrés de chocolat ; cette équation se transpose à l'unité « carré·s-de-chocolat ». Par abstraction, il est donc possible de considérer l'unité comme un symbole substituable : « $6m \times 12m = 72m^2$ », soit « $6X \times 12X = 72X^2$ ». En termes logiques, on pourrait dire que cette unité, le mètre, peut être considéré soit comme une grandeur (la valeur 1 m, p. ex. un mètre étalon), soit comme une unité idéale de mesure (le « *rod* » abstrait d'Einstein), soit comme une unité physique (cf. [5] pour plus de finesse), soit simplement comme un terme substituable. Est-ce un simple jeu d'écriture ? Pouvons-nous toujours faire cela ? Si l'on reprend un exemple précédent, qu'est-ce qui donne le droit de généraliser de « 5 jeton·s + 8 jeton·s = 13 jeton·s » à « 5 ce-qu'on-veut + 8 ce-qu'on-veut (le même) = 13 ce-qu'on-veut » ? À un premier niveau de réponse, on le peut parce qu'une analyse du raisonnement nous permet de montrer qu'on n'a pas utilisé de propriétés des jetons pour obtenir la première équation. Mais réfléchissons plus avant ; en réalité, on ne peut pas généraliser complètement : cette équation n'aurait pas de sens pour des dates ou des températures. En seconde analyse, il convient donc d'admettre que l'on a bien utilisé une propriété d'additivité : la grandeur « nombre de jeton·s », qui appartient à la nature de grandeur « cardinal », est une grandeur de rapport.

Épistémologie des natures de grandeur La question de la substitutabilité nous amène donc à considérer celle des natures de grandeurs.

En sciences, une grandeur regroupe essentiellement trois choses : un nombre, une unité de mesure [12] et une nature de grandeur. La définition standard de cette nature est d'être l'« aspect commun à des grandeurs mutuellement comparables » [3]. Nous retrouvons là une composante essentielle des grandeurs déjà soulignée par Euclide [11, cit.]. Notons que la nature est une question d'interprétation ; elle a une valeur ontologique. Par exemple, faut-il considérer que la longueur et la durée sont fondamentalement de même nature ? L'interprétation einsteinienne de la relativité suggère que oui, celle de Bell [1,10] que non. La question de la nature se pose aussi pour les grandeurs sans dimension comme le nombre d'Avogadro ou la constante de Boltzmann, qu'on peut voir, essentiellement, comme des facteurs de proportionnalité. Pour certains physiciens, le fait de se ramener à cette situation constitue même un objectif, ainsi Einstein : « Imagine that this has been realized [elimination of two universal constants to the benefit, for example of the mass and the radius of the electron] ; then there appear in the fundamental equations of physics only dimensionless constants » [5,

12. Ou plus généralement une référence, cf. [3], § 1.1, n. 2.

cit.]. On pourrait ainsi pousser à l'extrême la commodité de calcul qui veut que l'on supprime les unités et la nature de grandeur, ramenant des équations physiques à leur seule composante numérique.

Dans l'enseignement scolaire français, l'usage actuel, qui n'est pas formalisé dans les programmes, est d'escamoter les unités dans les calculs et de les faire réapparaitre dans les résultats. Dans le supérieur ceci peut aller jusqu'à poser certaines constantes comme égales à 1. Il n'y a là aucune difficulté théorique (c'est une application de la substitution) ni pratique (l'écriture est plus concise). Mais, pédagogiquement, « de nombreux débats ont eu lieu et sont toujours d'actualité dans la communauté des didacticiens des mathématiques sur la façon dont on doit écrire les calculs sur les grandeurs [4, p. ex.] » [17].

Nature, type et substitution Adoptons maintenant le point de vue de l'informatique. En termes de spécifications et de démonstrations, la nature des grandeurs spécifie quels traitements sont possibles, elle correspond donc précisément au *type computationnel* d'une variable.

Cette notion de type est associé à un résultat très puissant et pourtant remarqué très tôt dans l'histoire de la discipline : la correspondance preuve–programme, dite de Curry–Howard. Ses nombreux enseignements montrent l'importance du type, dont la formalisation peut aller jusqu'à un outil de preuve d'un programme, d'un côté, et la production d'un algorithme, de l'autre. Les types peuvent être ainsi conçus, notamment, comme des contraintes sur les données (par exemple de domaine de validité) ou, à l'inverse, comme la spécification des opérations que l'on peut effectuer dessus.

Implications didactiques La spécification des types est un outil essentiel de validité des algorithmes en informatique. Ceci suggère l'importance de bien garder trace de la nature d'une grandeur au cours d'un calcul. À titre personnel, nous avons été saisi, en corrigeant un jour une copie de statistiques, de lire qu'un·e étudiant·e avait trouvé la réponse « 3 » comme âge moyen du chef de famille dans un quartier de la ville (exprimé en années). Ce cas est probablement extrême, mais pas isolé. Un type ou une nature ont pour fonction essentielle de prévenir de tels écueils. L'histoire de la physique suggère une conclusion similaire : l'équation aux dimensions, qui est essentiellement une algèbre de natures à gros grains, a parfois eu une grande portée heuristique.

Une première approche scolaire pourrait consister à *rappeler* systématiquement unités ou natures de grandeur au cours d'un calcul. Certes, cela peut s'avérer parfois malcommode, p. ex. avec l'unité carré·s-de-chocolat, mais on pourrait imaginer une formulation iconique. Une autre, plus cohérente, qui se pratique dans l'enseignement scolaire allemand, consiste à *calculer avec les grandeurs* [21]. Par exemple, si ℓ est une variable représentant la longueur d'une tige, on peut écrire $x = 3 \times \ell$ et x désigne alors une grandeur et non une mesure ni un nombre. Dans un tel calcul, on peut alors substituer une valeur à une telle variable, qu'il s'agisse d'une mesure ou d'une valeur idéale : si l vaut 6 cm, x vaut $3 \times 6\,\mathrm{cm} = 18\,\mathrm{cm}$.

Avec cette conceptualisation, c'est l'écriture sans unité « $3 \times 6 = 18\,\text{cm}$ » qui devient formellement incorrecte : un nombre sans unité ne peut être égal à une valeur avec unité. Pour des élèves à l'aise avec l'abstraction numérique, on peut, bien entendu, admettre qu'ils formulent des calculs sans unité, en particulier pour les rendre plus lisibles ; il est important, dans ce cas, que le calcul soit *intégralement* sans unité : $3 \times 6 = 18$ », l'unité n'étant rétablie que dans l'énoncé du résultat, pas dans le calcul (y compris sa dernière étape). Quoi qu'il en soit, un calcul ne devrait jamais se résumer à "pousser du bois" : les éléments qui le composent doivent faire sens à tout moment.

Dans une perspective de didactique de l'informatique, l'écriture des unités dans les calculs, jusqu'au cycle 4 au moins, apparait comme un moyen de développer très tôt chez les élèves une première conscience du type computationnel.

3.2 Abstraction et indirection

Concluons cet article en approfondissant la question de l'abstraction, qui est l'apprentissage le plus radicalement nouveau apporté par le cycle 4.

Les différents processus d'abstraction Toute représentation théorique, en science, en mathématique, comme en informatique, est fondamentalement une *abstraction*, c'est-à-dire le résultat d'un processus visant à isoler une notion par rapport à d'autres. Si l'on suit Colburn et Shute [6], « Mathematics, being primarily concerned with developing inference structures, has information neglect as its abstraction objective. Computer science, being primarily concerned with developing interaction patterns, has information hiding as its abstraction objective »[13]. Nous pourrions ajouter que les sciences expérimentales ont le contrôle du contexte par un protocole comme objectif d'abstraction. Ces trois approches heuristiques ne peuvent que peser sur la représentation savante des grandeurs et des variables.

Du point de vue de ce processus d'abstraction, une variable, pour l'essentiel, « hides the details of how a symbolic token can serve as a location for changing data values in memory » [6]. Ceci nous conduit donc à examiner le statut épistémologique des variables informatiques sous l'angle du concept majeur d'indirection.

Indirection et pédagogie L'indirection est certainement une des opérations les plus profondes utilisées en informatique, en témoigne l'aphorisme de David

13. Nous ne suivrons pas ces auteurs quand ils considèrent ces approches comme systématiques : il y a, en mathématiques, du masquage d'information dans la pratique de l'utilisation de théorèmes, en particulier de construction ; à l'inverse, en informatique il est parfois nécessaire de négliger de l'information, par exemple dans certains choix d'indicateurs ou de représentation de données (résolution, compression avec pertes...). De la même façon, il peut y avoir une démarche de contrôle de contexte en mathématiques (la thèse classique de Lakatos [14] en atteste abondamment) comme en informatique (particulièrement en automatique ou en intelligence artificielle).

Wheeler, co-inventeur du sous-programme : « all problems in Computer Science can be solved by another level of indirection ». Si l'indirection est une notion d'informatique, elle est, en revanche, d'un usage très général en tant qu'opération intellectuelle. Il conviendrait, selon nous, de l'intégrer au socle fondamental.

Zilles [25], qui examine la manière d'enseigner la notion d'indirection en informatique, est gêné par ses définitions habituelles : « indirection is the ability to reference something using a name, reference, or container instead the value itself » est trop abstraite, quand les définitions concrètes (p. ex. « manipuler une donnée *via* son adresse ») sont trop étriquées. Nous proposons d'envisager l'indirection sous un autre angle. Pour nous, une indirection est le fait de désigner quelque chose, par exemple une grandeur, par un *signe arbitraire* qui peut fonctionnellement la remplacer et à quoi, pour exécuter un calcul, on *substitue* une valeur qui lui est attachée. L'indirection est donc une généralisation de l'arbitraire du signe linguistique, tel qu'il apparait, notamment, dans l'écriture alphabétique [14]. Comme pour le signe linguistique, l'indirection permet une forme de virtualisation : mutabilité de la valeur au cours du temps et démultiplication (plusieurs valeurs traitées parallèlement ou au choix), généralisation (subsomption, factorisation de plusieurs cas sous une même référence générale). En pratique, l'indirection permet aussi la non-référentialité (il n'est pas nécessaire que la référence, par exemple la variable, ait effectivement une valeur pour effectuer certains raisonnements ou certains calculs [15]), ainsi que les définitions récursives (`fact(n):=n*fact(n-1)`). L'indirection est donc une forme d'abstraction, de ce point de vue également. Par ailleurs, [24] note que l'ajout d'une couche d'indirection permet la flexibilité et une meilleure organisation — par exemple, l'accès systématique aux objets via des pointeurs permet le ramasse-miette (*garbage collection*). Ajoutons, enfin, que l'indirection peut présenter un intérêt cognitif : le nom, puisqu'il est arbitraire, permet d'expliciter une intention du concepteur.

Dans le cas des variables, il pourrait être intéressant, pédagogiquement, d'expliciter l'indirection qu'elles mettent en œuvre. En informatique : mutabilité, généralisation et choix judicieux du nom. En physique : mutabilité, généralisation et standardisation du nom. En mathématiques : généralisation, abstraction et non-référentialité. Ceci permettrait de ne pas laisser sous silence la forme d'abstraction mise en œuvre dans chaque discipline : négliger l'information est une dimension essentielle de l'épistémologie des mathématiques, la masquer est fondamental dans la compréhension de l'encapsulation et, plus généralement, de la modularisation en informatique, et le contrôle du contexte est une dimension essentielle de ce qui fait la méthode expérimentale en sciences. Incidemment, cela permettra de clarifier un point tout à fait problématique pour nombre d'élèves, et même encore d'étudiants : la polysémie du signe « = », instruction d'affectation dans nombre de langages de programmation (« `largeur=15;` »), relation dans

14. Pour autant, la sémiotisation, le recours à un signe, n'est pas toujours arbitraire, donc ne fait pas toujours recours à l'indirection, par exemple dans le cas des écritures idéographiques et syllabiques.

15. Pensons aux noms de domaines Internet, par exemple, ou à l'utilisation des URI par le web sémantique.

les équations (« $x^2+2x=4$ »), fait observé ou anticipé en contexte expérimental (« $\ell = 3\,\mathrm{cm}$ »).

4 Annexe : recommandations curriculaires pour le cycle 4

Cette annexe rassemble nos principales suggestions et recommandations, qu'il conviendrait de tester sur le terrain, ainsi que des définitions susceptibles de servir de guide aux enseignants.

4.1 Un *caractère* est une propriété d'un phénomène, d'un corps ou d'une substance, en principe observable. Une *grandeur* est un caractère « que l'on peut exprimer quantitativement sous forme d'un nombre et d'une référence » [3]. Pour les grandeurs ordinales, la référence est une échelle (p. ex. l'échelle sismologique de Richter) ; pour les grandeurs quantitatives, la référence est une *unité*. Le nombre et son unité constituent, ensemble, sa *valeur* ; il peut s'agir d'une *mesure*, résultat d'un *mesurage* ou plus généralement d'une observation, ou d'une valeur idéale, supposée ou hypothétique. Des « grandeurs mutuellement comparables » sont dites de même nature [3] ou de même *type*. L'opération de comparaison est la base de la compréhension du type d'une grandeur, indépendamment de tout mesurage. L'unité est un indice de ce type. Une *quantité* est une grandeur de type « cardinal ».

4.2 Une *variable* est un signe arbitraire, son *nom* ou identificateur, associé à une donnée mutable, sa *valeur*, et à un *type* computationnel définissant les valeurs possibles. Le mot valeur s'entend au sens large, incluant le sens précédent (4.1) ; il ne s'agit pas seulement de nombres, mais de n'importe quels types d'objets ou de valeurs de grandeurs. L'exécution d'un calcul sur une variable *substitue* sa valeur à son identificateur. Son type définit les calculs applicables à une variable.

4.3 Nous recommandons d'adopter une approche transdisciplinaire unifiée en appliquant les définitions 4.1 et 4.2 aussi bien en informatique qu'en sciences ou qu'en mathématiques. Ceci permet de présenter les variables comme un moyen *abstrait* de raisonner en termes de flux de données (d'information) : variable d'entrée ou de sortie, variable (observable) expérimentale, variable (paramètre) d'une modélisation. Une variable représentant une grandeur, par indirection, peut donc signifier aussi bien sa valeur, une ou plusieurs mesures ou encore une valeur hypothétique. Cette définition est compatible avec une transposition scolaire des variables statistiques et aléatoires au lycée. Notre analyse suggère, en revanche, de séparer résolument les variables des *indéterminées* et des *inconnues*.

4.4 Quand une variable est quantitative, elle *représente* une grandeur et son type computationnel peut être assimilé à la nature de cette grandeur. Nous recommandons que cette notion soit explicitée dans l'enseignement scolaire en utilisant le mot « type ». Il est de bonne pratique en informatique, en mathématiques comme en sciences qu'un identificateur soit « parlant » : le nom d'une variable devrait toujours évoquer le type de celle-ci ou de la grandeur qu'elle représente (hauteur h, vitesse v, liste d'éléments `elements`...). En particulier, un nom n'est pas nécessairement réduit à une lettre.

4.5 Une variable ou une grandeur *a* une valeur, mais elle n'*est* pas cette valeur. Nous recommandons que les grandeurs elles-mêmes puissent faire l'objet de calculs. Si ℓ est une variable représentant la longueur d'une tige, on peut donc écrire $x = 3 \times \ell$, le résultat étant une grandeur. Ce choix a deux conséquences. D'une part, on peut alors *substituer* une valeur à cette variable, qu'il s'agisse d'une mesure ou d'une valeur idéale : si ℓ vaut 6 cm, $x = 3 \times 6\,\text{cm} = 18\,\text{cm}$. D'autre part, il convient de conserver l'indication de l'unité au cours des calculs. Une explication de calcul pourrait même aller à rappeler les types, au-delà des seules unités. Bien sûr, on pourrait admettre qu'un élève à l'aise avec l'abstraction numérique des grandeurs puisse s'épargner l'écriture de ces unités, en particulier pour rendre plus lisibles certains calculs, mais, dans ce cas, ce choix doit être homogène tout au long d'un calcul.

4.6 L'abstraction du raisonnement est un des apprentissages majeurs du collège, dans la plupart des disciplines. Son abord est difficile pour de nombreux élèves. Il pourrait être intéressant pédagogiquement d'expliciter une partie du processus d'abstraction, afin d'en favoriser l'appropriation. Dans le cas des variables, en s'appuyant sur notre analyse de l'indirection, il pourrait être intéressant de montrer des exemples des mécanismes suivants : généralisation ; mutabilité, en informatique et en sciences ; définition par récurrence, en informatique et en mathématiques. En sciences, les grandeurs peuvent être vues comme les paramètres d'une situation expérimentale qu'il s'agit soit de contrôler soit d'observer. On peut ainsi faire le lien avec les variables d'entrée et de sortie d'un programme ou d'un modèle (physique ou informatique).

4.7 Nous suggérons d'expérimenter l'usage de la substitution pour aider à la compréhension de la nature conventionnelle de l'étalon, des conversions d'unités et des calculs de proportionnalité, par exemple sur le modèle de l'apprentissage de la numération au cycle 2, mais d'une façon qui serait explicite et plus adaptée au cycle 4.

Références

1. Bell, J.S. : How to teach special relativity. Progress in Scientific Culture **1**(2), 1–13 (1976), reprint in Speakable and unspeakable in quantum mechanics, Cambr. Un. Pr. (1987), p. 67–80.
2. Benavoli, A., Facchini, A., Zaffalon, M. : Quantum mechanics : The Bayesian theory generalized to the space of hermitian matrices. Phys. Rev. A **94**, 042106 (Oct 2016). https://doi.org/10.1103/PhysRevA.94.042106
3. BIPM (ed.) : Vocabulaire international de métrologie – Concepts fondamentaux et généraux et termes associés (VIM). JCGM, 3 edn. (2012), éd. 2008 corr.
4. Chevallard, Y., Bosch, M. : Les grandeurs en mathématiques au collège. Partie I : une Atlantide oubliée. Petit X pp. 5–32 (2001), http ://yves.chevallard.free.fr/spip/spip/IMG/pdf/Les%5Fgrandeurs%5Fau%5Fcollege%5FI.pdf
5. Cohen-Tannoudji, G. : Lambda, the fifth foundational constant considered by Einstein. Metrologia **55**(4), 486–498 (2018)
6. Colburn, T., Shute, G. : Abstraction in computer science. Minds & Machines **17**, 169–184 (2007). https://doi.org/10.1007/s11023-007-9061-7

7. Cook, B.F. : Le cunéiforme. In : Bonfante, L., Chadwick, J., Cook, B.F., Davies, W.V., Healey, J.F., Hooker, J.T., Walker, C.B.F. (eds.) La naissance des écritures, pp. 24–99. Seuil (1994)
8. Coppé, S., Grugeon, B. : Le calcul littéral au collège. Quelle articulation entre sens et technique ? In : Colloque de la CORFEM (2009), https ://halshs.archives-ouvertes.fr/halshs-00959612
9. Delmas-Rigoutsos, Y. : Proposition de structuration historique des concepts de la pensée informatique fondamentale. In : Parriaux, G., Pellet, J.P., Baron, G.L., Bruillard, É., Komis, V. (eds.) Didapro 7 – DidaSTIC. De 0 à 1 ou l'heure de l'informatique à l'école. Peter Lang, Bern, Lausanne (2018)
10. Delmas-Rigoutsos, Y. : Quantum modalities, interpretation of Quantum Mechanics and Special Relativity, and an experimental test for multiple worlds. soumis (2020), https ://hal.archives-ouvertes.fr/hal-01983757
11. Dhombres, J., Dahan-Dalmedico, A., Bkouke, R., Houzel, C., Guillemot, M. : Mathématiques au fil des âges. Gauthier-Villars (1987), recueil de textes commentés.
12. Favrat, J.F., Munier, V. : évolution de l'enseignement des grandeurs à l'école élémentaire depuis 1945. Repères IREM **68**, 51–75 (2007)
13. Girard, J.Y., Taylor, P., Lafont, Y. : Proofs and types. No. 7 in Cambridge tracts in theoretical computer science, Cambridge univ. pr. (2003 (1989)), http ://www.paultaylor.eu/stable/prot.pdf, web reprint
14. Lakatos, I. : Preuves et réfutations. Hermann (1984 (1976))
15. MEN : Programme du cycle 2. Éduscol (2018), arrêté du 17-07-2018
16. MEN : Programme du cycle 4. Éduscol (2018), arrêté du 17-07-2018
17. Munier, V., Passelaigue, D. : Réflexions sur l'articulation entre didactique et épistémologie dans le domaine des grandeurs et mesures dans l'enseignement primaire et secondaire. Tréma **38** (2012)
18. Norton, J.D. : General covariance and the foundations of general relativity : eight decades of dispute. Reports on Progress in Physics **56**(7), 791–858 (jul 1993). https://doi.org/10.1088/0034-4885/56/7/001
19. Passelaigue, D. : Grandeurs et mesures à l'école élémentaire. Des activités de comparaison à la construction des concepts : le cas de la masse en CE1. Ph.D. thesis, Université Montpellier 2 (2011)
20. Perdijon, J. : La mesure. No. 186 in coll. Dominos, Flammarion (1998)
21. Pressiat, A. : La place des grandeurs dans la construction des mathématiques. Bulletin de l'APMEP (483), 467–490 (2009)
22. Rovelli, C. : Quantum spacetime : what do we know. In : Callender, C., Huggett, N. (eds.) Physics meets philosophy at the Planck scale. Cambridge Univ. Pr. (2001)
23. Séré, M.G. : La mesure dans l'enseignement des sciences physiques : évolution au cours du temps. Aster **47**, 25–42 (2008)
24. Spinellis, D. : Beautiful code : leading programmers explain how they think, vol. 17, chap. Another level of indirection, pp. 279–291. O'Reilly (2007), http ://www.academia.edu/download/30699875/Spi07g.pdf
25. Zilles, C. : "What does a CPU have in common with a fast food restaurant ?" A reflection on emphasizing the big ideas of computer science in a computer organization class. In : Proceedings Frontiers in Education 35th Annual Conference. pp. S3C–11 (Oct 2005). https://doi.org/10.1109/FIE.2005.1612263

Initier des jeunes élèves à la robotique/informatique : gestes professionnels et agir enseignant

Katell Bellegarde[1] et Julie Boyaval [2]

[1] CIREL (EA 4354), Université de Lille, France
[2] Circonscription de Carvin, Académie de Lille, France

Abstract. Cette contribution rend compte d'une étude comparative des médiations cognitives à l'œuvre dans un dispositif pédagogique visant à initier des élèves de grande section de maternelle[1] à la robotique/informatique en utilisant le dispositif *Blue Bot* dans le cadre de jeux sérieux déclinés sur supports corporel, robotique et tablette numérique. Cette étude éclaire la manière dont les instruments-médiateurs influent sur l'agir enseignant interrogés à partir de l'identification, chez les praticiens, de leurs gestes, postures et conceptions professionnels.

Keywords : robotique pédagogique, médiation cognitive, analyse de l'activité, grande section de maternelle, gestes et postures professionnels.

1 Introduction

Le jeu sérieux numérique constitue une ressource pédagogique attrayante et sans doute intéressante pour assurer des enseignements à l'école maternelle (Alvarez et al., 2016). Toutefois, l'exposition d'enfants de premier cycle à des écrans dans l'enceinte scolaire vient se rajouter à celle domestique. Face au temps de consommation écran des élèves et à ses dangers sur le développement de plusieurs capacités, l'usage du jeu sérieux numérique à l'école est à questionner (Tisseron, 2013). Pour bénéficier du potentiel du jeu sérieux sans pour autant arriver à une surexposition écran, l'usage de la robotique apparaît alors pertinent auprès de jeunes élèves.

Depuis une quarantaine d'années, un courant éducatif nommé « robotique pédagogique » a ainsi fait l'objet d'applications intéressantes dans le champ de l'enseignement. A destination de différents types de public, de l'école maternelle à la formation des adultes, le dispositif robotique est associé à un langage de programmation, dans un objectif d'initiation à la robotique/informatique. Toutefois, le jeu sérieux sur terminaux numériques offre des avantages en termes de coût (souvent gratuit à l'instar du l'instar du jeu en ligne Blue Bot) et de diffusion en contexte scolaire. Du point de vue de la recherche, il convenait donc d'étudier si les changements de supports pédagogiques influaient, d'une part, sur la réception et les apprentissages des élèves (Bellegarde, Boyaval, Alvarez, 2019), d'autre part, sur l'agir enseignant.

Le projet de recherche Blue Bot que nous avons réalisé et dont nous présentons une partie des résultats ici porte sur un dispositif pédagogique visant à initier des élèves de Grande Section de Maternelle (GSM) à la robotique/informatique en utilisant le dispositif Blue Bot dans le cadre de jeux sérieux déclinés sur supports corporel, robo-

1 Ces élèves sont âgés de 5 ou 6 ans.

tique et tablette numérique. Cette communication s'intéresse particulièrement à comprendre l'influence des instruments-médiateurs sur l'agir enseignant interrogés à partir de l'identification, chez les praticiens ayant participé à l'expérimentation Blue Bot, de leurs gestes, postures et conceptions professionnels.

2 Enseigner la programmation à des jeunes élèves de cycle 1

Cette première partie rend compte des investigations théoriques menées en vue d'approcher la question de l'enseignement de la programmation à des élèves de GSM. Nous examinerons les trois considérations scientifiques suivantes : (1) La robotique pédagogique favorise une appréhension précoce des concepts de programmation chez les jeunes enfants ; (2) programmer en GSM questionne la place et le statut à donner aux sciences informatiques à l'école, en tant qu'outil d'enseignement et/ou objet d'enseignement à part entière ; (3) Apprendre avec des robots ne doit pas occulter l'action de l'agents médiateur humain.

2.1 La robotique pédagogique : vers une appréhension précoce des concepts de programmation

Le courant éducatif de la robotique pédagogique a été initié par Papert (1981) dans les années 80 avec la tortue de plancher, Turtle, associée au langage de programmation, Logo . Ce dipositif s'inscrit dans le modèle socioconstructiviste des apprentissages. En tant qu'« objets pour penser avec » (Papert, 1981), les robots permettent une manipulation et une expérimentation à partir de situations réelles, dans un contexte de résolutions de problème et de développement de la pensée algorithmique : « l'élève est l'artisan de sa propre formation en se posant lui-même ses problèmes. […] Un système comme LOGO est un des moyens de donner à l'élève un comportement actif dans un processus d'acquisition de concept ou de prise de conscience d'un mécanisme » (Vivet, cité par Denis, 2000, p 196). Considérés comme « les bâtisseurs de leurs savoirs » , de leurs propres structures intellectuelles, les enfants sont également « épistémologues » (Papert, 1981) dans le sens où ils seraient amenés à entrer dans une étude critique de leur propre réflexion : La robotique pédagogique contribuerait ainsi à débarrasser la notion d'erreur du sentiment de sanction intellectuelle : la recherche du bug du programme, son analyse, sa compréhension et sa correction font partie intégrante de l'activité de programmation, constitue une étape du processus d'apprentissage.

Le robot programmable Blue bot (TTS group, 2016), que nous avons choisi pour les expérimentations que nous allons présenter dans le cadre de cette communication, est l'héritier du langage logo et s'inscrit dans cette philosophie de l'éducation. Ses aspects ludiques et interfaces tangibles favorisent par ailleurs une appréhension précoce des concepts de robotique et de programmation, la motivation des élèves et leur implication dans les activités porteuses de significations (Komis, Misirli, 2013).

En définitive ces jouets programmables constituent de véritables outils de médiation, qui permettent aux enfants de s'y identifier par « effet miroir ». Linard (1996)

nous met en garde contre le mythe de l'autogenèse cognitive qui consisterait à négliger le rôle de la médiation humaine vis-à-vis des médiations techniques. Or, le soutien humain sera toujours nécessaire pour relayer l'information médiatisée et ainsi aider les élèves à faire, à penser, à comprendre, à réfléchir sur leurs actions et finalement, les aider à apprendre (Leroux, 2003, Vivet, 2000). Bucheton et Soulé relèvent ainsi l'intérêt à porter à la grammaire de l'agir enseignant dans l'optique de penser la formation des enseignants.

2.2 Les sciences de l'informatique à l'école maternelle : un objet d'enseignement au service d'autres apprentissages

A l'école maternelle, il n'existe pas de compétences à proprement dits qui visent la programmation et le codage informatique. En effet, la seule référence du programme de l'école maternelle en France en lien avec la programmation et le codage informatique concerne l'identification du principe d'un algorithme et la poursuite de son application (BO spécial n°2 du 26 mars 2015). Mais, il s'agit ici de compléter une suite à partir de critères préalablement identifiés. Cette simple référence à l'algorithme peut néanmoins justifier l'initiation à la robotique/informatique en classe de GSM. Dans cette perspective, initier des élèves à la programmation apparaît cohérent avec l'introduction, en 2016, de la science informatique dans les programmes de L'Education Nationale à l'école élémentaire et au collège et avec le nouveau socle commun de compétences (BO 2015). Et pourtant, si l'on considère l'école maternelle comme un socle sur lequel va se construire l'ensemble des apprentissages de la vie de l'élève, on sent bien la nécessité d'amorcer dès le cycle 1 certains savoirs, qui se concrétiseront par un apprentissage « systématisé » quelques années plus tard, notamment en cycle 3 et 4. La logique de continuité et fluidité des parcours de l'élève tout au long de sa scolarité donne ainsi tout son sens à une initiation précoce aux concepts de robotique et de programmation.

Un débat persiste cependant autour de l'apprentissage de l'informatique à l'école : s'agit-il d'un outil d'enseignement au service d'autres disciplines ou d'un objet d'enseignement à part entière ? S'agit-il d'apprendre à programmer ou de programmer pour apprendre ? Nous adopterons dans cette communication un positionnement intermédiaire qui envisage la programmation comme un objet d'enseignement, qui favoriserait par ailleurs l'acquisition de certains apprentissages premiers, langagiers et culturels, qui dépassent les simples notions algorithmiques : la communication orale, la construction du nombre, la structuration de l'espace et du temps, la résolution de problèmes, la collaboration ou encore l'abstraction (Greff, 2004).

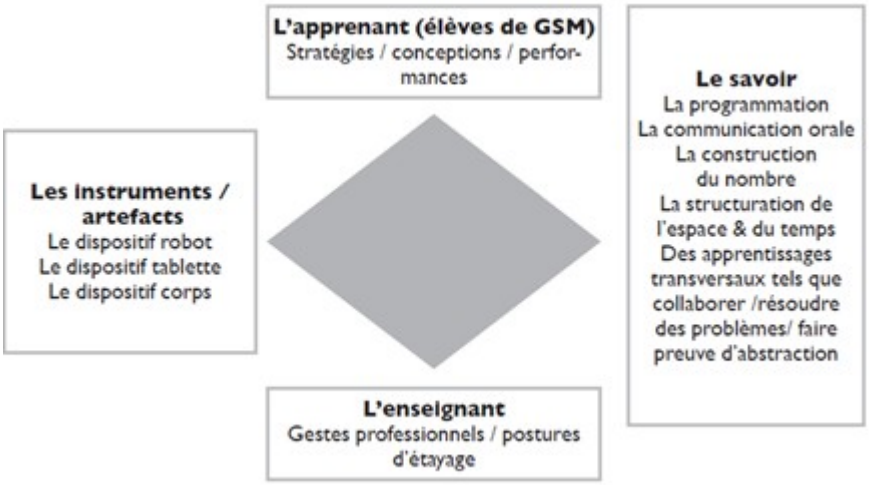
2.3 La médiation humaines au service de la robotique pédagogique : la grammaire enseignante en questions

S'approprier un savoir suppose toujours un processus d'objectivation qui n'est possible que par l'action d'un système médiateur qui va jouer le rôle d'intermédiaire entre le sujet et le savoir. Composé d'aspects très hétérogènes, ce système médiateur comprend des personnes ayant des statuts différents (médiateur–apprenant) et des ar-

tefacts culturels (Weil-Barais & Resta-Schweitzer, 2008). La fonction de médiation de l'enseignant renvoie à celle d'étayage, à la manière dont un adulte, plus expert, organise le monde pour l'enfant dans l'optique d'assurer la réussite de ses apprentissages (Bruner, 1983).

La recherche que nous présentons ici implique l'utilisation d'artefacts culturels de type robotique, numérique et corporel qui suppose une activité de didactisation exercée par l'enseignant à travers ces artefacts. Dans cette perspective, nous envisageons la médiation comme un processus résultant soit de l'action directe d'une personne, soit de son action indirecte exercée par le biais des instruments-médiateurs. À l'instar de Rézeau (2002), nous proposons de penser le système médiateur dans le projet de recherche Blue Bot (cf Schéma 1) sous la forme d'un carré pédagogique qui met en évidence le rôle d'agents-médiateurs joué à la fois par l'instrument et l'enseignant.

Schéma 1. Le système médiateur dans le projet Blue Bot



Cette communication focalise son attention sur trois pôles du carré, l'instrument-médiateur, l'enseignant et le savoir. Il s'agit ainsi de comprendre l'influence du numérique et de la robotique sur l'agir enseignant, agir analysé dans le cadre d'une initiation à la programmation.

Bucheton et Soulé (2009) proposent d'appréhender la grammaire de l'agir enseignant comme *« l'organisation modulaire, systémique et dynamique des gestes professionnels (le jeu des postures d'étayage des maîtres) »* (p. 45). A leur instar, nous appréhendons l'agir enseignant sous le prisme d'un ajustement permanent des gestes et postures en fonction des situations. Cet ajustement est un principe fondateur du geste professionnel

3. Présentation de l'enquête de terrain

Cette communication propose de comprendre l'influence des instruments médiateurs, robotique et numérique, sur l'agir enseignant. Au démarrage de cette recherche, nous avons fait l'hypothèse que la robotique, par sa dimension tangible et extracorporelle et les possibilités de décentration et de manipulation de l'objet-robot qu'elle procure à l'élève, conduirait à une mise en retrait de l'enseignant au profit d'apprentissages autodirigés chez les élèves. Nous étudierons l'agir enseignant sous le prisme des gestes, postures et conceptions professionnels construits à travers la conduite de séances d'initiation à la programmation.

Les informations exposées ci-dessous entendent procurer au lecteur les éléments de repérage sur la manière dont a été mise en œuvre l'enquête de terrain.

3.1 Dispositif expérimental mis en œuvre

Trente-cinq classes de GSM et vingt-huit écoles du Nord et du Pas-de-Calais ont participé à l'expérimentation Blue Bot à travers la mise en œuvre du scénario pédagogique construit dans le cadre de cette recherche. Elaboré selon une progression en trois temps, ce scénario a été pensé de manière à être transposable aux trois supports (robot , tablette , corps) : (1) introduction à l'algorithmique et aux instructions de programmation (prise en main des supports pédagogiques et des fonctions des commandes), (2) introduction progressive des commandes (« avancer », « tourner à droite », « tourner à gauche ») avec création d'une séquence de codage, (3) création d'une séquence de codage avec contraintes supplémentaires (« passer par une ou plusieurs cases » « éviter une ou plusieurs obstacles », « répondre aux deux contraintes simultanément»).

Les situations-problème proposées aux élèves sont construites autour de l'histoire Vibot – le robot (Romero, 2016), personnage que l'enfant doit programmer pour le conduire à différents lieux. Les activités de programmation se réalisent sur un damier de 24 cases (6 x 4) avec une barre et des cartes de programmation ; l'élève conserve alors la trace du programme et un travail sur l'erreur est possible. Les élèves réalisent les activités par groupe de quatre maximums sur un temps d'environ trente minutes.

Afin de mesurer l'influence des instruments-médiateurs (robot, tablette, corps) sur l'agir enseignant, l'expérimentation s'est réalisée de façon à ce que les enseignants mettent en œuvre le scénario pédagogique en expérimentant les supports dans des ordres différents[2].

2 Lors de la citation de verbatim, cet ordre sera indiqué à partir d'un nombre adossé au nom du support correspondant au rang (1/2/3) occupé par le support dans la pratique d'enseignement. Par exemple « robot_1 » signifie que l'enseignant à eu le support robot comme premier support pour initier ses élèves à la programmation.

3.2 Modalités d'élaboration du corpus : l'analyse de l'activité enseignante au cœur de l'investigation

Cette étude s'inscrit dans une approche compréhensive de la grammaire de l'agir enseignant dans le cadre de séances d'initiation à la programmation. L'analyse de l'activité enseignante a constitué le cœur de la démarche d'investigation mise en œuvre. Trois modalités d'élaboration du corpus ont été retenues :

(1) des captations vidéo des situations éducatives ont été réalisées de manière à déconstruire-reconstruire la dynamique de la situation d'enseignement et à identifier les invariants significatifs de l'activité enseignante. Sept classes ont été filmées.

(2) des documents de suivi ont été mis à la disposition des enseignants dans l'optique de saisir leur regard porté sur leur activité et celle des élèves. Dix-sept évaluations intermédiaires et quatorze carnets de bords ont ainsi été recueillis.

(3) dans cette même visée, des entretiens semi-directifs ont été réalisés avec sept enseignants après l'expérimentation de chacun des supports dans leur classe.

4. Enseigner la programmation à des élèves de grande section de maternelle : des gestes, postures et conception professionnels en (trans)formation

L'analyse de l'activité enseignante met en évidence la grammaire de l'agir des professionnels confrontés à l'enseignement de la programmation en classe de GSM, à travers des gestes, postures et conception professionnels qui se (trans)forment au fil de l'expérimentation. Elle souligne également l'influence des instruments-médiateurs sur ces configurations enseignantes.

4.1 Des gestes professionnels de soutien aux activités de programmation des élèves :

Six gestes professionnels majeurs ont été observés chez les enseignants dans le cadre d'une initiation à la robotique en classe de GSM. Ceux-ci sont orientés vers le soutien des activités de programmation des élèves.

Le rappel

Le rappel est le premier geste professionnel observé chez les enseignants à travers la verbalisation des consignes, règles et principes propres à l'activité de programmation. Ce geste professionnel apparaît également dans l'agir enseignant au travers du rappel de ce qui a été fait lors des séances précédentes. Dans ce cas, on se situe dans une action de tissage (Bucheton, 2009) ; en début de séance, l'enseignant met en relation la tâche en cours avec celle qui précède.

A travers le rappel, il s'agit pour l'enseignant de « dire » au sens de rendre explicite pour les élèves ce qu'ils doivent faire et comment. Il s'agit également de rendre disponible pour les élèves les connaissances antérieures qui devront être mobilisées dans l'activité.

« J'ai eu des actions de rappel, je commence toujours par un rappel de ce qui a été fait les fois précédentes. […] J'étais vraiment là pour donner des consignes. » (Entretien enseignante M)

La reformulation

Le geste de reformulation observé chez les enseignants prend trois formes : (1) celle d'une reformulation des consignes, des règles et principes de la programmation. (2) celle d'un modèle verbal suscité par l'enseignant ; (3) celle de la reprise des propos d'un élève pour les mettre en relief et ceci, notamment pour faire avancer le raisonnement de l'ensemble du groupe.

Dans le second cas, il peut s'agir d'insister sur les connecteurs de temps pour travailler la chronologie des actions en utilisant un vocabulaire précis.

Ce qui est alors visé par l'enseignant à travers ce second geste professionnel, c'est de mettre l'accent sur des éléments de la tâche pour les élèves, jugés importants pour comprendre, réfléchir, réussir et surtout apprendre.

« Reformulation du parcours en insistant sur les connecteurs de temps : d'abord, après, enfin. » (Evaluation intermédiaire H, robot 3)

Le questionnement

Le questionnement est un geste professionnel tout particulièrement convoqué par les enseignants. Quatre visées principales de ce geste ont été observées : (1) solliciter chez les élèves la verbalisation de leurs stratégies, les amener à communiquer avec leurs camarades sur leurs manières de faire et de penser ; (2) les inciter à convoquer des savoirs scolaires, par exemple, utiliser les notions de « droite » et de « gauche » pour donner les instructions quand il est plus facile d'indiquer la direction avec la main ; (3) les inviter à utiliser un procédé particulier, notamment en leur demandant le chemin qu'ils ont choisi pour les encourager à le montrer avec le doigt avant de coder le programme.

Par ce geste professionnel du questionnement, l'enseignant va guider au sens de chercher à ce que les élèves rendent explicite ce qu'ils font, leur raisonnement mais également à ce qu'ils utilisent des savoirs et savoir-faire scolaires et enfin à ce qu'ils réfléchissent à ce qu'ils font pour accompagner la réussite de l'activité.

« Le maître doit encourager les élèves à parler, les questionner pour les amener à expliquer ce qu'ils font et comment ils font. Il faut aussi veiller à interroger les élèves qui font sans parler. » (Carnet de bord enseignant H-B, tablette1)

« Je leur demande de changer le message de celui qui guide sans changer les picto[grammes] pour les amener à utiliser le nombre. » (Eval. Intermédiaire enseignant C, Corps2)

L'aide dans l'activité

Bienveillants et soucieux de la réussite de leurs élèves, les enseignants ont également développé un certain nombre de gestes afin d'aider leurs élèves dans la réalisation de l'activité : (1) la simplification de la tâche (retrait d'une contrainte, d'un dé-

placement etc… ou ajout d'un outil supplémentaire[3]) dans l'optique de permettre sa réussite ; (2) la prise en charge d'une partie de la tâche (« le faire avec » : « je commence et tu finis) ou (3) sa totalité (« le faire à la place de ») à travers le geste du « contre-étayage », l'enseignant pour avancer plus vite, si la nécessité s'impose, pouvant aller jusqu'à faire à la place de l'élève ou donner la réponse.

Tous ces gestes permettent à l'enseignant de guider au sens de rendre possible pour les élèves la réussite de l'activité, ne pas les mettre en situation d'échec et redonner à l'erreur un statut positif.

« Les cartes ''tourne à gauche'' ont été enrichies d'une gommette colorée afin de rappeler le chouchou porté par les élèves au poignet gauche [lors des activités préliminaires] et faciliter ainsi le transfert. » (Carnet de Bord enseignant B., corps1)

« Aidée par l'enseignant pour la verbalisation de chaque déplacement » (Eval. Intermédiaire enseignant SO, tablette1)

Le maintien dans l'activité

Un autre geste particulièrement observé chez les enseignants est celui du maintien dans l'activité des élèves. Ces praticiens encourageaient les élèves, valorisaient leurs réussites, régulaient le travail collectif à travers la distribution des tâches, le fait de veiller à ce que chacun joue son rôle dans l'activité, par exemple, ou, encore, proposaient une différenciation pédagogique en fonction du niveau des élèves (simplification ou complexification de la tâche).

Derrière tous ces gestes de maintien dans l'activité, l'objectif est celui de conserver et renouveler la motivation des apprenants.

« C'est aussi le rôle de l'enseignant de dire ''c'est bien, tu as bien travaillé aujourd'hui. C'est un peu difficile, on reprendra demain''. […] A chaque fois c'était une évaluation positive. Je disais ''tu t'es trompé mais vas-y corrige toi, tu vas y arriver''. Donc, J'ai essayé d'être bienveillante. […] Le fait de les encourager, c'est vraiment important. » (Entretien enseignant W, robot1)

La mise en retrait

Le dernier geste professionnel observé est celui de la mise en retrait volontaire de l'enseignant. Ce dernier se met alors en position d'observateur et invite les élèves à collaborer, à co-construire la solution au problème de programmation posé, ce travail entre pairs autorisant une analyse et une correction du programme par les élèves eux-mêmes. Ce geste, expert, permet à l'enseignant d'observer ses élèves et d'entrer dans un rôle de modérateur.

« J'essaye toujours de me mettre en retrait mais j'essaie que ce soit vraiment les interactions des enfants qui leurs permettent de trouver eux-mêmes les solutions aux problèmes qui leur sont proposés. Donc, moi, je suis là plus pour réorienter leurs réflexions mais j'essaie de pas trop les guider. » (Entretien Enseignant MLP, tablette2)

3 Nous avons, par exemple, pu observer une enseignante qui proposait aux élèves d'utiliser un quadrillage papier et des petits bonhommes pour simuler les déplacements du robot.

4.2 Des postures professionnelles différenciées en fonction des instruments-médiateurs :

Les gestes professionnels dégagés précédemment s'organisent autour de trois postures professionnelles endossées par les enseignants dans le cadre d'activités de programmation. En référence aux travaux de Bucheton (2006), nous envisageons une posture professionnelle comme une manière cognitive et langagière de s'emparer d'une tâche en fonction des obstacles liés à l'acquisition du savoir ou aux difficultés ressenties par les élèves.

L'accompagnement et le support robot

La posture d'accompagnement se réalise dans le cadre d'un pilotage de classe souple et ouvert et dans une atmosphère détendue et collaborative. L'enseignant apporte une aide de manière ponctuelle individuelle ou collective en fonction des obstacles à surmonter et de l'avancée de la tâche. Il évite de donner la réponse, il provoque les discussions entre les élèves. Il se retient d'intervenir, observe plus qu'il ne parle. Les élèves, quant à eux, sont amenés à faire et à discuter sur ce qu'ils font et donc à s'inscrire dans une posture réflexive et créative (Bucheton, Soulé, 2004). Les gestes professionnels les plus représentés ici sont le rappel (le plus souvent réalisé par l'élève accompagné par l'enseignant), la reformulation et le questionnement.

« On n'avait même pas besoin de gérer le groupe, ils se géraient tout seul. […] Y a peut-être des groupes où c'était un petit peu plus difficile donc, ceux-là, plus les accompagner. […] Un rôle de pilotage. Si on voit que ça ne part pas dans le bon sens on essaie de corriger » (Entretien Enseignant V, tablette_2)

L'accompagnement dans toutes ses dimensions prédomine sur support robot. Les enseignants évoquent un pilotage de la classe facilité par ce support qui s'intègre facilement au système de fonctionnement en atelier. L'adoption de cette posture est encouragée par une prise en main facile de l'instrument par les enfants et le système de réglette qui autorise un repérage autonome et aisé de l'erreur par les élèves.

Le contrôle et le support corps

Une autre posture observée est celle du contrôle. Ici, le pilotage de l'enseignant est très serré et en synchronie, c'est-à-dire que les élèves doivent travailler au même rythme. L'atmosphère est plutôt tendue et hiérarchique. Les gestes professionnels les plus représentés pour cette seconde posture sont ceux de l'aide dans l'activité de programmation. L'enseignant va, par exemple, inciter voire imposer un procédé à suivre, faire avec ou à la place de l'élève ce qui suppose bien une forme de contrôle sur l'activité de l'élève.

« Et alors ? qu'est-ce qu'on fait maintenant ? Il faut tourner ? On n'a pas dit de tourner ? […] [l'enseignante intervient alors corporellement pour replacer l'élève dans le bon sens] » (Extrait captation vidéo, enseignante M, corps_1)

Quelle que soit la place du support dans le processus d'apprentissage, on peut remarquer une prédominance de la posture du contrôle sur le support corps avec une intervention verbale et corporelle davantage marquée des enseignants.

Le lâcher-prise et le support tablette

Enfin, la posture du lâcher prise se caractérise par un pilotage confié au groupe, avec une atmosphère de confiance et de refus d'intervention du maître. L'enseignant confie aux élèves la responsabilité de leur travail et les autorise à expérimenter, les laisse libre de leurs choix. Les élèves sont, ici, dans une posture réflexive et « du faire » puisqu'ils sont amenés à agir, créer et à développer une réflexion sur leur propre action. Le geste professionnel le plus représentatif de cette posture est la mise en retrait.

« C'est plus un rôle […] de coordinateur. […] Donc, l'application est le support d'apprentissage et on plus là à veiller, d'une part, que l'élève ne s'écarte pas de la tâche et, d'autre part, à réorienter l'élève. […] C'est très individuel comme activité, […] ça permet à chacun de bien réfléchir et au maître de bien réorienté ou de laisser filer si on voit que l'élève réussit. On n'est plus dans celui qui transmet le savoir, on est plus dans celui qui aide à avancer sur le chemin qui mène au savoir. » (Enseignant H, tablette_2)

La posture du lâcher-prise est plus prégnante sur support tablette. Le caractère intuitif de ce support mais aussi son format réduit implique une autonomie des élèves et l'adoption par l'enseignant d'un rôle d'observateur, de modérateur.

4.3 Des conceptions de la programmation signe d'une appropriation des instrument-médiateur par l'enseignant

L'analyse des données nous ont également permis d'approcher les conceptions construites par les enseignants à propos de l'enseignement de la programmation en maternelle.

Les objets robot et tablettes sont tout d'abord plébiscités pour leurs aspects ludiques, dimension particulièrement importante aux yeux des enseignants intervenants auprès de jeunes élèves. En effet, « jouer pour apprendre » s'inscrit dans la philosophie des programmes et recommandations d'enseignement à l'école maternelle[4] qui considèrent le jeu comme essentiel au bon développement physique, psychologique et social de l'enfant. Pour autant, selon les enseignants, cet aspect ludique ne serait pas suffisant en lui-même et des objectifs pédagogiques doivent être clairement identifiés :

« Je pense que c'est un outil [le robot] qui est amusant pour les enfants. C'est l'aspect ludique de l'outil qui est intéressant. Cela correspond à notre société de maintenant aussi. Je pense qu'il faut y trouver des objectifs pédagogiques pour l'utiliser. » (Entretien enseignante A, robot_2)

Ainsi, conscients que les activités de programmation ne sont pas au programme du cycle1, les enseignants y voient la possibilité de réinvestir les compétences travaillées dans d'autres domaines d'apprentissage ce qui justifie en soi cet enseignement. On se situe ici dans « programmer pour apprendre », une initiation précoce à la programmation s'inscrivant dans la logique de parcours et de continuité des apprentissages promue par l'école :

4 https://cache.media.eduscol.education.fr/file/Apprendre/30/3/Ress_c1_jouer_jouerapprendre_458303.pdf

« Ça leur permet aussi de travailler toutes ces compétences de […] mise en espace mais réflexif. Ils réussissent à imaginer ce que peut être une forme, une direction sans forcément l'avoir tout de suite sous les yeux. [….] Le fait de développer la logique des élèves, le retour sur la réflexion, etc » (Entretien enseignant H, robot_1)

« Dans les évaluations nationales et celles de notre bassin qui est quand même défavorisé, on voit que les évaluations CE1 ce qui pêche, c'est la résolution de problèmes. Donc, je me dis plus tôt on met en place des activités qui vont mettre en route des images mentales, des schémas chez l'élèves, plus tôt, on va l'entraîner à ça, plus vite ça portera ses fruits. Et c'est ça la grande force de ce projet, c'est la construction des images mentales, la résolution de problèmes. Il est pas du tout illogique de travailler des compétences de codage puisqu'elles seront reprises des années plus tard. » (Entretien enseignant C, robot_1)

Les limites d'initier de jeunes élèves à la programmation sont moins évoquées par les enseignants. La question du rapport entre le niveau cognitif de l'enfant et les tâches à accomplir (notamment les tâches qui ont attrait à l'abstraction), est néanmoins soulevée par un enseignant. Il ne faudrait pas trop anticiper certains apprentissages ou, du moins, ne pas forcer l'entrée dans ces apprentissages, le propre de l'école maternelle étant aussi cette faculté de « laisser du temps » aux enfants, de les préparer aux apprentissages futurs mais sans anticiper ceux-ci trop prématurément en les mettant en situation d'échec :

« Peut-être certains se sentiraient en échec face à ça et ensuite, ça se répercuterait sur différentes activités qu'on leur proposerait pour la suite de leur scolarité. En même temps, j'ai pas vu trop d'élèves en échec sur ce qu'on a fait. » (Entretien Enseignante H, robot_1)

Le pilotage de la classe à travers l'accompagnement du travail entre pairs constitue, en revanche, une difficulté mentionnée par plusieurs enseignants :

« Les inconvénients c'est que c'est un travail par petits groupes […] on doit le faire en dirigé pour le mettre en place, tout au moins au début…parce qu'ils font par essaie-erreur je pense…dons ils peuvent peut-être être plus en autonomie mais on sait bien qu'avec certains enfants cela ne va pas être facile à faire. C'est donc plutôt des difficultés de l'ordre du pilotage de la classe. » (Entretien Enseignante L, robot_1).

Enfin, notons que les enseignants de maternelle n'ont pas été formés à l'enseignement de la programmation et aux outils dédiés à cet enseignement et leurs connaissances théoriques liées aux sciences informatiques apparaissent alors fragiles.

« J'ai fréquemment utilisé le mot « programme » mais beaucoup moins les termes « instruction » et « code ». Je pense ne pas être au clair moi -même avec les nuances entre ces termes. » (Carnet de Bord enseignant C, robot_3)

5 Comprendre l'influence des médiations cognitives sur la grammaire de l'agir enseignant à partir de la matérialité différente des supports

Précédemment, nous avons procédé à l'analyse de la grammaire de l'agir enseignant confronté à la mise en œuvre de scénarios pédagogiques d'initiation à la pro-

grammation auprès d'élèves de GSM. Nous avons alors analysé leurs gestes, postures et conceptions professionnels construits lors de ces séances dédiées à la programmation. Nous avons ainsi identifié la mise en œuvre de trois postures professionnelles différentes : la posture de contrôle sur support corps, la posture d'accompagnement sur support robot et la posture de lâcher prise sur support tablette. A chacune de ces postures a été adossé des gestes professionnels particuliers. Maintenant, nous proposons de comprendre plus particulièrement l'influence des instruments médiateurs en jeu sur cette grammaire de l'agir enseignant.

La matérialité des instruments médiateurs constitue un élément de compréhension d'un agir enseignant différencié dans les activités de programmation. A l'instar de Laparra et Margolinas (2016), nous pensons que *« la matérialité d'une situation d'apprentissage n'est jamais indifférente »* (p 30), La dimension matérielle des tâches et objets scolaires structurant l'organisation de l'espace de classe et les conduites des élèves et enseignants au sein de cet espace. Dans de précédentes analyses (Bellegarde, Boyaval, Alvarez, 2019), nous avons montré les incidences de la matérialité des supports médiateurs sur l'appropriation de la programmation chez élèves de GSM au travers : (1) des dimensions des supports (tangible (robot et corps) versus virtuelle (tablette), intracorporelle (corps) versus extracorporelle (robot et tablette)) et (2) de leur écart à la réalité physique de l'élève[5]. Nous avons ainsi appréhendé les différences entre ces dispositifs pédagogiques en termes d'activités cognitives prises en charge et de compétences travaillées par les élèves. Demandons-nous maintenant en quoi ces matérialités des instruments médiateurs peuvent transformer les qualités de la situation d'enseignement/apprentissage du point de vue de l'enseignant ?

D'une part, la dimension tangible, caractéristique des supports corps et robots, a autorisé une intervention de l'enseignant qui peut agir corporellement sur le robot Blue Bot ou l'enfant-robot : par exemple, les repositionner dans le bon sens sur la case « départ ». A l'inverse la dimension virtuelle du support tablette permet une moindre intervention de l'enseignant sur l'objet qu'il ne peut être saisi, sur lequel il ne peut influer.

« Je les ai laissés libres de manipuler. Je leur rappelais le scénario et puis comme ils étaient en binôme en fait ils se débrouillaient tout seul. […] [C'était] du tutorat et puis j'écoutais si la verbalisation était correcte » (Entretien enseignant L, tablette3)

La dimension intracorporelle du support corps est venue également renforcer le positionnement interventionniste de l'enseignant. Les schèmes d'utilisation de ce dispositif à travers un élève qui joue le rôle de l'enfant-robot explique cette posture de contrôle : on observe alors des interventions corporelles et verbales plus marquées pour inciter l'enfant-robot à suivre les instructions données.

« Il y a eu un peu plus d'interventions de ma part […] [sur support corps] puisque là, le robot est faillible par essence puisque c'est un enfant. Donc, j'avais ce rôle de vérificateur. » (Enseignant H, corps3)

Inversement, la dimension extracorporelle des supports robot et tablette, leurs schèmes d'utilisation invitent à une mise en retrait de l'enseignant. « Ces objets pour penser avec » (Papert, 1981) inscrivent les élèves dans un contexte de résolution de

5 Identifier l'écart du média à la réalité physique de l'élève revient à mesurer l'adéquation entre le corps de l'élève et le support utilisé.

problèmes, la recherche, le contrôle et le débogage du programme par les élèves, étant facilité par la mise en œuvre du programme par les dispositifs pédagogiques eux-mêmes et le système de réglette adossé.

« Le système réglette permet à l'enfant de revoir sa programmation parce qu'elle reste là. [Sans], il n'aurait pas vu le résultat de sa programmation, où est-ce qu'il se serait trompé. […] Là, au moins, y avait une trace de son essai. » (Enseignant W., robot1)

Enfin, l'écart du média à la réalité physique de l'élève vient renforcer la prévalence des postures enseignantes observées sur les trois supports expérimentés lors des activités de programmation. Le dispositif tablette, de par sa taille réduite, apparaît proche de la réalité de l'élève, il la tient entre ses doigts juste devant ses yeux. A l'inverse, on observe, à travers la grandeur du quadrillage sur support robot et particulièrement sur support corps, un écart plus important entre ces dispositifs et la réalité physique de l'enfant. Dans le premier cas, l'enseignant se trouve naturellement mis en retrait de l'activité de programmation de l'élève, dans le second, l'enseignant est autorisé à s'impliquer, intervenir dans cette activité.

6 Conclusion

Au démarrage de ce travail, nous supposions que le support robot conduirait particulièrement à une mise en retrait de l'enseignant au profit d'apprentissages autodirigés chez les élèves. Nos analyses nuancent ce positionnement supposé de l'enseignant.

Effectivement, la matérialité du support robot procure à l'enfant des possibilités de décentration, de manipulation de l'objet-robot intéressantes tout en l'inscrivant dans un contexte de résolution de problème. Toutefois, la dimension tangible de cet instrument-médiateur et son écart à la réalité physiques de l'élève structurent l'agir enseignant autour de gestes professionnels relevant de la posture d'accompagnement. La matérialité du support robotique autorise l'intervention de l'enseignant, et moins sa mise en retrait. Nous avons ainsi montré que c'est la dimension virtuelle de la tablette et sa proximité à la réalité physique de l'élève qui encouragent l'adoption de la posture du lâcher-prise chez l'enseignant.

A l'instar de Laparra et Margolinas (2016), ces résultats relèvent l'intérêt à porter à la matérialité des tâches et objets scolaires dans l'optique de comprendre la manière dont celle-ci peut transformer les qualités des situations scolaires et ainsi influer sur l'agir enseignant. Ce travail met alors en évidence les enjeux de formation et d'accompagnement des enseignants relatifs à l'introduction de nouveaux outils pédagogiques et des sciences informatiques à l'école.

Références

1. Alvarez, J., Djaouti, D., Rampnoux, O. Apprendre avec les Serious Games ? France : Réseaux Canopé (2016).

2. Bellegarde, K., Boyaval, J., Alvarez, J. S'initier à la robotique/informatique en classe de grande section de maternelle - une expérimentation autour de l'utilisation du robot Blue Bot comme jeux sérieux, RESMICTE, 13, 1, p. 51-72 (2019).
3. Bucheton D. L'agir enseignant : des gestes professionnels ajustés. Ocatres éditon. (2014)
4. Bucheton D., Soulé, Y. Les gestes professionnels et le jeu des postures de l'enseignant dans la classe : un multi-agenda de préoccupations enchâssées. Education et didactique (2009).
5. Bruner, J-S. Le développement de l'enfant, savoir-faire, savoir dire. Paris : PUF (1983).
6. Denis, B. Vingt ans de robotique pédagogique, *Sciences et techniques éducatives*, (7),1, pp. 195-206 (2000).
7. Greff, E. Le corps d'abord ! Education enfantine, 1056, 62-63 (2004).
8. Komis, V., Misirli Etude des processus de construction d'algorithmes et de programmation par les petits enfants à l'aide de jouets programmables. In G-L. Baron, E. Bruillard, V. Komis (pp. 271-281). Sciences et technologies de l'information et de la communication en milieu éducatif. Clermont-Ferrand, France : edutice-00875628 (2013).
9. Linard, M. Des machines et des hommes. Apprendre avec les nouvelles technologies. Paris : L'Harmattan (1996).
10. Lappara, M. Margolinas, C. *Les premiers apprentissages scolaires à la loupe : des liens entre énumération, oralité et littératie*. De Boeck. (2016)
11. Leroux, P). *Machines partenaires des apprenants et des enseignants - Étude dans le cadre d'environnements supports de projets pédagogiques* (HDR). Université du Maine (2002).
12. Papert, S. Jaillissement de l'esprit. Ordinateur et apprentissage. Paris : Flammarion (1981).
13. Rézeau, J. Médiation, médiatisation et instruments d'enseignement : du triangle au carré pédagogique, ASP, Varia (35-36), p183-200 (2002).
14. Romero, M. Vibot – le robot. Québec : les publications du Québec (2016).
15. Tisseron, S. Apprivoiser les écrans pour grandir. Paris : Erès (2013).
16. Vivet, M. Des robots pour apprendre, *Sciences et techniques éducatives*, (7), 17-60 (2000).
17. Weil-Barais, A., Reste-Schweitzer, M. Approche cognitive développementale de la médiation en contexte d'enseignement-apprentissage, la nouvelle revue de l'adaptation et de la scolarisation, 42, 83-98(2008).

Robots scolaires et configuration spatiale de la classe
Effets sur les interactions sociales dans le cadre de séquences pédagogiques au CE2

Rawad Chaker[1][0000-0003-4616-3246] et Théodore Njingang Mbadjoin[2]

[1,2] Laboratoire Education, Cultures, Politiques (EA 4571), Université Lumière Lyon 2
[1]rawad.chaker@univ-lyon2.fr
[2]theodore.njingangmbadjoin@univ-lyon2.fr

Résumé. Cet article étudie les effets de l'intégration des artefacts tels que les robots et les tablettes en salle de classe sur la reconfiguration de l'espace et les interactions entre les acteurs. Les résultats montrent que les activités proposées par l'enseignant dans le cadre des scénarios pédagogiques jouent un rôle important dans la répartition spatiale des élèves (en îlots hermétiques ou poreux, alignés ou dispersés) et des enseignants (face au groupe-classe, face à un groupe d'élève, excentré ou au milieu de la salle). Nous déduisons que les facteurs qui conduisent à reconfigurer l'espace-classe ne sont pas les artefacts numériques mais la prescription scénaristique ainsi que les consignes données par l'enseignant. Ces dispositions entraînent des effets sur les interactions entre acteurs : entre élèves, entre l'élève et les groupes d'élèves, entre l'élève et l'enseignant, entre l'enseignant et le groupe-classe, et enfin entre l'enseignant et les groupes d'élèves. Ces résultats conduisent à leur tour à un niveau d'engagement dans la tâche différencié de la part des apprenants. Nous concluons par des recommandations à l'intention du public enseignant dans le cadre sa préparation de séquences pédagogiques intégrant des outils numériques.

Keywords: Robots scolaires, nouveaux espaces d'apprentissage, configuration spatiale, interactions.

1 Introduction : les robots dans les espaces scolaires, des questions contemporaines et légitimes

Nous observons aujourd'hui une forte tendance à repenser les nouvelles configurations des espaces scolaires (intérieurs ou extérieurs), pour tenter de répondre à des besoins éducatifs soulevant la question de leur appropriation pédagogique pour l'enseignement-apprentissage en petit et grand effectif [1]. C'est à juste titre que [2] pose cette question qui reste d'actualité : « quelles formes de classes pour quelles pédagogies ? ». Des auteurs avancent que la performance des élèves aurait un rapport avec le type de configuration de l'espace, les cours dispensés et activités réalisées en classe [3, 4]. Les recherches qui ont été menées sur le fonctionnement des classes en

considérant l'environnement d'apprentissage évoquent le fait que les organisations spatiales influenceraient le modèle pédagogique déployé par l'enseignant ou encore le comportement d'apprenants en cours d'activité de classe [3 ; 4 ; 5]. La configuration de l'espace en situation sert à accompagner les enfants dans les recherches de solution et contribuer à faire exprimer leurs capacités créatives [6]. Par exemple, des recherches proposent de repenser l'architecture de l'école pour que les bâtiments soient à la fois des médias spatiaux, des artefacts et des nœuds de relations en facilitant une exploitation du travail en îlot via la mobilité des équipements, [1 ; 7]. En outre, ces dernières années, ont été publiées différentes études portant sur les artefacts technologiques virtuels ou tangibles à l'instar des robots éducatifs dans la salle de classe, dans le cadre d'activités d'apprentissage des mathématiques et des langues [8]. Parallèlement, des questions de recherche portant sur l'apprentissage du code à l'école et la robotique éducative prennent de l'ampleur, notamment en France depuis que les robots ont fait leur entrée dans les programmes scolaire en 2015, dans la perspective du développement de la pensée computationnelle comme compétence-clé du XXI^{ème} siècle [9 ; 10]. Dans cette optique, certains auteurs analysent les interactions d'apprentissage entre acteurs (élèves, enseignants) et robots [11], et peu se préoccupent du rapport à l'environnement et notamment l'impact des modifications apportées dans l'espace sur la stratégie pédagogique et l'apprentissage [12], tel [13] qui étudie l'effort des espaces sur le développement de la pensée computationnelle dans le cadre d'activités robotiques scolaires. De ce point de vue, notre étude est complémentaire de ces travaux avec l'intérêt d'interroger en contexte pédagogique l'effet de l'intégration des robots scolaires sur la configuration spatiale à l'école primaire. Il s'agit d'étudier non pas les espaces d'apprentissage conçus ou aménagés en amont, mais plutôt la manière dont l'espace-classe se reconfigure sous l'effet conjugué de l'intégration d'objets numériques inédits pour les acteurs et de scénarios pédagogiques prévus en conséquence. D'où cette double question de recherche : est-ce que l'introduction d'objets tangibles (robots, tablettes) dans le cadre de scénarios pédagogiques oblige à la reconfiguration de l'espace classe ? Comment cette disposition impacte les interactions entre les élèves d'une part, et entre l'enseignant et les élèves d'autre part ? Pour y répondre, nous présentons d'abord le contexte de l'étude, le cadre théorique, notre démarche méthodologique puis nos résultats à partir de l'analyse des vidéos réalisées lors des séances d'activités en groupe des élèves de CE2.

2 Contexte

Notre étude se déroule dans le contexte du projet PREP (Programmation du Robot à l'Ecole Primaire), financé par la région Rhône-Alpes (2018-2021). Le but de cette expérimentation, à laquelle participent plus de 1200 élèves et 75 enseignants, est de mettre en œuvre des séquences pédagogiques intégrant la robotique scolaire, dont l'objectif est l'enseignement-apprentissage du code informatique via des robots et des tablettes. Plusieurs ateliers ont été mis en place pour la formation des enseignants en amont de l'expérimentation. Le projet est mené en étroite collaboration avec la cir-

conscription académique de Villeurbanne 1, où se situent cinq des six établissements participants au projet (une autre école est lyonnaise). Durant ces séances de formation, les enseignants ont également été en charge de produire les séquences pédagogiques qu'ils devaient par la suite eux-mêmes mettre en œuvre dans leurs classes. Les trois séquences annuelles comprennent chacune quatre séances s'étalant sur une semaine, pour chaque niveau du CP au CM2. Le présent article se penche plus précisément sur le cas d'une classe de CE2 d'une école villeurbannaise. Il s'agit d'une partie du bilan de la première année d'expérimentation (2018-2019). Comme le montre le tableau 1, l'organisation pédagogique se présente de la manière suivante : une première semaine porte sur l'introduction au langage Scratch Jr sur tablette Android au mois d'octobre, suivi d'une deuxième semaine portant sur des exercices de programmation à réaliser avec le robot Dash à l'aide d'une tablette, puis une dernière et troisième semaine au mois de mars suivant avec le robot Thymio programmé à l'aide d'un PC.

Tableau 1. Planning annuel de la classe de CE2

Numéro de la séquence	Nombre de séances	Période de l'année scolaire	Matériel
1	4	1	Tablette (logiciel Scratch Jr)
2	4	2	Tablette (logiciel Blockly) + robot Dash
3	4	4	PC (logiciel VPL) + robot Thymio

3 Cadre théorique

Afin d'adresser nos questions de recherche, nous nous basons sur les dimensions liées à la structure spatiale de la salle de classe et non pas l'échelle, autrement dit les éléments liés à la taille de la salle [4]. Il est établi que « l'espace exerce une influence liée à la situation sur les activités et expériences humaines au fur et à mesure de leur adoption et ressenties dans les environnements » [5]. Les auteurs montrent que si l'on essaie de réaliser la même tâche dans les limites de différents espaces, elles produiraient des effets différents. Pour étudier cet effet, ils proposent qu'une structure est liée aux arrangements physiques de base, aux configurations de l'espace et la connectivité des objets et acteurs entre eux. Alors que la notion d'échelle relève du rapport entre l'étendue de l'espace physique et sa taille. En isolant les effets de l'espace sur le comportement des apprenants et de l'enseignant, les recherches quasi expérimentales mettent en évidence le rôle de l'espace d'apprentissage où se déroulent les cours et activités. Elles montrent que la salle exerce un effet significativement positif sur l'apprentissage en différenciant entre : 1. la configuration traditionnelle de l'espace que [4] nomme Active Traditional Classroom (ATC) disposant d'un tableau blanc, d'un écran de projection et d'une estrade pour l'enseignant à l'avant de la salle, face à des rangées de sièges et de tables tournés vers l'avant ; et 2. l'environnement ALC (Active Learning Classroom) pour qualifier les espaces modulables, organisation disposant les élèves physiquement autour du déroulement de l'activité et regroupés en

table ronde ou en îlots [4]. Pour analyser les interactions selon chaque configuration, l'auteur distingue les variables liées aux activités en salle (leçon magistrale, activités de groupe, discussions, questions-réponses, présentation) des variables « mode de transmission des contenus » (vidéoprojecteur, tableau, papier), comportement de l'enseignant (ex : consultation des étudiants, placement dans la salle), le niveau d'engagement des apprenants dans la tâche ainsi que les conditions environnementales (température, niveaux sonores, éclairage). Des études s'appuient sur les variables comportements et le niveau d'interaction collaborative pour caractériser une organisation spatiale dynamique plus ou moins favorable à la communication, au contact et l'interaction sociale entre individus [14]. La flexibilité de l'espace comme réponse aux besoins de la vie scolaire et de la pédagogie demandent dans certains cas différentes organisations de la salle de classe. Par exemple, le type alternatif facilite la pédagogie active et notamment la collaboration interactive entre élèves et entre élèves et enseignants, ainsi que la facilité de mobilité dans l'espace [15 ; 7]. Elle permet d'organiser l'interaction et la créativité avec des espaces polyvalents (dimensions variables, grandes/petites tailles, équipent mobile, etc.) impliquant la modularité des équipements. Un espace d'enseignement par équipe interdisciplinaire et un apprentissage coopératif en petits et grands groupes avec des dispositions en îlots intégrant divers mobiliers mobiles avec des agencements et regroupements de tables rondes, carrées, ou ovales pour quatre à six élèves [16 ; 17 ; 18].

4 Problématisation

Partant de ce cadre théorique, nous formulons la problématique suivante : dans quelle mesure un scénario pédagogique introduisant des artefacts tangibles (tablette, ordinateurs et robots scolaires dans notre cas) conduit à reconfigurer l'espace classe ? Ce faisant, jusqu'où la recomposition de l'espace classe avec des artefacts tangibles modifie-t-elle la nature et la quantité des interactions entre les acteurs (élèves – élèves et élèves – enseignants) ? Nous émettons deux hypothèses générales (figure 1):

- H1 : Le scénario pédagogique mène à la reconfiguration spatiale de la classe (H1a). Dans cette perspective, l'intégration d'artefacts tangibles (tablettes, ordinateurs et robots scolaires) dans le scénario déclenche ou mène à la reconfiguration spatiale de la classe (H1b).
- H2 : Cette reconfiguration spatiale impacte à son tour les interactions sociales au sein de l'espace classe.

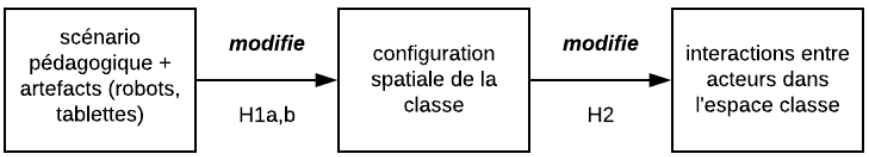


Fig. 1. Schéma de notre problématisation

5 Méthodologie

5.1 Terrain

Le public est constitué d'élèves (N=28, âge moyen 8 ans) en CE2 ainsi que de leur enseignante. Comme nous l'avons vu précédemment, l'étude porte sur trois séquences (tableau 1) de la première année scolaire de l'expérimentation PREP, ayant eu lieu entre octobre 2018 et mars 2019.

5.2 Procédure

Une caméra GoPro© installée sur trépied a été utilisée pour capter systématiquement l'ensemble des trois semaines (quatre jours par séquence, une séance par jour), à raison d'une durée de 50 minutes par séance. Ce qui nous donne en tout douze séances, ou 600 minutes d'observations vidéo au total. L'angle de vue a été ajusté afin d'intégrer dans le champ de la caméra l'ensemble de salle de classe. La séquence 1 (Scratch Jr sur tablette) a eu lieu en salle de classe, la séquence 2 (robot Dash et tablette) en salle informatique, ainsi que la séquence 3 (robot Thymio et PC).

Fig. 2. Captation vidéo de gauche à droite : séquence 1, séquence 2 et séquence 3

5.3 Mesures

Pour vérifier nos hypothèses, nous prenons en considération des indicateurs inspirés et adaptés de la méthode de [4], qui caractérise et quantifie l'investissement spatial en classe selon différentes dimensions, décrites dans le cadre théorique (tableau 2) : 1. le type d'activité prévu par le scénario: questions/réponses, leçon magistrale, activité de groupe ; 2. la modalité de transmission des contenus : vidéoprojecteur, tableau ; 3. Positionnement des élèves dans la classe : îlots hermétiques, îlots poreux, dispersés, alignés, mixte ; 4. Positionnement de l'enseignant dans la classe : face ou au milieu du groupe-classe, excentré ; 5. l'engagement de l'élève sur la tâche (bas, haut, mixte) ; et 6. les interactions entre acteurs (élèves-élèves, élèves-groupe d'élève, élève-enseignant, enseignant-élève, enseignant-groupe d'élèves et enseignant-groupe-classe). Puisque nous cherchons à étudier le rôle des artefacts pédagogiques, nous avons inclus dans notre observation les objets prévus par le scénario pédagogique et come supports pour les élèves : les robots, la tablette, les PC, support écrit (papier).

Tableau 2. Dimensions et observables relevés

Dimensions	Sous dimensions	Description des indicateurs
Activité en classe	Leçon magistrale	Consignes, contenus, ressources, etc
	Activité de groupe	Formation groupes d'interactions, reconstitution des petits groupes d'échanges d'au moins deux membres en cours d'activité.
	Question/réponses	Questions et réponses de l'enseignant ou des élèves
Mode de contenu	Vidéoprojecteur	Exposé de contenus via un vidéoprojecteur
	Tableau	Exposé ou illustration via un tableau
Elèves (Position spatiale)	Ilots hermétiques	Elèves regroupés en groupes fixes, peu de mobilité
	Ilots poreux	Elèves regroupés en groupes avec mobilité
	Alignés	Alignement en « U » ou rangées
	Dispersés	Pas de positionnement particulier, groupements non respectés
Enseignant (Position spatiale)	Face au groupe-classe	Position magistrale
	Au milieu du groupe-classe	Déplacement au milieu de la salle, vers les groupes
	Excentré	Déplacement vers les extrémités de la salle
Engage-ment tâche (élèves)	Haut	haute concertation/concentration sur tâche (de tout le groupe)
	Mixte	concentration sur tâche d'environ la moitié des membres
	Bas	concentration sur la tâche (moins de la moitié des membres
Interactions	Enseignant/élève	réaction enseignant vers un élève (initie, questionne, répond)
	Enseignant/groupe classe	réaction enseignant vers le groupe classe (initie, questionne, répond, propose)
	Enseignant/groupe d'élèves	réaction enseignant vers le groupe d'élèves (initie, questionne, répond, propose)
	Elève/enseignant	réaction élève vers l'enseignant (initie, questions, réponses)
	Elève/élève	réaction élève vers un élève (initie, questionne, répond, propose)
	Elève/groupe	réaction élève vers le groupe (initie, questionne, répond, propose)

Nous avons utilisé le logiciel Actograph© (figure 3) afin de procéder à l'indexation vidéo. Il permet de créer des catégorie (nos dimension) et à l'intérieur de celles-ci des observables (nos indicateurs).

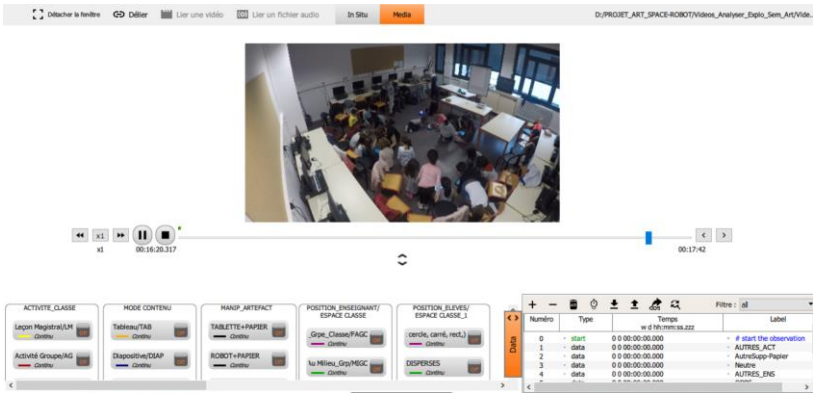


Fig. 3. Ecran d'indexation vidéo d'Actograph©, avec l'enregistrement vidéo, les catégories et leurs observables, et l'horodatage de l'indexation des observable.

Comme plusieurs protocoles d'observation vidéo de salles de classe, nous avons structuré nos unités d'indexation en segments de cinq minutes [4 ; 19]. Ainsi, nous

obtenons des données simultanées pour l'ensemble des nos catégories et observables, reportées en nombre de millisecondes au sein de chaque intervalle. Nous obtenons un total de 125 intervalles, pour une durée totale de 600 minutes (intervalle moyen=5,05 minutes, écart-type=0,2). Cette méthode de recueil de données permet de mesurer simultanément l'ensemble des indicateurs pour chaque dimension. Le test statistique d'accord inter-juges (kappa de Cohen) entre deux observateurs produit une moyenne générale d'indice d'accord de 0,8 sur toutes les dimensions. Nous estimons alors que notre protocole de recueil de données est suffisamment fiable pour poursuivre nos analyses.

6 Résultats

6.1 Scénario pédagogique, robots, tablettes et configurations spatiales

La méthode de clustering permet d'analyser les données en les regroupant selon des ensembles de variables ayant des variances homogènes [20]. Cette technique de traitement des données permet de vérifier si les différences entre les groupes sont significatives. A l'aide du logiciel IBM SPSS 23©, nous avons d'abord utilisé la méthode « Two-step cluster » en intégrant dans la liste des variables l'ensemble des données relevant de la disposition spatiale des élèves et de l'enseignant, ainsi que le mode de transmission des contenus et l'activité pédagogique. Le résultat nous donne deux clusters avec un indice de cohésion de 0,7, ce qui indique une qualité satisfaisant d'une solution à deux clusters. Nous avons par la suite procédé à la méthode « *k*-means clusters » en imposant une solution finale à deux clusters, dans le but de vérifier la qualité des regroupements par l'inspection en détails de la significativité de l'appartenance de chaque variable à son groupe. Nous avons d'abord introduit l'ensemble des variables, puis retiré progressivement ceux dont le test F d'appartenance au cluster confirmait l'hypothèse nulle d'indépendance (p>0,05), autrement dit, dont l'appartenance au groupe 1 ou 2 n'était pas significative. Le résultat final produit bien deux groupes distincts par leur composition (figure 4): le premier, composé des variables « enseignant face au groupe-classe », « leçon magistrale », « questions/réponses » et « tableau blanc », que nous associons à la classe traditionnelle active (ATC) [3 ; 4], dans la mesure où ces éléments correspondent à une configuration de classe où, même si les activités intègrent des outils numérique, en l'occurrence le logiciel de programmation Scratch Jr sur tablettes, le mode transmission des contenus et le positionnement de l'enseignant correspondent à une modalité pédagogique classique. Le deuxième groupe, que nous associons à la Active Learning Classroom (ALC) de [3 ; 4], est composé des variables « groupes d'élèves en îlots poreux » et « activité de groupe », puisqu'il s'agit des configurations où les apprenants sont constitués en groupe non hermétiques, c'est-à-dire qu'il leur est possible de circuler entre les groupes, dans le cadre d'une consigne où les tâches sont à effectuer de manière collaborative. Le tableau 3 montre la significativité de l'appartenance de chaque variable à son cluster.

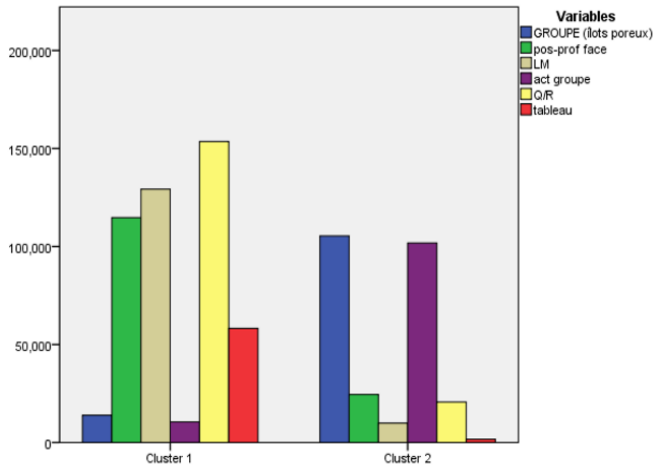


Fig. 4. Clustering des observables (cluster 1=ATC; cluster 2=ALC)

Table 3. Significativité selon le test *F* d'appartenance de chaque observable à son cluster et corrélations entre observables

Clusters	Observables	F	*p*	1	2	3	4	5
1	1.Positionnement de l'enseignant	8,12	,007					
(ATC)	face au groupe-classe							
	2.Leçon magistrale	23,7	,000	,34*				
	3.Questions/réponses	23,3	,000	,27	,37*			
	4.Tableau blanc	7,83	,008	,37**	,39*	,43**		
2	5.Groupes (îlots poreux)	15,1	,000	-,09	-,48**	-,33*	-,27	
(ALC)	6.Activités de groupe	8,5	,006	,17	-,20	-,05	-,07	,44**

*ddl=33 ; *p<0,05 ; **p<0,01*

Ces résultats confirment bien l'existence de deux configurations spatiales et scénaristiques différentes par la disposition des acteurs dans la salle, qui est interdépendante avec la nature de l'activité et la modalité de diffusion des contenus. En effet, les variables « îlots poreux » et « activités de groupe », appartenant au groupe 2, sont corrélées (r_s=0,44 ; p<0,01). Nous observons d'autres corrélations entre les observables du groupe 1 : « leçon magistrale » avec le « positionnement de l'enseignant face au groupe-classe » (r_s=0,34 ; p<0,05), « questions-réponses » avec le « tableau blanc » (r_s=0,43 ; p<0,01), qui est corrélé avec le positionnement de l'enseignant (r_s=0,37 ; p<0,01), et « leçon magistrale » (r_s=0,39 ; p<0,05). Sont également mis en évidence des corrélations négatives entre certaines d'entre elles et certaines du groupe 2 (« îlots poreux » avec « leçon magistrale » : r_s=-0,48 ; p<0,01 ; et avec « questions/réponses » : r_s=-0,33 ; p<0,05), ce qui marque une différenciation statistique supplémentaire entre les deux groupes et leurs variables respectives. Nous pouvons valider notre H1a en y répondant plus précisément : il existe un effet du scénario pédagogique, en termes de prescription, sur la configuration spatiale de la classe. Pour-

tant, le rôle des artefacts tangibles tels que les robots et les tablettes sur ces configurations de l'espace reste encore à préciser. Nous cherchons d'abord à vérifier si la configuration spatiale ATC ou ALC correspond aux séquences (1, 2 ou 3) : le test du χ^2 nous mène à retenir l'hypothèse d'indépendance entre numéro de séquence et configuration spatiale (p>0,05). Autrement dit, la configuration spatiale n'est pas constante sur l'ensemble d'une seule séquence, quand bien-même ces dernières sont différenciées en fonction du matériel et du scénario. Cela voudrait dire alors que la configuration se modifie tout au long des séances ou au sein d'une séance. Y aurait-il alors un effet temporel, lors du déroulé d'une séance, sur la configuration spatiale ? En effet, il apparaît lors des visionnages que différentes reconfigurations peuvent se mettre en place dans le cadre d'une seule séance de 50 minutes, quelle que soit la séquence, la disposition des acteurs en début d'une séance se différenciant de la fin. Nous avons ainsi introduit une variable binaire (première ou deuxième moitié de séance) à laquelle chaque intervalle appartient. Le test exact de Fisher nous indique que la configuration ATC correspond aux observations en première moitié de séance et ALC à la deuxième moitié : χ^2=6,88 ; p=0,015. Il s'agit donc d'un élément de réponse supplémentaire : la configuration spatiale dépendrait donc de la scénarisation pédagogique, des consignes données et du moment dans la séance, plutôt qu'uniquement du matériel utilisé. Afin de vérifier notre H1b, nous allons analyser si les artefacts tangibles utilisés par les élèves varient en fonction de la configuration spatiale. Les distributions des résidus de ces variables indiquant qu'elles violent les lois de normalité (p<0,001), nos tests statistiques de variance seront donc non-paramétriques. Les résultats des tests U de Mann-Whitney nous mènent à retenir l'hypothèse d'indépendance entre la configuration spatiale et chaque objet tangible intégré dans le scénario (0,096<p<1). Il n'y a donc pas de lien entre l'artefact utilisé dans le scénario et la configuration (H1b) mais bien entre cette dernière et le scénario pédagogique en situation ou prescrit.

6.2 Configuration spatiale et interactions

Nous vérifions à présent s'il existe un effet de la configuration spatiale sur les interactions entre acteurs tels que nous les avons relevés. Pour ce faire, nous opérons à une modélisation d'équations structurelles intégrant la variable configuration ATC ou ALC comme prédictive des variables d'interactions, en ne retenant que les liens significatifs (p<0,05). Nous avons utilisé le test du ratio Chi² sur le degré de liberté du modèle (χ^2/df), le « Comparative Fit Index » (CFI), le « Tucker Lewis Index » (TLI) et le « Root Mean Squared Error of Approximation » (RMSEA) pour évaluer l'ajustement de nos données au modèle obtenu. Il est communément accepté [21] qu'une valeur supérieure ou égale à 0,95 pour le CFI et le TLI. Une valeur inférieure ou égale à 0,06 pour le RMSEA est préférable mais acceptable en deçà de 0,08. Notre modèle s'ajuste bien à nos données : $\chi^2(13)$=1,16 ; p=0,3 ; TLI=0,96 ; CFI=0,97 ; RMSEA=0,07. La figure 5 illustre le modèle et les relations entre les variables. Avec ATC=1 et ALC=2, les coefficients de régression indiquent que le « tableau blanc » prédit la configuration ATC (β=-0,44 ; R^2=0,2), qui prédit les interactions de l'enseignante vers le groupe classe (β=-0,68 ; R^2=0,46), les interactions des élèves vers l'enseignante (β=-0,82 ; R^2=0,67) et absence d'interactions entre élèves (β=-

0,48 ; R^2=0,23). La configuration ALC prédit les interactions entre l'enseignante et les groupes d'élèves (β=0,43 ; R^2=0,20 et de l'élève vers les groupes d'élèves (β=0,49 ; R^2=0,24). Ces résultats démontrent que les configurations spatiales prédisent avec des niveaux de variances relativement importants ($|0,43|<\beta<|0,82|$; $0,2<R^2<0,67$) les interactions entre acteurs (H2) : la configuration ATC prédit les interactions correspondant aux modalités « classiques », autrement dit de l'enseignante vers le groupe classe et pas d'interactions, mais permet également un niveau important d'échanges depuis l'élève qui interpelle l'enseignante. Puisque le ATC contient les variables d'activité prévues par le scénario « leçon magistrale » et « questions/réponses », nous pouvons déduire que cette modalité permet à l'élève d'interroger davantage l'enseignante (notamment en début de séance, comme nous l'avons vu) que dans la configuration ALC, qui elle, permet davantage d'interactions entre élèves. Comme attendu selon la modalité ATC, c'est l'outil « tableau blanc » qui est significativement concerné. Nous confirmons ainsi notre H2 : la configuration spatiale influence les relations entre les acteurs.

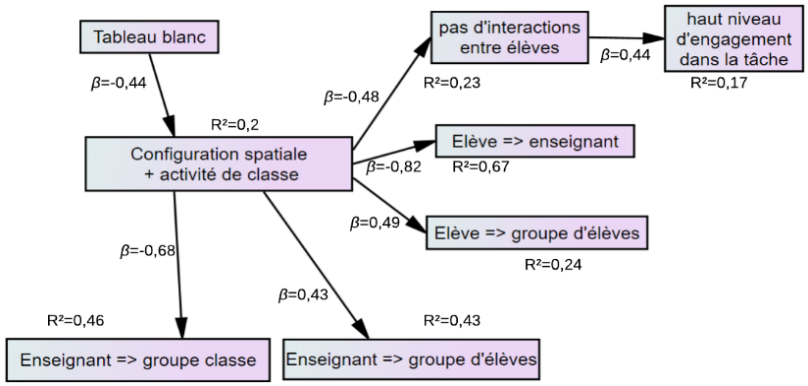


Fig. 5. Modélisation d'équations structurelles entre la configuration spatiale (ATC =1 ; ALC=2), les interactions entre acteurs et le haut niveau d'engagement de l'élève dans la tâche

A défaut de pouvoir mesurer la performance des élèves dans le cadre de cette étude, nous cherchons toutefois à vérifier si ce modèle mis au jour permet de prédire le niveau d'engagement de l'élève dans la tâche. Nous intégrons donc à notre modèle la variable « haut niveau d'engagement de l'élève dans la tâche ». La nouvelle version du modèle s'ajuste toujours correctement à nos données : $\chi^2(18)=1,16$; p=0,28 ; TLI=0,96 ; CFI=0,97 ; RMSEA=0,068. Il ressort que le seul lien significatif depuis nos variables d'interactions vers la variable d'engagement haut est depuis la variable « pas d'interactions entres les élèves » : β=0,44 ; R^2=0,17. Ainsi, c'est la configuration ATC, par le biais de la non-interaction entre acteurs, qui prédit le haut niveau d'engagement dans la tâche de la part de l'élève. Ces résultats paraissent contre-intuitifs : nous pouvions en effet nous attendre à ce que la configuration ALC, par le biais des interactions entre élèves et la personnalisation de l'intervention de l'enseignant vers un groupe d'élève qu'elle permet, va conduire les apprenants à s'engager dans leur tâche davantage que dans une configuration classique. Pourtant, c'est l'absence d'interactions, caractéristique de cette dernière configuration, qui va

permettre cela. Par ailleurs, il apparait que le lien direct entre la configuration spatiale et le haut niveau d'engagement dans la tâche n'est pas significatif. C'est donc bien indirectement, par le biais de l'absence d'interactions entre les élèves, que le niveau d'engagement haut des élèves est permis.

7 Discussion et conclusion

Ces résultats confirment en partie ceux de l'étude de [4] : dans son cas, la configuration ATC a permis un plus haut niveau d'engagement des élèves dans la tâche qu'ALC. Même si nous n'avons pu établir de lien direct entre la configuration spatiale et le haut niveau d'engagement dans la tâche, il apparaît que ce dernier dépend de l'absence d'interactions, qui lui est permis par la configuration ATC. Si le cadre de ce papier ne permet pas de mesurer la performance des élèves, d'après une étude américaine au niveau national, un niveau élevé d'engagement des élèves augmente l'apprentissage et la rétention des contenus [22]. Nous avons démontré que le niveau d'engagement dépend des interactions, et non pas de la configuration spatiale directement. Nous pouvons interpréter ces résultats par le fait qu'avant l'activité, c'est-à-dire en amont de la consigne donnée par l'enseignant, les élèves ne peuvent prédire de l'efficacité de l'outil, surtout s'il s'agit d'un objet nouveau tel que le robot et pour certains la tablette. Ainsi, nous pourrions imaginer que peu importe l'outil que l'enseignant pourrait proposer, les élèves vont apprendre à l'utiliser en situation, cette dernière étant cadrée par le niveau de prescription du scénario pédagogique. Ainsi, ce ne sont pas les outils intégrés dans le scénario qui vont conditionner la configuration spatiale (puis les interactions), mais bien l'orchestration [23] des activités dans l'espace classe que l'enseignant souhaite mettre en place. Il est à souligner l'importance de la mise en œuvre du scénario en situation : c'est bien l'environnement, donc le niveau de prescription [24], qui va permettre ou non l'engagement de l'élève dans la tâche, la prise en compte du robot ou de la tablette seuls ne pouvant la prédire. Nous établissons alors une série de recommandations pour les praticiens et pédagogues : il est d'abord important de réaliser que ce n'est pas l'outil seul qui va assurer de l'efficacité d'une situation d'apprentissage (notamment en termes d'engagement et de concentration des élèves), ni de la bonne conduite du scénario pédagogique. Il est en effet nécessaire de prendre en compte le degré de prescription du scénario pédagogique : ce qu'il permet de faire, ce qu'il doit permettre de faire, et le degré de flexibilité pour parer à d'éventuels obstacles en situation. Enfin, anticiper le fait que ce scénario, autrement dit l'ensemble des consignes, vont modifier l'espace-classe (et donc affecter les interactions entre acteurs), et donc prévoir les activités en conséquence.

Références

1. Blyth, A. (2013). Perspectives pour les futurs espaces scolaires. *Revue internationale d'éducation de Sèvres, (64)*, 53-64. http://ries.revues.org/3606

2. Forster, S. (2004). Quelles formes de classe pour quelles pédagogies ? *L'architecture scolaire, 7*.
3. Brooks, D. C. (2011). Space matters: The impact of formal learning environments on student learning. *British Journal of Educational Technology*, *42*(5), 719-726.
4. Brooks, D. C. (2012). Space and consequences: The impact of different formal learning spaces on instructor and student behavior. *Journal of Learning Spaces, 1*(2).
5. Amedeo, D., Golledge, R. G., & Stimson, R. J. (2008). *Person-environment-behavior research: Investigating activities and experiences in spaces and environments*. New York: Guilford
6. Cohen, B. (2010). Des espaces pour se développer : Comment l'architecture peut jouer un rôle essentiel dans la vie des jeunes enfants. *CELE Echanges, 6*. OCDE.
7. Jeannin, L. (2017). La mobilité, clé de nouvelles pratiques ? *Éducation et socialisation. Les Cahiers du CERFEE, (43)*. http://edso.revues.org/1950
8. Merdan, M., Leputchitz, Koppensteaner, G., Balogh, R., Obdrzaleg (2017) Robotics in education, (AISC) *Advances in Intelligent Systems and Computing*, *1023*, Springer.
9. Lehmans, A. (2017). De l'informatique éducative au robot émancipateur. *Hermès, La Revue*, *78*(2), 132-138. https://www.cairn.info/revue-hermes-la-revue-2017-2-page-132.htm
10. Parriaux, G., Pellet, J. P., Baron, G. L., Bruillard, É., & Komis, V. (2018). *De 0 à 1 ou l'heure de l'informatique à l'école. Actes du colloque Didapro 7–DidaSTIC*. Peter Lang.
11. Lytridis, C., Bazinas, C., Papakostas, G., A., Kaburlasos, V. (2017). On Measuring Engagement Level During Child-Robot Interaction in Education in Kacprzyk, J.(eds). *Advances in Intelligent Systems and Computing, 1023*, Springer, Poland.
12. Oblinger, D. G. (2006). Space as a change agent. *Learning spaces, 1*, 1-2.
13. Chiasson, M. (2019). *Etude des caracteristiques de l'espace d'apprentissage favorisant le processus de la pensée informatique chez les eleves de l'ecole intermediaire*. Doctoral dissertation, Universite de Moncton (Canada).
14. Issaadi, S., & Jaillet, A. (2017). Proxémie d'apprentissage. *Éducation et socialisation. Les Cahiers du CERFEE, (43)*. http://edso.revues.org/1960
15. Schneider R. (2004). Tendances de l'architecture scolaire en Allemagne au XXè siècle. *Histoire de l'éducations, (102),* 137-155.
16. Mazalto M., Bonnault M-C., Zahra, B. (dir.) (2008). *Architecture scolaire et réussite éducative*. Paris: Fabert, 191.
17. Musset M. (2012). De l'architecture scolaire aux espaces d'apprentissage : au bonheur d'apprendre ? *Dossier d'actualité Veille et Analyses de l'IFé, n° 75.*
18. Martin, S. H. (2002). The classroom environment and its effects on the practice of teachers. *Journal of Environmental Psychology*, *22*(1-2), 139-156.
19. Stipek, D., & Byler, P. (2004). The early childhood classroom observation measure. *Early Childhood Research Quarterly*, *19*(3), 375-397.
20. Scott, A. J., & Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 507-512.
21. Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural equation modeling: a multidisciplinary journal*, *6*(1), 1-55.
22. Kuh, G. D. (2001). Assessing What Really Matters to Student Learning. Inside The National Survey of Student Engagement, *Change: The Magazine of Higher Learning, 33*:3, 10-17.
23. Dillenbourg, P. (2013). Design for classroom orchestration. *Computers & Education*, *69*, 485-492.

24. Simonian, S. (2014). *L'Affordance socioculturelle: une approche éco-anthropocentrée des objets techniques*. Habilitation à diriger des recherches. Université Rennes 2.

6 leçons de robotique pour les sciences

Ashwarya Arora[1], Julie Boulc'h[1], Céline Caza[2], Magali Gourvil[3], Ikhlass Karchaoui[1], Amandine Lemeunier[3], Cécile Plaud[4], Vincent Ribaud[1], Lisa Rolland[1], Noémie Schall[1], and Elsa Segalen[1]

[1] Université de Bretagne Occidentale, Brest, France
{Prénom.Nom}@etudiant.univ-brest.fr
[2] Ecole Lucien-Guilbault, Montréal, Canada
c.caza@lucien-guilbault.ca
[3] Académie de Rennes, France
Prénom.Nom@ac-rennes.fr
[4] ENSTA Bretagne, Brest, France
cecile.plaud@ensta-bretagne.fr

Résumé « Les filles qui... » forment une communauté éducative qui enseigne la programmation en primaire autour de Brest. Pour appuyer l'enseignement des sciences, un cours de robotique avec Scratch, en six leçons, a été conçu, développé et évalué dans deux classes de cycle 3. Cet article présente, leçon par leçon, les enjeux, les liens avec le socle commun, les apports de la robotique éducative, et les difficultés rencontrées.

Keywords: Scratch · robotique éducative · modèle précurseur

1 Introduction

Le dispositif « les filles qui... » est un projet de communauté éducative. Son but principal est de montrer l'exemple de la programmation Scratch au féminin. « Les filles qui... » enseignent la programmation dans les écoles primaires sous le couvert d'une convention *Partenaires scientifiques pour la classe* établie entre l'inspection académique du Finistère et l'université de Bretagne occidentale.

Trois types de cours sont proposés : Scratch ou mBlock (robots mbot) du CE2 au CM2, Scratch Junior de la moyenne section au CE1. En 2018-2019, environ 70 filles de licence ont animé les cours pour 46 classes et près de 900 élèves. Lors des cours de robotique, nous avons pris conscience du potentiel de la robotique éducative pour l'enseignement des sciences. Ce cours a donc été complètement refondu en 6 leçons indépendantes, chacune destinée à l'acquisition de connaissances et de compétences d'éléments significatifs pour les sciences : mouvements (prélude aux forces), couleurs (vision et synthèse), musique (prélude aux ondes), capteur de son (caractéristiques, mesures), lumière (caractéristiques, seuil), trajectoires (géométrie, puissance des forces). Cet article expose, leçon par leçon, les enjeux, les liens avec les connaissances du socle commun, les apports de la robotique, les difficultés rencontrées. En conclusion, nous présentons les évaluations réalisées sur certaines leçons dans deux classes, une de CM1 en zone rurbaine, et l'autre de CM2 en zone rurale.

2 La robotique à l'école primaire

Les robots utilisés dans les cours sont des robots mbot, programmés dans un environnement mBlock dérivé de Scratch. Les deux classes pilotes ont appris Scratch lors de 6 séances préalables aux cours de robotique, basées sur les concepts définis par Brennan et Resnick [2] dans leur cadre pour la pensée informatique : séquence, boucle, événement, parallélisme, conditionnelle, opérateurs, données. Cependant, les 6 leçons de robotique mobilisent une version simple des concepts ; les enjeux d'apprentissage portent peu sur ces concepts mais sur les phénomènes physiques mis en jeu et les situations scientifiques abordées.

2.1 Objectifs du socle commun et la robotique éducative

Les objectifs officiels (présentés en italique ci-dessous) de connaissances et de compétences liées a la robotique éducative font partie de trois domaines du socle commun. Le site eduscol propose des situations d'évaluation d'éléments signifiants (présentés en caractères droits ci-dessous) pour ces domaines.

Domaine 1 - Composante 3 : Comprendre, s'exprimer en utilisant les langages mathématiques, scientifiques et informatiques. *L'élève effectue des calculs et modélise des situations. L'élève produit et utilise des représentations d'objets, d'expériences, de phénomènes naturels.*

1.1 Utiliser les nombres entiers, décimaux et les fractions simples.

1.2 Reconnaitre des solides usuels et des figures géométriques.

1.3 Se repérer et se déplacer.

Domaine 2 : Les méthodes et outils pour apprendre. *L'élève est amené à résoudre un problème, [...] effectuer une prestation ou produire des objets.*

2.1 Mobiliser des outils numériques pour apprendre, échanger, communiquer.

Domaine 4 : Les systèmes naturels et les systèmes techniques *L'élève sait mener une démarche d'investigation. [...] l'élève formule des hypothèses, les teste et les éprouve ; [...] procède par essais et erreurs ; l'élève modélise pour représenter une situation ; [...] L'élève résout des problèmes impliquant des grandeurs variées, en particulier des situations de proportionnalité.*

4.1 Mener une démarche scientifique, résoudre des problèmes simples.

2.2 Les apports de la robotique en primaire

Thymio Thymio est un robot éducatif qui possède plusieurs capteurs et actuateurs et se programme facilement dans l'environnement Aseba en programmation visuelle, Blocky ou textuelle. Thymio est généralement introduit en classe en demandant aux élèves d'observer son comportement puis en émettant des hypothèses qu'il s'agit de valider en les confrontant à la réalité. Grâce aux rétroactions données par le robot, cette méthode d'apprentissage par observation est complétée par un apprentissage par l'erreur, soit par simple mauvaise utilisation des fonctions, soit lorsqu'une erreur de codage provoque un fonctionnement inadéquat du robot : l'erreur doit être identifiée, analysée puis corrigée.

Lego Mindstorms Dans [1], Brandt et Colton mettent en avant les Lego Mindstorms comme robots éducatifs grâce à plusieurs bonnes caractéristiques. *Polyvalence* - les bricks Lego permettent aux élèves de concevoir, construire et tester facilement leurs programmes sans demander de talents d'assemblage ; *attirance pour les élèves* - les élèves ont une satisfaction immédiate dès qu'ils et elles arrivent à construire un robot avec succès ; *caractéristiques riches* - le kit de Lego Mindstorms contient tous les composants nécessaires pour construire une grande variété de robots.

Eléments de comparaison Dans [3], Chevalier et al. comparent les caractéristiques clés de 5 robots éducatifs : mbot, BeeBot, Finch, Thymio et EV3 (ex-LEGO Mindstorms). Sur la base d'un questionnaire rempli par 43 professeur.es, les auteurs - parties prenantes du projet Thymio à l'EPFL Lausanne - placent Thymio en tête pour l'offre capteurs-actuateurs, le temps d'apprentissage, les interactions d'entrée-sortie, la richesse de l'environnement de programmation. Les robots mbot - que nous avons choisi - ont le prix le plus bas et le meilleur look, une offre correcte en capteurs-actuateurs, mais sont évalués pauvrement pour les autres caractéristiques. Les professeur.es constatent l'intérêt de programmer un robot pour les mathématiques et les sciences, sans donner plus de détails.

2.3 Un cadre théorique

Ravanis propose un cadre théorique pour la construction du monde physique des 5-7 ans [7], articulés sur des concepts que nous reprenons pour les 7-11 ans.

Représentations. Les enfants se construisent dès la naissance et peuvent élaborer tout seuls des représentations scientifiques. Ces représentations sont généralement élaborées à partir des apparences plutôt qu'à partir de raisonnements logiques. Harlen [4] écrit : « Les idées des enfants tendent à différer notablement des idées scientifiques acceptées, et elles influencent fortement la signification que les enfants accordent aux activités qu'ils entreprennent. » Il s'agit donc de découvrir quels sont les idées des enfants, et de chercher à les modifier progressivement vers des représentations et des formes mentales explicatives.

Objectifs-obstacles. Selon Martinand, l'idée de la formulation des objectifs-obstacles est basée sur deux hypothèses fondamentales. La première est qu'il « est possible de trouver un nombre limité de progrès décisifs, non acquis spontanément mais qui ont une signification du point de vue de la pensée scientifique ou technologique, des attitudes et des capacités correspondantes » [6]. La seconde hypothèse est que dans une activité, il existe à un moment donné du parcours éducatif « un obstacle décisif, dont l'aspect dominant se situe dans une des grandes catégories d'objectifs, attitudes méthodes, connaissances, langages, savoir-faire » [6]. L'enjeu des leçons est donc le franchissement des obstacles.

Modèles-précurseurs. Le concept du modèle précurseur, proposé par Weil-Barais et Lemeignan [5], peut être fructueux pour le travail sur le progrès cognitif des enfants : « Le qualificatif précurseur associé au mot modèle signifie qu'il s'agit de modèles préparant l'élaboration d'autres modèles. En conséquence, les modèles précurseurs comportent un certain nombre d'éléments caractéristiques des modèles savants vers lesquels ils tendent. » Ravanis propose d'utiliser le

modèle précurseur comme cadre de référence d'éléments stables et articulés au niveau cognitif, qui constituent un système relationnel et qui pourraient favoriser plus tard l'élaboration et la maîtrise des modèles scientifiques [7].

3 Les concepts des 6 leçons de robotique

L'attrait des élèves pour des séances de robotique est indéniable mais il faut se concentrer sur ce que Harlen appelle la "dimension questionnante" c'est-à-dire la conception que se font les enfants de l'attente du professeur.e [4]. Harlen postule que bien des questions que se posent les enfants ne sont pas des points de départ efficaces pour la science et que ce sont les questions de l'enseignant.e qui déclenchent l'activité scientifique. Il s'agit donc de travailler à partir de questions *fécondes* et nous avons donc reconstruit les leçons de robotique dans cet esprit. Toutes les leçons sont sous licence libre CC-BY-SA à l'url http ://lesfillesqui.org/

3.1 Mouvements

L'objectif de la séance est d'effectuer des séquences et des répétitions de déplacements en distance ou angulaire ainsi que d'appréhender la notion de mouvement avec un ou deux moteurs, un préalable aux concepts de force.

Symétrie. Dans l'exercice *Promenade*, les enfants doivent modifier un programme pour que le robot revienne d'où il est parti en marche arrière. L'objectif-obstacle est d'utiliser des opérations de symétrie : avancer doit être remplacé par reculer, tourner à gauche par tourner à droite et inversement. Peu d'enfants y arrivent sans aide. Un modèle précurseur possible serait de regarder le déplacement du robot dans un miroir et de faire énumérer les mouvements réalisés.

Déplacements angulaires. Les déplacements angulaires du robot se font en paramétrant un temps de rotation, et ce temps varie en fonction de l'usure des piles. Même si un enfant a trouvé expérimentalement le temps pour faire un tour complet et peut en déduire le temps pour un quart de tour, l'enfant n'a aucune garantie que ce soit le même temps pour un autre robot, ni même pour le sien à un autre moment. Deux objectifs-obstacles sont à franchir : penser systématiquement à l'étalonnage du temps d'un tour ; établir la relation entre la rotation demandée et le temps nécessaire. Comme modèle-précurseur possible, nous présentons dans la section 3.6 un tableau de proportionnalité.

3.2 LED

L'objectif de la séance est de découvrir les LED, d'expérimenter la synthèse des et de programmer des petites applications lumineuses et sonores.

Synthèse de couleurs. Le robot a 2 diodes électroluminescentes appelées LED (*light-emitting diode*). Chaque LED est composée d'une diode Rouge, d'une diode Verte et d'une diode Bleue. Dans l'exercice *La couleur orange*, les enfants doivent utiliser le bloc qui permet de régler l'intensité des 3 diodes et déterminer comment produire les couleurs orange, jaune et blanc, une introduction à

la synthèse additive. Dans l'exercice *Palette de couleurs*, la palette de la synthèse additive est présentée puis l'enfant doit écrire un programme qui allume successivement les couleurs de l'arc-en-ciel, les intensités RVB de chacune des 6 couleurs étant données dans un tableau. Certains enfants font bien la relation entre la palette, les intensités RVB et la couleur attendue ; d'autres ne la font pas. La synthèse soustractive est la synthèse apprise à l'école, nous ne savons pas si cette représentation entre en collision avec celle de la synthèse soustractive qu'il faut comprendre (ou tout au moins appliquer) ou bien si la nature abstraite des nombres associés à l'intensité RVB est un objectif-obstacle à la compréhension.

Gérer deux actuateurs simultanément. Dans les exercices *Un feu tricolore* et *Un feu tricolore et sonore*, on emploie une boucle infinie. La difficulté est de comprendre comment Scratch temporise les actions. C'est assez simple dans le premier exercice où la séquence répétée indéfiniment est : allumer le vert - temporiser, allumer l'orange - temporiser, allumer le rouge - temporiser. Dans le deuxième exercice (un feu pour mal-voyant.es), le robot doit jouer une séquence de sons plus ou moins rapide pendant l'allumage du vert et du orange, rien pendant le rouge. Beaucoup d'enfants n'y arrivent pas sans aide et ne comprennent pas qu'on peut remplacer la temporisation par une suite d'actions dont la somme des durées a la même durée que la durée prévue pour la couleur. Les enfants n'ont pas de représentation du temps qui leur permet de franchir l'objectif-obstacle qu'est le parallélisme d'exécution des actuateurs. Lorsque cette notion de temporisation a été introduite par les accompagnatrices, peu d'enfants sont capables de calculer combien il faut de notes d'une certaine durée pour obtenir la durée totale allouée à la couleur. La démarche par essai-erreur ne fonctionne pas bien ici car l'observation fait appel à deux sens simultanés. On envisage l'utilisation d'une frise temporelle avec plusieurs couloirs comme modèle-précurseur.

Programmation continue d'un arc-en-ciel Dans l'exercice *Un arc-en-ciel*, il faut utiliser trois variables `ROUGE`, `VERT`, `BLEU` et incrémenter et/ou décrémenter les variables pour passer d'une couleur à l'autre. On donne à expérimenter un programme qui passe du rouge au jaune en incrémentant le `VERT`. L'application demandée est de passer du jaune au vert en décrémentant le `ROUGE`, la moitié des enfants n'y arrivent pas sans aide. On donne ensuite à expérimenter un programme qui passe du vert au bleu en décrémentant le `VERT` et en incrémentant le `BLEU`. L'application demandée est de passer du bleu au rouge sans autre indication (il faut décrémenter le `BLEU` et incrémenter le `ROUGE`), très peu d'enfants y arrivent sans aide. Nous ne savons pas s'il s'agit de difficultés venant de la programmation (incrémenter et décrémenter des variables dans une boucle répéter jusqu'à) ou de difficultés de représentation d'un continuum de couleurs.

3.3 En musique

L'objectif de la séance est de comprendre les notions de hauteur et de durée des notes, de découvrir que certains sons (les ultrasons) ne s'entendent pas, de comprendre la relation entre le son réfléchi et la distance et de programmer des petites applications utilisant le capteur ultrason et produisant des sons.

Durée et vitesse. Dans l'exercice *Au clair de la lune*, les enfants programment une suite de blocs jouant la mélodie. On leur demande ensuite de jouer l'air deux fois plus vite ou deux fois plus lentement, à quoi elles et ils arrivent par essai-erreur. Peu d'enfants font la relation entre la vitesse de la mélodie et la durée des notes. Généralement les enfants multiplient la durée (au lieu de la diviser) pour aller plus vite et la divisent (au lieu de la multiplier) pour aller plus lentement. La proportionnalité inverse est une notion de cycle 4 mais par l'expérimentation, les élèves l'appréhendent. Un modèle-précurseur serait de leur faire construire un tableau à deux lignes avec la vitesse qui croît sur une ligne et le temps sous forme de fraction qui décroit sur l'autre ligne.

Capteur ultrasons. Le "look" du robot mbot fait penser aux enfants que les deux cylindres en face avant sont ses "yeux". Cependant, les enfants connaissent assez bien la manière dont les chauve-souris se repèrent et il est facile de donner une représentation du capteur ultrason par analogie. Bien que le capteur mesure un temps d'aller-retour du son, le bloc mettant en œuvre ce capteur le présente comme une distance *distance mesurée sur le capteur ultrasons*. Comme cette distance est proportionnelle au temps, les enfants ne sont pas surpris par la mesure. Dans l'exercice *Le capteur ultrasons*, le robot doit avancer jusqu'à être près d'un obstacle et l'obstacle provient plutôt du concept de boucle (répéter jusqu'à) que de compréhension du fonctionnement du capteur. Dans l'exercice *Éviter les obstacles*, on introduit pour la première fois la notion d'algorithme de comportement du robot qui doit tourner quand un obstacle est trop près. La plupart des enfants arrivent à traduire l'algorithme en blocs Scratch sans aide.

```
Répéter indéfiniment
  Si distance mesurée par le capteur ultrasons du port 3 < 10 alors
    Tourner à gauche (ou à droite)
  Sinon
    Avancer
Fin répéter indéfiniment
```

Le thérémine est un instrument composé d'un boîtier électronique équipé de deux antennes. La main droite commande la hauteur de la note, en faisant varier sa distance à l'antenne verticale. Dans l'exercice *Programmer le robot comme théremine*, on simule l'antenne verticale avec le capteur ultrasons. L'obstacle vient de la programmation avec des si alors sinon imbriqués pour détecter dans quelle plage de distance est située la main qui arrête l'ultrason. Malgré l'exemple des deux premières notes, la programmation imbriquée est trop difficile. Le modèle envisagé est un organigramme et la simulation pas-à-pas du fonctionnement.

3.4 Capteurs

L'objectif de la séance est d'introduire un nouveau capteur qui mesure l'intensité d'un son, de comprendre la notion de mesure et de seuil, d'appréhender la différence entre fréquence et intensité d'un son et de programmer une petite application qui enregistre une suite de mesures d'intensité de sons (cet exercice s'est révélé trop ambitieux).

S'approprier des outils et des méthodes. L'emploi d'un capteur pour résoudre un problème travaille la compétence "s'approprier des outils et des méthodes". On donne la documentation du capteur de son dans un tableau (cf. table 1) et on demande de faire "dire" au robot si le son est calme ou bruyant. Sans aide, les enfants ont beaucoup de difficultés de passer du tableau à un programme. L'objectif-obstacle est l'emploi de la conditionnelle ; la représentation algorithmique ne convient pas, il faudrait un organigramme. Dans la même veine, on donne les fréquences des notes de la gamme dans un tableau et on demande aux enfants d'utiliser une variable `FREQUENCE` dans un programme qui joue le début de « Au clair de la lune » (Do – Do – Do – Ré – Mi – Ré) et sans aide, le passage du tableau à la suite de notes est difficile.

Table 1. Documentation du capteur de son et du capteur de lumière

Son ambiant	Valeur mesurée		Lumière ambiante	Luminosité - Valeur mesurée
Calme	De 0 à 499		Sombre	De 0 à 499
Bruyant	500 et plus		Clair	500 et plus

Caractéristiques du son. Dans l'exercice *Un robot musicien*, l'enfant mesure avec le capteur de son les différentes notes d'une gamme montante et doit répondre s'il est vrai que l'intensité est plus forte quand la note est plus haute ou bien que l'intensité est plus forte quand la note est plus basse. Les enfants ont généralement du mal à se représenter qu'il n'y a pas de rapport entre la fréquence du son (de grave à aigu) et l'intensité du son (de calme à bruyant). En cycle 3, il est préconisé de présenter le son comme une vibration. En représentant le son comme une vague, on dispose d'un modèle-précurseur où on peut donner du sens aux mots. Le nombre de vagues par seconde est la fréquence des vagues, un son grave a peu de vagues, un son aigu a beaucoup de vagues. La taille des vagues est l'amplitude, l'intensité, le volume sonore, un son fort a une grande taille de vague, un son faible a une petite taille de vague. Ce sont des notions fragiles car il y a beaucoup de mots voisins (amplitude, intensité, volume) qui perturbent la mémorisation des concepts, il faut donc faire attention à la cohérence du vocabulaire employé pour les différents capteurs.

3.5 Lumière

L'objectif de la séance est d'introduire un nouveau capteur qui mesure l'intensité de la lumière, de retravailler la notion de mesure et de seuil, et d'utiliser un tableau de LED pour programmer de petites animations.

Vocabulaire scientifique. Le capteur de son utilise le bloc *son mesuré sur le port x* pour obtenir l'intensité du son alors que la notion équivalente avec le capteur de lumière utilise le bloc *luminosité mesurée sur le port x*. La science emploie des termes (intensité, volume, amplitude, luminosité, etc.) reliés entre eux par des relations de synonymie, de hiérarchie et d'association. Établir une carte de concepts permet d'évaluer comment la classe se représente ces termes.

Fonctionnement "générique" d'un capteur. On donne la documentation du capteur de lumière dans un tableau similaire à celui du capteur de son (cf. table 1) et on demande de faire "dire" au robot si la luminosité ambiante est claire ou sombre. Certains enfants font la relation avec le programme du capteur de son et réutilisent ce programme sans aide. Se représenter que la lumière est une vague (une onde) comparable au son permettrait d'établir des parallèles fructueux entre l'amplitude, l'intensité d'un son et la luminosité. De même la fréquence d'une note pourrait être fructueusement comparée à la couleur.

Figure 1. Programme Scratch de trajectoire polygonale

3.6 Trajectoire

L'objectif de la séance est de comprendre la relation entre la vitesse des mouvements du robot et le temps, de programmer le robot pour qu'il effectue des trajectoires polygonales ou en suivant une ligne ou le modèle d'une lettre.

Table 2. Tableau à remplir sur les trajectoires polygonales

Nombre de côtés	3	4	5	6	8
Nom	Triangle	Carré	Pentagone	Hexagone	Octogone
Angle intérieur	60	90	108	120	135
Angle extérieur	120	90	72	60	45

Trajectoire, vitesse et temps. Les enfants doivent programmer un tour complet du robot et donner le temps nécessaire. Dans l'ensemble, les enfants ont beaucoup de patience pour arriver à obtenir le temps d'un tour. Un exemple de programme exploitant le temps d'un tour dans une variable leur est donné pour que le robot fasse un angle de 60 degrés puis on leur demande que le robot suive une trajectoire en triangle équilatéral. On explore ensuite les différentes polygones avec le programme de la figure 1. L'objectif-obstacle est de comprendre la relation entre l'angle intérieur de la figure et l'angle extérieur du déplacement du robot. Peu d'enfants sont capables de faire la relation entre le nombre de côtés et la valeur affectée par l'expression `360/réponse`. Cependant, en expérimentant le programme de la figure 1, la plupart des enfants arrivent à remplir correctement le tableau demandé (cf. tableau 2).

Suivi de ligne et tracé de lettres. Le robot mbot dispose d'un mode pré-programmé de suivi de lignes. Ce programme utilise le capteur appelé suiveur de ligne qui est composé de deux capteurs parallèles de couleur. Ces deux capteurs permettent de déterminer si le robot est sur une ligne noire, sorti de la ligne à gauche, sorti de la ligne à droite ou sur du blanc ; la mesure de cette situation s'obtient avec le bloc *état du suiveur de ligne* et peut aussi être vérifiée visuellement grâce à deux petites LED qui s'allument sur du blanc et s'éteignent sur du noir. L'objectif est que les enfants arrivent à programmer le même comportement que le programme pré-programmé : le suivi d'une ligne noire en forme de 8. La programmation se fait par étapes inductives ; (i) observer les quatre situations, l'état des petites LED et la valeur renvoyée par le bloc ; (ii) comprendre sur un dessin le comportement attendu du robot dans chaque situation (cf. figure 2) ; (iii) observer le mode pré-programmé du robot mbot ; (iv) remplir le tableau représentant l'algorithme du robot (cf. figure 3) ; (v) compléter l'ossature du programme de suivi de lignes.

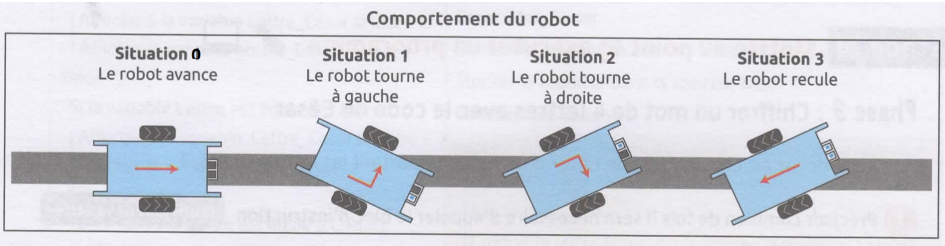


Figure 2. Actions à effectuer par le robot dans chaque situation

Dans une première version de cet exercice, on présentait un algorithme plus compliqué de suivi de ligne que les enfants devaient traduire en Scratch. L'algorithme étant présenté de manière déductive et descendante, il s'est avéré que l'algorithme restait abstrait et la traduction constituait un objectif-obstacle hors de portée des enfants. L'objectif-obstacle a été remédié grâce au schéma inductif exposé plus haut, et dans l'ensemble, les enfants viennent à bout de ce travail avec un besoin d'aide très variable.

Dans l'exercice où on demande que le robot trace une lettre, chaque segment de la lettre est numéroté et on demande aux enfants de compléter (avant de programmer) un programme "à trous", ce qui fonctionne bien en général.

Gauche et droit sur noir	Gauche sur noir et droit sur blanc	Gauche sur blanc et droit sur noir	Gauche et droit sur blanc
0	1	2	3
si SuiveurLigne = alors à la vitesse	si SuiveurLigne = alors à la vitesse	si SuiveurLigne = alors à la vitesse	si SuiveurLigne = alors à la vitesse

Figure 3. Actions à effectuer par le robot dans chaque situation

3.7 Synthèse

Les leçons ont été conçues, réalisées et testées au fil de l'eau en mai-juin 2019. La rédaction de cet article et les suggestions judicieuses des relecteurs et relectrices placent les exercices dans le cadre théorique présenté en section 2.3.

Un.e professeur. des écoles expérimenté.e, même sans connaissances en didactique de l'informatique, identifie rapidement les objectifs-obstacles d'une leçon et les classe facilement en difficulté. Cependant, à notre avis, ce n'est pas la difficulté de l'objectif-obstacle qui impacte le plus sur la capacité de réussite des élèves mais c'est plutôt la pertinence et l'ergonomie des modèles-précurseurs proposés, s'ils existent, qui conditionnera le succès. Par exemple, après une activité guidée d'étalonnage du robot et la construction d'un tableau de proportionnalité entre les rotations angulaires et le temps nécessaire, les enfants arrivent à programmer une trajectoire polygonale du robot. Par contre, comme nous n'avons pas su fournir un modèle pour représenter l'activité lumineuse et l'activité sonore en parallèle, la plupart des enfants n'ont pas su passer d'un programme de feu tricolore à un programme de feu tricolore et sonore.

4 Évaluations

Nous avons voulu évaluer la capacité des élèves à se représenter le comportement des robots suivant un programme donné ainsi que les connaissances de sciences acquises dans les leçons de robotique. Les résultats donnés dans cette section sont ceux de tests réalisés sans l'aide du robot, donc conceptuels.

4.1 Première évaluation : mouvements

Cette évaluation est composée de 4 exercices composé d'une situation et de questions. L'enfant peut répondre vrai, faux ou qu'elle ou il ne peut pas décider. Dans l'exercice 1, le robot est représenté vu de côté et on indique que les roues tournent dans le même sens et à la même vitesse. L'enfant doit répondre si le robot avance, recule, tourne. Dans l'exercice 2, le robot est représenté vu de côté et on indique qu'une seule roue tourne et dans quel sens. L'enfant doit répondre si le robot tourne, recule, tourne vers nous. Dans l'exercice 3, le robot est représenté vu de dessus et on indique que les roues tournent dans le même sens et à la même vitesse. L'enfant doit répondre si le robot avance, recule, tourne à gauche, tourne à droite. Dans l'exercice 4, le robot est représenté vu de dessus et on indique que les roues tournent dans des sens inverses et à des vitesses inconnues. L'enfant doit répondre si le robot avance, recule, tourne à gauche, tourne à droite, tourne autour d'une de ses roues.

La première évaluation a été réalisée entre les leçons 1 - Mouvement et 2 - LED dans deux classes, une classe de CM1 à Guipavas en zone rurbaine (14 filles et 14 garçons), et l'autre classe de CM2 à Loperhet en zone rurale (12 filles et 13 garçons). Les résultats des tests pour la classe entière sont donnés dans la table 3 et la ventilation filles/garçons dans la table 4.

Table 3. Tests sur les mouvements : classe

Ecole / Exercice	1	2	3	4	Total
Guipavas (28 CM1)	65%	57%	61%	45%	55%
Loperhet (25 CM2)	80%	75%	85%	38%	59%

Les résultats sont très satisfaisants. Le modèle-précurseur de forces, sur lequel les enfants ont raisonné, consiste en un robot vu de côté ou de dessus et des flèches de sens de rotation des roues. Avec ce modèle, les enfants anticipent correctement le déplacement du robot. L'exercice 4 est beaucoup moins bien réussi car les vitesses sont inconnues et l'élève ne dispose pas d'une représentation des forces permettant de raisonner. Ls tests ont été réalisés en toute fin d'année scolaire, et nous supposons que la plus grande réussite de la classe de CM2 est le reflet de la maturité d'enfants qui s'apprêtent à quitter l'école élémentaire.

Table 4. Tests sur les mouvements : filles / garçons

Ecole / Ex.	1		2		3		4		Total	
	F	G	F	G	F	G	F	G	F	G
Guipavas	69%	62%	51%	54%	52%	73%	38%	45%	51%	57%
Loperhet	82%	77%	73%	77%	85%	75%	38%	36%	66%	63%

D'après le tableau ci-dessus, en CM2 les filles ont mieux réussi que les garçons et en CM1 c'est l'inverse. Il n'a pas été constaté, lors des leçons, de différences significatives selon le genre, nous imputons le décalage à la procédure de test.

4.2 Deuxième évaluation : LED et son

Cette évaluation est composée de 4 exercices composé d'une situation et de questions. L'enfant peut répondre vrai ou faux. Dans l'exercice 1, les palettes des deux systèmes de synthèse (additive et soustractive) sont présentées. L'enfant doit répondre à des questions de compréhension sur ces systèmes. Dans l'exercice 2, le spectre des ondes électromagnétiques est donné. L'enfant doit répondre à des questions sur l'effet du bloc *régler la LED de la carte en composantes RVB* et à des questions de compréhension sur la lumière. Dans l'exercice 3, l'enfant doit répondre à des questions comparant les sons produits (hauteur et durée) par deux blocs *jouer la note en fréquence et en durée*. Dans l'exercice 4, on donne 4 programmes utilisant soit le capteur de son, soit le capteur ultrason pour afficher un seuil ou une distance. L'enfant doit répondre si les programmes sont corrects.

La deuxième évaluation a été réalisée après les leçons 2 - LED et 3 - En musique et avant la leçon 4 - Capteurs. Faute de temps, cette évaluation n'a pas été réalisée à Loperhet. Les résultats des tests pour la classe de Guipavas sont donnés dans la table 5 et la ventilation fille/garçons dans la table 6.

Les résultats sont aussi très satisfaisants. Les blocs Scratch *régler la LED de la carte en (composantes RVB)* et *jouer la note (hauteur) et (durée)*, qui sont des

Table 5. Tests sur les LED et son : classe

Ecole / Exercice	1	2	3	4	Total
Guipavas (28 CM1)	71%	61%	76%	38%	62%

Table 6. Tests sur les LED et son : filles / garçons

Ecole / Ex.	1		2		3		4		Total	
	F	G	F	G	F	G	F	G	F	G
Guipavas	64%	79%	55%	71%	59%	93%	41%	51%	53%	72%

abstractions du comportement lumineux ou sonore du robot, sont bien assimilés par les enfants. L'exercice 4 utilise deux capteurs différents et deux types de mesure ; c'est donc un exercice compliqué qui a eu moins de succès.

5 Conclusion et perspectives

Pour appuyer l'enseignement des sciences, un cours de robotique avec Scratch, en six leçons, a été conçu, développé et évalué dans deux classes de cycle 3. Les séances ont eu un grand succès. Cependant, comme le remarque Harlen, les questions que se posent les enfants ne sont pas des points de départ efficaces pour la science [4] et il s'agit donc de provoquer et de travailler avec des représentations *fécondes* et d'identifier les objectifs-obstacles à franchir. Ceci permet de construire ou d'utiliser des modèles-précurseurs fournissant des pistes de résolution de problèmes à la portée des enfants.

Nous avons constaté que certains enfants, identifié.es en difficulté par leur professeure des écoles, pouvaient réussir seul.e ou avec l'aide de camarades. La programmation des robots est une activité où il n'y a pas de bonnes ou mauvaises réponses immédiates ; où on expérimente, on peut se tromper, on tâtonne et on y arrive. Cette gratification rapide, motivante pour les enfants, doit être inscrite dans des tentatives à plus long terme, un effort qui peut être devenu hors de portée de certains enfants.

En partenariat avec l'école Lucien Guilbault à Montréal (école pour enfants avec des grandes difficultés d'apprentissage) et quelques classes de Brest et ses environs dont une unité localisée pour l'inclusion scolaire (ULIS), nous développons un dispositif de Robolympiades à partir des exercices bien réussis par tous les enfants. Au cours de cette année scolaire, les élèves s'entrainent à maitriser des "figures" comme *avancer le robot d'un mètre* ou *passer du vert à l'orange* ou encore *détecter un obstacle à moins de 10 cm*. Pendant les entrainements, animés à Montréal par la professeure d'informatique de l'école Lucien Guilbault et en France par les filles qui..., les enfants ont l'objectif de réussir les figures afin d'être qualifié.es pour les Robolympiades. La Robolympiade aura lieu en fin d'année scolaire sous la forme d'une compétition entre équipes de deux enfants. Les équipes devront réaliser des figures imposées qui seront notées par un jury. Une cérémonie de remise des prix couronnera les efforts et les réussites des enfants. Nous espérons vous raconter la suite dans un prochain Didapro !

Références

1. Brandt, A.M., Colton, M.B. : Toys in the classroom : Lego mindstorms as an educational haptics platform. In : 2008 symposium on haptic interfaces for virtual environment and teleoperator systems. pp. 389–395. IEEE (2008)
2. Brennan, K., Resnick, M. : New frameworks for studying and assessing the development of computational thinking. In : Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada. pp. 1–25 (2012)
3. Chevalier, M., Riedo, F., Mondada, F. : Pedagogical uses of thymio ii : How do teachers perceive educational robots in formal education ? IEEE Robotics & Automation Magazine **23**(2), 16–23 (2016)
4. Harlen, W., Elstgeest, J., Jelly, S. : Enseigner les sciences : comment faire ? Le Pommier, Paris (2004)
5. Lemeignan, G., Weil-Barais, A. : Construire des concepts en physique : l'enseignement de la mécanique. Hachette, Paris (1993)
6. Martinand, J.L. : Connaître et transformer la matière. Peter Lang, Berne (1986)
7. Ravanis, K. : Représentations, modèles précurseurs, objectifs-obstacles et médiation-tutelle : concepts-clés pour la construction des connaissances du monde physique à l'âge de 5-7 ans. Revista Electrónica de Investigación en Educación en Ciencias **5**(2) (2010)

TurtleTablet : un jeu collaboratif et tangible sur tablette pour l'initiation à la programmation

Iza Marfisi-Schottman, Sébastien George et Marc Leconte

Le Mans Université, LIUM - EA 4023, 72085 Le Mans, France
{Iza.Marfisi, Sebastien.George, Marc.Leconte}@univ-lemans.fr

Abstract. Les jeux numériques sont souvent utilisés au collège pour initier les élèves aux bases de la programmation informatique. Par ailleurs, le travail en groupe tient une place très importante dans l'apprentissage des sciences et ces jeux sur ordinateur sont couramment effectués en groupe de deux élèves. Cependant, les interactions clavier-souris ou tactile, et la conception même de ces jeux freinent la collaboration. Le plus souvent, seul un élève est engagé dans l'activité à un moment donné. Pour favoriser une réelle pédagogie s'appuyant sur la collaboration, nous proposons *TurtleTablet*, un jeu sur tablette pour aider les élèves à comprendre comment se déroule l'exécution d'un algorithme. Les activités de ce jeu sont conçues pour un binôme d'élèves, de façon à ce que chacun ait un rôle actif. De plus, les élèves doivent manipuler des objets tangibles pour interagir avec la tablette. Ce mode d'interaction, déjà éprouvé sur tables interactives, est inédit sur tablette.

Keywords : informatique, programmation, algorithme, objet tangible, collaboration, serious games

1 Jeu numérique collaboratif pour s'initier aux concepts de la programmation

Depuis septembre 2016, les principes de la programmation informatique sont inscrits au programme des collèges en France. Le CAPES "Numérique et sciences informatiques" venant tout juste d'être mis en place en 2019, les enseignants qui donnent actuellement les cours d'informatique sont, le plus souvent, non-spécialistes. Pour les aider, il existe de nombreux blogs d'enseignants ou d'associations [1][2] qui proposent des fiches d'activités à réaliser, notamment des animations débranchées, qui peuvent se faire sans ordinateur ni matériel spécifique [3]. On y trouve par exemple le jeu collaboratif du *robot idiot* qui se joue à trois : un élève construit un algorithme avec des cartes « avancer », « tourner-droite » et « tourner-gauche », un autre l'exécute en pointant la carte en cours et le dernier élève effectue les déplacements sur un plateau ou un sol carrelé.

À l'image de ce jeu, toutes les activités débranchées sont faites en équipe de deux ou trois élèves et donnent une part importante à la **collaboration**. En effet, comme il

est inscrit dans le décret du 31 mars 2015, définissant le socle commun de connaissance, de compétences et de culture : « l'élève travaille en équipe, partage des tâches, s'engage dans un dialogue constructif, accepte la contradiction tout en défendant son point de vue, fait preuve de diplomatie, négocie et recherche un consensus » [4]. Cette démarche collaborative accroit la motivation et améliore les résultats des élèves de façon significative [5]. De plus, la collaboration est également inscrite dans le programme officiel du collègue dans la section sciences et technologie [6].

Les enseignants utilisent également des **outils numériques** gratuits pour que les élèves apprennent à programmer à partir de dessins [7], d'organigrammes [8], de blocs [9] ou même directement dans un vrai langage [10]. Ces outils reposent, en grande partie, sur des activités ludiques dont le but est d'écrire un programme pour déplacer une entité (un robot, un personnage ou un animal) dans un environnement. Ces outils numériques ont l'avantage de corriger automatiquement les élèves et de leur proposer plusieurs niveaux de difficulté, ce qui permet de rendre les élèves autonomes et de libérer du temps pour les enseignants afin qu'ils puissent aider les groupes en difficulté.

Cependant, les activités et les interactions clavier-souris proposées par ces applications numériques ne favorisent pas la collaboration au sein d'un groupe. En effet, lorsque deux élèves sont assis devant l'ordinateur, les activités ne sont pas conçues pour donner un rôle à chacun et seul l'élève qui tient la souris ou le clavier est actif [11]. Le constat est le même pour les applications tactiles sur tablette : l'expérience que nous allons décrire dans la suite montre qu'au bout d'un certain temps, un seul des élèves va prendre le contrôle intégral de la tablette. **Peut-on garder les avantages du numérique tout en ayant des activités qui encouragent la collaboration entre élèves ?**

Dans cet article, nous présentons *TurtleTablet* : un jeu collaboratif sur tablette pour s'initier aux concepts de base de l'algorithmique. Le jeu se joue par groupe de deux élèves en autonomie, même si un enseignant peut superviser une classe pour inciter à la discussion et mieux ancrer les apprentissages. Dans les parties suivantes, nous expliquons comment la conception du jeu et l'utilisation d'objets tangibles ont été pensées pour favoriser la collaboration.

2 Conception du jeu collaboratif TurtleTablet

Comme nous l'avons indiqué dans l'introduction, les activités collaboratives peuvent améliorer l'appropriation de concepts, mais les applications numériques pour l'initiation à l'informatique ne supportent pas vraiment une pédagogie de l'apprentissage collaboratif. Dans la suite de cet article, nous présentons le principe du jeu *TurtleTablet* et la façon dont il a été conçu pour favoriser la collaboration.

2.1 Un jeu pour apprendre à exécuter un programme pas à pas

L'objectif pédagogique de *TurtleTablet* est d'apprendre à exécuter un programme pas à pas. Le principe, inspiré du jeu du robot idiot, est simple : exécuter un programme présenté (sur la gauche de l'interface de la fig. 1) en déplaçant les objets sur une grille

(à droite sur la fig. 1). Le fait de s'entrainer à exécuter un programme pas à pas, comme une machine, est complémentaire aux activités d'écriture d'algorithmes proposées par les autres outils numériques [12]. Dans *TurtleTablet*, le programme est écrit en pseudo langage et contient des instructions introduites selon une difficulté croissante : des instructions simples (*ex. déplacer obj_1 de 3 cases vers l'avant, tourner obj_2 de 45° vers la gauche*), des variables (*ex. y = 3*), des conditions (*ex. si x < 2 {…}*) et des boucles (*ex. pour i allant de 1 à 3 {…}*).

Si les élèves ne déplacent pas l'objet correctement, une animation indique que l'objet doit être déplacé à nouveau. *TurtleTablet* offre plusieurs types d'aides visuelles : l'instruction en cours est en blanc, celle exécutée avec une erreur est en rouge et celles qui ont été exécutées correctement sont en vert. Toutes ces aides sont optionnelles et peuvent être désactivées par l'enseignant, ce qui rend les niveaux beaucoup plus complexes, surtout ceux avec des boucles.

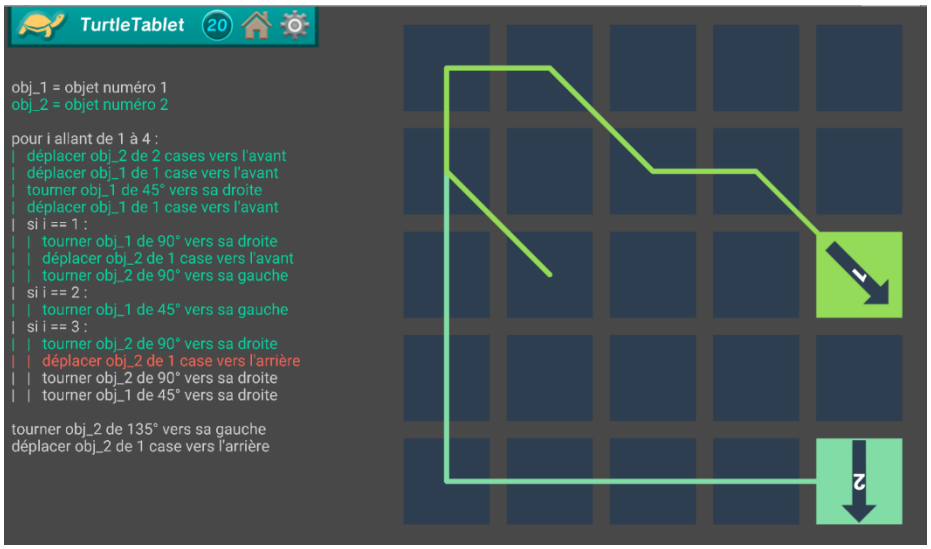


Fig. 1. Interface du niveau 20 de TurtleTablet

Chaque niveau possède un compteur d'erreurs. L'objectif est d'inciter les élèves à prendre le temps de réfléchir et à se concerter avant de déplacer les objets. Une fois que toutes les instructions sont réussies, les joueurs gagnent un quart de dessin qui correspond aux lignes tracées par les objets sur la grille (*eg.* le niveau 20 correspond à la tête d'un caméléon). Ils peuvent alors passer au niveau suivant. Quand les quatre quarts d'un dessin sont gagnés, le dessin complet apparait dans le menu (fig. 2). Ce principe de récompenses est un ressort ludique qui fonctionne bien avec un public jeune.

TurtleTablet est composé de 20 niveaux de jeu avec des difficultés progressives (fig. 2). Les notions d'algorithmiques (variables, boucles et conditions) sont introduites progressivement avec des fenêtres de tutoriel. Pour introduire ces concepts du programme officiel, nous nous sommes inspirés des tutoriels des jeux de *CODE.org*

[13], qui sont largement utilisés dans les écoles américaines notamment. La difficulté augmente également avec des codes de plus en plus longs et des concepts qui se combinent (comme dans le dernier niveau présenté sur la fig. 1).

Afin d'inciter les élèves à refaire les niveaux sans erreur, le menu affiche les niveaux qui peuvent être améliorés en orange ainsi de leur nombre d'erreurs. Le jeu affiche également des messages d'encouragement pour les inciter au sans-faute.

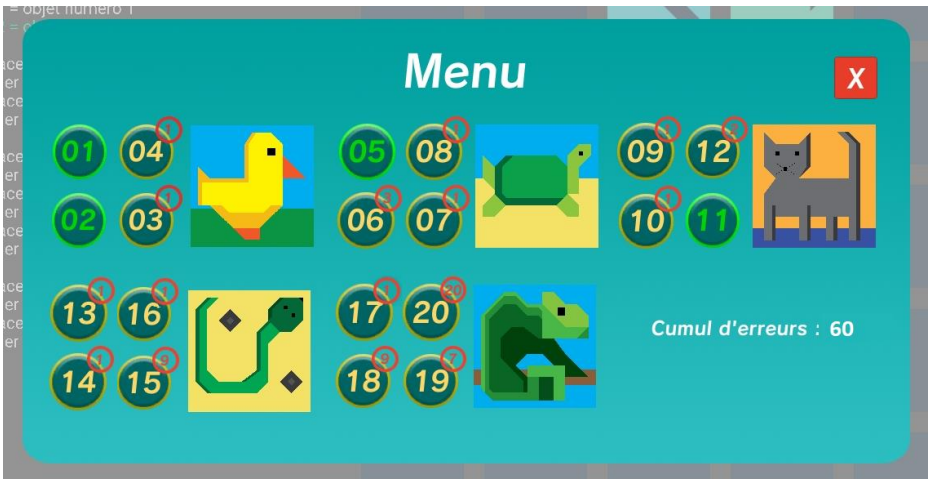


Fig. 2. Interface du menu avec les 20 niveaux finis

2.2 Un jeu collaboratif

TurtleTablet intègre des mécaniques de jeu qui favorisent la réflexion et la concertation entre les deux équipiers pour réussir les niveaux avec le moins d'erreurs possible. Tous les niveaux sont conçus pour que chaque élève ait un rôle à jouer en manipulant un objet particulier (objet_1 ou objet_2). Le niveau de difficulté et le type d'instruction sont répartis équitablement entre les deux joueurs. Pour finir, afin que les deux joueurs prennent le temps de comprendre chaque nouveau concept, les fenêtres de tutoriel se ferment uniquement quand les deux ont indiqué qu'ils sont prêts.

TurtleTablet est conçu pour favoriser la collaboration, mais cela ne suffit pas toujours à engager les deux élèves dans l'action. Dans la suite de cet article, nous présentons le mode d'interaction innovant proposé par le jeu pour renforcer l'engagement et la collaboration de la part des deux élèves.

3 Interactions avec des pièces tangibles

Les interactions classiques clavier-souris sont des freins à la collaboration entre les élèves. Plusieurs études ont montré qu'il est possible de favoriser la collaboration en proposant les interactions avec deux ou trois souris [11]. Les équipiers ont alors une souris chacun, mais contrôlent toujours le même curseur sur l'écran. Comme nous

allons voir dans la suite de cet article, la collaboration sur une tablette n'est pas non plus forcément facilitée ni incitée, car une personne peut être tentée de réaliser toutes les actions mêmes si des consignes sont données pour se répartir le travail. D'autres études en Interaction Homme Machine (IHM) ont prouvé que l'utilisation d'objets tangibles (i.e. objet 3D qui interagissent avec l'application) pouvait également faciliter la collaboration et l'appropriation de concepts, de par l'engagement corporel et l'invitation à l'action [14]-[18]. L'utilisation d'éléments physiques est particulièrement efficace pour les enfants qui ont l'habitude d'apprendre en manipulant des objets.

L'utilisation de ces pièces tangibles se fait habituellement sur une table avec un système de Réalité Augmentée par projection [19] ou en posant directement les pièces sur une table interactive. Ces surfaces offrent l'avantage d'être large et donc de faciliter le travail collaboratif en groupes [20].

Dans la section suivante, nous présentons *TurtleTable*, la première version du jeu développée pour table interactive avec des pièces tangibles ainsi que les résultats obtenus sur l'impact de ses interactions concernant la collaboration et la motivation des élèves. Cependant, l'achat et l'installation de système de projection en réalité augmenté (environs 1 500 euros pour le kit projecteur et camera de profondeur) ou d'une table interactive (environ 7 000 euros) ne sont pas abordables pour une diffusion large dans les établissements d'enseignement. Par contre, la plupart des collèges en France disposent déjà de tablettes. Nous expliquons alors comment nous avons conçu une version de l'application sur tablette en conservant le principe d'interaction par objets tangibles.

3.1 TurtleTable : objets tangibles sur table interactive

TurtleTable reprend le même principe de jeu que celui présenté précédemment, mais avec des grilles de 8 sur 8 cases (alors que celles de *turtletablet* font seulement 5 sur 5 cases). Cette différence de taille a eu un impact sur le *game design* puisque que les élèves effectuaient un dessin complet à la fin de chaque niveau alors qu'ils doivent compléter quatre niveaux dans *turtletablet* avant de voir un dessin complet. De plus, les niveaux non pas deux mais trois objets puisqu'une table interactive est assez grande pour accueillir aisément une équipe de trois élèves. Pour faire tourner et avancer les objets, les élèves doivent faire glisser une pièce tangible sur la surface de la table (fig. 3 en haut à gauche.). Nous avions mis à disposition trois pièces numérotées 1, 2 et 3 sur chaque table et les élèves ont naturellement choisi une pièce pour bouger l'objet virtuel portant le même numéro. Afin de mesurer l'impact de ce mode d'interaction sur l'apprentissage, nous avons développé trois autres modes d'interactions : un mode tactile sur table interactive, un mode tactile sur tablette et un mode clavier-souris sur ordinateur (fig. 3). Pour bouger les objets avec les interactions tactiles, les apprenants doivent cliquer dessus, pour le sélectionner, puis cliquer sur les cases devant ou derrière. La rotation des objets est faite avec deux doigts, comme un compas. Avec les interactions clavier-souris, la rotation de l'objet se fait avec les flèches directionnelles du clavier.

Afin de mesurer l'impact des interactions sur la motivation, la collaboration et l'apprentissage, nous avons expérimenter ses différences versions de *TurtleTable* avec une soixantaine de collégiens. Plusieurs types de données ont été collectés. Premièrement, nous avons conçu des pré-tests et des post-tests afin de mesurer le niveau de connaissances des élèves sur les concepts vus dans le jeu (variables, boucles, conditions). Ce test se compose de 10 exercices, qui sont similaires aux niveaux dans le jeu : l'objectif est d'exécuter le programme en dessinant le chemin parcouru par les objets sur la grille. Les groupes ont également été filmés afin d'analyser les interactions entre sujets. Certains groupes ont également été questionnés en *focus group* pour comprendre leurs interactions. Pour finir, les sujets ont également répondu à un questionnaire final dans lequel ont leur demandé de classer les versions par ordre de préférence et de les qualifier avec des mots clés.

Fig. 3. TurtleTable avec ses 4 modes d'interactions

L'expérimentation a été conduite sur 59 collégiens du collège La Salle (Laval, France), dont 30 filles et 29 garçons, de 14 à 16 ans. Ils sont venus au laboratoire, dans une salle d'innovation pédagogique équipée de caméra, pendant une demi-journée en décembre 2017. Seul cinq des sujets avaient déjà des connaissances en

programmations, puisqu'ils suivaient des cours pendant la pause de midi. Pour tous les autres, il s'agissait de leur première initiation à la programmation.

Les sujets ont été séparés en quatre groupes et assignés une version spécifique de TurtleTable. Les enseignants ont composé les groupes de façon à ce qu'ils soient équitables en termes de niveau. L'expérimentation été planifiée comme suit :

- 15 minutes pour répondre au pré-test
- 60 minutes pour jouer à *TurtleTable*, sur la version assignée, sans l'aide des enseignants
- 15 minutes pour répondre au post-test
- 20 minutes pour tester les autres versions de TurtleTable, avec l'aide des enseignants, s'ils le souhaitaient
- 20 minutes pour répondre au questionnaire final

Tout d'abord, le jeu semble avoir beaucoup plus aux élèves et aux enseignants. Presque tous les groupes ont réussi à finir les 20 niveaux de jeu en 60 minutes et étaient en compétition pour faire le moins d'erreurs possible en refaisant les niveaux. Les post-tests montrent qu'après le jeu, plus de la moitié des sujets maitrisaient les concepts des variables et des boucles. Étant donné que seul cinq des élèves avaient des connaissances en algorithmique, c'est un résultat qui semble satisfaisant pour un premier cours. Les résultats détaillés de cette expérimentation sont décrits dans un autre article [21] mais nous reprenons ici, les points importants.

Du point de vue de la **motivation**, 2/3 des élèves ont préféraient la version avec les objets tangibles, 1/5 la version sur tablette et les quelques élèves restant les versions sur ordinateur et Table avec les interactions tactiles. Ces résultats sont cohérents avec les mots qu'ils ont utilisés pour qualifier ces versions : la version avec les objets tangibles est qualité d'« amusante », « intéressante » alors que ces termes disparaissent progressivement, au profit de « difficile » et « ennuyeux » pour les autres versions.

D'un point de vue de **l'apprentissage**, les tests statistiques montrent qu'il n'y a pas de différences significatives entre les pré-tests et les post-tests des quatre groupes, même si, les sujets ayant joué sur tables interactives (avec les objets tangibles ou avec les interactions tactiles) ont déclaré avoir plus appris que les autres.

Pour finir, la différence la plus significative entre les versions se trouve dans la **collaboration**. En effet, les sujets sur tables interactives ont beaucoup plus interagi avec leurs coéquipiers. De plus, seuls les sujets ayant joué avec les objets tangibles ont utilisé les mots « convivial » et « travail d'équipe ». Le fait que les sujets aient un objet tangible assigné semble leur apporter beaucoup plus d'assurance et de légitimité pour participer à l'activité, même s'ils ne sont pas sûrs d'eux. Cette observation est cohérente avec une étude, dans laquelle les élèves étaient assignés soit le clavier, soit la souris lors d'un travail collaboratif sur ordinateur [22]. Ce contrôle individuel d'une entité semble favorable à l'engagement des sujets dans la tâche collaborative. Ce mode d'interaction semble également éviter qu'un élève ne fasse le niveau tout seul. Dans l'expérience réalisée, un seul élève a pris les pièces des mains de ses coéquipiers pour faire à leur place. Le comportement le plus courant observé est des échanges et des conseils apportés pour aider à réaliser le bon mouvement. Inversement, les élèves utilisant le mode tactile, sur table interactive ou sur tablette, ont plus eu tendance à

faire les manipulations à la place de leur coéquipier au lieu de leur expliquer. La collaboration s'est avérée être la pire chez les groupes utilisant le mode clavier-souris : dans le meilleur des cas, les élèves faisaient les niveaux chacun leur tour pendant que les autres regardaient ailleurs, y compris quand il fallait lire les tutoriels introduisant les nouveaux concepts algorithmiques.

Le mode d'interaction avec les pièces tangibles sur la table interactive montre donc des avantages en termes de motivation et de collaboration, mais les établissements scolaires n'ont pas les moyens d'acheter de telles tables. Afin de favoriser une diffusion large du jeu pour apprendre les bases de l'algorithmique, nous avons eu pour objectif de recréer les mêmes conditions d'interactions tangibles sur des dispositifs déjà existants dans les collègues : les tablettes.

3.2 TurtleTablet : objets tangibles sur tablette

L'utilisation de pièces tangibles sur tablette est très récente et concerne essentiellement des jeux vidéo ou des jeux de société qui proposent une extension numérique [23].

Fig. 4. Interactions avec des pièces tangibles sur tablette

D'un point de vue technique, les mouvements des pièces tangibles sont reconnus par la tablette grâce à la position de ses pieds. La pièce est recouverte d'une couche de peinture conductrice [24] qui conduit la charge électromagnétique, produite naturellement par le corps humain, vers l'écran capacitif de la tablette. Des petits patins en mousse conductrice sont également collés en dessous des pieds pour éviter de rayer l'écran. Seuls trois pieds sont utiles pour reconnaitre les mouvements de la pièce. Nous utilisons l'API fourni par *Volumique* [23] pour détecter la position de ces trois points de contact comme étant une pièce. Il est possible de créer des pièces avec des positionnements de pieds uniques de façon à ce que l'objet virtuel n°1 ne puisse être déplacé qu'avec la pièce n°1. Cependant, les premiers tests menés lors de la fête de la science 2019 montrent que l'inscription des numéros sur les pièces suffit à dissuader les joueurs d'utiliser la même pièce pour déplacer les deux objets virtuels (même si cela est possible). Les pièces que nous avons utilisées pour *TurtleTablet* (Fig. 4) sont imprimées avec une imprimante 3D, mais il est possible de fabriquer des pièces avec carton recouvert d'aluminium, qui possède les mêmes propriétés conductrices. Afin

de stabiliser les pièces, nous avons ajouté un quatrième pied sans le recouvrir de peinture conductrice.

L'application et le guide de création des pièces, à imprimer en 3D, ou à faire en carton, sont disponibles sur le site du projet [25].

4 Conclusion et perspectives

Les applications numériques offrent de nombreux avantages pour l'apprentissage de l'informatique : ils permettent aux élèves de travailler en autonomie en leur proposant des exercices adaptés à leurs niveaux. Comme la plupart des activités en sciences et technologies, les enseignants souhaitent que les élèves travaillent sur ces applications par petit groupe, afin de favoriser la collaboration, qui est une compétence clé du cursus scolaire. Cependant, les interactions proposées par un ordinateur ou une tablette ne facilitent pas la collaboration. Dans cet article, nous avons présenté *TurtleTablet*, une application numérique sur tablette, pour apprendre les bases de la programmation. Ce jeu, qui se joue par équipe de deux, permet de comprendre comment un programme s'exécute pas à pas, en introduisant les notions de variables, boucles et conditions. *TurtleTablet* vise à favoriser la collaboration de deux façons. Tout d'abord, les activités du jeu sont conçues de façon à impliquer les deux joueurs en leur demandant de bouger chacun un objet à tour de rôle. Ce jeu propose également une Interaction Home-Machine qui consiste à glisser des pièces tangibles sur l'écran de la tablette pour bouger les objets virtuels. Ce type d'interaction a déjà montré sa capacité à favoriser la collaboration des élèves sur une table interactive. Nous allons prochainement conduire des expérimentations pour évaluer *TurtleTablet* afin de mesurer si, malgré une taille d'écran plus petite, l'application tangible conserve des propriétés propices à une collaboration entre élèves.

Par la suite, nous envisageons également d'ajouter une interface d'édition à *TurtleTablet* pour que les enseignants ou les élèves puissent eux-mêmes créer de nouveaux niveaux. Il serait également intéressant d'afficher le programme sous différentes formes (organigrammes, blocs de type Scratch ou différents langages tels que python ou java) pour introduire la notion de syntaxe propre à chaque langage et proposer un plus grand éventail de niveau de difficulté.

References

1. Site Web de Pixees, Ressources pour les sciences numériques, https://pixees.fr/utiliser-une-ressource/, vu le 16/10/2019.
2. Site Web du Concours Castor, http://concours.castor-informatique.fr/, vu le 16/10/2019.
3. Calmet C., Hirtzig M., et Wilgenbus D., *1, 2, 3... codez ! enseigner l'informatique à l'école et au collège*, Le pommier, Paris, France, 358 p., (2016).
4. Journal officiel de la République française (2015), Décret n° 2015-372 relatif au socle commun de connaissances, de compétences et de culture, https://www.legifrance.gouv.fr/eli/decret/2015/3/31/2015-372/jo/texte

5. Johnson D., Murayama G., Jonson R.T., Nelson D. et Skon L., « Effects of cooperative, competitive, and individualistic goal structures on achievement: A meta-analysis », In: *Psychological Bulletin*, vol. 89, pp. 47–62, (1981).
6. Site officiel du programme du collège français, https://www.education.gouv.fr/cid81/les-programmes.html, vu le 16/10/2019.
7. Site Web de Lightbot, https://lightbot.com/, vu le 16/10/2019.
8. Site Web de RobotProg, http://www.physicsbox.com/demorobotprog.html, vu le 16/10/2019.
9. Sites Web de Scratch, https://scratch.mit.edu/, vu le 16/10/2019.
10. Site Web de la tortue-logo, http://tortue-logo.fr/fr/tortue-logo, vu le 16/10/2019.
11. Scott S. D., Shoemaker G. B. D., et Inkpen K. M., « Towards Seamless Support of Natural Collaborative Interactions », In: *Proceedings of Graphics Interface*, pp. 103–110, (2000).
12. George S., Marfisi-Schottman I., et Leconte M., « TurtleTable : apprendre les bases de la programmation avec des interfaces tangibles », In: *Atelier Apprentissage Instrumenté de l'Informatique, Orphee RDV*, Font-Romeu, France, (2017).
13. Site Web de CODE.org, https://code.org/learn, vu le 16/10/2019.
14. Schneider B., Jermann P., Zufferey G. et Dillenbourg P., « Benefits of a Tangible Interface for Collaborative Learning and Interaction », In: *IEEE Transactions on Learning Technologies*, vol. 4, nº 3, pp. 222–232, (2011).
15. Price S. et Pontual Falcão T., « Where the attention is: Discovery learning in novel tangible environments », In: *Interacting with Computers*, vol. 23, nº 5, pp. 499–512, (2011).
16. Kubicki S., Wolff M., Lepreux S., et Kolski C., « RFID interactive tabletop application with tangible objects: exploratory study to observe young children' behaviors », In: *Personal and Ubiquitous Computing*, vol. 19, nº 8, pp. 1259–1274, (2015).
17. Da costa J., Szilas N., et Müller A., « Réalité augmentée pour l'apprentissage conceptuel en sciences : quels principes de conception pour les EIAH ? Cas du dispositif DEAPE Learn en électromagnétisme », In: *Actes d'Environnements Informatiques pour l'Apprentissage Humain, EIAH*, Paris, France, pp. 181–192, (2019).
18. Palaigeorgiou G., Tsolopani X., Liakou S., et Lemonidis C., « Movable, Resizable and Dynamic Number Lines for Fraction Learning in a Mixed Reality Environment », In: *The Challenges of the Digital Transformation in Education*, pp. 118–129, (2019).
19. Laviole J. et Hachet M., « PapARt : interactive 3D graphics and multi-touch augmented paper for artistic creation », In: *Proceedings of the IEEE Symposium on 3D User Interfaces*, Costa Mesa, CA, United States, pp.3–6, (2012).
20. Shaer O. *et al.*, « The design, development, and deployment of a tabletop interface for collaborative exploration of genomic data », In: *Int. J. Human-Computer Studies*, vol. 70, nº 10, pp. 746–764, (2012).
21. Marfisi-Schottman I., George S., et Leconte M., « TurtleTable: Learn the Basics of Computer Algorithms with Tangible Interactions », In: *Proceedings of the Games and Learning Alliance Conference, GALA*, Palermo, Italy, pp. 291–300, (2018).
22. Angeli C., Tsaggari A., « Examining the effects of learning in dyads with computer-based multimedia on third-grade students' performance in history», In: *Computers & Education*, vol. 92-93, pp. 171-180, (2016).
23. Site Web de l'entreprise Volumique, https://volumique.com/, vu le 16/10/2019.
24. Page Web du produit *Conductive Paint*, vendu par Bare Conductive https://www.bareconductive.com/shop/electric-paint-50ml/, vu le 16/10/2019.
25. Page Web du projet TurtleTable, https://lium.univ-lemans.fr/en/turtletable/, vu le 16/10/2019.

Vers un modèle de débriefing : une étude de cas avec le jeu *Programming Game*

Maud Plumettaz-Sieber[1], Catherine Bonnat[2] et Eric Sanchez[3]

[1-2-3] Université de Fribourg, LIP/CERF, P.-A. de Faucigny 2, CH-1700 Fribourg, Suisse
maud.sieber@unifr.ch, catherine.bonnat@unifr.ch,
eric.sanchez@unifr.ch

Résumé. Notre recherche porte sur l'apprentissage de la programmation avec le jeu *Programming Game*. Dans ce papier, nous proposons un modèle de débriefing que nous mettons à l'épreuve pour analyser quand et comment les connaissances développées par les joueurs sont transformées en savoirs transférables. Cette recherche exploratoire concerne une étude de cas d'un enseignant d'informatique au secondaire qui a utilisé le jeu avec des novices en programmation, dans une classe de première année de niveau gymnasial (15-16 ans) à Fribourg, en Suisse. Pour répondre à nos questions, nous avons utilisé un logiciel d'annotation pour catégoriser les interactions entre les élèves et l'enseignant. Les résultats montrent que l'enseignant a mis en place une stratégie de débriefing qui ne prend pas en compte toutes les dimensions de notre modèle.

Mots clés : Débriefing/Institutionnalisation, Jeu numérique, Didactique de l'informatique.

1 Introduction

La phase de débriefing après une session de jeu est cruciale pour le transfert des savoirs issus de l'activité [1], mais elle demeure difficile à mettre en place et à réaliser pour les enseignants. Dans le cadre de notre recherche, nous cherchons à modéliser la phase de débriefing des savoirs issus d'une séquence de jeu *Programming Game*, un jeu numérique d'apprentissage des bases de la programmation [2]. L'utilisation du jeu *Programming Game* vise le développement et l'acquisition de savoirs en programmation. Pour ce jeu numérique, les élèves sont amenés à écrire des algorithmes et à développer une « pensée informatique » (ou *computational thinking)*. Le scénario pédagogique consiste pour l'enseignant à accompagner ses élèves dans l'utilisation du jeu, mais également à mettre en place une phase de débriefing sur les variables, séquences d'instructions et conditions pour permettre une généralisation et un transfert des savoirs dans des situations futures.

Nous utilisons une méthodologie de type recherche orientée par la conception (RoC) [3] pour concevoir, analyser et améliorer le jeu, le scénario pédagogique et le modèle de débriefing. Dans une précédente étude [4], nous présentions un modèle de débriefing

comprenant quatre dimensions. Les derniers travaux que nous avons menés nous ont conduits à ajouter une nouvelle dimension à notre modèle.

Dans le présent article, nous décrivons notre modèle qui se fonde sur les apports scientifiques liés à la pensée informatique et au débriefing. Puis, nous présentons nos questions de recherche et la méthodologie adoptée. La dernière partie porte sur les résultats que nous discutons, ainsi que sur la conclusion et nos perspectives.

2 Fondements théoriques

Dans la partie qui suit, nous présentons les fondements théoriques sur la pensée informatique et le débriefing.

2.1 Enseignement de l'informatique et pensée informatique

Pour Orange [5], l'informatique devrait être abordée en termes de résolution de problèmes et non pas de méthodes ou d'outils décontextualisées. Dans la littérature, cette méthodologie de résolution de problème porte plusieurs noms : démarche informatique, pensée informatique, démarche algorithmique ou encore pensée computationnelle [6]. Cette pensée informatique, c'est l'expression que nous retiendrons dans ce papier, constitue une compétence parfois qualifiée de compétence du 21ème siècle [7] en raison des enjeux liés au développement du numérique et à son influence dans toutes les sphères de l'activité humaine. Il s'agit de penser en prenant en compte de multiples niveaux d'abstraction et d'utiliser les outils et méthodes de l'informatique pour résoudre des problèmes qui se posent dans d'autres disciplines [7]. La pensée informatique est souvent décrite à travers six concepts : la décomposition du problème en sous-problèmes, l'abstraction, la modélisation, la généralisation (reconnaissance de pattern), la pensée algorithmique et logique, ainsi que l'évaluation [7, 8] ; et cinq techniques : la réflexion, le codage, le design, l'analyse et l'application [7]. La capacité à mobiliser une pensée informatique permet, dans le cadre de la résolution de problèmes, une plus grande efficacité, un meilleur résultat, et une analyse de son activité pour l'améliorer [7]. Face à un problème, deux questions peuvent se poser : est-il difficile de le résoudre et quelle est la meilleure façon de l'aborder. La programmation est une des pratiques qui permet le développement de cette pensée informatique [9].

2.2 Débriefing et institutionnalisation

Le débriefing est une étape importante de l'apprentissage par le jeu [10]. Nous distinguons débriefing et institutionnalisation. Le débriefing se rapporte aux apprentissages réalisés dans le jeu, en lien avec les comportements des joueurs et leurs ressentis durant l'activité [10]. Le concept d'institutionnalisation recouvre une réalité plus étroite. C'est « une situation qui se dénoue par le passage d'une connaissance de son rôle de moyen de résolution d'une situation d'action, de formulation ou de preuve, à un nouveau rôle, celui de référence pour des utilisations futures, personnelles ou collectives » [11, p. 4].

Le débriefing se rapporte donc d'une part à l'expérience des participants et, d'autre part, à la transformation des connaissances subjectives développées dans le cadre du jeu en savoirs objectifs mobilisables dans d'autres contextes, c'est-à-dire à l'institutionnalisation de ces savoirs. Selon notre approche, l'institutionnalisation des savoirs est le processus qui, dans le cadre du débriefing, concerne plus particulièrement les savoirs.

Selon la Théorie des situations didactiques élaborée par Guy Brousseau [12], l'institutionnalisation permet la transformation des connaissances en savoirs. Dans une situation d'apprentissage telle que le jeu, l'enseignant opère une dévolution du problème. Selon Rouchier [13], cette dévolution peut être définie comme le processus de conversion des savoirs en connaissances. Les élèves deviennent des joueurs et interagissent avec le jeu-*game*. Brousseau [12] distingue trois situations : les situations d'action, de formulation et de validation. Dans la situation dite d'action, les joueurs prennent des décisions qui les amènent à développer des connaissances subjectives et contextualisées. Les rétroactions du jeu amènent les joueurs à réfléchir à leurs erreurs et à réviser leurs choix. Les situations de formulation et de validation, permettent aux joueurs, dans le cadre d'un jeu d'équipe de discuter et de valider leurs choix stratégiques avec leurs partenaires. Dans ces situations, les connaissances développées lors de la phase de jeu sont formulées et validées. L'enseignant intervient ultérieurement, lors de l'institutionnalisation, qui est une situation dans laquelle l'enseignant vient décontextualiser, dépersonnaliser et légitimer les savoirs [14], afin de rendre leur transfert possible. Cette phase d'institutionnalisation est cruciale : elle vise « la formulation, la formalisation, la mémorisation, la reconnaissance d'une valeur culturelle et sociale […] nécessaires pour permettre de qualifier la connaissance comme savoir » [14, p. 146]. Ces processus de contextualisation et de décontextualisation sont complémentaires [14].

Pour élaborer une situation d'apprentissage, trois aspects doivent être pris en compte : la topogenèse, la chronogenèse et la mésogenèse [15]. La topogenèse renvoie au rôle de l'enseignant et de l'élève pendant le débriefing : qui effectue le débriefing ? Qui pose les questions et qui y répond ? La chronogenèse se réfère à la gestion du temps : quand institutionnaliser les savoirs ? Quelle progression des savoirs pendant le débriefing ? Quant à la mésogenèse, elle renvoie aux milieux didactiques de l'institutionnalisation : Quel matériel est utilisé ?

3 Proposition de modèle de débriefing

À partir d'éléments issus de la littérature et des premières expérimentations que nous avons conduites, nous avons proposé un modèle de débriefing en contexte de jeu que nous avons fait évoluer (Figure 1). Il comprend 5 dimensions que nous présentons dans la figure 1.

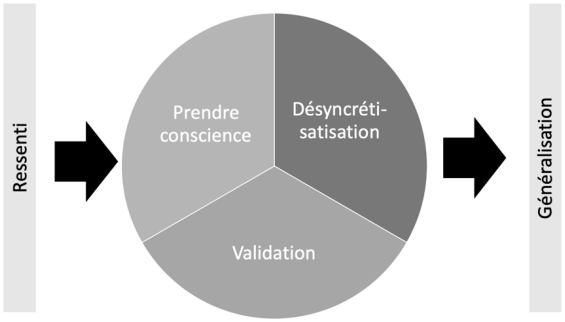


Fig. 1. Modèle de débriefing en cinq dimensions

Ressenti : Cette dimension regroupe tout ce qui se rapporte au ressenti, aux émotions, à la motivation, à la satisfaction, ainsi que l'envie de recommencer l'expérience de jeu. L'enseignant s'intéresse au vécu des participants.

Prendre conscience : C'est une première étape vers la métacognition. Dans une situation d'apprentissage par le jeu, les objectifs d'apprentissage sont implicites. L'enseignant et/ou les élèves vont donc rendre ces apprentissages visibles en identifiant les apprentissages réalisés et les savoirs en jeu, ainsi que les échecs et les réussites. Des connaissances, subjectives et contextualisées (qui font référence à la situation de jeu), parfois erronées et incomplètes, vont émerger.

Désyncrétiser : L'attention est portée sur la désyncrétisation des actions effectuées dans le jeu et les savoirs mobilisés pour les effectuer. Il s'agit de montrer que les tâches mobilisent des savoirs spécifiques. Un travail sur les erreurs et les incompréhensions est également réalisé. Dans cette dimension, les savoirs sont décontextualisés du jeu, puis recontextualisés.

Valider : Cette dimension permet de nommer les savoirs, de leur donner une définition, d'expliquer ce savoir d'un point de vue théorique, par le recours à des analogies, ainsi qu'en expliquant l'utilisation de ce savoir. Le savoir est légitimé et décontextualisé.

Généraliser : Des liens sont établis entre les apprentissages réalisés dans le jeu et les apprentissages passés, voire futurs. L'enseignant amène les élèves à identifier d'autres situations, d'autres problèmes pour lesquels ces savoirs peuvent être utilisés, ainsi qu'à argumenter sur comment les utiliser.

Selon ce modèle, le débriefing comprend un retour sur l'expérience de jeu et les affects du joueur ainsi qu'un travail spécifique sur les savoirs en jeu qui concerne plus spécifiquement ce qui relève de leur institutionnalisation. Il est généralement conduit par l'enseignant après le temps consacré au jeu mais certains éléments peuvent apparaître lors de la phase de jeu elle-même, quand l'enseignant est amené à interagir avec les joueurs.

Bien que la littérature montre l'importance du débriefing [1, 12], celui-ci est difficile à concevoir et à mettre en œuvre. Ainsi, ce papier vise à identifier, en contexte de jeu,

(Q1) Quelles dimensions du modèle de débriefing apparaissent dans les pratiques de l'enseignant ? Et à quels moments ? ; (Q2) Comment l'enseignant prend en compte chaque dimension en termes de topo- (rôle), chrono-(temps) et mésogenèse (milieu) ; (Q3) Comment s'articulent les différentes dimensions du modèle ?

4 Méthodologie

Dans cette étude exploratoire, la mise à l'épreuve de notre modèle repose sur une étude de cas. L'expérimentation s'est déroulée dans une classe du secondaire (15-17 ans) du canton de Fribourg (Suisse) au printemps 2019 avec un enseignant expérimenté. L'ensemble de la séance, comprenant une introduction, une phase de jeu et un débriefing, a duré 1h45 et a été filmé. Les échanges enregistrés pendant la séance ont été codés toutes les 15 secondes avec le logiciel d'annotation Elan[1] selon une grille d'indicateurs définis en amont et issue d'une précédente étude [4]. En effet, relativement aux questions de recherche, nous avons défini 16 catégories et sous-catégories afin d'identifier les cinq dimensions du modèle de débriefing. Leur élaboration s'est également appuyée sur les travaux de Sensevy [15]. Pour chaque interaction, nous identifions la dimension concernée et ses caractéristiques (cf. table 1). De plus, pour chaque dimension, nous identifions : 1) le rôle de l'enseignant et celui de l'élève, ainsi que le type d'interaction ; 2) les paramètres du milieu, à savoir, la nature du support d'intervention (écran de l'élève, le système de vidéoprojection, etc.), le type de support (le jeu, des diapositives, le code des élèves, etc.) et le type de production (préparée à l'avance ou récolté pendant la séance). Nous identifions les choix temporels pour chaque dimension, ainsi que pour les savoirs en comptabilisant les pas dans ELAN. Pour compléter les résultats quantitatifs issus des données ELAN, nous avons également relevé les *verbatims* de l'enseignant.

Table 1. Exemples d'indicateurs implémentés dans Elan pour identifier les dimensions et la nature de la dimension (caractéristiques)

Dimension	Nature de la dimension (caractéristiques)
Ressenti	Ressenti, émotion, motivation, satisfaction, envie de recommencer le jeu
Prendre conscience	Objectifs d'apprentissage, difficultés, réussites, savoirs en jeu
Désyncrétiser	Action/tâche, stratégie de résolution, erreurs et/ou incompréhension
Valider	Nommer, donner une définition, expliquer, analogies, expliquer l'utilisation, légitimer
Généraliser	Lien avec les apprentissages passés, lien avec les apprentissages futurs, liens avec le quotidien, identifier des liens avec d'autres situations, d'autres problèmes, argumenter comment agir dans ces autres situations

[1] ELAN est un logiciel d'annotation de fichiers multimédia développé par le groupe technique de l'Institut Max Planck de Psycholinguistique (Nijmegen, Pays-Bas).

5 Résultats et discussion

Dans cette partie, nous discutons les résultats qui ont émergé de notre analyse pour répondre à nos trois questions de recherche : (Q1) Quelles dimensions du modèle de débriefing apparaissent dans les pratiques de l'enseignant ? Et à quels moments ? ; (Q2) Comment l'enseignant prend en compte chaque dimension en termes de topo- (rôle), chrono-(temps) et mésogenèse (milieu) ; (Q3) Comment s'articulent les différentes dimensions du modèle ?

Toutes les dimensions du modèle sont présentes dans les pratiques de l'enseignant (Q1). En effet, les dimensions du modèle de débriefing sont observables durant le temps qui succède au jeu, mais également, et dans une moindre mesure, lors de l'introduction du jeu et de la phase de jeu elle-même. Le tableau 2 montre le nombre d'apparitions de ces phases et sa répartition en temps. Toutefois certaines sont plus présentes que d'autres. Sur les 90 minutes de la séance, les dimensions « ressenti », « prendre conscience » et « généralisation » constituent 1,9% du temps total (1'45''), alors que les dimensions « désyncrétiser » (11,9%) et « valider » (9,2%) représentent 21,1% du total. Le temps restant correspond à des moments de jeu silencieux.

La désyncrétisation consiste dans des interactions qui portent sur les actions à réaliser dans le jeu et les stratégies à appliquer, par exemple : "[...] dans votre cas, vous vous retrouviez ici sur le parcours devant un panneau sur lequel il y avait quelque chose d'écrit, un nombre, qui changeait à chaque fois. [...] c'est impossible d'écrire un programme et pis d'savoir avant c'que l'nombre va être [...] Et pis l'idée ici c'est d'aller prendre ce nombre [...] et pis on le met dans la boîte." ; et sur les erreurs et les incompréhensions, par exemple : "chez certains d'entre vous quand on dit tourner vers la droite, il tourne vraiment vers la droite, il avance plus après. Donc c'est pas tourner vers la droite et avancer. "

La validation concerne des interactions qui consistent à nommer les savoirs, à les définir, par exemple "La variable c'est une boîte dans laquelle on peut aller stocker quelque chose qu'on peut aller rechercher plus tard" ; et à les expliquer (théorie et utilisation), par exemple "[..] cette boîte, on peut en avoir plein. On peut faire plein de boîtes dans l'ordinateur, c'est c'qu'on appelle la mémoire de l'ordinateur. Et pis cette boîte, c'qu'on fait en programmation c'est qu'on lui donne un nom.". Toutefois, la caractéristique « légitimer » (dans la validation) est peu prise en compte (15'' auprès d'une seule élève). Il en est de même pour la « prise de conscience » (1'), étape importante pour rendre explicites les objectifs d'apprentissage et les savoirs en jeu [12].

Table 2. Dimensions identifiées : nombre d'apparitions et répartition du temps

	Ressenti	**Prendre conscience**	**Désyncrétiser**	**Valider**	**Généraliser**
Intro	30'' (2x)	15'' (1x)	30'' (1x)	-	-
Jeu	-	15'' (1x)	5'30'' (22x)	2'45'' (10x)	15' (1x)
Débriefing	-	30'' (2x)	4'45'' (19x)	5'30'' (19x)	-
Total	***30'' (2x)***	***1' (4x)***	***10'45'' (45x)***	***8'15'' (29x)***	***15' (1x)***

En référence aux travaux de Brousseau [12], la phase de jeu constitue la situation d'action, de formulation et de validation. L'institutionnalisation intervient pour transformer les connaissances en savoir. Dans le cadre de l'expérimentation que nous avons conduite, des phases de débriefing sont menées en parallèle. L'enseignant intervient alors pour aider des élèves en difficulté et leur permettre de poursuivre le jeu. De plus, au cours du jeu, les objectifs sont implicites. Un travail de prise de conscience est nécessaire pour rendre explicites les savoirs. Selon nos observations, ce travail de prise de conscience est peu représenté. Par conséquent, il est possible que le travail de désyncrétisation et de validation des savoirs n'aboutisse pas. En conclusion, il semblerait que l'enseignant cherche en priorité à transmettre des savoirs et à les décontextualiser. La prise de conscience du savoir, sa valorisation comme élément appartenant à une communauté scientifique, ainsi que le transfert de ces savoirs dans d'autres situations est peu pris en compte.

Concernant la prise en compte de chaque dimension en termes de rôles, de milieu et de temps (Q2), l'analyse de la vidéo montre que :

Du point de vue du rôle [15], pour chaque dimension, la discussion est conduite par l'enseignant face au groupe classe lors de l'introduction et du débriefing. Lors de la phase de jeu, un début d'échange s'observe entre 1 ou 2 élèves et l'enseignant lorsqu'une question est posé à l'enseignant. Dans l'ensemble, il y a donc peu d'échanges entre les élèves et l'enseignant.

Du point de vue du milieu [15], l'enseignant se sert du vidéoprojecteur et du jeu pour désyncrétiser, et de l'*openboard* pour désyncrétiser et valider devant le groupe classe. Le débriefing sur le « ressenti », la « prise de conscience » et la « généralisation » n'ont nécessité aucun matériel. Durant les 90 minutes, les élèves se trouvent à leur place, devant leur écran. Lors de l'introduction et du débriefing, l'enseignant se trouve principalement à son bureau ou devant le tableau, alors que lors de la phase de jeu, il effectue des allers-retours entre son bureau et les élèves. Il est probable que cette disposition ait favorisé une poste de transmission magistrale des savoirs au détriment d'échanges plus symétriques.

En termes de temps [15], sur les 90 minutes de la séance, l'introduction a duré 6 minutes (6,7% du temps total), la phase de jeu 53 minutes (58,9%) et le débriefing 11

minutes (12,2%). Les élèves ont répondu à un questionnaire de 15 minutes avant la phase de débriefing. Le tableau 2 montre que les interventions de l'enseignant sont en priorité orientées vers la désyncrétisation lors de l'introduction la phase de jeu (5'30), ainsi que vers la validation (2'45). Cette tendance s'inverse lors du débriefing (validation = 5'30 ; désyncrétisation = 4'45). Nous en concluons que l'enseignant intervient pour contextualiser les savoirs lors de la phase de jeu et pour les décontextualiser lors du débriefing. Toutefois, la désyncrétisation reste encore très présente lors du débriefing et le travail sur le transfert est absent.

L'analyse du type d'intervention montre que les savoirs abordés par l'enseignant dépendent des questions des élèves, alors que pour la phase de débriefing, l'enseignant aborde les savoirs dans l'ordre proposé dans le jeu, à savoir sur les instructions (durée 1'15''), les variables (7'15''), et les conditions (2'30''). Aucun concept et/ou technique [7, 8] de la pensée informatique n'est abordé dans le débriefing. Nous soulignons toutefois que l'enseignant ne parvient pas à la fin du débriefing en raison de la sonnerie de fin de séance. Lors de ces interventions, l'enseignant privilégie l'apport d'informations (table 3). Il a toutefois choisi de poser des questions et de faire des rétroactions pour la désyncrétisation (table 3). Selon nous, la gestion du temps pourrait être un facteur impactant le type d'interventions, mais également la manière de mener le débriefing. Le temps imparti pour le débriefing devrait donc être augmenté pour permettre à l'enseignant de débriefer sur l'ensemble des savoirs issus du jeu, mais également pour permettre une variation dans la manière dont les interactions sont menées.

Table 3. Type d'intervention selon les dimensions du modèle de débriefing

	Ressenti	**Prendre conscience**	**Désyncrétiser**	**Valider**	**Généraliser**
Consigne	1	1	2	-	-
Question		1	6		
Rétroaction			6		-
Information		3	24	31	1

Nous n'avons pas été en mesure d'identifier des patterns entre les dimensions du modèle de débriefing (Q3), hormis entre les dimensions de désyncrétisation et de validation qui se font le plus souvent de manière alternée. Il est parfois difficile de distinguer la limite entre ces deux dimensions, car la transition de l'une à l'autre se fait rapidement. Toutefois, nous identifions un pattern « nommer – donner une définition - expliquer la théorie » à trois reprises dans la validation des savoirs du débriefing. Cela pourrait traduire une méthode de validation du savoir de la part de l'enseignant. Toutefois, l'analyse de cas ne permet pas de généraliser un tel résultat.

6 Conclusion et perspectives

Notre objectif était de tester les cinq dimensions de notre modèle. À travers notre recherche exploratoire, basée sur une étude de cas, nous souhaitions utiliser notre modèle pour analyser une leçon d'apprentissage de la programmation avec le jeu *Programming Game*, ainsi que de voir quand et comment les connaissances développées dans le jeu étaient transformées en savoirs transférables.

Notre analyse montre que, dans le cadre de cette expérimentation, les interactions enseignant-élèves visent principalement la validation des savoirs et leur désyncrétisation. Le travail sur la prise de conscience des objectifs d'apprentissage et des savoirs en jeu est limité ce qui pourrait entraver l'appropriation des savoirs. De plus, les aspects qui relèvent de la légitimation, ainsi que la généralisation n'ont pas été pris en compte par l'enseignant, ce qui pourrait mettre en péril le transfert des savoirs dans d'autres contextes. Dès lors, l'analyse de la situation d'apprentissage par le jeu à partir de notre modèle de débriefing donne des pistes pour améliorer le travail de l'enseignant afin d'accompagner les élèves dans la transformation des connaissances en savoirs réutilisables dans d'autres contextes.

Les apports de ce travail exploratoire nous semblent résider dans une première validation de notre modèle en tant que cadre d'analyse du débriefing. Ils nous semblent également résider dans la méthodologie que nous avons élaborée et en particulier la grille d'analyse des vidéos qui cependant devra subir certaines révisions. Il est notamment difficile d'identifier la limite entre les dimensions désyncrétisation et validation. Ces résultats sont issus d'une étude de cas et devront être confirmés lors des expérimentations planifiées pour la seconde itération de notre projet. La suite de notre recherche visera à adapter et valider la grille d'analyse. En vue de la seconde itération, nous travaillerons avec les enseignants pour implémenter le modèle développé dans le scénario pédagogique et ainsi tester sa valeur pour penser la question du débriefing dans le cadre de l'usage du jeu pour l'enseignement de l'informatique. Une analyse des savoirs informatiques en jeu dans *Programming Game* est en cours et viendra compléter cette première analyse du débriefing.

Remerciements. Nos remerciements vont à la Fondation Hasler qui a financé le projet durant l'année 2018-2019, l'équipe AlbaSim (HEIG VD, Suisse), ainsi qu'aux enseignants d'informatique, au graphiste et au *game designer* qui ont contribué au projet.

References

1. Sanchez, E. « Game-Based Learning », in *Encyclopedia of Education and Information Technologies*, A. Tatnall, Éd. Cham: Springer International Publishing, 2019, p. 1-9.
2. AlbaSim, « Programming Game », *Serious Game*, 2018. [En ligne]. Disponible sur: https://www.albasim.ch/fr/nos-serious-games/. [Consulté le: 14-janv-2019].

3. Sanchez, E. et Monod-Ansaldi, R. « Recherche collaborative orientée par la conception. Un paradigme méthodologique pour prendre en compte la complexité des situations d'enseignement-apprentissage », *Education et didactique*, vol. 9, nº 2, p. 73-94, 2015.
4. Plumettaz-Sieber, M., Bonnat, C. et Sanchez, E. « Debriefing and Knowledge Processing. An Empirical Study about Game-Based learning for Computer education », soumis à Game And Learning Allianz (GALA), Athènes, à paraître.
5. Orange, C. « Didactique de l'informatique et pratiques sociales de référence », *Bulletin de l'EPI (Enseignement Public et Informatique)*, nº 60, p. 151–161, 1990.
6. Drot-Delange, B. et Bruillard, É. « Éducation aux TIC, cultures informatique et du numérique : quelques repères historiques », *Études de communication. langages, information, médiations*, nº 38, p. 69-80, juin 2012.
7. Wing, J. « Computational thinking », *Communications of the ACM*, vol. 49, nº 3, p. 33–35, 2006.
8. Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. et Woollard, J., *Computational Thinking: A guide for teachers*, Computing at School. Hodder Education, 2015.
9. Malyn-Smith, J. et Angeli, C. « Computational Thinking », in *Encyclopedia of Education and Information Technologies*, A. Tatnall, Éd. Cham: Springer International Publishing, 2019, p. 1-7.
10. Lederman, L. C. « Debriefing: Toward a Systematic Assessment of Theory and Practice », *SAGE journals*, vol. 23, nº 2, p. 145-160, 1992.
11. Brousseau, G. « Glossaire de quelques concepts de la théorie des situations didactiques en mathématiques ». 2010.
12. Brousseau, G. *Théorie des situations didactiques*, La pensée sauvage. Grenoble, 1998.
13. Rouchier, A. « Etude de la conceptualisation dans le système didactique en mathématiques et informatique élémentaires : proportionnalité, structures itéro-récursives, institutionnalisation », Thèse d'état, Université d'Orléans, 1991.
14. Laparra, M. et Margolinas, C. « Milieu, connaissance, savoir. Des concepts pour l'analyse de situations d'enseignement », *Pratiques, Centre de recherche sur les médiations (CREM)*, p. 145-146, 2010.
15. Sensevy, G. « Théories de l'action et action du professeur », in *Théories de l'action et éducation*, J. Baudouin et J. Friederich, Éd. Bruxelles : De Boeck Supérieur, 2001.

Introduire la concurrence en début de secondaire ?

Cédric Libert et Wim Vanhoof

Université de Namur, Belgique
{cedric.libert,wim.vanhoof}@unamur.be

Résumé Dans la littérature, plusieurs auteurs encouragent l'enseignement de la concurrence, car beaucoup d'évolutions en informatique font appel à ce concept. Nous rapportons dans cet article le résultat d'une expérience menée auprès de 101 adolescents et adolescentes de 12 à 15 ans à qui nous avons dispensé deux séances de 100 minutes sur la concurrence par passage de messages. Nous y commentons l'évolution des élèves face au concept de concurrence et face à la réalisation d'un problème de synchronisation.

Keywords: concurrence · secondaire · programmation concurrente · passage de messages

1 Introduction

En Fédération Wallonie Bruxelles, les cours de programmation dispensés en secondaire se concentrent sur l'enseignement des paradigmes impératif et orienté objet. On considère généralement ces paradigmes comme plus simples à comprendre [2,3], notamment grâce à leur nature séquentielle. D'un autre côté, la programmation concurrente, permettant de programmer l'exécution de plusieurs tâches simultanées, fait généralement partie de cours plus avancés, après le secondaire. Certains auteurs prétendent toutefois que la programmation concurrente peut être au moins aussi intuitive pour les étudiants que la programmation séquentielle [11,12]. De plus, dans son rapport de 2013 sur l'enseignement de l'informatique, l'ACM recommande que le cours introductif à la programmation, destiné à des novices, s'inscrive dans un modèle où la concurrence est la norme, et l'aspect séquentiel un cas particulier [1, p.44]. En effet, l'évolution de trois grands domaines de l'informatique -matériel, logiciel et données- implique la concurrence.

Pour vérifier la possibilité d'enseigner la programmation concurrente en secondaire, nous avons mené une expérimentation auprès de 101 étudiants de secondaires en Fédération Wallonie Bruxelles. Nous y avons élicité leurs préconceptions lexicales sur le terme "concurrence" et examiné comment ils tentent de résoudre un problème de synchronisation, au moyen d'un test initial [7]. Nous avons ensuite donné aux élèves deux séances de 100 minutes de cours sur la concurrence dans le cadre du projet School-IT [9], qui consiste à développer,

tester en classe et regrouper des matériaux d'enseignement liés à différentes facettes de l'informatique. Finalement, nous leur avons fait passer un test final, correspondant au test initial.

Dans cet article, nous vérifions l'évolution de ces analyses préalables après ces deux séances de cours en comparant les réponses aux questionnaires prétest et post-test. Nous présentons d'abord le contexte scientifique de cette recherche : quelle est la conception de certains auteurs sur la pertinence d'enseigner la concurrence à des novices et quel est l'état de la recherche sur les préconceptions des apprenants. Nous décrivons ensuite l'expérience que nous avons menée avec les élèves et comment nous avons traité les données récoltées. Puis, en nous basant sur ces données, nous discutons des résultats quantitatifs et qualitatifs des questionnaires que nous avons soumis aux étudiants avant et après les deux séances de cours.

2 Contexte

La concurrence est un concept de programmation qui permet de programmer plusieurs actions pour qu'elles aient lieu en même temps [8] dans la poursuite d'un même but. Elle est présente intrinsèquement dans le monde, où des évènements simultanés ont lieu en permanence. La programmation concurrente capture cette caractéristique en s'affranchissant de l'obligation de séquentialité et en permettant l'exécution simultanée de plusieurs instructions.

Cette propriété, en fonction du paradigme auquel elle est intégrée, se réalise sous la forme d'un modèle particulier, qui décrit comment les instructions parallèles peuvent communiquer. Dans cet article, nous utiliserons un modèle de concurrence en particulier qui est le modèle acteur. Dans ce modèle, le système programmé est constitué d'acteurs qui agissent en même temps et peuvent envoyer des messages à d'autres acteurs ou recevoir des messages d'autres acteurs.

L'idée d'enseigner la programmation concurrente relativement tôt dans le cursus n'est pas neuve. Dès 1997, Feldman et Bachus décrivent en quoi il serait possible de l'expliquer à des néo-programmeurs [4]. En 1998, Lynn Andrea Stein, quant à elle, plaide pour qu'on passe du modèle de programmation Turing/Von Neumann, qui considère qu'un programme est une fonction et donc que son exécution est réductible à un calcul, à un modèle d'interaction où un programme est une communauté d'agents qui communiquent [10].

C'est dans ce contexte qu'en 2001, Yifat Ben-David Kolikant [5] analyse les préconceptions de ses étudiants, dans une approche constructiviste de l'apprentissage. Elle réalise cette recherche dans le cadre d'un cours de programmation concurrente dispensé à des élèves de fin du secondaire (17-18 ans), en orientation informatique avancée. Elle souhaite, grâce à ses conclusions de recherche, adapter son cours aux problèmes rencontrés par ses élèves. Elle y traite principalement du problème de synchronisation entre les différentes actions concurrentes. Les étudiants ont ainsi dû résoudre le problème suivant :

Pour vendre un ticket dans un cinéma, un vendeur réalise l'algorithme suivant :

```
Pour chaque client
    Chercher la meilleure place libre: C
    Si C existe (s'il restait encore au moins une place libre)
       Marquer C comme occupée
       Imprimer le ticket correspondant à C
```

Les étudiants doivent décrire un système et un algorithme qui permettent à deux vendeurs de vendre des tickets en même temps. Ils doivent en sus expliquer en quoi leur solution permet d'éviter que deux tickets identiques soient vendus à des clients différents.

Les chercheurs ont ensuite classé les réponses en cinq catégories : trois (C1, C2, C3) qui incluent une gestion centralisée de la concurrence ; deux (D1, D2) qui proposent une solution centralisée au problème :

— C1 : une entité centrale gère l'entrelacement des actions ;
— C2 : l'étudiant fait des suppositions sur l'ordre d'exécution, de sorte qu'il n'existe qu'un seul scénario possible ;
— C3 : certaines ressources communes sont privatisées au sein d'entités concurrentes ;
— D1 : les vendeurs communiquent de façon implicite, via une mémoire partagée ;
— D2 : les vendeurs communiquent de façon explicite, via des messages.

De ces réponses, Kolikant conclut que :

— les étudiants trouvent plus de solutions centralisées que décentralisées
— ceux qui ont suggéré des solutions décentralisées utilisent principalement la communication pour bloquer une tâche tant qu'une autre n'est pas réalisée
— beaucoup simplifient le problème, pour le résoudre plus facilement
— un système concurrent distribué sur plusieurs ordinateurs est plus intuitif pour eux que la concurrence sur un seul ordinateur

Quelques années après, en 2007, Lewandowski et al. [6] a mené une expérience similaire, avec les mêmes exercices, sur des étudiants de première année d'étude supérieure en sciences informatiques. Il modifie le problème pour l'adapter au manque de connaissance de ses étudiants et ne leur demande ainsi pas de (pseudo-)code ni de définition précise du matériel utilisé, mais uniquement une description en anglais de leur système. Il y retrouve les même catégories élicitées par Kolikant et met au jour deux problèmes saillants : les étudiants éprouvent des difficultés à repérer et éviter les *race conditions*, c'est-à-dire le caractère non-déterministe de leur solution ; et certains simplifient à outrance la tâche pour en résoudre une version plus simple. Il conclut de son analyse que manifestement les étudiants sur lesquels il a réalisé son expérience ont des préconceptions très variées, de sorte qu'un enseignant en programmation concurrente pourrait introduire son cours par la résolution d'un problème du type "vente de tickets" et la comparaison des différentes solutions proposées par les étudiants et soulever les mauvaises conceptions sur l'exécution concurrente. Finalement, il remarque que la distribution des solutions de ses étudiants est différente de celle obser-

vée par Kolikant. En particulier, plus de la moitié des solutions proposées sont centralisées (C1, C2, C3) contre un tiers chez Kolikant.

Notre article s'inscrit dans la lignée de ces deux contributions en fournissant aux élèves le même problème à résoudre. Toutefois, nous nous démarquons sur trois points. D'une part, les étudiants sur lesquels nous avons mené notre recherche sont plus jeunes et n'ont, pour beaucoup, jamais programmé. D'autre part, nous avons adapté le problème à résoudre, à la manière de Lewandowski, en leur demandant de décrire leur solution en français, et nous avons ajouté d'autres exercices qui permettent d'en savoir plus sur les conceptions lexicales du terme "programmation concurrente". Finalement, notre but n'est pas de déterminer une classification des solutions proposées (comme l'a fait Kolikant), ni de déterminer si les étudiants sont capables de reconnaitre les problèmes de concurrence dans le cas où il y a des ressources partagées (comme l'a fait Lewandowski), mais de déterminer à quel point des élèves de 12 à 15 ans sont capables de comprendre et de résoudre des problèmes de concurrence, et comment.

3 Méthodologie

Notre recherche s'articule en trois phases : un questionnaire pré-test, pour évaluer les préconceptions des étudiants ; deux séances de 100 minutes de cours données dans chacune des classes ; un questionnaire post-test, identique au questionnaire pré-test, pour évaluer l'évolution des conceptions.

Le public de notre étude est constitué de 101 élèves répartis sur trois écoles, en première (27), deuxième (20) et troisième (54) année, soit 73 adolescents et 28 adolescentes de 12 à 15 ans.

3.1 Pré-test et post-test

Le pré-test et post-test se sont déroulés de façon identique, respectivement en début de la première séance de cours et en fin de la seconde. Les élèves avaient 20 minutes pour répondre au questionnaire. Puisqu'il s'agissait d'une inquiétude récurrente, nous leur avons précisé d'emblée qu'il ne s'agissait pas d'une évaluation certificative, mais d'une recherche scientifique. 35% de ces étudiants avaient déjà programmé avant, principalement avec des systèmes de programmation visuelle tels que Scratch ou Micro:bit.

Dans la première partie du questionnaire, nous demandons aux étudiants s'ils ont déjà programmé et d'expliquer, le cas échéant, ce qu'ils ont déjà fait. Cette explication est importante, car elle nous permet de nous rendre compte de ce que les étudiants considèrent comme étant de la programmation.

Dans la deuxième partie, nous cherchons à évaluer les préconceptions lexicales de la notion de programmation concurrente, à travers 3 questions. Nous leur demandons, d'abord, de définir la programmation concurrente. Ensuite, nous leur montrons sept systèmes (Android, jeu vidéo, calculatrice, programme Scratch, distributeur de boissons, Facebook, Whatsapp) et nous leur demandons d'indiquer, pour chacun, s'ils utilisent la programmation concurrente (oui, non, je

ne sais pas, je ne connais pas ce système). Finalement, ils doivent expliquer, pour l'un des systèmes où ils ont indiqué "oui", pourquoi ils considèrent que c'est concurrent. Ils doivent faire de même pour un système où ils ont indiqué "non".

Finalement, dans la troisième partie, nous leur soumettons le problème des vendeurs de tickets tels que décrit dans la section 2. Nous leur indiquons que, dans la configuration actuelle, deux vendeurs peuvent être amenés à vendre le même ticket à deux clients différents et nous leur demandons d'imaginer quatre solutions qui permettraient de résoudre le problème en modifiant le système. Nous leur signalons qu'ils sont invités à donner leurs solutions en français et pas dans un langage de programmation.

3.2 Séances de cours

La première séance de cours s'entame, suite au test, sur une discussion avec les élèves à propos de la concurrence. Il en ressort qu'un système concurrent est un système où plusieurs évènements peuvent avoir lieu simultanément ou quasi-simultanément. Les élèves donnent ainsi différents exemples de systèmes et discutent de leur aspect concurrent ou pas.

S'ensuit une mise en situation où certains élèves reçoivent une fiche avec une partie du visage (œil, sourcil, bouche), certains reçoivent une fiche *cerveau*, et enfin d'autres reçoivent une fiche sens (vue ou audition). Ils reçoivent également des rectangles de papier qui représentent des messages. Nous leur expliquons que chacune des fiches décrit les actions qu'ils doivent réaliser en fonction du message qu'ils reçoivent. Nous appelons cette fiche le *comportement*. En examinant les *messages*, nous parvenons à une définition avec eux de ce qu'est un message simple : un destinataire et un contenu. Dans le cas qui nous concerne, le destinataire est un élément du visage et le contenu une action qu'il doit effectuer.

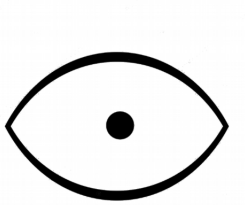
Nous réalisons ensuite une première exécution de ce système-visage : un premier stimulus est perçu par l'audition (un claquement de mains), qui envoie un message au cerveau, qui lui-même envoie des messages à différentes parties du visage. Après quelques secondes, un visage surpris se met en place. Nous réitérons l'opération, avec d'autres élèves et un autre stimulus, pour obtenir un autre visage, puis nous passons à une phase de discussion : forment-ils, ensemble, un système concurrent ? Qu'observent-ils ?

Après cette discussion, nous leur demandons, par groupe, de modifier les fiches de comportement des acteurs, en ajoutant le cas échéant des messages, pour créer un nouveau visage qui réagit à un autre stimulus, qu'ils doivent définir également. Ils peuvent, à cet effet, écrire directement sur les fiches qu'ils ont à leur disposition. À la fin de cette étape, nous testons ensemble le nouveau scénario élaboré par chacun des groupes, en identifiant, le cas échéant, les imprécisions dans la description des actions ou des messages.

Généralement, la première séance de 100 minutes s'arrête là. Au début de la séance suivante, nous leur demandons de réaliser un schéma qui représente le système-visage qu'ils ont exécuté la dernière fois, au moyen de flèches qui représentent les messages envoyés entre les différents agents.

Figure 1. Fiche de comportement pour l'œil.

ŒIL



Actions à faire pour chacun des messages reçus :

Message : **OUVRE**
À faire :
tourner l'œil du côté ouvert

Message : **FERME**
À faire :
tourner l'œil du côté fermé

Nous passons finalement à la partie de programmation sur les Micro:Bit. Nous leur expliquons d'abord le fonctionnement de l'appareil, de l'environnement de programmation et de l'envoi de messages. Nous leur demandons ensuite d'implémenter chacun sur son Micro:Bit la partie de visage qui lui est attribuée, en se basant sur le schéma qui a été réalisé. Puisqu'ils n'ont encore jamais programmé, cette phase est partiellement guidée. En fin de séance, nous mettons en commun les Micro:Bit pour qu'ils forment un visage et nous vérifions si chaque élément du visage réalise bien ce qui est demandé.

3.3 Analyse et traitement des données

Nous avons encodé les réponses des étudiants au pré-test et au post-test et les avons traitées de la façon suivante, en prélude à l'analyse.

En ce qui concerne la définition de la concurrence demandée aux élèves, nous avons élicité les grandes catégories suivantes à partir de leurs définitions et des explications qu'ils ont données à la question de classification des systèmes : programmer des actions simultanées, plusieurs actions en même temps, concurrence économique, envoyer des messages, plusieurs actions (sans préciser plus), aucune idée. Nous n'avons pas retenu les 10% de réponses qui constituent des catégories singletons.

Pour ce qui est de l'exercice sur la vente de tickets, nous avons classifié chaque solution en l'une des cinq catégories de Kolikant (voir section 2). Nous avons également évalué, pour chacune des réponses, si elle est raisonnable, c'est-à-dire si elle propose une idée pour résoudre le problème présenté, même si l'étudiant n'explique pas en détail comment l'implémenter. Par exemple "empêcher les deux vendeurs d'utiliser le système en même temps" est considéré comme une solution

raisonnable pour résoudre le problème, bien qu'elle puisse conduire à d'autres problèmes et que l'étudiant ne spécifie pas une procédure qui permettrait d'atteindre cet objectif. Finalement, certaines ne sont simplement pas des solutions au problème soumis.

4 Résultats

4.1 Évolution de la compréhension terminologique

Grâce au post-test, nous avons remarqué qu'à la suite des deux séances de 100 minutes chacune sur la programmation concurrente, la perception du terme "programmation concurrente" a significativement changé : les définitions données (question 3) sont extrêmement différentes du pré-test, la catégorisation des systèmes concurrents ou non (question 4) témoigne d'un plus grand sentiment de connaissance et les justifications qu'ils donnent pour cette classification (question 5) sont cohérentes avec les nouvelles définitions données.

Concernant les définitions, le pré-test met en évidence deux grandes catégories de réponses chez les élèves : ceux qui avouent ne pas savoir (71) et ceux qui évoquent la concurrence dans le sens de compétition économique ou sportive (19). Les autres catégories sont relativement négligeables, mais nous notons tout de même que 3 élèves évoquent l'idée d'exécution d'actions simultanées. Après les deux séances de cours, dans le post-test, seuls 17 déclarent encore ne pas savoir ce qu'est la programmation concurrente. Parmi eux, 14 se trouvaient déjà initialement dans cette catégorie. Pour 36 étudiants, la programmation concurrente consiste à programmer un système où plusieurs actions peuvent avoir lieu simultanément. Et 23 autres évoquent également l'idée d'actions simultanées. Les étudiants qui se retrouvent dans ces deux catégories au post-test sont principalement ceux qui avaient déclaré initialement qu'ils ne savaient pas. A contrario, seuls 2 étudiants considèrent toujours la concurrence comme une forme de compétition économique ou sportive. Nous observons finalement que 10 étudiants considèrent que la programmation concurrente est définie par le fait qu'il y a un échange de messages entre entités, ce qui est dû au modèle de concurrence par passage de messages utilisé dans le cadre de ce cours. Nous concluons donc que les préconceptions terminologiques de départs ont été rapidement corrigées chez les étudiants, car un peu plus de la moitié d'entre eux considèrent au post-test que la programmation concurrente concerne des actions qui ont lieu en même temps.

Dans la question suivante, où il s'agit pour l'élève de déterminer pour chacun des systèmes mentionnés s'il est concurrent ou non, nous remarquons une diminution significative des réponses vides ou des réponses où ils indiquent ne pas savoir. Cela montre un possible sentiment de maitrise plus élevé chez les étudiants. Par ailleurs, au pré-test, les étudiants excluaient les systèmes qui n'étaient pas considérés comme commercialement en concurrence avec d'autres (la calculatrice et le distributeur de boissons). Ces systèmes demeurent, dans le post-test, peu sélectionnés par les étudiants comme étant concurrents, relativement aux autres. Toutefois, au regard des justifications données par les étudiants

Table 1. Nombre d'étudiants par type de définition de la programmation concurrente.

Catégorie de réponse	Pré-test	Posttest
Systèmes avec plusieurs actions simultanées	0	36
Actions en même temps	3	23
Compétition	19	2
Courant	2	0
Meilleure, grande, complexe	4	1
Aucune idée	71	17
IA	2	0
Plusieurs acteurs	0	1
Plusieurs programmeurs en même temps	0	3
Envoyer des messages	0	10
Ordinateur fait ce qu'on demande	0	1
Plusieurs choses se passent	0	5
Serveur	0	1
Compétition	0	1

dans la question subséquente pour expliquer leurs choix, nous en déduisons que la raison n'est plus liée à la compétition commerciale, mais à la réalisation ou non d'actions simultanées dans le système.

Table 2. Classification des systèmes par l'élève : ce système est-il concurrent ?

	Pré-test				Posttest			
	Concurrent	Pas concurrent	Ne sait pas	Rien	Oui	Non	Ne sais pas	Rien
Android	51	9	21	14	78	6	11	5
Jeu vidéo	49	7	26	13	88	6	3	4
Calculatrice	20	32	28	15	37	47	12	5
Programme Scratch	21	11	26	14	39	6	17	4
Distributeur	17	34	26	15	32	42	22	5
Facebook	49	12	21	13	79	9	9	4
Whatsapp	46	10	23	14	79	6	9	4
Totaux	253	115	171	98	432	122	83	31

L'exercice suivant met également en évidence ce plus grand sentiment de compétence. En effet, là où, dans le pré-test, 48 étudiants n'ont rien répondu quand il s'agit d'expliquer, pour deux des systèmes préalablement sélectionnés, pourquoi ils sont ou pourquoi ils ne sont pas concurrents, seulement 13 étudiants n'ont rien répondu dans le post-test. Mieux, 43 étudiants justifient, dans le post-test au moins un de leurs deux choix au moyen de la définition de la concurrence comme permettant l'exécution simultanée de plusieurs actions, contre aucun dans le pré-test. Par ailleurs, très peu d'étudiants (3) ont conservé la conception que la concurrence était liée à une forme de compétition économique. Nous notons finalement que 10 étudiants associent concurrence et envoi de messages,

comme nous l'avions déjà remarqué dans les définitions qu'ils ont données à la première question.

Table 3. Raisons pour avoir indiqué qu'un système est concurrent / n'est pas concurrent

	PRE		POST	
	Une réponse	Deux réponses	Une réponse	Deux réponses
Rien	71	48	45	13
Informatique	15	7	7	1
Concurrence économique	17	9	3	2
Complexe	7	2	1	0
En même temps	0	0	43	13
Plusieurs actions	0	0	26	3
Messages	0	0	8	0

De ces analyses, nous concluons que les élèves, après deux séances de 100 minutes de cours, étaient déjà capables de définir de façon assez juste ce qu'était la programmation concurrente et d'appliquer cette définition à des systèmes existants, dans le but de les classer en justifiant deux de leurs réponses.

4.2 Évolution des solutions au problème de la vente de tickets

Nous avons examiné les solutions proposées par les élèves pour résoudre le problème qui leur était soumis. En les comparant avec les réponses fournies dans le pré-test, nous remarquons deux points importants. D'une part, le nombre de solutions raisonnables fournies est significativement supérieur à ce qui se trouvait dans le pré-test. Nous passons ainsi de 90 solutions raisonnables à 114, et nous doublons le nombre d'étudiants qui proposaient au moins 3 de ces solutions : de 6 nous passons ainsi à 14.

Table 4. Solutions raisonnables

# solutions	# étudiants pré-test	# étudiants post-test
4	1	3
3	5	11
2	21	21
1	29	28
0	45	38

En revanche, lorsque nous observons les catégories de réponses fournies par les étudiants, nous remarquons qu'il y a relativement peu de changement entre le pré-test et le post-test. Il y a toutefois davantage de réponses qui citent explicitement l'envoi et la réception de messages. Cela s'est traduit par une augmentation

légère du nombre de réponses qui se trouvent dans la catégorie D2 (communication explicite par messages envoyés entre les entités).

Cette faible différence trouve sans doute son explication dans le fait que l'activité de programmation *per se* comprenait une large part de temps guidé. En effet, puisque deux tiers des étudiants n'avaient jamais programmé, il a fallu consacrer un certain temps à les accompagner dans l'implémentation du visage. Un autre facteur, conséquent au précédent, est qu'ils n'ont pas eu l'occasion, faute de temps, de créer eux-mêmes un système concurrent pour résoudre un problème.

Table 5. Nombre de solutions données par catégorie (voir section 2 pour la description des catégories)

Catégorie de solutions	solutions pré-test	solutions post-test
C1	13%	12%
C2	20%	21%
C3	28%	30%
D1	27%	20%
D2	12%	17%

Finalement, nous remarquons une différence assez nette avec la distribution des réponses observées par Kolikant (qui s'explique probablement par la différence d'âge et d'expérience de programmation), et une différence légèrement moins marquée avec celle rapportée par Lewandowski qui, comme nous, a réalisé son expérience sur des étudiants sans expérience de programmation (mais moins jeunes). Nous y remarquons une grande baisse des solutions de type décentralisée, déjà observée chez Lewandowski qui l'expliquait par une plus grande facilité des étudiants à ordonner des actions de façon centralisée.

Table 6. Distribution des types de réponses donnée

	Kolikant	Lewandowski	Libert
C1	21.5%	7%	7%
C2	5.9%	9%	13%
C3	5.9%	22%	18%
D1	25.2%	31%	12%
D2	25.2%		10%
Pas raisonnables	16.3%	31%	40%

4.3 Réactions des élèves à l'activité

Lors des différentes activités, nous avons relevé deux réactions intéressantes qui se sont produites dans certaines classes : l'identification d'un goulot d'étran-

glement qui limite les performances du système (*bottleneck*) et la création spontanée de procédures.

Ainsi, dans 3 des 7 classes, après l'activité d'exécution manuelle du visage, des élèves nous ont indiqué que le cerveau contribuait à retarder la transmission des messages, parce que tous les messages lui arrivent, qu'il doit en envoyer plusieurs et qu'il ne sait pas faire plusieurs choses en même temps. Cela se traduit physiquement par la difficulté qu'a l'élève qui exécute les instructions du cerveau à gérer les différents messages qui lui parvient et à envoyer les messages correspondants. Cette discussion a conduit les élèves à se demander s'il ne valait pas mieux envoyer directement les messages aux acteurs concernés, sans passer par un cerveau central. Cette remarque met en évidence que certains jeunes élèves sont capables de se rendre compte qu'un goulot d'étranglement limite les performances d'un système concurrent.

Dans 2 autres classes, lorsqu'ils devaient déterminer un nouveau comportement pour le visage, plusieurs groupes ont choisi de définir de nouveaux mouvements possibles pour différentes parties du visage. Et lorsque ce comportement était semblable dans plusieurs acteurs, ils ont défini des procédures. Ils ont ainsi nommé l'action qui devait en résulter, ils l'ont définie à part et y ont fait référence dans les différents acteurs concernés. Ainsi, le concept de procédure semble relativement aisé à faire émerger chez les élèves au sein d'un cours sur la concurrence.

5 Conclusion

Au regard des résultats obtenus, nous tirons les conclusions suivantes.

Tout d'abord, comme le souligne la partie terminologique du test, la notion de concurrence semble relativement rapide à assimiler pour eux : ils sont capables après deux séances de 100 minutes d'en donner une définition correcte et d'utiliser cette définition pour classifier des systèmes. En outre, les justifications qu'ils donnent pour classifier ces systèmes semblent cohérentes avec la définition donnée. En particulier, ils mettent rapidement de côté les préconceptions comme quoi la concurrence serait liée à une forme de compétition économique ou sportive.

Ensuite, pour relativiser un peu, nous constatons que les deux séances de cours ne suffisent pas à modifier fondamentalement la façon dont ils tentent de résoudre le problème de concurrence fourni. En effet, la répartition des réponses en différentes catégories est relativement semblable au post-test et au pré-test. Nous notons juste une augmentation du nombre de réponses raisonnables et une légère augmentation du nombre de réponses impliquant de la communication explicite entre les agents du système. Nous pensons qu'une ou deux séances supplémentaires permettant la réalisation d'un système concurrent de façon moins guidée permettrait d'améliorer leur capacité à résoudre des problèmes de ce type, car ils y seraient confrontés directement.

Finalement, nous concluons que, étant donné les résultats obtenus et les réactions des élèves face à cette activité sur la concurrence, rien n'indique que l'ap-

prentissage de la programmation concurrente ne soit pas adaptée à des jeunes de 12 à 15 ans. Ils sont capables, en très peu de temps, de comprendre le concept de concurrence, de l'utiliser, de participer à l'implémentation d'un système concurrent par passage de messages et de donner des solutions simples à un problème de synchronisation. Nous observons d'ailleurs que, bien qu'ayant un nombre de réponses raisonnables bien inférieur à ceux rapporté dans les précédentes études (60%, contre 69% et 83.7%), nous avons plus de 60% des étudiants qui ont été capables de fournir au moins une réponse raisonnable. Toutefois, ces résultats devraient être approfondis au moyen de séances supplémentaires qui permettraient aux élèves de se confronter réellement aux problèmes d'un système concurrent et de développer par eux-mêmes des solutions de plus grande taille.

Références

1. ACM/IEEE-CS Joint Task Force on Computing Curricula : Computer science curricula 2013. Tech. rep., ACM Press and IEEE Computer Society Press (2013)
2. Brabrand, C. : Constructive Alignment for Teaching Model-Based Design for Concurrency, pp. 1–18. Springer Berlin Heidelberg (2008)
3. Carro, M., Herranz, A., Mariño, J. : A model-driven approach to teaching concurrency. Trans. Comput. Educ. **13**(1), 5 :1–5 :19 (2013)
4. Feldman, M.B., Bachus, B.D. : Concurrent programming can be introduced into the lower-level undergraduate curriculum. SIGCSE Bull. **29**(3), 77–79 (1997)
5. Kolikant, Y.B.D. : Gardeners and cinema tickets : High school students' preconceptions of concurrency. Computer Science Education **11**(3), 221–245 (2001)
6. Lewandowski, G., Bouvier, D.J., McCartney, R., Sanders, K., Simon, B. : Commonsense computing (episode 3) : concurrency and concert tickets. In : Proceedings of the third international workshop on Computing education research. pp. 133–144. ACM (2007)
7. Libert, C., Vanhoof, W. : Analysis of students' preconceptions of concurrency. In : Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence. pp. 9–12. ACM (2019)
8. Sadowski, C., Ball, T., Bishop, J., Burckhardt, S., Gopalakrishnan, G., Mayo, J., Musuvathi, M., Qadeer, S., Toub, S. : Practical parallel and concurrent programming. In : Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education. pp. 189–194. SIGCSE '11, ACM (2011)
9. Smal, A., Frenay, B., Henry, J. : School-it : une «mallette numérique» pour éduquer à l'informatique (descriptif de poster). In : Didapro 7–DidaSTIC. De 0 à 1 ou l'heure de l'informatique à l'école (2018)
10. Stein, L.A. : Challenging the computational metaphor : Implications for how we think (1999)
11. Sutter, H. : The Free Lunch Is Over : A Fundamental Turn Toward Concurrency in Software. Dr. Dobb's Journal **30**(3) (2005)
12. Van Roy, P., Armstrong, J., Flatt, M., Magnusson, B. : The role of language paradigms in teaching programming. SIGCSE Bull. **35**(1), 269–270 (2003)

Une preuve pour le lycée de l'indécidabilité du problème de l'arrêt

Matthieu Journault[1], Pascal Lafourcade[2], Rémy Poulain[1], and Malika More[2]

[1] Sorbonne Université, LIP6
[2] Université Clermont Auvergne, LIMOS, IREM

Résumé L'objectif de cette activité est de faire démontrer par des élèves à partir de la troisième qu'il existe des problèmes informatiques qui sont *indécidables*, c'est-à-dire pour lesquels il n'existe pas d'algorithme pour les résoudre. Pour montrer qu'il existe de tels problèmes, l'activité en exhibe un, *le problème de l'arrêt* : étant donnée une paire constituée de l'encodage d'une machine de Turing M et d'un paramètre d'entrée w, l'exécution de M sur w s'arrête-t-elle ? L'activité suit la preuve proposée par Turing, mettant ainsi en avant le principe du raisonnement par l'absurde et la notion de disjonction de cas. Tout au long de l'activité, les élèves se servent d'une version papier d'un modèle simplifié d'ordinateur pour exécuter à la main des programmes (ici écrits en Scratch). Ceci leur permet d'abord de découvrir des programmes simples, mais aussi de se familiariser avec un modèle de calcul proche de celui d'une machine à registres. Certains de ces programmes simples sont ensuite composés ensemble pour obtenir le résultat d'indécidabilité.

1 Introduction

La conception de programmes est une activité omniprésente à la fois dans toutes les sous-disciplines de l'informatique, mais aussi à tout les niveaux de compétences, c'est d'ailleurs souvent par la conception de programmes que débute l'apprentissage de l'informatique. Certaines activités d'initiation peuvent prendre la forme d'activités utilisant les langages de programmation Scratch ou bien Snap ! [Ber11]. D'autres peuvent se faire sans ordinateur (ou algorithmique débranchée [BRC12]), on peut citer par exemple l'activité du crêpier, au cours de laquelle un tas de crêpes est classé à l'aide d'une spatule [Bul13], ou encore les activités développées à l'IREM/MPSA de Clermont-Ferrand dans le groupe Informatique Sans Ordinateur [3] auquel appartiennent deux des auteurs. Cet article propose une activité sans ordinateur, utilisant comme support le langage de programmation Scratch. Les programmes utilisés peuvent sans aucune difficulté être traduits dans un autre langage de programmation, en fonction des connaissances des élèves.

Il nous semble important de montrer rapidement aux élèves que certains problèmes informatiques n'ont pas de solution algorithmique. Ce résultat fondamental de l'informatique, démontré en 1936 par Alan Turing [Tur36], peut être

3. http ://www.irem.univ-bpclermont.fr/informatique-sans-ordinateur-140

présenté assez simplement à des élèves de fin de collège et de lycée. Notons à ce propos qu'il sera attendu des élèves de la spécialité NSI en classe de Terminale de « *Comprendre que tout programme est aussi une donnée. Comprendre que la calculabilité ne dépend pas du langage de programmation utilisé. Montrer, sans formalisme théorique, que le problème de l'arrêt est indécidable* » [Nat19]. Le but de cet article est de présenter une activité débranchée qui fait travailler les élèves sur chacun de ces points. Cette activité peut être proposée à partir de la troisième. Pour des élèves plus grands, les programmes Scratch peuvent être remplacés par des programmes dans un autre langage de programmation comme Python ou bien en pseudo code.

Cette activité aborde de plus les problématiques suivantes :

— Lecture de programmes ;
— Exécution pas à pas d'un programme ;
— Composition de programmes ;
— Raisonnement par l'absurde ;
— Raisonnement par disjonction de cas ;
— Démonstration en informatique ;
— Modèles de calcul.

Cette activité ne nécessite pas d'ordinateur, mais un peu de matériel imprimable pour représenter la machine ainsi que des programmes Scratch à imprimer [4]. Afin de permettre les interactions entre les élèves, ils peuvent être répartis en groupes de trois. La description faite dans cet article suit la progression pédagogique créée afin de prouver l'indécidabilité du problème de l'arrêt. Lors de la création de cette activité nous nous sommes inspirés d'une vidéo [5] de la chaîne Youtube *Udiprod*. Il est intéressant de montrer cette vidéo aux élèves une fois l'activité finie, afin de synthétiser la preuve, ceci en sept minutes et cinquante et une secondes.

Plan : Dans la section 2, une présentation du matériel nécessaire pour l'activité est faite et des activités préparatoires sont proposées pour introduire progressivement les programmes et notions nécessaires pour la preuve d'indécidabilité. Dans la section 3, une activité simple pour sortir d'un labyrinthe est décrite afin d'introduire les notions de preuve par disjonction de cas mais aussi de preuve par l'absurde. Dans la section 4, l'activité en elle-même est décrite. Enfin, avant de conclure dans la dernière section, nous décrivons dans la section 5 différents retours d'expérience pour cette activité dans trois situations différentes, au collège, au lycée et en formation continue d'enseignants.

2 Activités préparatoires

Tout d'abord nous présentons le matériel nécessaire pour l'activité et ensuite nous décrivons chacun des programmes préparatoires à la preuve du théorème d'indécidabilité.

4. http ://www.irem.univ-bpclermont.fr/Indecidabilite-du-probleme-de-l
5. https ://www.youtube.com/watch ?v=92WHN-pAFCs

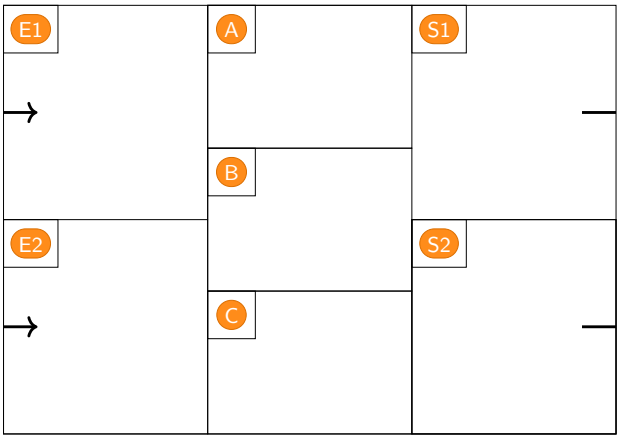


FIGURE 1: Représentation d'un ordinateur.

2.1 Matériel

Afin de représenter l'exécution de programmes sur un ordinateur, nous adoptons un modèle très simplifié de celui-ci. L'ordinateur est modélisé par une "ardoise" sur laquelle est dessiné le schéma de la Figure 1 ou bien par une feuille de papier sur laquelle est imprimé ce même schéma. L'activité dure une heure en comptant le temps pour faire des remarques d'introduction et de conclusion.

Cet ordinateur simplifié comporte deux entrées dénotées E1 et E2, deux sorties dénotées S1 et S2, et trois cases mémoires dénotées A, B et C. L'objectif de la prochaine section est de se familiariser avec le fonctionnement de cet ordinateur simplifié à l'aide de programmes Scratch simples qui sont réutilisés dans la preuve de la section 4. De plus, c'est une bonne façon de familiariser les élèves avec un modèle de calcul abstrait. Le modèle d'ordinateur utilisé dans cette activité est proche de ce qu'un informaticien appellerait une machine à trois registres.

2.2 Exécution de programmes simples

Le terme *machine* désigne un groupe de trois participants, équipé d'une ardoise sur laquelle est dessiné le schéma de la Figure 1. Cette ardoise sert à représenter l'évolution de l'état de la machine lors de l'exécution d'un programme.

Nous expliquons, via des exemples simples de programmes Scratch donnés dans les Figures 2 à 7, comment des programmes s'exécutent sur notre modèle d'ordinateur. Pour chaque exemple, le programme est donné à chaque groupe d'élèves. Ils doivent exécuter le programme sur des entrées choisies par le groupe. L'ordre dans lequel ces exemples sont présentés aux élèves correspond à une progression pédagogique afin de leur faciliter la compréhension du fonctionnement du modèle d'ordinateur utilisé dans l'activité.

INCRÉMENT *(Figure 2) :* Chaque élève du groupe choisit à son tour un entier qu'il écrit en entrée E1, et observe qu'au terme de l'exécution de ce programme, il obtient l'entier suivant écrit en sortie S1. L'objectif est simplement de se familiariser avec l'utilisation du matériel sur un programme simple qui ajoute 1 à l'entrée choisie.

INCRÉMENT *puis* INCRÉMENT *:* Ce deuxième temps de l'activité nécessite deux ardoises par groupe d'élèves. L'objectif est d'attirer l'attention sur le fait que les ardoises distribuées aux élèves (voir Figure 1) comportent des demi-flèches, sortantes sur les cases S1 et S2 et entrantes sur les cases E1 et E2. En accolant deux machines M_1 et M_2, on en obtient ainsi une nouvelle machine M, dont le comportement est le suivant :

- les entrées de la machine M sont les entrées de M_1 ;
- les sorties de la machine M sont les sorties de M_2 ;
- exécuter M revient à exécuter M_1, recopier les résultats obtenus dans les sorties de M_1 dans les entrées de M_2 (S1 dans E1, S2 dans E2), puis à exécuter M_2.

Il s'agit ici d'appréhender la notion de composition de programmes : chaque groupe d'élèves utilise deux machines, chacune exécutant le programme INCRÉMENT. Il est important de tester la machine composée ainsi obtenue sur plusieurs entiers et d'insister sur la transmission des sorties de l'une dans les entrées de l'autre.

PHOTOCOPIE *(Figure 3) :* Ce programme est à exécuter d'abord en choisissant une entrée numérique, et ensuite en choisissant comme entrée un programme. Pour cela, les fichiers fournis sur le site [6] contiennent une version du programme INCRÉMENT imprimé en petit pour qu'il rentre dans les cases E1, S1 et S2. Il en faut trois exemplaires pour cet exercice. Ensuite, le programme PHOTOCOPIE est exécuté en prenant comme entrée le programme PHOTOCOPIE lui-même, il faut donc aussi trois exemplaires d'une "petite version" de ce programme. Le but de ces exercices est de présenter un point crucial de la la preuve d'indécidabilité : le fait que les ordinateurs peuvent accepter en entrée non seulement des nombres mais aussi des programmes. Ce comportement surprend au premier abord les élèves mais, après discussion avec l'enseignant, ne leur pose pas de problème de compréhension.

MOINS *(Figure 4) :* Dans chaque groupe, chaque élève à son tour choisit deux entrées avec $E1 > E2$, et exécute ce programme. L'objectif est de se familiariser avec un programme qui utilise des variables et une boucle (et dont l'exécution se termine, voir ci-dessous).

MOINS* *(Figure 5) :* Dans chaque groupe, chaque élève à son tour choisit deux entrées avec $E1 > E2$, et exécute ce programme. Il s'agit souvent de la

6. http ://www.irem.univ-bpclermont.fr/Indecidabilite-du-probleme-de-l

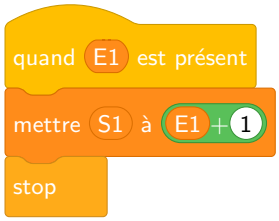


FIGURE 2: Programme INCRÉMENT

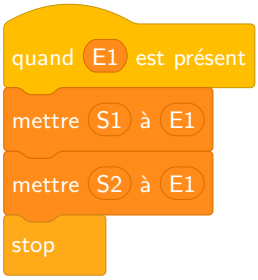


FIGURE 3: Programme PHOTOCOPIE

première rencontre avec un programme qui boucle indéfiniment, c'est-à-dire dont l'exécution ne se termine pas. Il est important d'expliquer que ce programme a été obtenu à partir du programme MOINS en "oubliant" simplement une instruction, et qu'il est par conséquent souvent difficile de déterminer si l'exécution d'un programme se terminera ou pas en lisant son code.

SUPER *(Figure 6) :* Dans chaque groupe, chaque élève à son tour choisit une entrée, et exécute ce programme. Après quelques essais, l'enseignant demande aux élèves d'expliquer sur quelles entrées l'exécution de ce programme ne se termine pas et sur quelles entrées elle se termine. L'exécution de ce programme se termine sur certaines entrées (les nombres négatifs ou nuls), mais ne se termine pas pour les nombres positifs. Trouver ces deux conditions demande un peu de réflexion pour les élèves, et cela montre qu'il n'est pas facile de déterminer si l'exécution d'un programme sur une entrée donnée se terminera ou pas, juste en lisant son code.

NÉGATION *(Figure 7) :* Il s'agit maintenant de demander aux élèves de chaque groupe de tester ce programme sur les entrées SE TERMINE puis NE SE TERMINE PAS (noter ici que SE TERMINE et ne NE SE TERMINE PAS sont des chaînes de caractères, s'ajoutant ainsi aux types de données que nos machines sont autorisées à manipuler). Ils peuvent constater que son comportement est inverse de ce qu'il reçoit en entrée : l'exécution du programme NÉGATION sur l'entrée SE TERMINE ne se termine jamais, alors que sur l'entrée NE SE TERMINE PAS (ou n'importe quelle autre valeur), elle se termine.
Les programmes NÉGATION et PHOTOCOPIE sont au cœur de la démonstration du théorème d'indécidabilité, il est donc important que les élèves comprennent bien leur fonctionnement.

3 Preuve par disjonction de cas et preuve par l'absurde

Afin d'introduire la notion de preuve par disjonction de cas, qui est très utilisée en informatique, et également la notion de preuve par l'absurde, nous

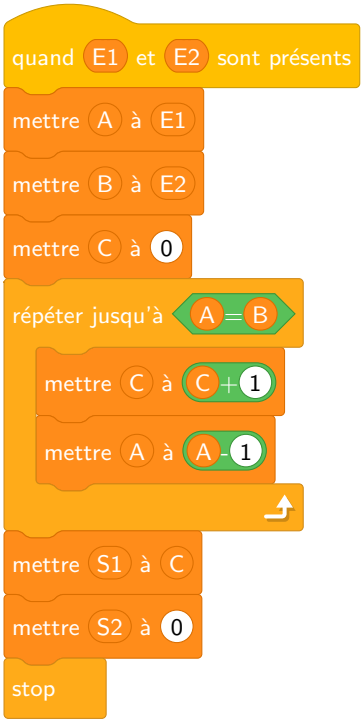


FIGURE 4: Le programme MOINS, sous la condition que $E1 > E2$.

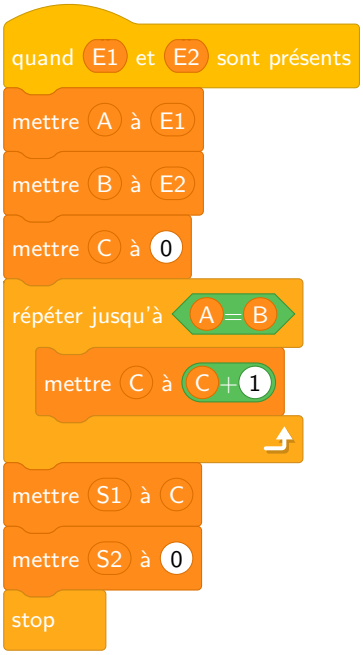


FIGURE 5: Le programme MOINS*, sous la condition que $E1 > E2$.

considérons dans cette section une situation simple et accessible sans mathématiques. Soit le labyrinthe de la Figure 8, qui contient un précipice mortel et infranchissable noté P et un monstre invincible noté M. Alice part du point A, et il s'agit de montrer qu'elle ne peut pas sortir vivante du labyrinthe. Les élèves se convainquent bien entendu facilement que c'est vrai, et l'activité a pour objectif d'expliciter une démonstration de ce résultat.

La preuve est une preuve par l'absurde. Supposons qu'Alice puisse sortir vivante du labyrinthe. Elle a seulement deux choix possibles à partir de sa position initiale A : partir vers la droite ou partir vers le bas.

Examinons les deux cas et montrons qu'ils sont tous deux impossibles.

1. **Vers la droite :** Si Alice part vers la droite, alors elle va finir par tomber dans le précipice P. Donc elle ne sortira pas vivante du labyrinthe par ce chemin. Cette solution est impossible.
2. **Vers le bas :** Si Alice part vers le bas, alors elle va se faire dévorer par le monstre M. Donc elle ne sortira pas vivante du labyrinthe par ce chemin. Cette solution est impossible également.

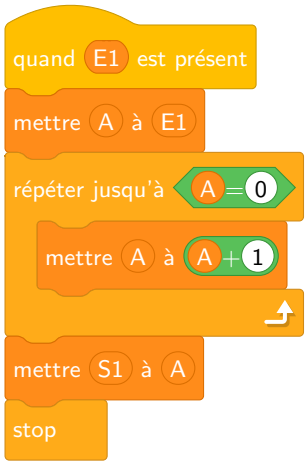


FIGURE 6: Programme SUPER

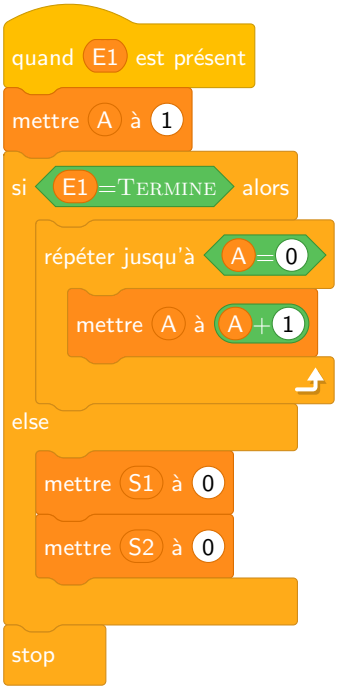


FIGURE 7: Le programme NÉGATION.

Finalement, nous pouvons conclure que l'hypothèse selon laquelle Alice peut sortir vivante du labyrinthe est fausse. Nous en déduisons qu'Alice ne peut pas sortir vivante du labyrinthe. Cette partie de l'activité peut prendre la forme d'une discussion interactive en classe entière avec les élèves, avec l'objectif d'expliquer pas à pas la preuve et de montrer comment utiliser les deux techniques de preuve sur un "théorème" simple.

4 Preuve d'indécidabilité

Il s'agit d'une preuve par l'absurde. Nous supposons l'existence d'un programme, appelé ARRÊT, ayant deux données d'entrée : la première représente un programme P et la seconde représente une donnée d'entrée E du programme P. Le programme ARRÊT détermine *sans jamais se tromper* si l'exécution du programme P sur l'entrée E se termine ou pas. Dans un premier temps, l'enseignant va manipuler ce programme ARRÊT avec les élèves, avant de montrer qu'un tel programme ne peut pas exister. La preuve est constructive : nous allons construire un programme qui contredit l'existence du programme ARRÊT à partir des trois programmes PHOTOCOPIE, ARRÊT et NÉGATION.

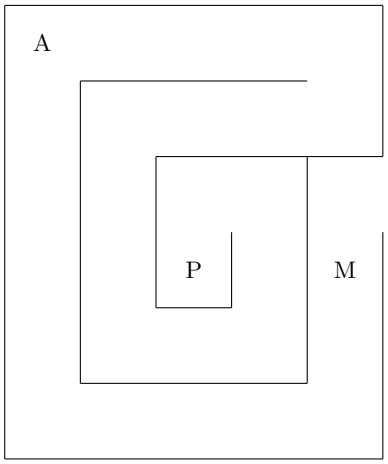


FIGURE 8: Labyrinthe.

4.1 Le programme Arrêt

Soit le programme, appelé ARRÊT, qui prend en entrées E1 et E2, et répond SE TERMINE si l'exécution du programme E1 sur l'entrée E2 se termine et NE SE TERMINE PAS sinon. Nous fournissons en Figure 9 ce programme ARRÊT écrit en "Scratch". Il est bien évidemment fallacieux, puisque l'activité démontre qu'il n'existe pas. Les élèves doivent être conscients que la ligne importante car difficile (en réalité impossible) à réaliser est le test :

l'exécution du programme E1 se termine sur l'entrée E2.

Dans chaque groupe, les élèves exécutent le programme ARRÊT sur les entrées suivantes afin de se familiariser avec son fonctionnement :

1. E1 est le programme INCRÉMENT et E2 est un nombre quelconque.
2. E1 est le programme SUPER et E2 est 5.
3. E1 est le programme SUPER et E2 est 0.

Pendant ces exécutions, il est important de faire remarquer aux élèves que, contrairement aux exécutions précédentes où il suffisait de suivre pas à pas les instructions des programmes, il faut ici réfléchir pour exécuter le test vert.

4.2 Preuve de l'indécidabilité

Une fois toute ces activités réalisées, nous sommes prêts pour démontrer par l'absurde que le programme ARRÊT ne peut pas exister. Pour cela, nous supposons que ce programme existe c'est-à-dire qu'il existe un programme qui se comporte comme nous l'avons vu à la section précédente. La preuve repose sur

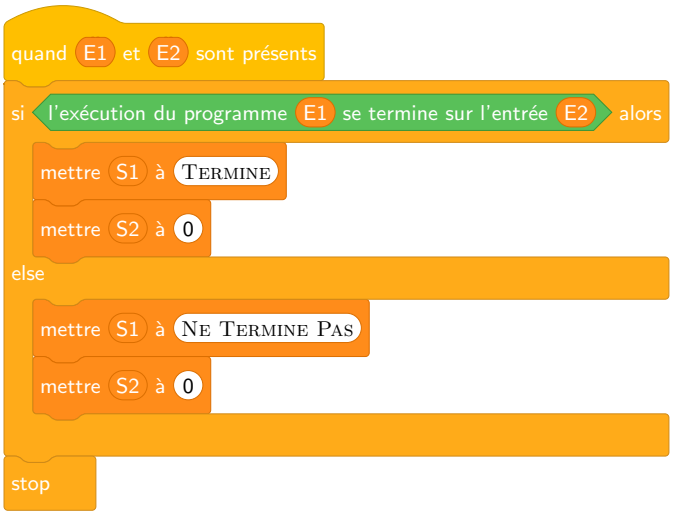


Figure 9: Le programme Arrêt.

la construction d'un programme, noté X, qui utilise le programme Arrêt et qui permet de contredire l'existence de ce même programme. Le programme X est le composé de trois programmes. Le premier programme est le programme Photocopie, le deuxième est le programme Arrêt, et le troisième est le programme Négation. L'idée est d'exécuter le programme X avec comme entrée le programme X lui-même. Cela revient, pour commencer, à exécuter le programme Photocopie en mettant X dans l'entrée E1. Ensuite, par construction du programme X, il faut déterminer si l'exécution du programme Arrêt donne pour résultat Se Termine ou Ne Se Termine Pas quand le programme X lui est passé à la fois en entrée E1 et en entrée E2. Il faut donc distinguer deux cas pour le résultat de l'exécution du programme Arrêt sur les entrées X et X.

1. L'exécution du programme Arrêt sur les entrées X et X donne pour résultat Se Termine. Dans ce cas, le programme Négation reçoit comme entrée Se Termine et son exécution ne se termine jamais. Finalement, dans ce cas, l'exécution du programme X appliqué à l'entrée X ne se termine pas, ce qui contredit le résultat donné par le programme Arrêt. Ce dernier programme étant réputé ne jamais se tromper, ce cas est impossible.

2. L'exécution du programme Arrêt sur les entrées X et X donne pour résultat Ne Se Termine Pas. Dans ce cas, le programme Négation reçoit comme entrée Ne Se Termine Pas et son exécution se termine. Finalement, dans ce cas, l'exécution du programme X appliqué à l'entrée X se termine, ce qui contredit le résultat donné par le programme Arrêt.

Ce dernier programme étant réputé ne jamais se tromper, ce cas aussi est impossible.

Ainsi, nous aboutissons dans les deux cas à une contradiction, ce qui montre l'impossibilité de l'existence du programme ARRÊT. Il n'existe donc pas de programme qui détermine d'une façon générale si un programme donné se termine sur une entrée donnée.

5 Retours d'expérience en classe

L'activité a été testée par les auteurs dans trois situations différentes :
- En classe de troisième, avec des groupes de 14 élèves.
- En classe de seconde, en classe entière de 35 élèves.
- En formation continue d'enseignants dans le cadre du Diplôme Inter-Universitaire Enseigner l'Informatique au Lycée, avec des professeurs du secondaire de plusieurs disciplines scientifiques (mathématiques, informatique et technologie).

Les élèves travaillaient par groupe de 3 ou 4 pendant une séance d'une heure. Ils commencent par manipuler les premiers programmes, ensuite pour la preuve le programme ARRÊT leur est présenté et enfin le programme X leur est donné avec la consigne de regarder ce qui se passe quand X s'exécute avec comme entrée le code de lui-même.

5.1 Classe de Troisième

Ce groupe a testé l'activité juste après sa création. Les élèves ont été très perturbés par la démonstration par l'absurde, difficile à appréhender pour des élèves si jeunes. Ainsi nous avons ajouté l'exemple du labyrinthe pour clarifier ce point. Nous pouvons citer quelques retours d'élèves : *"C'était intéressant car j'y ai compris l'ordinateur et le fonctionnement de l'intérieur."*. Pour cet élève, plus que de comprendre la démonstration elle-même, le grand apport de cet atelier a été de comprendre l'exécution des programmes. Mais certains autres ont bien suivi la preuve et compris la portée du résultat : *"Cela nous a permis de comprendre que l'ordinateur ne peut pas tout faire et qu'il est bien moins intelligent que ce que l'on peut supposer."*. Il est satisfaisant de constater que dans la majorité des retours, les élèves avaient réussi à dégager une bonne intuition de ce théorème fondamental de l'informatique.

5.2 Classe de Seconde

Dans le cadre de l'option Informatique et Création Numérique (ICN) en Seconde, l'un des auteurs est intervenu dans une classe de 35 élèves pour réaliser cette activité en présence du professeur. Les élèves avaient déjà écrit des programmes en Scratch, ce qui a permis d'aller plus rapidement sur les programmes simples. Par contre, ils n'avaient jamais vu de programmes dont, involontairement, l'exécution ne se termine pas. L'activité avec le labyrinthe a permis

d'introduire le raisonnement par l'absurde et par disjonction de cas. La réussite de cette étape est cruciale pour que les élèves soient ensuite bien outillés pour faire la preuve du résultat principal d'indécidabilité. En reprenant pas à pas ce que nous avions fait pour le labyrinthe, les élèves sont arrivés par eux même à concevoir la preuve attendue, bien entendu une fois que nous leur avons présenté la machine X.

Le résultat démontré a vraiment surpris les élèves et les a fait beaucoup réfléchir sur ce qu'il est possible de faire ou non avec un ordinateur.

5.3 Formation continue de professeurs du secondaire

L'activité a été présentée dans le cadre du Diplôme Inter-Universitaire Enseigner l'Informatique au Lycée, destiné à former les professeurs de la spécialité Numérique et Sciences Informatiques (NSI) en première et en terminale. La plupart des enseignants présents ont découvert ce résultat fondamental lors de cette séance. Ils ont été surpris par la puissance de ce résultat, et ont aussi pris conscience qu'il faut faire attention lorsqu'ils construisent des énoncés pour leurs devoirs car certains pourraient ne pas avoir de solution. À cette occasion, nous avons parlé d'autres problèmes indécidables classiques, comme le théorème de Rice [Ric53,Wol06], le pavage du quart de plan par les tuiles de Wang [Rob71], la résolution des équations diophtantiennes [Mat93] [7], le problème de la mortalité des matrices [CHHN14] [8], ou encore le problème des correspondances de Post (PCP) [Sip06]. Les professeurs ont été convaincus de l'utilité de présenter la notion d'indécidabilité à leurs élèves. Ils ont aussi apprécié le fait que la démonstration proposée utilise des programmes Scratch simples qui peuvent facilement être traduit dans un autre langage, la rendant ainsi accessible pour des lycéens.

6 Conclusion

Dans cet article, nous avons présenté une activité pouvant être proposée à des élèves de fin de collège, de lycée ou au niveau post-bac. Cet activité établit l'existence de problèmes informatiques pour lesquels aucun programme ne peut donner le résultat en toute généralité. Il est à noter que la démonstration de ce résultat est maintenant au programme de la spécialité NSI du lycée en classe de Terminale. L'activité peut-être faite sans recours à l'outil informatique et s'inscrit donc une démarche d'informatique sans ordinateur. Malgré l'absence d'ordinateurs, l'activité fait usage de programmes écrits en Scratch qui sont facilement transposables dans un autre langage. Ceux-ci sont exécutés à la main par les élèves mais aussi manipulés comme des données (à l'image d'entiers ou de chaînes de caractères). Bien que l'activité soit faite sans ordinateur, il nous

7. Aussi connu sous le nom de 10ème problème de Hilbert.

8. Il s'agit de savoir si, à partir d'un ensemble de matrices à coefficients entiers, il est possible d'en trouver un sous-ensemble qui, une fois multipliées entre elles, donne la matrice nulle.

a semblé que faire jouer à l'élève le rôle de la machine qui exécute un programme lui permet de comprendre un peu plus le fonctionnement d'un ordinateur. En effet, il est attendu des élèves qu'ils exécutent divers programmes sans prendre d'initiative, à la manière d'un processeur. Notre activité est aussi l'occasion d'aborder une autre notion fondamentale en informatique : la composition de plusieurs programmes, mais aussi de manière duale : la décomposition d'un problème en sous-problèmes. Les notions de preuve par l'absurde et par disjonction de cas sont aussi abordées ce qui est à notre avis le premier obstacle dans la compréhension de la preuve. L'exemple du labyrinthe peut servir pour créer des activités préliminaires pour que les élèves acquièrent ces mécanismes souvent utilisés en informatique. Le second obstacle vient de la compréhension de la contradiction dans chaque cas, cela nécessite de prendre du recul sur ce que la machine fait et ce qu'elle a dit qu'elle allait faire. Enfin, comme souligné en Section 5 le résultat obtenu pendant l'activité donne souvent lieu à des discussions intéressantes sur ce qui peut être attendu ou pas d'un ordinateur, mais aussi sur la distinction entre la puissance d'un ordinateur (sa rapidité) et son pouvoir d'expression (ce qu'il peut faire).

Références

[Ber11] Berkley. Snap !, 2011. https ://snap.berkeley.edu.

[BRC12] Tim Bell, Frances A. Rosamond, and Nancy Casey. Computer science unplugged and related projects in math and computer science popularization, 2012.

[Bul13] Laurent Bulteau. *Ordre et désordre dans l'algorithmique du génome*. PhD thesis, Université de Nantes, Faculté des sciences et des techniques, 2013.

[CHHN14] Julien Cassaigne, Vesa Halava, Tero Harju, and Francois Nicolas. Tighter undecidability bounds for matrix mortality, zero-in-the-corner problems, and more, 2014.

[Mat93] Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, Cambridge, MA, USA, 1993.

[Nat19] Education Nationale. Nouveau programme nsi. Journal officiel, 2019.

[Ric53] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2) :358–366, 1953.

[Rob71] Raphael M Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones mathematicae*, 12(3) :177–209, 1971.

[Sip06] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, second edition, 2006.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42) :230–265, 1936.

[Wol06] Pierre Wolper. *Introduction à la calculabilité*. Dunod, 2006.

Analyse didactique d'un jeu de recherche : vers une situation fondamentale pour la complexité d'algorithmes et de problèmes ⋆

Antoine Meyer[1] and Simon Modeste[2]

[1] LIGM (UMR 8049), UPEM, CNRS, ESIEE, ENPC, Université Paris-Est, 77454 Marne-la-Vallée, France
antoine.meyer@u-pem.fr
[2] IMAG, Université de Montpellier, CNRS, Montpellier – France
simon.modeste@umontpellier.fr

Résumé En analyse d'algorithmes, on s'intéresse souvent à la notion de complexité en temps et dans le pire cas : étant donné un algorithme résolvant un certain problème, peut-on estimer le nombre d'opérations effectuées pour résoudre la pire instance d'une certaine taille ? Une autre notion importante s'intéresse à la complexité du problème lui-même, indépendamment d'un algorithme : peut-on démontrer que tout algorithme résolvant ce problème effectue au moins un certain nombre d'opérations ? Nous questionnons la possibilité d'aborder à divers niveaux d'enseignement (scolaire ou supérieur) les notions de complexité au pire d'un algorithme et de complexité intrinsèque d'un problème. Après une brève présentation des notions visées, nous présentons une famille d'activités de type « débranché » inspirées du classique « jeu de la devinette », dont nous faisons l'hypothèse qu'elles sont de nature à faire émerger ces notions, voire de constituer une situation fondamentale à leur égard. Ces activités reposent en particulier sur la notion d'argument d'adversaire, utilisée en théorie de la complexité. Nous fournissons une analyse a priori de cette famille d'activités et présentons un plan d'expérimentation.

Keywords: complexité · algorithme · problème · ingénierie didactique · jeu · informatique débranchée

1 Introduction

Selon Modeste [10], qui s'appuie sur une revue de littérature informatique et didactique, un algorithme est « une procédure de résolution de problème, s'appliquant à une famille d'instances du problème et produisant, en un nombre fini d'étapes constructives, effectives, non-ambigües et organisées, la réponse au problème pour toute instance de cette famille ». Les algorithmes occupent en France une place croissante dans les programmes scolaires. Ils interviennent en mathématiques et en technologie au cycle 4 (fin du collège, 12 à 15 ans), en

⋆. Travail réalisé avec le soutien financier de l'ANR (projet DEMaIn, référence <ANR-16-CE38-0006-01>).

mathématiques au lycée (seconde, première et terminale, 15 à 18 ans), et surtout dans les nouvelles disciplines SNT [3] et NSI [4], qui identifient les algorithmes comme l'un des quatre « concepts fondamentaux » et « universels » de la science informatique. Déjà au cycle 3 (9 à 12 ans), les élèves y sont familiarisés par le biais d'algorithmes simples de déplacement, de construction de figures, etc.

Suite à la popularisation par Wing [13] du terme *pensée algorithmique*, divers travaux ont tenté d'apporter des précisions sur les compétences exactes que recouvre ce terme. S'appuyant sur une revue de la littérature parue entre 2006 et 2013, Selby et Woolard [11] proposent de retenir cinq compétences principales, que l'on pourrait traduire par (1) la capacité à penser par abstractions, (2) à décomposer un problème, (3) à planifier et exécuter une procédure algorithmique, (4) à évaluer un algorithme selon divers critères, et enfin (5) à généraliser. Dans cet article, nous nous intéresserons dans une certaine mesure à la compétence (3), mais plus particulièrement à la compétence (4), que nous comprendrons ici dans le sens spécifique d'*évaluation de l'efficacité d'un algorithme*, préoccupation généralement associée au thème de la *complexité algorithmique*.

La question de l'enseignement de cette notion dans le cadre de cours d'initiation à l'informatique et à la programmation apparaît dans la littérature. Par exemple, [6] plaide pour une introduction précoce, juste après la présentation des boucles. Ceci est illustré par trois exemples de problèmes principalement arithmétiques. Dans [5], les auteurs décrivent un nouveau type d'exercices destiné à sensibiliser des étudiants de première année d'informatique à la notion de complexité et à l'existence possible d'algorithmes d'efficacité variées pour un même problème. L'originalité de notre travail est d'accorder une importance particulière à l'étude de la complexité de *problèmes* algorithmiques, non spécifique à un ou plusieurs algorithmes particuliers. Ceci permet de poser la question de l'existence d'algorithmes efficaces (ou plus efficaces que d'autres) pour un problème donné, et parfois de démontrer qu'il n'en existe pas.

Dans la suite de cet article, après une brève discussion sur le contexte institutionnel (section 2), nous présentons nos objectifs, hypothèses et questions (3). Après un rappel des notions algorithmiques visées (section 4), nous exposons une famille de situations d'apprentissages liée au problème de la recherche d'élément dans une collection triée (section 5), dont nous faisons l'hypothèse qu'elle est susceptible de constituer une situation fondamentale pour la notion de complexité algorithmique en général, et celle de complexité de problème en particulier. Enfin, nous présentons quelques perspectives de recherche, incluant un plan d'expérimentation et d'analyse à divers niveaux scolaires (section 6).

2 La complexité algorithmique dans les programmes

Pour des élèves découvrant la notion de complexité au pire d'un algorithme, une motivation possible est la recherche d'efficacité et d'économie de moyen. Il est entendu qu'un « bon » algorithme est un algorithme efficace (performant,

3. SNT : Sciences numériques et technologie (seconde)
4. NSI : Numérique et Sciences Informatiques (première et terminale)

rapide, économe en ressources). Ceci rejoint d'ailleurs une tendance naturelle de la recherche en algorithmique. Ce thème de la complexité algorithmique est très nettement pris en compte par les programmes officiels de NSI. Par exemple, dans le programme de première [8], on peut lire :

> Quelques algorithmes classiques sont étudiés. L'étude de leurs coûts respectifs prend tout son sens dans le cas de données nombreuses (...).

Cette exigence se reflète dans les attendus relatifs à plusieurs points de la partie algorithmique du programme. Par exemple, dans le cas du parcours séquentiel d'un tableau, « [o]n montre que le coût est linéaire », tandis que pour les algorithmes de tri par sélection ou par insertion « [o]n montre que leur coût est quadratique *dans le pire cas* ». Le programme aborde également l'algorithme de « recherche dichotomique dans un tableau trié », sans cependant mentionner sa complexité. Dans le programme de NSI de terminale [9], le thème de la complexité d'algorithmes est repris en des termes plus précis :

> On continue l'étude de la notion de coût d'exécution, en temps ou en mémoire et on montre l'intérêt du passage d'un coût quadratique en n^2 à $n \log_2 n$ ou de n à $\log_2 n$. [5]

Il y a ici un implicite important : on compare des algorithmes *résolvant le même problème* : ainsi, on peut supposer que le passage « d'un coût quadratique en n^2 à $n \log_2 n$ » fait allusion au problème du tri et au passage d'algorithmes dits naïfs à des algorithmes plus efficaces (tri par fusion notamment, qui est évoqué dans la suite du programme), et que le passage « de n à $\log_2 n$ » fait allusion au problème de la recherche d'élément dans une collection triée et au passage de l'algorithme de recherche séquentiel à la recherche dite « dichotomique » (même si ce dernier n'apparaît explicitement que dans le programme de première).

Il est donc en réalité question ici d'établir des faits à propos de la complexité de *problèmes*, et non seulement d'algorithmes : en passant de première à terminale, il est par exemple attendu que l'élève comprenne que le *problème* du tri, dont on savait en première qu'il était de complexité $O(n^2)$, est en fait en $O(n \log_2 n)$, comme en attestent les algorithmes étudiés en terminale.

Ainsi, il est légitime de poser la question de la limite de ce phénomène de perfectionnement progressif des algorithmes : pour un problème donné, y a-t-il une borne inférieure à l'efficacité des algorithmes qui le résovent ? Autrement dit, est-on en mesure de démontrer que *tout* algorithme résolvant le problème *doit* avoir une complexité supérieure à un certain seuil ? Si oui, est-il possible de mettre au point un algorithme *optimal*, dont la complexité dans le pire cas est précisément égale à cette complexité minimale ? On peut arguer du fait que l'existence d'algorithmes optimaux pour certains problèmes (dont au moins deux, la recherche par dichotomie et le tri par fusion, sont explicitement visés par les programmes de NSI) justifie l'intérêt d'étudier ces algorithmes, même si cette justification n'apparaît pas de manière explicite dans les programmes.

5. Le programme précise ici que « Le logarithme en base 2 est ici manipulé comme simple outil de comptage (taille en bits d'un nombre entier). », ce qui est inévitable puisque la fonction logarithme n'est pas toujours connue à ce niveau de classe.

3 Hypothèses de recherche

Les observations ci-dessus nous amènent à formuler les hypothèses suivantes :

H1 : L'étude de la complexité algorithmique par le biais des *problèmes*, et non seulement des algorithmes, est signifiante pour la compréhension des enjeux de la pensée algorithmique.

H2 : Il est possible d'aborder, même de manière informelle, les notions de borne supérieure et de borne inférieure de complexité d'un problème dans le cadre scolaire, par l'intermédiaire du jeu à deux joueurs.

H3 : Le problème de la recherche d'élément dans une suite triée est un objet d'étude propice à l'émergence des notions de complexité d'algorithme et de problème à différents niveaux scolaires.

L'hypothèse H1, de nature plutôt épistémologique, est soutenue par le fait que c'est l'étude de la complexité d'un problème (et non d'un unique algorithme) qui fournit à la fois le cadre et l'enjeu de l'analyse comparée d'algorithmes, comme le sous-entendent les programmes de NSI (cf. section 2). C'est également le point de vue général sur le problème algorithmique en tant qu'objet qui donne du sens à la notion d'algorithme optimal.

L'hypothèse H2 s'appuie sur des idées issues de la théorie algorithmique des jeux. Dans ce cadre, les joueurs qui s'opposent cherchent à atteindre des objectifs antagonistes, qui peuvent dans certains cas être formulés comme des problèmes d'optimisation : chaque joueur cherche à maximiser son gain et à minimiser celui de l'adversaire. Nous verrons à la section 4.2 que la question de la complexité d'un problème peut être présentée dans le cadre d'un « jeu » entre l'algorithme, qui cherche à minimiser par exemple le nombre d'opérations utilisées, et son environnement qui cherche au contraire à maximiser cette quantité.

En endossant le rôle de l'algorithme, on peut faire l'hypothèse que l'élève renforce sa compréhension de la notion même d'algorithme interagissant avec un contexte d'exécution donné. En endossant le rôle de l'adversaire, l'élève qui joue « contre » l'algorithme cherche à faire apparaître l'un des pires cas pour celui-ci. On peut penser que cela permet de faire émerger ou de renforcer chez l'élève la notion de complexité dans le pire cas, tout en préparant l'idée de borne inférieure sur la complexité du problème.

Dans le prolongement des hypothèses H1 et H2, l'hypothèse H3 repose en particulier sur la nature du problème considéré et des algorithmes qui permettent de le résoudre, ainsi que sur leur place dans les programmes scolaires. En effet, le problème de recherche d'élément dans une liste triée peut être qualifié de fondamental pour le domaine de l'algorithmique : c'est un cas d'école par sa simplicité, mais son immense importance pratique en fait également un ingrédient essentiel dans un grand nombre d'applications. Par ailleurs, ses propriétés sont bien comprises, et abordables avec un bagage mathématique modéré. Enfin, tout comme le problème du tri, il s'agit d'un exemple de cas où l'on sait démontrer l'existence d'algorithmes *optimaux*, dans un sens que nous préciserons.

D'un point de vue plus didactique, nous défendrons à la section 5.3 l'idée que l'exploitation de ce problème dans l'enseignement peut être propice à l'émergence

chez les élèves des notions visées, en complément (au collège) ou en conformité avec les exigences immédiates des programmes (au lycée et après). Par ailleurs l'algorithme de dichotomie, qui joue un rôle central dans l'étude de ce problème, est également identifié comme un contenu visé par les programmes.

L'objectif de ce travail est de commencer à mettre à l'épreuve ces hypothèses, en élaborant tout d'abord une proposition d'activité d'enseignement autour du problème de la recherche d'élément dans une liste triée.

4 Recherche d'élément dans une liste triée

Dans cette section, nous présentons quelques éléments théoriques qui sous-tendent la situation d'apprentissage dont il sera question à la section 5.

On considère une suite finie indexée $L = (a_1, \ldots, a_n)$ – c'est à dire une liste – d'objets comparables deux à deux selon une relation d'ordre $\prec$. On note $L(i) = a_i$ le i-ème élément de L. On suppose en outre que ces listes sont *triées* selon $\prec$, c'est-à-dire que si $i < j$ alors $L(i) \prec L(j)$. On se place dans un modèle de calcul où les seules opérations disponibles sont des questions de la forme C_i : « Comment se comparent e et $L(i)$? » que nous appellerons simplement « opérations de comparaison », où e est une valeur quelconque, et $i \in \{1, ..., n\}$ est un indice valide de la liste L. Les réponses possibles à cette question sont : $e \prec L(i)$, $e = L(i)$ ou $e \succ L(i)$. On considère le problème fondamental suivant :

Problème 1 (recherche dans une liste triée)
Instance : *couple* (L, e)*, où* $L = (a_i)_{1 \leq i \leq n}$ *est une suite non vide et croissante selon la relation d'ordre* $\prec$.
Résultat : $i \in \{1, \ldots, n\}$ *tel que* $a_i = e$*, ou* 0 *si* $e \notin L$.

4.1 Algorithme et borne supérieure

On remarque qu'un algorithme naïf de parcours exhaustif, comparant e tour à tour à chacun des $L(i)$, résout ce problème :

Algorithme 1 (recherche exhaustive)
RechercheExhaustive***(L, e) :*** *Pour* i *allant de* 1 *à* n*, déterminer si* $a_i = e$*. Si c'est le cas, répondre* i*. Sinon, continuer l'itération. Répondre* 0 *si l'itération se termine sans succès.*

On voit donc que n questions de type C_i peuvent suffire à résoudre le problème à tous les coups. On dit alors que la complexité du *problème* 1 est majorée par $n = |L|$ dans le modèle de calcul considéré. Cette majoration est cependant très grossière, et il existe un algorithme de complexité bien meilleure :

Algorithme 2 (recherche binaire [6])
RechercheBinaire***(L, e) :***
- *On pose* n *la longueur de* L*,* $g = 1$ *et* $d = n$.
- *Tant que* $g \leq d$*, faire ce qui suit :*

  - *On pose* $m = (g + d)/2$ *(arrondi à l'entier inférieur) ;*
  - *Si* $a_m = e$*, l'algorithme répond* m *et s'arrête ;*
  - *Sinon, si* $a_m < e$*, on pose* $g = m + 1$ *;*
  - *Sinon, on pose* $d = m - 1$.
- *Si l'on sort de la boucle avec* $g > d$*, l'algorithme répond* 0 *et s'arrête.*

La complexité dans le pire cas de cet algorithme est bien connue :

Theorem 1. *Sur une instance* (L, e) *telle que* L *contient* n *éléments, l'algorithme 2 effectue dans le pire cas* $\lfloor \log_2(n + 1) \rfloor$ *opérations de type* C_i.

La notation $\lfloor x \rfloor$ utilisée ici désigne la partie entière de x, pour x un réel positif. On peut démontrer ce théorème par récurrence sur n, ou plus informellement observer qu'à chaque itération le nombre de réponses encore possibles (qui sont les entiers de l'intervalle fermé $[g, d]$) est diminué de 1 et divisé par deux, et que l'algorithme s'arrête au plus tard quand cet intervalle devient vide. Quelle que soit la réponse attendue, l'algorithme ne peut donc pas effectuer plus de $\log_2(n + 1)$ itérations, et donc comme chaque itération effectue une et une seule opération C_i, pas plus de $\log_2(n + 1)$ opérations de comparaison en tout. Un examen attentif montre qu'il existe pour tout n des cas où ce nombre d'opérations est également *nécessaire*. On en conclut que la complexité dans le pire cas de l'algorithme de recherche binaire est précisément égal à cette quantité.

L'existence de l'algorithme 2 montre que la complexité du problème 1 est majorée par $\log_2(n + 1)$. Il est en fait possible de démontrer que ce problème ne *peut pas* être résolu à coup sûr en moins de $\lfloor \log_2(n + 1) \rfloor$ opérations de comparaison sur une liste de longueur n.

4.2 Borne inférieure par argument d'adversaire

Comme annoncé, nous allons établir le résultat suivant :

Theorem 2. *Dans le modèle de calcul considéré (opérations de comparaison de type* C_i*), tout algorithme résolvant le problème 1 effectue au moins* $\lfloor \log_2(n+1) \rfloor$ *opérations de comparaison dans le pire cas.*

En conjonction avec le théorème 1, ceci implique le corollaire suivant :

Corollary 1. *Dans le modèle de calcul considéré, l'algorithme 2 de recherche binaire est optimal pour le problème 1.*

Démontrer l'existence d'une borne inférieure sur la complexité d'un problème (et non d'un algorithme particulier) est un exercice un peu délicat. En effet, il faut raisonner de manière générale sur la complexité de *n'importe quel* algorithme le résolvant. On veut en fait démontrer que le meilleur algorithme possible pour

6. Plutôt que « recherche par dichotomie », nous choisissons une appellation proche de l'appellation anglophone pour éviter une possible confusion avec la méthode de dichotomie en mathématiques. Voir [7] pour une discussion à ce sujet.

ce problème doit effectuer, dans certains cas, au moins un certain nombre d'opérations. Ceci n'a de sens que si l'on fixe au préalable un modèle de calcul précis (ce que nous avons pris soin de faire au paragraphe précédent).

Nous allons démontrer ce résultat par un argument appelé argument d'adversaire. Pour cela, on se place dans une situation imaginaire où l'algorithme n'a pas accès directement aux valeurs des éléments qui constituent la liste L, mais est confronté à un environnement antagoniste, appelé *adversaire*, dont le rôle est de fournir les résultats de chaque opération de comparaison de type C_i effectuée.

La taille de la liste L dans les instances (L, e) considérées étant fixée, l'adversaire a pour objectif de pousser l'algorithme à effectuer le plus grand nombre possible d'opérations. En d'autres termes il cherche à « orienter » l'exécution vers l'un des pire cas pour cette taille d'instance. Pour cela, l'adversaire choisit la réponse à apporter à chaque requête de l'algorithme, avec la seule contrainte de ne jamais fournir d'informations contradictoires.

Voici une stratégie possible pour l'adversaire, dans le cadre du problème 1. L'adversaire maintient pour son propre compte une mémoire des bornes (g, d) de l'intervalle des positions de L pour lesquelles l'algorithme ne possède encore aucune information. À chaque nouvelle opération C_i effectuée par l'algorithme, l'adversaire répond selon la procédure suivante :

Algorithme 3 (Stratégie d'adversaire.)
RÉPONSE(i, g, d) *:*

- *Si* $i < g$*, répondre «* $e \succ L(i)$ *» ;*
- *Sinon, si* $i > d$*, répondre «* $e \prec L(i)$ *» ;*
- *Sinon, si* $i \leq (g + d)/2$*, poser* $g = i$ *et répondre «* $e \succ L(i)$ *» ;*
- *Sinon, si* $i > (g + d)/2$*, poser* $d = i$ *et répondre «* $e \prec L(i)$ *».*

On peut montrer qu'au fur et à mesure des appels à RÉPONSE(i, g, d), on a toujours $g \leq d$, les valeurs successives de g croissent et celles de d décroissent. D'autre part, l'adversaire n'a jamais répondu « $e \prec L(i)$ » pour un indice i inférieur à g, ni « $e \succ L(i)$ » pour un indice i supérieur à d. L'adversaire ne contredit donc jamais une de ses réponses précédentes.

Supposons maintenant qu'un algorithme, confronté à cet adversaire, propose un résultat après avoir effectué strictement moins de $\lfloor \log_2(n+1) \rfloor$ comparaisons. On peut montrer que dans ce cas, $g \neq d$. Quel que soit le résultat avancé par l'algorithme, l'adversaire peut alors le mettre en échec en exhibant soit une liste triée L' dans laquelle $L(g) = e$, $L(i) \succ e$ pour tout $i > g$ et $L(i) \prec e$ pour tout $i < g$, soit une liste L'' dans laquelle $L(d) = e$, $L(i) \succ e$ pour tout $i > d$ et $L(i) \prec e$ pour tout $i < d$. Ces deux listes sont cohérentes avec toute l'information précédemment fournie par l'adversaire, mais la réponse de l'algorithme ne peut être juste sur chacune d'elles. Ceci permet de conclure que tout algorithme résolvant le problème 1 doit poser au moins $\lfloor \log_2(n + 1) \rfloor$ questions sur certaines instances (L, e) où L est de taille n.

5 Situation d'apprentissage autour des jeux de recherche

Dans cette section, nous présentons un schéma de situation d'apprentissage de type *informatique débranchée*, dont nous faisons l'hypothèse qu'il peut s'instancier à différents niveaux scolaires, et qu'il est susceptible de constituer une situation fondamentale pour la notion de complexité de problème.

Le terme *informatique débranchée* fait référence à un courant éducatif apparu dans les années 1990 mais d'essor relativement récent (voir par exemple [4,1,12], ainsi que les travaux de plusieurs groupes IREM [7]) dont l'objet est l'élaboration de situations d'apprentissage de l'informatique sans recours à des dispositifs électroniques (ordinateurs, robots, etc.) mais plutôt à du matériel tangible « traditionnel ».Ce courant a été à l'origine de la création d'un grand nombre d'activités portant sur des sujets variés, des plus théoriques aux plus appliqués et s'adressant à tous les niveaux d'apprentissage. Parmi ces activités, le jeu à un ou plusieurs joueurs tient un rôle particulier, comme c'est le cas dans d'autres disciplines comme les mathématiques.

5.1 Présentation générale de la famille de situations

Déroulement du jeu et conditions de victoire. Dans sa version la plus élémentaire, le jeu que nous considérons implique deux rôles, le *devinant* et le *répondant*. Le devinant cherche à déterminer un nombre entre 1 et n, en proposant un par un des candidats $i \in \{1, \ldots, n\}$. Le répondant indique, pour chaque proposition du devinant, si le nombre cherché est égal au candidat proposé (réponse dite « positive »), s'il est plus petit ou s'il est plus grand (réponses « négatives »). La partie s'arrête dès que le répondant donne une réponse positive.

Un objectif secondaire du devinant est d'obtenir une réponse positive en proposant le moins de candidats possibles, tandis que le répondant cherche à lui en faire poser le plus possible. Ceci est concrétisé par un nombre k convenu à l'avance : si le répondant donne une réponse positive après moins de k propositions, le devinant gagne la partie. Si plus de k candidats ont été proposés, le répondant gagne la partie. Si exactement k questions ont été posées, la partie est nulle. On considérera en outre que la partie est immédiatement gagnée par le devinant si le répondant fait une réponse contradictoire.

Lien avec le problème de la recherche triée. La situation telle qu'elle est proposée au joueur devinant peut être formulée comme la recherche d'un élément e (de valeur non directement accessible à l'algorithme) dans la liste contenant les entiers de 1 à n dans l'ordre. L'objectif d'apprentissage, suscité en particulier par le choix des conditions de victoire, est d'une part de faire émerger une stratégie de recherche efficace chez le devinant (l'algorithme de recherche binaire présenté au paragraphe 4.1) et une stratégie de réponse efficace chez le répondant (la stratégie d'adversaire présentée au paragraphe 4.2), et d'autre part de susciter

7. IREM : Institut de Recherche sur l'Enseignement des Mathématiques (http://www.univ-irem.fr/). Voir par exemple les IREM de Grenoble ou Clermont-Ferrand.

une réflexion sur les issues possibles des parties en fonction de la valeur du paramètre k, autrement dit sur la complexité exacte du problème.

Pour des élèves ou étudiants plus avancés, l'étude peut aller jusqu'à une mise en situation mettant en jeu la formulation générale du problème de la recherche d'élément dans une liste triée, la programmation des stratégies de chacun des deux rôles, la recherche d'une expression générale de k en fonction de n, voire jusqu'à une exigence de démonstration, mais ce ne sont évidemment pas des objectifs pour les plus petites classes. Selon les cas, les objectifs d'apprentissages secondaires peuvent donc inclure des éléments de programmation, la maîtrise des fonctions $\log_2(n)$ et/ou 2^n, la recherche et la résolution d'un problème mathématique, la preuve (par récurrence ou par invariant), etc.

5.2 Exemple de mise en œuvre

Voici un exemple d'instanciation de cette situation, pour des élèves plutôt jeunes (cycle 3 par exemple, élèves de 9 à 12 ans environ). Le dispositif peut être adapté selon un grand nombre de variables, dont nous détaillerons un certain nombre dans le paragraphe suivant.

Temps 1 : On fixe $n = 15$ et $k = 4$. Les élèves jouent par trois, un devinant, un répondant et un arbitre. L'arbitre demande à l'élève répondant d'inscrire un nombre sur une feuille de papier, et place cette feuille à l'écart face cachée. Au cours de la partie, il note chaque proposition du devinant et contrôle la véracité des réponses du répondant. À l'issue de la partie il compte le nombre de propositions, dévoile la feuille de papier préalablement cachée, et désigne le vainqueur (non par son prénom mais par son rôle, devinant ou répondant). Les rôles changent et les élèves recommencent jusqu'à échéance du temps imparti.

Temps 1' : On confronte les scores des deux rôles dans chacun des trinômes, et on met en commun les stratégies de recherche trouvées. Une question est posée : le joueur devinant est-il certain de pouvoir trouver en 4 coups ou moins ? Quelques parties sont jouées entre l'enseignant (dans le rôle de devinant) et les élèves de la classe (en concertation).

Temps 2 : On conserve $n = 15$ et $k = 4$. Le dispositif est le même qu'au temps 1, mais on ne demande plus au répondant de consigner un nombre à l'avance. L'arbitre vérifie seulement que les réponses sont cohérentes entre elles. Si nécessaire, l'enseignant indique que le répondant n'est plus obligé de choisir un nombre, du moment qu'il ne se contredit pas.

Temps 2' : On confronte les scores des deux rôles dans chacun des trinômes, et on met en commun les stratégies de réponse trouvées. Une question est posée : le répondant est-il certain de pouvoir forcer une partie nulle ou de gagner ? Quelques parties sont jouées entre l'enseignant (dans le rôle de répondant) et les élèves de la classe (en concertation), sous le contrôle d'un arbitre (toutes les parties devraient être nulles si les élèves appliquent une stratégie de recherche binaire correcte, et gagnées par l'enseignant sinon).

Temps 3 : On dresse un bilan des observations, en cherchant à expliquer le choix du paramètre $k = 4$. Que se passe-t-il si l'un des deux joueurs dévie des stratégies mises au point ? Que se passerait-t-il si l'on choisissait un nombre k plus petit ou plus grand ? Quel valeur faudrait-il choisir pour k pour garantir que toutes les parties soient nulles quand $n = 7$? Quand $n = 31$? Quand $n = 20$?

5.3 Analyse de la situation proposée

Le dispositif central est la phase de jeu antagoniste illustrée par le temps 2 décrit ci-dessus : les élèves jouent l'un contre l'autre sous le contrôle d'un arbitre et cherchent à gagner la partie. Il s'agit donc d'une situation *adidactique* au sens de la théorie des situations didactiques [3,2]. On peut arguer du fait que cette situation est fondamentale pour la notion de complexité de problème, au sens où son émergence est requise pour établir l'optimalité de la stratégie des joueurs.

En effet, l'enjeu de la situation est de discuter la possibilité pour chaque acteur (algorithme ou adversaire) de gagner ou de faire partie nulle *à tous les coups*. Acquérir la certitude de ne jamais perdre nécessite donc de la part de l'élève qui joue l'un des deux rôles d'avoir pris conscience du caractère indépassable de la borne k utilisée au cours du jeu : indépassable par l'algorithme, qui ne peut espérer gagner en moins de k questions, et indépassable pour l'adversaire, qui ne peut forcer l'algorithme à poser plus de k questions. Cette situation de status quo est précisément le résultat d'optimalité recherché : on a montré que le problème n'était pas résoluble en moins de k questions, et on a trouvé un algorithme qui peut atteindre cette borne inférieure.

On remarque que la stratégie initiale de recherche exhaustive ne permet pas au devinant d'atteindre son objectif, ce qui justifie l'élaboration d'une stratégie plus avancée. De même, la version du dispositif dans laquelle le répondant se contraint à ne pas changer de nombre ne lui permet pas de garantir un match nul. Par ailleurs, cette situation a un caractère fondamental au sens où elle autorise une grande variaté d'implémentations différentes selon les valeurs données aux variables didactiques qui la constituent. En voici quelques exemples :

Choix de n. L'exemple ci-dessus propose au devinant de déterminer un nombre entre 1 et $n = 15$. Dans ce cas, les propositions successives d'un joueur appliquant strictement la stratégie de recherche binaire sont faciles à anticiper (d'abord 8, puis 4 ou 12, puis 2, 6, 10 ou 14, et enfin une des positions impaires). D'autres choix possibles pour n sont (1) une autre valeur de la forme $2^p - 1$ avec $p \geq 0$; (2) une valeur quelconque. Un choix de type (1) autorisera (ou non) la possibilité de pré-calculer ou de calculer de tête toutes les suites de propositions possibles selon la valeur du nombre choisi. Un choix de type (2) complique la recherche en introduisant la nécessité de procéder à des arrondis. Dans les deux cas, une valeur plus grande de n change aussi la difficulté de la stratégie du répondant, qui doit à chaque tour calculer le nombre médian parmi les candidats restants.

Choix de k. Si l'on suppose que n est compris entre 2^{p-1} et $2^p - 1$ avec $p \geq 1$, l'analyse préalable du problème nous apprend que la borne inférieure de complexité est égale à p comparaisons. Ainsi, en fixant $k = p$, deux joueurs suivant

rigoureusement leur stratégie optimale peuvent forcer une partie nulle à chaque fois. Choisir une valeur différente de k introduit un déséquilibre dans les chances de gain des deux joueurs. Une valeur plus petite de k condamnera le devinant à perdre toutes ses parties contre un répondant optimal, et vice-versa. Une manipulation de cette variable met donc en jeu la capacité des élèves à détecter ce déséquilibre et à prévoir les issues des parties.

Statut des joueurs. plusieurs combinaisons de statuts des joueurs sont envisageables. On a déjà évoqué la possibilité pour l'enseignant d'assumer un des rôles face à un élève ou un groupe d'élèves. Il est également possible de faire intervenir un joueur artificiel (programme informatique) jouant parfaitement.

Nature de l'espace de recherche. La formulation générale du problème de recherche dans une collection triée autorise a priori d'autres espaces de recherche qu'un simple intervalle de nombres. On peut envisager toute collection d'éléments comparables deux à deux, triée vis-à-vis d'un ordre quelconque mais non nécessairement constituée d'éléments consécutifs.

Nature de la question. De même, il est possible de modifier la question exacte posée à l'élève devinant. Par exemple, on peut le charger de chercher un indice i tel que le $L(i) = i$, ou encore le plus petit i tel que $L(i) \geq e$.

Une forme un peu plus éloignée, qui rappelle le problème de la méthode de bissection en analyse, demande étant donnés une fonction croissante f, un entier n et un entier e de chercher un entier $1 \leq i \leq n$ tel que $f(i) = e$. Ce cas peut servir de transition vers une étude de la méthode de bissection en elle-même, mais des écueils peuvent surgir dans le passage du discret au continu. Une discussion sur ce sujet est apparue dans [7].

Matériel de jeu. Plusieurs types de matériel de support sont possibles. Pour favoriser une visualisation de l'espace de recherche, on pourra par exemple munir les joueurs d'une bande numérique plastifiée et de feutres effaçables, leur permettant de délimiter l'intervalle de recherche restant à explorer. Pour de grandes valeurs de n ou pour rechercher par tâtonnement la valeur de la borne de complexité, on peut également fournir une calculatrice ou un ordinateur. L'impact exact de ce choix d'organisation reste à analyser.

6 Conclusion et perspectives

Nous avons présenté des hypothèses sur la possibilité d'aborder en classe, par le biais du jeu à deux joueurs, les notions de borne inférieure et de borne supérieure de complexité de problèmes algorithmiques. En particulier, nous avons présenté une situation d'apprentissage autour du problème de la recherche d'élément dans une liste triée, susceptible de permettre un grand nombre d'adaptations à différents niveaux de classe.

Nous souhaitons à court terme finaliser l'analyse a priori de cette situation d'apprentissage et mettre en œuvre un plan d'expérimentation permettant d'évaluer ses effets et la pertinence des variables didactiques identifiées. Nous envisageons de procéder à une expérimentation préalable à petite échelle dans trois

niveaux de classes différents : une classe de cycle 3 (de préférence élémentaire), une classe de cycle 4 et une classe de NSI. Il sera également nécessaire de poser la question de l'évaluation des apprentissages visés par cette activité, en particulier sur la notion de « coût » d'un algorithme, et le cas échéant de proposer des exemples d'évaluations. On pourra se référer en particulier aux futures épreuves de NSI du baccalauréat français ainsi qu'aux évaluations du supérieur, mais la question reste ouverte pour les plus petites classes.

Parmi les perspectives à plus long terme, nous souhaitons élargir l'étude à d'autres situations proches, en particulier autour des d'algorithmes de type « diviser pour régner ». Enfin, nous souhaitons continuer à explorer le thème des interactions entre mathématiques et informatique dans l'enseignement, d'un point de vue épistémologique et didactique. Nous pensons que le présent travail ouvre par exemple des questions intéressantes sur la fonction logarithme en mathématiques et en informatique, ainsi que sur le raisonnement et la preuve.

Références

1. Bell, T., Alexander, J., Freeman, I., Grimley, M. : Computer science unplugged : School students doing real computing without computers. The New Zealand Journal of Applied Computing and Information Technology **13**(1), 20–29 (2009)
2. Bessot, A. : Une introduction à la théorie des situations didactiques (master "mathématiques, informatique" de grenoble 2003-2004). Les cahiers du laboratoire Leibniz **91** (2003)
3. Brousseau, G. : Théorie des situations didactiques. La pensée Sauvage, Grenoble (1998)
4. Fellows, M., Koblitz, N. : Kid krypto. In : Annual International Cryptology Conference. pp. 371–389. Springer (1992)
5. Gal-Ezer, J., Vilner, T., Zur, E. : Teaching algorithm efficiency at cs1 level : A different approach. Computer Science Education **14**(3), 235–248 (2004)
6. Ginat, D. : Efficiency of algorithms for programming beginners. SIGCSE Bull. **28**(1), 256–260 (1996)
7. Meyer, A., Modeste, S. : Recherche binaire et méthode de dichotomie, comparaison et enjeux didactiques à l'interface mathématiques - informatique. In : Espace Mathématique Francophone (2018)
8. Ministère de l'Éducation nationale et de la Jeunesse : Programme de numérique et sciences informatiques de première générale. BO de l'éducation nationale (2019)
9. Ministère de l'Éducation nationale et de la Jeunesse : Programme de numérique et sciences informatiques de terminale générale. BO de l'éducation nationale (2019)
10. Modeste, S. : Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ? Ph.D. thesis, Université de Grenoble (2012)
11. Selby, C., Woollard, J. : Computational thinking: the developing definition. Project report, University of Southampton (2013)
12. Vincent, J.M., Collectif Tangente : Informatique débranchée. No. 42/43 in Tangente Éducation, Pole (2017)
13. Wing, J.M. : Computational thinking. Commun. ACM **49**(3), 33–35 (2006)

Reconnaissance et synthèse de motifs redondants avec des élèves de 6-7 ans MOTIFS.MOTIFS.MOTIFS. ⟺ 3 x MOTIFS.

Marielle LÉONARD[1,2], Yvan PETER[1] & Yann SECQ[1]

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France (prenom.nom@univ-lille.fr)
[2] Association France-IOI

Résumé. Cet article présente une expérimentation d'initiation à la pensée informatique pour de jeunes élèves de classe préparatoire (6-7 ans). L'objectif de la séquence pédagogique proposée est de développer la capacité de reconnaissance de motifs redondants et leur expression sous la forme de répétitions. Pour cela, des activités débranchées et en ligne sont proposées aux élèves, puis leurs traces d'activités sur la plateforme sont ensuite analysées. Cette étude illustre le fait que la détection de motifs et son expression sous forme de répétition est accessible pour cet âge dans un contexte où le code utilisé et les éléments du motif sont similaires visuellement. Lorsque le contexte implique en plus des compétences d'orientation spatiale, la détection de motif devient difficile pour la plupart des élèves de cet âge.

Mots-Clés: Pensée informatique, reconnaissance de motifs redondants, notion de répétition, programmation.

1 Introduction

La pensée informatique (PI) regroupe des compétences d'abstraction, de conception et de résolution de problèmes en lien avec les fondamentaux de la discipline informatique. En 2006, Wing [14] recommandait d'introduire l'enseignement de la PI aussi tôt dans les curricula que celui de la lecture, de l'écriture et de l'arithmétique. En France, le retour dans les programmes scolaires d'un enseignement de l'informatique dans l'ensemble des cycles date de 2016. À l'école primaire, au collège, et de manière plus conséquente au lycée avec la réforme du baccalauréat, des éléments de pensée et compétences informatiques sont successivement mobilisés afin d'initier l'ensemble d'une classe d'âge aux bases de l'informatique. Les problématiques sont nombreuses, tant en terme de démarches pédagogiques et activités proposées aux élèves que de formation continue des enseignants concernées par ces enseignements.

Les travaux présentés dans cet article portent sur les premières bases conceptuelles et les compétences en PI, qu'il nous semble crucial de développer dès le plus jeune âge, au début de la phase d'apprentissage de la lecture et l'écriture. Plus précisément, la présente expérimentation est focalisée sur la capacité de reconnaissance de motifs redondants et leur expression synthétique sous la forme d'une répétition. La question de recherche que nous nous posons est de savoir si cette compétence est accessible pour des élèves âgés de 6 à 7 ans, au moment de leur entrée dans l'apprentissage de l'écriture et la lecture.

Nous présenterons dans un premier temps les liens entre notre préoccupation et les programmes scolaires, puis dans un second temps un état de l'art de travaux proches de notre problématique. Ensuite, la séquence pédagogique sera détaillée avant d'étudier les résultats de l'analyse des traces d'activités des élèves sur la plateforme en ligne. Finalement, nous formulerons les interrogations subsistant et les perspectives envisagées à l'issue de cette première étude.

2 État de l'art et liens avec les programmes scolaires

Dans les programmes scolaires actuels, on trouve une référence en lien avec l'identification de motifs dans le programme scolaire de maternelle[1] à travers l'identification du principe d'un algorithme et la poursuite de son application. Il est à noter que le mot « algorithme » est souvent employé en maternelle comme synonyme de « répétition d'un motif ». La répétition d'instructions n'apparaît ensuite explicitement qu'à partir de la classe de 6ème : « *La construction de figures géométriques de simples à plus complexes, permet d'amener les élèves vers la répétition d'instructions.* »[2] .

La capacité à identifier et exprimer des structures itératives est un attendu du premier niveau en fin de 5ème : « *L'utilisation progressive des instructions conditionnelles et/ou de la boucle « répéter ... fois » condition « si ... alors » et boucle permet d'écrire des scripts de déplacement, de construction géométrique ou de programme de calcul.*" .

La capacité à détecter des motifs, n'est cependant plus mentionnée. Or, cette capacité à identifier des objets et des itérations est une composante de la pensée informatique, elle-même compétence clé du XXIème siècle [11]. De nombreux articles apportent une contribution à la définition de la PI et à ses composantes principales. Parmi les travaux récents, on peut citer Romero et al. [11] et Shute et al. [13]. Dans ces travaux, on retrouve une référence à la capacité d'abstraction incluant la capacité de reconnaissance de motifs et le traitement algorithmique correspondant (itération).

Plusieurs auteurs ciblent le premier degré dans leurs études. Baratè et al [2] abordent les notions d'algorithmique et de représentation de l'information en primaire à travers des activités visant à représenter des musiques par l'intermédiaire de briques LEGO. Brackmann et al. [4] ont trouvé un effet significatif des activités débranchées

1 BO spécial n°2 du 26 mars 2015
2 BO du 29 mai 2019

les compétences liées à la PI. Ils concluent que ce type d'activité constitue une bonne introduction à la PI mais que des activités "en ligne" sont nécessaires pour aller plus loin. Aggarwal et al. [1] ont comparé l'effet des modalités tangible et numérique sur l'acquisition des compétences. Ils concluent que si la modalité tangible facilite l'appropriation de la syntaxe, la modalité numérique permet de mieux développer la capacité à visualiser et prévoir l'effet de la séquence d'instruction. Ils suggèrent donc une utilisation limitée de la modalité tangible comme introduction avant de passer au numérique.

Peter et al. [10] ont mis en évidence la reconnaissance de motifs et son expression sous forme de répétition chez des élèves de 8 à 10 ans. Ils recourent à une séquence pédagogique alternant activités tangibles et activités sur support numérique dans un contexte de déplacement d'un robot sur une grille . Pour cette même tranche d'âge, Declercq et Tort [5] mobilisent un contexte de création de dessins numérique nommé *Pixel Art* dans lequel l'utilisateur doit déplacer un curseur orienté pour peindre des cases dans une grille.

Concernant l'introduction de la PI auprès d'un public plus jeune (5-7 ans), plusieurs travaux portent sur l'utilisation de robots pédagogiques : Bellegarde et al. [3], Komis et al. [7], Nogry [9]. Dans ces différentes études, seule la séquence d'instructions a été abordée, l'introduction de la répétition ne faisant pas partie du protocole. Komis et al. [7] pointent la difficulté de latéralisation pour 30% des sujets de l'étude, difficulté qui est un « frein persistant à la maîtrise des commandes de pivotement ». Bellegarde et al. [3] font une étude comparative de différentes modalités (corps-robot, robot tangible, robot virtuel).

Komis et al. [8] ont étudié les concepts de programmation pouvant être abordés avec le logiciel *ScratchJr*. Ils montrent que la sémantique de *ScratchJr* n'est « ni simple, ni toujours intuitive » pour cette tranche d'âge.

Dans les travaux précédents, on retient que l'introduction de la structure itérative a été étudiée pour des élèves de fin d'école élémentaire, dans un contexte où celle-ci doit être gérer de manière concomitante à la gestion des déplacements d'un personnage ou d'un curseur. Pour des élèves âgés de 6-7 ans, nous n'avons pas trouvé de travaux spécifiques sur cette compétence pourtant centrale dans la pensée informatique. Pour l'ensemble de la tranche d'âge de l'école élémentaire, une complémentarité des modalités est préconisée par les différents auteurs.

3 Cadre expérimental

La séquence pédagogique qui suit concerne l'introduction de la pensée informatique chez des élèves de cours préparatoire (6-7 ans). Elle vise à explorer si la détection de motifs redondants et son expression sous forme synthétique est accessible pour cette tranche d'âge. Elle constitue une première introduction aux bases de la pensée informatique.

Différentes modalités repérées dans l'état de l'art sont mobilisées : activités corporelles, sur supports tangibles et numériques.

3.1 Description de la séquence pédagogique

La séquence est composée de trois séances d'environ une heure trente (figure 1).

La première séance a pour objectif d'introduire les notions de motif et de répétition dans un contexte de reproduction de frises colorées (MOTIF ART). La deuxième séance reprend les notions de motif et de répétition dans un contexte de déplacements de robot sur une grille (ROBOT). Une troisième séance, de renforcement, revient sur le contexte MOTIF ART. Une activité créative est proposée dans ce contexte, qui permet d'appréhender si les élèves mobilisent les notions de motif et de répétition dans un cadre beaucoup moins contraint.

La notion de motif est d'abord introduite dans un contexte visuel (séance 1), dans lequel l'orientation dans l'espace n'intervient pas. Il s'agit de pouvoir dissocier les éventuelles difficultés de détection de motif de difficultés d'orientation dans l'espace (séance 2).

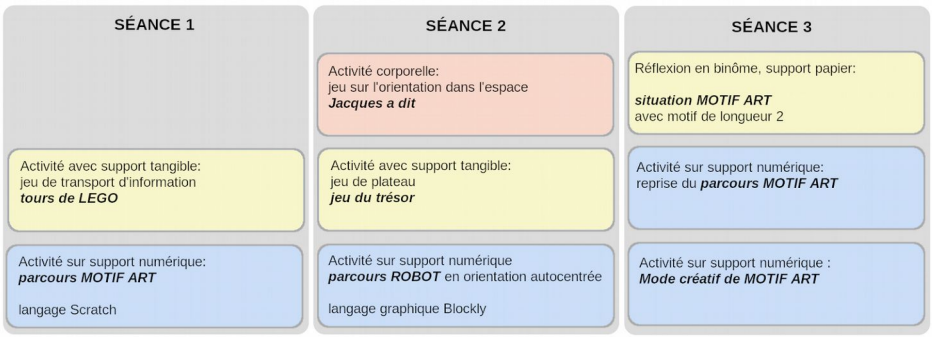


Figure 1. Organisation de la séquence pédagogique en trois séances mobilisant les modalités kinesthésique (rouge), tangible (jaune) et numérique (bleu)

Chaque séance est structurée en deux parties. Avec cette organisation pédagogique, chaque activité sur support numérique est précédée d'activités tangibles qui l'introduisent.

Une première partie en classe entière d'environ 30 minutes permet d'introduire les notions avec une ou deux activités sur support tangible. La deuxième partie de chaque séance, sur ordinateur, a eu lieu en demi-classe, pendant que l'autre moitié de la classe vaque à une autre activité en autonomie. Après 30 minutes, les rôles sont inversés.

Nous détaillons dans la suite le contenu des deux premières séances.

Séance 1 : introduction de la notion de motif et de répétition

Cette séance a pour objectif d'introduire les notions de motif et de répétition dans un contexte de reproduction de frises colorées.

Dans la première partie de la séance, un binôme doit reproduire à l'identique une tour en LEGO de 8 à 12 briques avec une alternance de couleur comme l'illustre la figure 2. Un des élèves est chargé de construire la tour, qui n'est pas visible de sa place. L'autre élève, le messager, se déplace pour aller voir le modèle, afin de guider oralement son partenaire dans sa reproduction. Le messager n'a pas le droit de toucher aux LEGO , ni de montrer ceux qu'il faut utiliser.

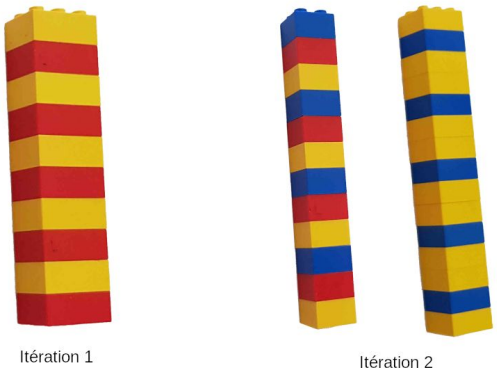


Figure 2. Exemples de modèles de tour en LEGO proposés aux élèves

Cette activité vise la prise de conscience de la nécessité de la précision des informations données et de l'importance du langage dans la description d'un processus.

Les élèves expérimentent l'ambiguïté ou l'imprécision du langage naturel. Ils se rendent compte qu'un très petit nombre d'informations est nécessaire pour réussir la tâche, mais que ces informations doivent être très précises.
Cette prise de conscience se fait grâce à la confrontation de la tour construite avec son modèle. Elle est renforcée par un temps collectif à l'issue de l'activité, au cours duquel l'enseignante aide à l'explicitation des erreurs commises.

Dans une seconde itération, on change les rôles dans les binômes et on introduit des tours avec des motifs de longueur trois. À l'issue de cette activité, après confrontation des tours, on fait émerger la notion de motif en demandant aux élèves de casser leur tour en morceaux identiques. Le but est d'obtenir le plus de morceaux identiques possibles. On a ainsi une matérialisation du motif et de son nombre de répétitions en décomptant le nombre de morceaux.

L'activité numérique qui suit reprend le même principe, sauf que l'élève doit transmettre les informations à l'ordinateur, dans le langage que celui-ci comprend, ici *Scratch*. Les motifs proposés sont ordonnées en complexité croissante, en faisant varier la longueur du motif à repérer (figure 3).

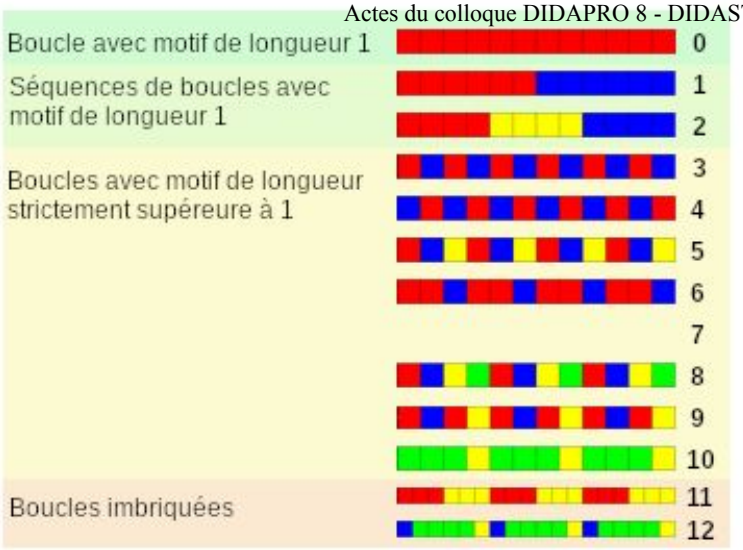


Figure 3. Parcours MOTIF ART sur support numérique, avec augmentation progressive de la complexité du motif à identifier et synthétiser.

L'interface est épurée, seuls les blocs utiles à l'activité sont disponibles (cf figure 4). Le nombre de blocs disponibles pour concevoir le programme contraint l'utilisation de la boucle (questions 0 à 10) puis de la boucle imbriquée (questions 11 et 12). Les briques LEGO restent à la disposition des élèves comme aide. Ils peuvent éventuellement reproduire la frise qu'ils voient à l'écran puis la casser pour retrouver le motif.

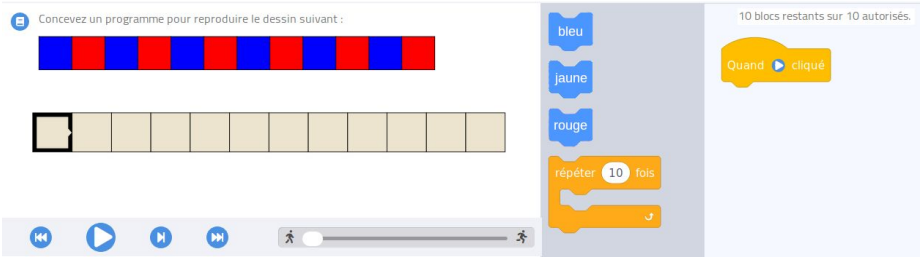


Figure 4. Interface MOTIF ART sur la plateforme

Séance 2 : gestion des déplacements d'un personnage sur une grille

Cette séance commence avec une activité corporelle, afin de renforcer la latéralisation. Dans les deux activités proposées ensuite, tangible et numérique, il s'agit de programmer le déplacement d'un robot sur une grille. Le but est d'amener le robot sur une case cible, éventuellement en contournant des obstacles. L'orientation est auto-

centrée, avec un jeu de trois instructions : « *avancer* » , « *tourner à droite* » et « *tourner à gauche* ».

Pour cette séance, la modalité tangible mobilise un jeu de plateau utilisé par groupes de quatre élèves, au cours duquel ils tiennent des rôles différents : codeur (qui écrit un code à l'aide du langage formel disponible), pointeur (qui montre avec le doigt la prochaine instruction à exécuter), lecteur (qui lit l'instruction pointée), processeur (qui exécute l'ordre donné). Après deux situations de séquences qui visent à entrer dans l'activité, la case de départ et la case cible sont éloignées afin d'introduire la notion de répétition de déplacements élémentaires (répétition avec motif de longueur 1, c'est-à-dire un corps de boucle ne contenant qu'une instruction). Au cours du jeu de plateau, la séquence de répétitions avec motif de longueur 1 a été introduite mais les répétitions avec plusieurs instructions dans le corps de la boucle (motif de longueur strictement supérieure à 1) n'ont pas été abordées.

Le même contexte est repris ensuite sur support numérique (ordinateur). Le parcours est proposé sur la même plateforme, dans un langage de programmation graphique similaire à *Scratch*. La figure 5 décrit la progression pédagogique proposée aux élèves.

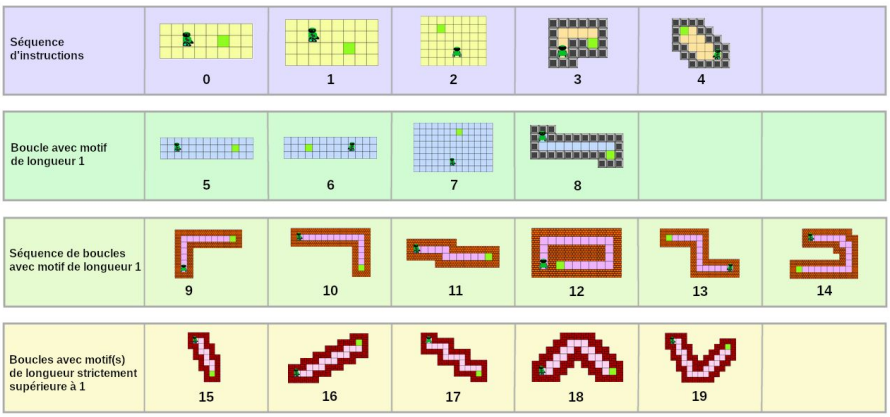


Figure 5. Parcours ROBOT sur support numérique, avec orientation autocentrée

3.2 Participants et organisation de l'expérimentation

Cette étude exploratoire a été menée dans une classe de CP de la circonscription de Lille sur 3 semaines en mai 2019. La classe comporte 26 élèves de 6 à 7 ans, mais seulement 24 élèves ont participé à la totalité de l'exploration et ont été comptabilisé dans l'étude. Tous les élèves sont lecteurs (au moins déchiffrent) sauf un.

L'expérimentation s'est déroulée dans les locaux de l'école, classe et salle informatique. La séquence pédagogique a été présentée en amont à l'enseignante de la classe, qui a mené les séances. L'expérimentatrice qui a été elle-même enseignante dans le premier degré est présente, dans une posture d'observateur, parfois aussi de co-enseignement.

3.3 Collecte des traces d'activités des élèves

Les activités numériques ont été réalisées sur une plateforme mise à disposition par l'association France-IOI[3]. Les élèves sont enregistrés avec un code de participant sur cette plateforme et toute l'activité enregistrée est rattachée à ce code, ce qui anonymise les données collectées tout en permettant leur traitement après extraction.

Lors de chaque évènement sur l'interface (changement de question, clic sur le bouton "*play*"), un enregistrement est déclenché. Cet enregistrement comprend le numéro de la question, le code de participant, la date, le type d'événement, le score réalisé et le programme soumis.

Il est à noter que les élèves ont reçu leur code de participant et ont pu retourner sur les parcours hors temps scolaire. L'extraction a été faite quelques semaines après l'expérimentation et les traces collectées comprennent l'activité pendant les séances en temps scolaires, mais aussi l'activité des élèves hors temps scolaire.

4 Résultats

4.1 Analyse des traces d'activités des élèves pour le parcours MOTIF ART

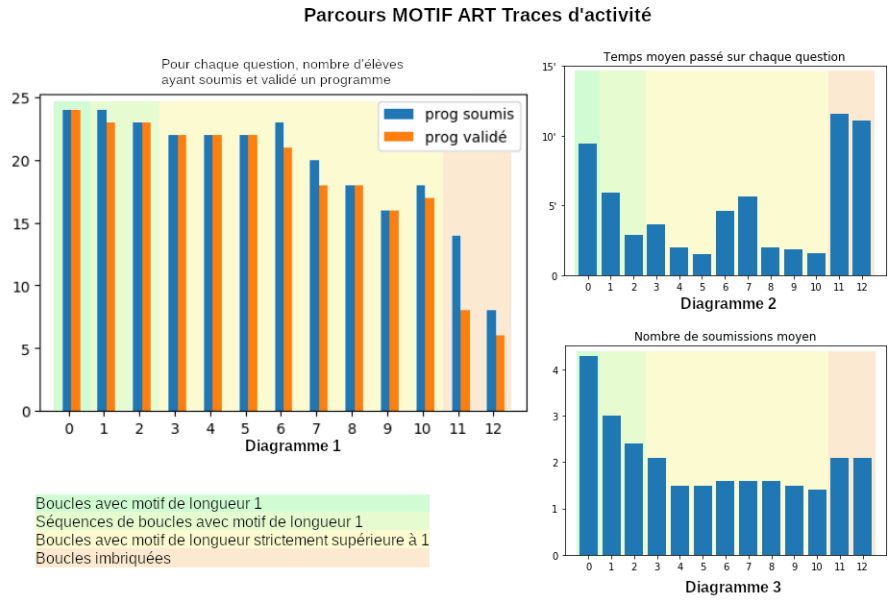


Figure 6. Traces d'activités des élèves sur le parcours MOTIF ART

Nous analyserons les diagrammes de la figure 6 afin de déceler la détection de motifs répétitifs et la mobilisation de la structure itérative par les élèves sur le parcours

3 Site de l'association France-IOI : `http://www.france-ioi.org`

MOTIF ART. Nous essaierons aussi de détecter les paliers des difficultés dans ce parcours.

Le diagramme 1 montre pour chaque question le nombre d'élèves qui ont au moins soumis une réponse (en bleu) et le nombre d'élèves qui ont réussi la question (en orange). Pour plus de la moitié du parcours, nous avons plus de 20 élèves qui soumettent un programme valide, ce qui fait un taux de réussite au-delà de 80% et permet d'affirmer que dans les conditions où il a été introduit, ce parcours est bien accessible à cette tranche d'âge.

Le diagramme 2 permet d'affiner ces résultats. Il présente la moyenne de temps passé sur chaque question en ne prenant en compte les élèves qui ont soumis au moins une réponse. Il est à mettre en parallèle avec le diagramme 3, qui montre le nombre moyen de soumissions pour chaque question.

L'analyse de ces diagrammes nous donne des éléments sur la difficulté du parcours pour les élèves.

Le temps passé sur la première question nous indique un temps de prise en main de l'interface de l'ordre de 10 minutes. Plusieurs programmes sont soumis, malgré la simplicité du puzzle. Pour les questions 2 et 3, le temps passé décroît fortement, de l'ordre 3 minutes pour la question 3. Le nombre de soumissions décroît aussi, ce qui montre que la notion de boucle simple est bien assimilée, et donc largement accessible dans ce contexte.

On observe une légère remontée de la durée pour la question 3 qui correspond au passage à la répétition avec plusieurs instructions dans le corps de la boucle. Par contre, le nombre moyen de soumissions continue à décroître. Ce motif de longueur 2 a été mobilisé lors de l'activité tangible "Tours de LEGO", ce qui semble aller dans le sens de l'efficacité de cette activité et d'une bonne transposition de la compétence du support tangible vers le support numérique. Le passage à un motif de longueur 3 (question 5) ne pose aucun souci, le temps passé diminuant encore.

La durée plus importante pour les questions 6 et 7, ainsi que le nombre d'élèves qui restent bloqués aux questions 6 (3 élèves) et 7 (2 élèves supplémentaires) nous indique un palier de difficulté à cet endroit du parcours. En effet, le motif de longueur 3 change de forme, avec deux fois la présence d'une même couleur dans le motif, ce qui le rend moins lisible et en complique la détection.

Un deuxième palier de difficulté est très visible aux questions 11 et 12. Il correspond au passage aux boucles imbriquées. 14 élèves sur les 24 arrivent jusqu'à la question 11 et 6 y restent bloqués. Le nombre moyen de soumissions remontent pour les questions 11 et 12. À cet âge et dans ce contexte particulier, on peut toutefois noter que les boucles imbriquées sont accessibles pour un tiers de la classe (8 élèves) qui ont réussi la question 11.

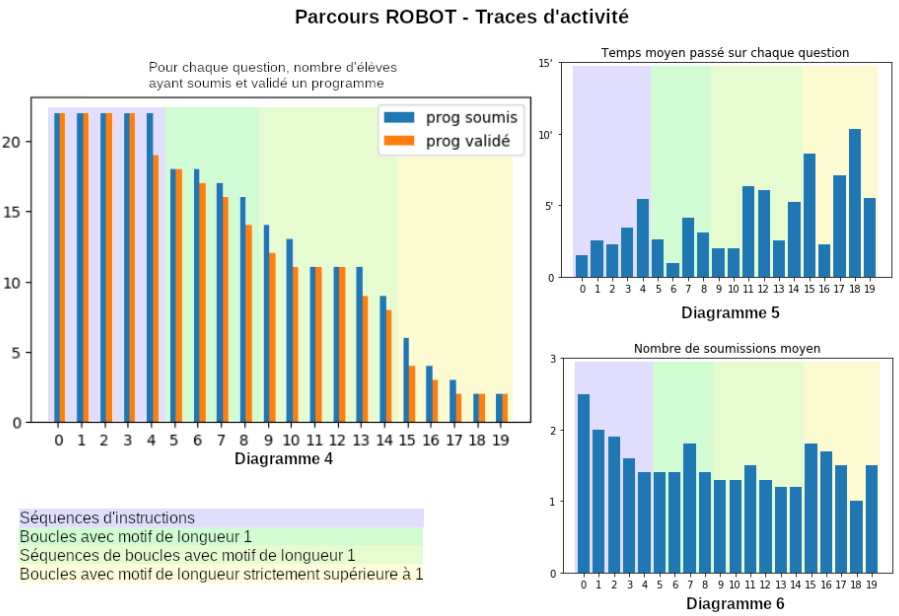


Figure 7. Traces d'activités des élèves sur le parcours ROBOT

En considérant la figure 7, nous reprendrons pour le parcours ROBOT le même type d'analyse que pour le parcours MOTIF ART. Nous comparerons aussi les résultats obtenus sur les deux parcours.

On ne remarque pas de temps significatif pour la prise en main du parcours, signe que le changement de contexte et de langage (*blockly* vs. *scratch*) sur la même plateforme s'est fait de manière fluide.

Les élèves obtiennent de moins bons résultats pour ce parcours, alors que la progression reprend la séquence d'instructions, et que la boucle avec des motifs de longueur strictement supérieure à 2 est abordée beaucoup plus tard.

Pour les questions sur la séquence d'instructions, le temps passé augmente avec le nombre de blocs nécessaires pour concevoir le programme. Cela n'est cependant pas interprété comme un signe de difficulté car parallèlement, le nombre de soumissions diminuent. On peut se poser la question de la pertinence de la question 4, qui bloque 3 élèves alors que l'objectif est de permettre aux plus grand nombre d'accéder aux questions sur la détection de motifs et la répétition.

On remarque un pic et sur le temps passé et sur le nombre de soumissions pour la question 7, signe d'une difficulté à cet endroit du parcours. En effet, il faut à la fois utiliser le bloc "*répéter*" avec une instruction dedans, mais aussi des blocs à l'extérieur de cette boucle, pour gérer le premier déplacement et le changement d'orientation du robot.

Le passage aux répétitions avec des motifs de longueur strictement supérieure à 2 est beaucoup plus discriminant que pour le parcours MOTIF ART. Seuls 4 élèves réussissent une question de ce type sur le parcours ROBOT, contre 22 élèves pour le parcours MOTIF ART.

Deux facteurs peuvent apporter des éléments d'explication. D'une part, la gestion de l'orientation du robot constitue une difficulté inhérente à ce contexte.

Mais surtout, la détection du motif nécessite un plus grand degré d'abstraction dans ce contexte que dans le contexte MOTIF ART. Dans le contexte MOTIF ART, le motif (de couleurs) et le code utilisé (blocs de couleur) sont très proches.

En revanche, dans le contexte ROBOT, le motif correspond aux déplacements du robot (blocs de déplacements), il est séquentiel et ne peut s'appréhender globalement qu'à travers la visualisation des cases que le robot doit parcourir. La difficulté supplémentaire est de replacer le robot dans la même orientation pour aborder chaque motif.

Il est courant de mobiliser ce contexte de programmation de déplacements d'un robot sur une grille (tangible ou numérique) pour l'initiation à la programmation. Il permet en particulier de viser des compétences en structuration de l'espace. Il semble en revanche qu'il soit moins efficace que le contexte MOTIF ART pour introduire la détection de motifs répétitifs et leur expression sous forme synthétique.

5 Conclusion et perspectives

Cet article porte sur l'initiation à la pensée informatique chez de jeunes élèves de 6-7 ans en cours d'apprentissage de la lecture. Il s'agit d'une expérimentation exploratoire menée en milieu scolaire. La séquence pédagogique initie ces élèves à la détection de motifs et à leur expression sous forme de répétition, dans un contexte de programmation de frises de couleurs, et dans un contexte de déplacement de robot sur une grille.

L'étude montre que la détection de motifs et son expression sous forme de répétition est accessible pour cet âge dans un contexte où le code utilisé et les éléments du motif sont similaires visuellement. Lorsque le contexte implique en plus des compétences d'orientation spatiale, la détection de motif devient difficile pour la plupart des élèves de cet âge. Lors de l'initiation à la pensée informatique, Il nous semble donc pertinent de dissocier la détection de motifs répétitifs et l'introduction de la structure de contrôle de répétition de la gestion des déplacements d'un personnage.

Cette étude fait aussi émerger un certain nombre d'autres problématiques. En effet, les élèves ont passé des pré-test et post-test, mais ces outils demandent à être affinés pour être exploitables. En particulier, la modalité papier de ces tests est à questionner.

Les 22 élèves présents lors de la séance 3 ont utilisé le mode créatif du parcours MOTIF ART et l'ont beaucoup apprécié. Une analyse plus fine des dessins produits reste à faire, pour appréhender dans quelle mesure les élèves utilisent des motifs répétitifs lors d'une activité de création.

Suite à cette expérimentation, une étude plus large est en cours. Le même parcours sera proposé à des élèves de Grande Section (4-5 ans). À cette fin, les blocs *Scratch* du parcours MOTIF ART ont été modifiés pour être accessibles aux non-lecteurs. La couleur de chaque bloc correspond désormais à la couleur codée.

Il s'agira dans cette nouvelle étude de confirmer ces résultats, mais aussi de faire une analyse plus fine des programmes soumis et des erreurs faites par les élèves.

Remerciements

Les auteurs tiennent à remercier l'enseignante qui nous a accueilli dans sa classe pour cette expérimentation, et l'association France-IOI pour les possibilités d'usages et d'adaptation de ses plateformes.

Nous remercions aussi l'INSPÉ Lille Nord-de-France et plus particulièrement Mme Dorothée HALLIER-VANUXEM, qui nous a informé de l'appel à projet initié par l'INSPÉ pour lequel notre projet de recherche a été retenu, ce qui nous permettra d'étudier les perspectives présentées dans cet article.

Références

1. Aggarwal, A., Gardner-McCune, C., & Touretzky, D. S. (2017). Evaluating the Effect of Using Physical Manipulatives to Foster Computational Thinking in Elementary School. In SIGCSE'2017.
2. Baratè, A., Ludovico, L.A., Malchiodi, D. (2017). Fostering Computational Thinking in Primary School through a LEGO-based Music Notation. Procedia Computer Science, 112, p. 1334–1344.
3. Bellegarde, K., Boyaval, J., & Alvarez, J. (2019). S'initier à la robotique/informatique en classe de grande section de maternelle. Une expérimentation autour de l'utilisation du robot Blue Bot comme jeux sérieux. Review of Science, Mathematics and ICT Education, 13(1), 51-72.
4. Brackmann, C. P., Román-gonzález, M., Robles, G., Moreno-león, J., Casali, A., & Barone, D. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. In WiPSCE'17, pp. 65–72
5. Declercq, C., & Tort, F. (2018, April). Organiser l'apprentissage de la programmation au cycle 3 avec des activités guidées et/ou créatives.
6. Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education - ITiCSE '14, p.57–62.
7. Komis, V., & Misirli, A. (2011, October). Robotique pédagogique et concepts préliminaires de la programmation à l'école maternelle: une étude de cas basée sur le jouet programmable Bee-Bot. Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques. Edutice-00676143, pp.271-281.
8. Komis, V., Touloupaki, S., & Baron, G.-L., (2017). Une analyse cognitive et didactique du langage de programmation ScratchJr. In Henry, J., Nguyen, A., Vandeput, E. (coordination éditoriale), L'informatique et le numérique dans la classe : qui, quoi, comment ?, Presses Universitaires de Namur, pp. 109–122.
9. Nogry, S. (2018). Comment apprennent les élèves au cours d'une séquence de robotique éducative en classe de CP?. De 0 à 1 ou l'heure de l'informatique à l'école, 235.
10. Peter, Y., Léonard, M., & Secq, Y. (2019, June). Reconnaissance de Motifs et Répétitions: Introduction à la Pensée Informatique. Environnements informatiques pour l'Apprentissage Humain, Paris, France, pp. 211-222.
11. Romero, M., Lille, B., & Patiño, A. (2017). Usages créatifs du numérique pour l'apprentissage au XXIe siècle. PUQ
12. Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research, p. 59–66.
13. Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22(September), p. 142–158.
14. Wing, J.M. : Computational Thinking. COMMUNICATIONS OF THE ACM 49 (3), 33#35 (2006)

La programmation par passage des messages dès l'école maternelle ? Le cas de ScratchJr.

Sevastiani Touloupaki[1] & Georges-Louis Baron[1]

[1]Laboratoire EDA, Université Paris Descartes, 45 rue saints-pères, 75006 Paris, France

sevina.touloupaki@gmail.com
georges-louis.baron@parisdescartes.fr

Abstract. Le travail qui suit s'inscrit dans le cadre d'une thèse élaborée en cotutelle entre l'Université Paris Descartes en France et l'Université de Patras en Grèce, portant sur l'apprentissage de l'informatique par de très jeunes enfants. L'objet de cette étude a été l'analyse de la manière dont les élèves de l'école maternelle peuvent mettre en œuvre des concepts de programmation à l'aide du logiciel ScratchJr. Dans ce contexte, nous avons réalisé une étude exploratoire dans une école maternelle grecque auprès de douze élèves de grande section. Pour les besoins de cet article nous avons choisi de nous concentrer sur l'analyse du concept de la communication par passage de messages. Les résultats présentés proviennent de l'analyse des programmes développés par les élèves pendant la séance d'évaluation. Ils soutiennent l'idée qu'il est possible, par la mise en œuvre d'un scénario pédagogique spécifique, d'utiliser dès un très jeune âge la communication par passage de messages, pour synchroniser les actions de personnages différents.

Keywords: Didactique de programmation, Apprentissage, ScratchJr, Message, École maternelle, Programmation visuelle.

1 Introduction

Depuis quelques années, il existe un mouvement international autour de l'éducation à l'informatique, et en particulier à la programmation, pour tous, depuis même la petite enfance. Plusieurs pays d´Europe ont ainsi introduit des enseignements d'informatique et de programmation en primaire ou en secondaire. À partir d'octobre 2016, les nouveaux programmes scolaires en France ont comporté une introduction de la robotique et de la programmation dès l'école élémentaire. Ce choix provient d'une volonté de développer chez les élèves des compétences en programmation, afin de les préparer à devenir des acteurs du monde numérique qui les entoure.

Des recherches ont été menées depuis longtemps sur l'apprentissage de la programmation par de très jeunes enfants, en particulier en Logo (Robert, 1985). Plus récemment, il a été confirmé que les jeunes enfants peuvent réaliser des programmes dès l'âge

de 4 ans (Bers, 2012). Fessakis, Gouli et Mavroudi (2013) ont montré qu'à l'aide d'un environnement de programmation de type Logo, des activités et du matériel approprié, les enfants de 5 à 6 ans peuvent être familiarisés avec des concepts de base de programmation et développer deux stratégies différentes de résolution de problème. Un peu plus tard, Misirli et Komis (2016), décrivent dans un ouvrage les résultats de l'expérimentation d'un scénario pédagogique fondé sur l'usage des jouets programmables Bee-Bot, pour enseigner aux élèves des notions de programmation tels que la séquence, le programme, la commande etc. Cette recherche a été menée dans 34 classes de maternelle en Grèce, auprès des très jeunes élèves de 4 à 6 ans.

L'apparition de l'environnement ScratchJr a suscité un nouvel intérêt pour l'apprentissage de rudiments de programmation dès la petite enfance. Ce système met en œuvre des éléments de différents paradigmes de programmation, tels que la programmation structurée, évènementielle et orienté-objet. Il s'agit d'un langage visuel et graphique, destiné aux enfants dès 5 à 7 ans, qui a été créé afin de permettre aux derniers d'apprendre à programmer pendant qu'ils créent leurs animations (Flannery, Kazakoff, Bonta, Silverman, Bers & Resnick, 2013).

Nous allons nous intéresser ici à une fonctionnalité peu étudiée au niveau de la petite enfance : la communication par passage de messages. Ce dernier nous rappelle le modèle de « passage de message synchrone » ou « synchronous message passing » en anglais, qu'on retrouve dans des systèmes concurrents (Ben-Ari, 1996). ScratchJr met en effet en œuvre des éléments de programmation concurrente, car il permet de lancer en même temps aussi bien les scripts de deux personnages, que les scripts d'un même personnage, à l'aide de l'événement « Quand drapeau vert appuyé ». La communication entre entités est implémentée par deux commandes sur ScratchJr : la commande « envoyer un message », qui envoie un message de couleur spécifique à un personnage et la commande de « quand message reçu », avec laquelle le script du personnage, dont la commande du message a cette couleur, est déclenché.

S'agissant de l'appropriation de la notion de message au niveau de la scolarité obligatoire, nous pouvons mentionner ici des travaux menés à l'aide du logiciel Scratch.

Meerbaum-Salant, Armoni et Ben-Ari (2010), présentent une étude exploratoire, auprès de deux classes de 3e (dix-huit et vingt-huit élèves), dont l'objectif a été d'évaluer la capacité pour les élèves à apprendre des concepts d'informatique. Pour cela, les auteurs ont formé les deux enseignants sur le processus de développement d'un programme, afin de leur permettre de mettre en place l'enseignement des concepts informatiques avec Scratch. Il est important de noter ici que les sujets de l'étude n'avaient aucune expérience préalable avec le logiciel Scratch. Les résultats de cette étude montrent que le taux de réussite au post-test était le deuxième meilleur après celui de boucle conditionnelle. En effet, 62% des participants sont arrivés à donner une définition correcte du concept des messages au post-test, tandis que le taux de réussite pour le concept d'initialisation était d'environ 20% et moins de 10% pour celui de variable.

Par la suite, dans une étude hors du contexte scolaire, pendant un camp des vacances d'une durée de deux semaines aux États-Unis, vingt-deux élèves de collège sans expérience préalable sur Scratch ont été évalués en fonction de leur performance aux concepts informatiques présentés. À la fin de cette expérience, les élèves ont pu développer

des compétences en programmation événementielle mettant en œuvre l'envoi et la réception de messages.

Plus précisément, la majorité des participants a utilisé la communication par messages, sauf un groupe qui a eu besoin d'aide et un autre qui a eu des problèmes avec le temps. Pourtant, les résultats montrent aussi qu'ils existent souvent des projets incomplets, où l'une de deux commandes de messages manque, soit parce que les participants ont oublié d'effacer les commandes non nécessaires de l'esce de programmation, soit parce qu'ils n'ont pas terminé leurs projets (Franklin, Konrad, Boe, Nilsen, Hill, Lern, Dreschler, …Suarez, Waite, 2013).

Dans un article plus récent, Fatourou, Zygouris, Loukopoulos et Stamoulis (2018), analysent l'apprentissage des concepts de programmation concurrente par des élèves de CM2, sans expérience préalable en programmation et de 6e, ayant un peu d'expérience en programmation. Cent vingt-trois élèves au total, de sept classes d'école primaire en Grèce ont participé à l'étude. Les résultats de cette étude montrent que le passage des messages a posé la plupart de problèmes aux élèves, car seulement un élève sur cinq a réussi de l'utiliser correctement sur son projet final. En effet, comme on pourrait le supposer, les élèves qui arrivent mieux à implémenter des messages sur leurs projets sont ceux de 6e.

1.1 La communication par passage des messages sur ScratchJr

Programmer avec ScratchJr consiste à créer des animations en choisissant des personnages et en rassemblant les commandes appropriées à l'ordre séquentiel correct, afin de faire réaliser aux personnages les actions souhaitées. (Komis, Touloupaki et Baron, 2017).

Comme on l'a vu plus haut, ScratchJr met en œuvre des éléments de programmation concurrente et événementielle. Lorsqu'un projet est lancé, les scripts des personnages qui ont l'instruction « Quand drapeau vert appuyé » au début de leur script, seront déclenchés en même temps. L'instruction « Quand drapeau vert appuyé » correspond à un événement qui est lancé lorsque le « drapeau vert global » est appuyé (Fig. 1).

ScratchJr permet également de contrôler le temps d'exécution des personnages. La communication entre personnages se fait grâce à l'envoi et la réception des messages. Pour cela nous avons besoin de deux commandes : la commande « Envoyer message », et la commande « Quand message reçu » (Fig. 2).

La commande « Envoyer message » se trouve dans la catégorie des blocs de déclenchements ou triggering blocs en anglais. Son seul paramètre est la couleur. C'est une instruction de contrôle, dont la forme est celle d'un bloc ordinaire. Elle peut être ajoutée à n'importe quel endroit dans une séquence des commandes. Sa position signale le moment de l'envoi du message par le personnage expéditeur. Après son exécution le personnage expéditeur suspend son script jusqu'à ce que celui du personnage destinataire soit terminé.

S'agissant de la réception de messages, elle se réalise à l'aide de la commande « Quand message reçu ». Il s'agit d'une commande, dont la forme nous rappelle celle de la commande « Quand drapeau vert appuyé » et elle ne se place qu'au début d'un

script. Lorsque la réception d'un message se produit, le programme qui suit la commande « Quand message reçu » s'exécute, sous la condition que les deux commandes sont de la même couleur.

Les commandes de messages permettent de définir l'ordre de l'exécution des scripts des personnages et les faire communiquer.

Fig. 1. Instruction « Drapeau vert » ou « Start on green flag » en anglais et icône « Drapeau vert global » ou « Green flag » en anglais.

Fig. 2. Les instructions des messages.

2 Problématique et questionnement de recherche

Dans un article précédent (Touloupaki et al., 2018) nous avons choisi d'étudier la manière dont les élèves de cours préparatoire arrivent à mettre en œuvre des messages à l'aide du logiciel ScratchJr. Pour le présent travail, nous avons choisi d'étudier la même question, mais auprès d'élèves plus jeunes, ceux de grande section de l'école maternelle.

Pour cette étude nous allons essayer de répondre au questionnement qui suit : Comment les élèves arrivent-ils à utiliser les commandes de messages de manière opérationnelle afin de pouvoir résoudre un problème d'évaluation ?

3 Méthodologie

Pour réaliser cette étude exploratoire nous avons choisi d'effectuer une recherche de conception (*design-based research*). Le scénario pédagogique (le design) est central dans cette approche, visant à évaluer sa capacité à transmettre les savoirs choisis par

son créateur. La recherche de conception permet une compréhension profonde du processus de l'apprentissage (Cobb, Confrey, DiSessa, Lehrer, & Shauble, 2003).

Selon Cobb et al. (2003), un des objectifs principaux de cette approche est d'améliorer le scénario du début à travers des évaluations itératives des différentes conjectures, en faisant en même temps des analyses autour du raisonnement des élèves et de l'environnement d'apprentissage.

À cette fin, notre scénario pédagogique a été modifié plusieurs fois pendant la mise en place du terrain, pour s'adapter aux besoins du public visé. Il a été de nouveau modifié à la fin de la première année de notre terrain, afin de correspondre aux besoins de notre terrain en France, l'année d'après. À noter ici que, pour pouvoir étudier la communication par passage des messages, nous sommes passés par un scénario pédagogique de onze séances.

3.1 Le scénario pédagogique

Pour concevoir le scénario pédagogique de cette étude, nous avons commencé par une analyse du langage de programmation ScratchJr, en termes de fonctionnalités. Ensuite, nous avons considéré les spécificités du groupe d'âge visé, c'est-à-dire le développement cognitif, social et affectif de ces élèves. Notre scénario pédagogique comprend trois types de séances, les séances d'initiation au logiciel et à la programmation, les séances de l'enseignement du contenu choisi et de mise en pratique et les séances de l'évaluation de la performance des élèves.

L'objectif de chaque séance d'enseignement était la production d'une animation. Le scénario a été réalisé en classe par nous-mêmes avec l'accord de l'enseignante de la classe. L'écran de notre tablette a été projeté à l'aide d'un mini vidéo- projecteur, pour que les élèves puissent mieux voir ce qui se passe dans notre projet, sur ScratchJr. L'étude a duré trois mois et chaque séance a duré une quarantaine de minutes.

Dans un premier temps, nous avons réalisé un entretien individuel (pré-test), pour détecter les idées initiales des élèves à propos de l'interface et des commandes étudiées. Nous avons aussi réalisé une activité de familiarisation avec la tablette, l'interface de ScratchJr et quelques principaux concepts de programmation.

Ensuite, nous sommes passés aux séances d'enseignement, pendant lesquelles nous avons présenté les commandes de mouvement, l'exécution séquentielle des commandes, l'initialisation de la position des personnages dans la scène, le déclenchement concurrent des scripts de deux personnages, la répétition d'une séquence des commandes un nombre des fois prédéfini et enfin, les messages. Neuf séances d'enseignement ont été réalisées au total et deux séances de ce scénario ont été consacrées à la communication par passage de messages.

Nous nous sommes fondés sur les observations de l'étude précédente (Touloupaki et al., 2018) et nous avons présenté des projets déjà prêts, contenant les commandes des messages et nous les avons laissés expérimenter et découvrir la fonctionnalité de chacune des commandes, à l'aide de nos questions. Nous avons choisi d'employer deux styles d'enseignement : l'expert et le facilitateur (Grasha, 1994). Selon Grasha (1994), le style d'expert, est focalisé à la manifestation des connaissances et à encourager les élèves à améliorer leurs compétences. Le style de facilitateur est centré aux besoins du

public visé et aux objectifs à atteindre. Il est caractérisé par une adaptation continue aux besoins du public, pour réussir l'objectif de l'enseignement. Strawhacker, Lee et Bers, (2017) montrent que la mise en œuvre de ces deux styles d'enseignement est corrélative d'une haute performance en programmation chez les élèves, sur l'environnement ScratchJr.

Pour faciliter la compréhension des élèves, nous avons choisi d'imprimer les images des commandes du logiciel. Nous avons donc construit les commandes du ScratchJr en version papier, que les élèves ont utilisées pour créer et observer en même temps les scripts de deux personnages communiquant par messages. Ensuite, nous sommes passés aux séances d'évaluation de la performance des élèves.

En effet, nous avons mené, un entretien individuel (post- test) et deux problèmes à résoudre pour tous les élèves. Le premier problème concernait entre autres la notion de messages. Pour résoudre le problème d'évaluation, les élèves ont été invités à regarder une animation en mode de présentation, accompagnée de la consigne du problème donné lue par nous-mêmes, parce que les élèves de l'école maternelle ne sont pas en mesure de lire, et reproduire le même projet final. Les élèves avaient la possibilité de revoir l'animation en plein-écran et de réécouter la consigne du problème s'ils avaient besoin.

Comme évaluation, on demande aux enfants d'observer le programme donné, d'identifier les commandes appropriées et de les utiliser dans l'ordre correct afin de reproduire les comportements observés, pour les personnages participant au projet. L'objectif de cette activité d'évaluation est d'évaluer aussi la capacité des élèves à réaliser ce qu'on appelle « goal-oriented programming ».

3.2 L'échantillon

Cette étude pilote a été menée en 2016-2017 dans une école maternelle publique de la ville de Patras en Grèce, auprès de 12 élèves, 6 filles et 6 garçons. Ces élèves (de 5 à 6 ans) ont été choisis en fonction de leurs réponses au pré-test et à l'aide de l'enseignante de la classe. Ils ont travaillé par groupe de quatre, avec quatre tablettes, pour réaliser les activités proposées par le scénario pédagogique élaboré à ce titre. Nous avons travaillé avec trois groupes de quatre élèves. Les groupes ont été constitués de manière à pouvoir représenter tous les niveaux de compétences scolaires différents présents dans la classe.

Notre échantillon est donc de faible taille, mais il est contrasté, ce qui nous a semblé suffisant pour notre objectif de repérer des faits intéressants qui devront être confirmés ou infirmés par des recherches ultérieures.

3.3 Techniques et outils de recueil des données

Pour recueillir nos données nous avons utilisé comme techniques l'observation participante et les entretiens semi-directifs. Pour faciliter le recueil de données nous avons utilisé un enregistrement audio-visuel du processus, ainsi que nos notes personnelles, en nous centrant sur les incidents critiques, c'est-à-dire les moments importants

pour l'élaboration de la solution du problème donné par les élèves. Nous avons également eu recours à des captures d'écran des programmes élaborés par les élèves tout au long du processus.

4 Résultats

L'analyse des programmes élaborés par les élèves face à un même problème d'évaluation a été conduite en fonction de deux critères : d'abord, s'ils ont utilisé les deux commandes de messages pour synchroniser les actions de deux personnages et ensuite, s'ils ont utilisé les commandes des messages selon l'ordre correct dans le script des personnages concernés.

L'ordre est dépendant de la consigne du problème donné. Dans le Tableau 1 nous présentons l'ordre correct des commandes des messages, selon le problème donné. Il est important de signaler ici que nous avons choisi de ne pas introduire le paramétrage des commandes des messages, c'est-à-dire la couleur.

Tableau 1. Ordre correct des commandes des messages.

Commande	Personnage	Ordre
Envoyer un message	Enfant	La commande se trouve après la fin du déplacement et avant la fin rouge.
Quand message reçu	Papillon	La commande se trouve au début du programme.

Ainsi, nous sommes arrivés à distinguer trois types de performance des élèves : une « utilisation opérationnelle », une « utilisation partielle », et « pas d'utilisation ». Plus précisément, 6 sur 12 élèves arrivent à une « utilisation opérationnelle » des messages (Fig. 3). Ces élèves utilisent les deux commandes des messages, dans l'ordre correct, sur les scripts des personnages demandés. Nous pouvons aussi remarquer que 3 sur 12 arrivent à une « utilisation partielle » des messages, car ils utilisent les deux commandes des messages, mais seulement l'une de deux commandes est à la bonne place (Fig. 4). Enfin, nous pouvons observer que 3 sur 12 élèves n'utilisent pas du tout les commandes des messages.

Fig. 2. Utilisation opérationnelle des commandes des messages.

Fig. 3. Utilisation partielle des commandes des messages.

Comme nous pouvons l'observer auparavant la moitié de notre échantillon (6/12) est arrivé à une « utilisation opérationnelle » du concept des messages. Il est intéressant ici de signaler que 5 sur 6 sont des filles. Nous pouvons alors constater que dans le contexte de cette classe et avec ce scénario pédagogique, ces dernières réussissent mieux que les garçons. Ce constat ne permet cependant pas de tirer de conclusions significatives car il pourrait être lié à des niveaux de développement différents entre filles et garçons.

En analysant les programmes des trois élèves qui arrivent à une « utilisation partielle » de la communication par envoi de messages, nous pouvons identifier une erreur qui se répète. L'erreur de ces trois projets concerne la commande « envoyer message » et plus précisément, sa position sur le script du personnage « enfant ». Il est important de noter ici que les trois élèves choisissent d'ajouter la commande « envoyer message » sur le script du correct personnage.

Aucun de ces élèves, n'a eu de problèmes avec la commande « Quand message reçu ». La morphologie de cette dernière est telle que la syntaxe du logiciel oblige l'utilisateur à la poser au début des programmes de personnages. La position de la commande « envoyer message » sur le script du personnage « enfant » dépend d'une part, de la compréhension de la consigne du problème et d'autre part, de la compréhension de l'exécution séquentielle, qui caractérise l'environnement ScratchJr.

D'après nos observations de terrain, nous avons pu remarquer que la compréhension de l'exécution séquentielle était très importante, pour l'enseignement des autres notions présentes dans notre scénario pédagogique. Nous pouvons alors supposer que ces trois élèves sont arrivés à un tel résultat, soit parce qu'ils n'ont pas bien compris l'énoncé du problème, soit parce qu'ils n'ont pas bien compris le concept d'exécution séquentielle.

Les trois élèves restants n'utilisent pas les commandes de messages. Il s'agit de deux garçons et une fille. Nous pouvons supposer que ces élèves n'avaient pas compris la fonctionnalité des commandes des messages et c'est pourquoi ils ne sont pas arrivés à les utiliser dans leur projet. Selon une étude de Flannery et al. (2013), seulement les élèves les plus avancés de leur classe arrivent à manipuler de manière opérationnelle les messages.

Nos résultats sont similaires que ceux provenant de l'étude que nous avons réalisée auprès des élèves de cours préparatoires (Touloupaki et al., 2018). Les élèves de maternelle font la même erreur que ceux de cours préparatoires concernant le positionnement de la commande « envoyer un message ». D'autre part, la commande « quand message reçu » ne pose pas de problèmes aux élèves de ces deux niveaux. Un tel résultat est assez logique du moment où il est bien connu que, le niveau développemental des enfants est assez similaire entre ces deux groupes d'âges (5-6 ans et 6-7 ans). Le taux de réussite en termes de manipulation de la notion en question est aussi assez similaire. Pourtant, les élèves de cours préparatoires réussissent un peu mieux que les élèves de maternelle (6/12 pour l'école maternelle et 8/12 pour le cours préparatoire).
Enfin, il convient de souligner l'importance du scénario pédagogique utilisé au cours de ce processus d'apprentissage. Les résultats de cette recherche sont dépendants du scénario pédagogique et des stratégies d'enseignement utilisées.

5 Discussion

Notre étude a eu un statut exploratoire et nous avons conscience qu'elle souffre de sérieuses limites. Tout d'abord, notre échantillon est de faible taille et nos observations ont duré seulement trois mois. Il n'est pas facile de mettre en place un scénario aussi élaboré dans des conditions réelles de classe. C'est pourquoi, nous avons eu besoin de nous adapter à la disponibilité du temps de l'enseignante de la classe et de réaliser onze séances au lieu de treize, ce qui était le programme initial.
D'autres recherches avec des échantillons plus larges seraient nécessaires, afin de pouvoir mieux étudier comment les jeunes enfants parviennent à se familiariser avec la notion de message.

Nous ne pouvons évidemment pas prétendre que les sujets considérés se sont approprié le concept de message, seulement qu'ils se sont montrés capables d'utiliser, dans les conditions d'un scénario donné, les commandes de messages de ScratchJr. Cela ouvre des possibilités dans la perspective de créer des programmes relativement sophistiqués pouvant gérer la communication entre entités différentes.

On a ainsi des perspectives permettant de diversifier à l'école primaire les possibilités de familiarisation pratique des jeunes avec l'informatique et de mettre en contact les élèves avec des notions qu'ils retrouveront sous une forme théorique plus tard dans leur scolarité. Il sera aussi nécessaire d'investiguer plus en détail les erreurs des élèves, afin de concevoir des moyens de les surmonter.

6 Références

1. Bers, M. U. : Designing Digital Experiences for Positive Youth Development: From Playpen to Playground. 1st edn.Oxford University Press, New York (2012).
2. Cobb, P., Confrey, J., DiSessa, A., Lehrer, R., & Shauble, L.: Design Experiments in Educational Research. Educational Researcher, 32(1), 9-13 (2003).
3. Fatourou, E., Zygouris, N., Loukopoulos, T., & Stamoulis, G.: Teaching concurrent programming concepts using Scratch in primary school: Methodology and Evaluation. International journal of engineering pedagogy,8(4), 89-105 (2018).
4. Fessakis, G.,Gouli, E., Mavroudi, E. : Problem-solving by 5-6 years old kindergarten children in a computer programming environement : A case study. Computers & Education, 63, 87-97 (2013).
5. Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., Dreschler, G., Aldana, G.: Assessment of computer science learning in a scratch-based outreach program. In Proceedings of the 44th SIGCSE technical symposium on computer science education, ACM, Denver (2013).
6. Flannery, L. P., Kazakoff, E. R., Bonta, P., Silverman, B., Bers, M. U., & Resnick, M.: Designing ScratchJr: Support for Early Childhood Learning Through Computer Programming. In: Proceedings of the 12th international conference on interaction design and children,pp.1-10.ACM,New York,(2013).
7. Komis, V., Touloupaki. S, Baron, G.-L. : Analyse cognitive et didactique du langage de programmation ScratchJr. In : Proceedings of the 6th conference Didapro-DidaSTIC, pp. 109-121.Presses universitaires de Namur. Namur (2017).

8. Meerbaum-Salant, O., Armoni, M., Ben-Ari, M.,: Learning computer science concepts with Scratch. In: Proceedings of the 6th international workshop on computing education research,
9. .pp. 69-76.ACM, New York,(2010)
10. Nikolos, D., Komis, V.,: Synchronisation in Scratch : A case study with education science students. Journal of computers in mathematics and science teaching 34(2),223-241(2015).
11. Robert, F. (1985). L'utilisation de l'ordinateur dans l'enseignement primaire : L'exemple de la France. Enfance, 38(1), 19-30. https://doi.org/10.3406/enfan.1985.2857.
12. Touloupaki, S., Baron, G.-L., & Komis, V. : Un apprentissage de la programmation dès l'école primaire: le concept de message sur ScratchJr. In : Proceedings of the 7th conference Didapro-DidaSTIC, pp.303-323.Peter Lang AG, Bern Switzerland (2018).

Vers un modèle de scénarisation pour l'enseignement de la pensée informatique à l'école primaire

Olivier Brunet[1], Amel Yessad[1], Mathieu Muratet[1,2], Thibault Carron[1,3]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[2] INS HEA, 92100 Suresnes, France
[3] Université Savoie Mont Blanc, F-73000 Chambéry, France
olivier.brunet@reseau-canope.fr
amel.yessad@lip6.fr
mathieu.muratet@lip6.fr
thibault.carron@lip6.fr

Résumé. De nombreuses initiatives institutionnelles et non institutionnelles proposent des ressources pour l'apprentissage de l'informatique à l'école primaire. Or, ces ressources sont difficiles à utiliser en classe d'une part, parce qu'elles ne sont pas suffisamment adaptables aux contextes hétérogènes des classes de primaire et n'explicitent pas clairement les conceptualisations informatiques sous-jacentes travaillées par les élèves et d'autre part, parce que les enseignants manquent de formation pour enseigner spécifiquement ces concepts informatiques. Nous avons mené deux ateliers de conception avec des enseignants de primaire pour travailler sur la construction d'un modèle de scénarisation. Ce dernier permet d'orchestrer des ressources existantes dans le but de faciliter leur réutilisation en classe de primaire. Ce travail a permis d'identifier les éléments de structure et de contenu nécessaires pour un scénario d'apprentissage de l'informatique à l'école primaire.

Mots clés : Pensée informatique, orchestration, scénarisation, ressources pédagogiques, théories de l'activité.

1 Introduction

Pour l'enseignement de l'Informatique à des jeunes élèves de l'école primaire, des sites institutionnels ainsi que de nombreuses initiatives (Class Code[1], 1,2,3 Codez ![2]) proposent des tâches pédagogiques sous forme de ressources avec des objectifs d'apprentissage dans les domaines de la science et de la pensée informatique plus ou moins explicités. Ces ressources peuvent être accompagnées de témoignages d'enseignants de primaire qui relatent et partagent les projets pédagogiques réalisés avec leurs élèves ; toutefois généralement sans véritable structuration ou harmonisation entre les témoi-

[1] https://pixees.fr/classcode-v2/ consulté le 30 octobre 2019
[2] https://www.fondation-lamap.org/fr/123codez consulté le 30 octobre 2019

gnages. Les sites des concepteurs de robots (Lego, Ozobot, Thymio, BlueBot...) proposent des ressources visant explicitement des apprentissages en informatique. Elles sont souvent organisées par difficulté qui, sans être articulées en scénarios pédagogiques, sont reliées à un curriculum en informatique. Il en résulte que l'exploitation de ces ressources reste difficile pour les enseignants. Malgré la multiplication de ressources visant l'apprentissage de l'informatique à destination des jeunes élèves, leur réutilisation par les enseignants de primaire reste très limitée et dépend énormément de l'appétence des enseignants pour les sciences informatiques et leur capacité à utiliser des artefacts techniques (robots, environnement de programmation, etc.).

Dans le cadre de cet article, nous abordons la question de recherche suivante : est-il possible d'outiller les enseignants de primaire pour favoriser et faciliter l'exploitation de ces ressources disponibles ?

Nous faisons l'hypothèse que l'orchestration de ces ressources au sein de scénarios pédagogiques adaptés permet de favoriser et de faciliter leur usage par les enseignants de primaire. Deux ateliers de travail sur un scénario-test d'apprentissage de l'informatique avec des enseignants ont été organisés pour identifier les éléments de structure et de contenu qu'un scénario devra intégrer afin de favoriser et faciliter son utilisation en classe.

2 Repères historiques sur l'enseignement de l'informatique à l'école

La question de l'enseignement de l'informatique, qui remonte aux années 70, se caractérise par un mouvement de balancier entre deux conceptions de ce qu'il faut enseigner. D'un côté, une vision d'une informatique-outil qui promeut la formation à l'usage des outils. De l'autre, une vision de l'informatique en tant qu'objet d'enseignement qui considère qu'il faut avant tout enseigner les concepts et méthodes de l'informatique [1].

Le travail présenté dans cet article s'inscrit dans cette dernière vision préconisée par l'Académie des sciences [2] et structurée autour de la pensée informatique : « L'informatique dans les sciences devient bien plus qu'un outil de calcul. Elle conduit à une nouvelle forme de pensée, appelée "pensée informatique" (computational thinking en anglais) ».

Selon Jeannette Wing [3], la pensée informatique met en jeu un répertoire de cinq capacités cognitives : (1) penser l'enchaînement séquentiel des actions pour résoudre un problème (pensée algorithmique), (2) sélectionner les informations pertinentes pour résoudre un problème (abstraction), (3) déterminer quelle solution, parmi plusieurs, est la mieux adaptée pour traiter un problème (évaluation), (4) décomposer un problème complexe en sous-problèmes simples (décomposition), et (5) inférer une solution à un problème général à partir de l'identification de régularités dans des instances ou d'éléments d'une solution existante à un problème analogue (généralisation).

Gilles Dowek [4] quant à lui propose une structuration de l'informatique en quatre concepts afin que les contenus enseignés donnent une image fidèle de la discipline elle-même : (1) l'information numérique, qui permet de représenter sous une forme unifiée des informations de nature et de domaine divers ; (2) les algorithmes, qui spécifient de

façon abstraite, sous la forme d'un enchaînement précis d'opérations, les traitements à effectuer sur l'information ; (3) les langages qui permettent de traduire des algorithmes abstraits en programmes exécutables par les machines ; et (4) les machines informatiques qui permettent d'exécuter les programmes, de stocker les données et de gérer les communications.

Ainsi, l'informatique ne se réduit pas au codage qui ne constitue que la phase finale de traduction dans un langage de programmation. Elle est avant tout une activité conceptuelle, qui peut en partie s'enseigner sans ordinateur (approche de l'informatique débranchée qui a pour objet de construire des concepts). Ainsi conçu, l'informatique quitte le statut d'outil pour ce qu'elle est en réalité : une science ayant ses spécificités et nécessitant un apprentissage propre.

3 Structuration de l'enseignement de l'informatique en France à l'école primaire

À l'école primaire, l'informatique disciplinaire réapparaît dans les programmes en 2016 sous la forme d'une sensibilisation à la programmation. Les textes préconisent explicitement la mise en œuvre d'activités de programmation pour déplacer un robot ou un personnage à l'écran dans une approche intégrée aux autres apprentissages, en particulier en mathématiques (repérage et déplacements dans l'espace, construction de figures géométriques).

Les conceptualisations traditionnellement associées à la programmation (notion de variable, structure conditionnelle, itération…) ne figurent pas comme des objectifs de connaissance. Et les programmes ne font aucune référence explicite à la pensée informatique. Elle n'est pas pour autant absente des préoccupations. La page d'accueil d'Eduscol consacrée au numérique dans le premier degré[3] propose un parcours M@gistere (Premières activités de programmation pour comprendre le numérique) élaboré par des acteurs extérieurs à l'Éducations Nationale (Class Code portée par l'INRIA et D-Clics numériques[4] portée par la Ligue de l'Enseignement) qui vise à transmettre les fondements historiques de la pensée informatique en préambule à des séances d'introduction à la programmation.

Le travail de Kradolfer [5] met en avant l'investissement particulièrement important demandé aux enseignants, du fait d'un manque de formation (initiale et continue) et de l'absence d'un répertoire de scénarios pédagogiques à proposer aux élèves.

Dans la suite de cet article, nous proposons un modèle de scénarisation pédagogique dont l'objectif est de construire un répertoire de scénarios à mettre à la disposition des enseignants de l'école primaire. A un niveau plus global, il s'agit de structurer et d'harmoniser la description des pratiques et faciliter ainsi la réutilisation de ressources existantes et par effet de bord, soutenir la démarche de capitalisation et de partage présente dans les pratiques actuelles et qui mérite d'être poursuivie et renforcée.

---

[3] https://eduscol.education.fr/pid29714/le-numerique-dans-le-premier-degre.html consulté le 30 octobre 2019

[4] https://d-clicsnumeriques.org/ consulté le 30 octobre 2019

4 Un modèle de scénarisation pédagogique pour orchestrer l'enseignement de l'informatique à l'école élémentaire

Le modèle de scénario que nous proposons s'appuie sur les théories de l'activité en prenant en compte les contraintes liées à l'orchestration des séances en classe de primaire.

4.1 Théories de l'activité et genèse instrumentale

Les théories de l'activité visent à comprendre l'activité d'un sujet lorsqu'il tente d'atteindre un objectif [6]. L'activité est donc différente de la tâche qui elle est prescrite. Une tâche précise ce qui doit être fait (l'objectif) et la procédure à suivre pour atteindre cet objectif. Dans un contexte scolaire, les enseignants ne proposent donc pas des activités à leurs élèves mais des tâches à réaliser avec des objectifs précis et des compétences à mobiliser (qui peuvent éventuellement être évaluées). L'activité est donc ce qui est fait par l'élève en fonction de ses compétences. L'activité est singulière, finalisée et médiatisée par un instrument [7]. L'instrument ici fait référence à un artefact associé à des schèmes d'utilisation [9]. Par exemple, un robot pédagogique (Lego, Ozobot, Thymio, BlueBot…) n'est qu'un artefact sur lequel les élèves doivent développer des schèmes en vue de s'en servir comme instrument à la résolution de la tâche proposée par l'enseignant. Ce processus progressif, appelé genèse instrumentale, permet aux élèves de s'approprier l'artefact (transformer l'artefact en instrument) et fait appel à deux types de transformation : l'instrumentalisation dans laquelle l'élève va adapter l'artefact à ses besoins et l'instrumentation dans laquelle l'artefact influence les actions de l'élève. L'enjeu du modèle de scénario que nous proposons est de permettre aux concepteurs de scénario de prendre en considération ces deux types de transformations en vue de favoriser la genèse instrumentale des artefacts manipulés lorsque les élèves sont en activité.

4.2 Orchestration de l'activité pédagogique

Introduite par Pierre Dillenbourg [10], l'orchestration a trait à la façon dont les enseignants gèrent, en temps réel l'activité pédagogique dans un contexte numérique qui induit à la fois des potentialités et des contraintes nouvelles. Les travaux menés par Pierre Dillenbourg dans le domaine des EIAH visent à prendre en compte, à côté des aspects didactiques, les contraintes très pratiques du fonctionnement de la classe dès la conception des tâches pédagogiques avec un objectif double : alléger la charge de l'enseignant (de manière à libérer son attention et à favoriser sa capacité à intervenir de manière adaptée auprès des élèves) et optimiser les potentialités d'exploitation pédagogique de l'instrumentation numérique.

Des aspects tels qu'organiser l'espace classe, s'assurer que chaque élève a avec lui les bons outils pour travailler et gérer le temps, constituent en effet une part importante du travail des enseignants et de leurs préoccupations et influencent la qualité des apprentissages. Cela est encore plus vrai lorsque les apprentissages sont accompagnés de

dispositifs informatiques qui n'ont, pour la plupart d'entre eux, pas été développés en cohérence avec les contraintes qui sont celles des enseignants et dont la mise en œuvre comporte des risques de complexification de la genèse instrumentale (pour une plus-value souvent faible). L'orchestration se préoccupe donc de répondre aux contraintes considérées comme de bas niveau (aspects logistiques, ergonomiques, organisationnels) et rejoint, en cela, nos préoccupations relatives à la description des tâches proposées dans un scénario.

4.3 Synthèse

Dans le modèle de scénarisation que nous proposons, l'objectif est d'orchestrer différentes tâches au sein de scénarios pédagogiques afin de favoriser et de faciliter leurs usages par les enseignants de primaire et d'asseoir auprès de leurs élèves des conceptualisations en informatique. Il s'agit de favoriser la genèse instrumentale auprès des élèves. Pour cette raison, le modèle de scénario s'articule autour des briques élémentaires des théories de l'activité à savoir d'une part la notion de tâche visant des objectifs centrés sur des connaissances en informatique et d'autre part la notion d'artefact.

5 Conception d'un modèle de scénarios d'apprentissage de l'informatique avec des enseignants

Nous avons mené deux ateliers avec des enseignants de primaire dont l'objectif était d'identifier les éléments de structure et de contenu que devra intégrer un modèle de scénarios et qui permettront de construire un scénario (c'est-à-dire une instance du modèle). Pour être efficaces et rendre les ateliers productifs, nous avons en amont construit un scénario-test où les tâches proposées aux élèves étaient décrites de manière sommaire et avons demandé aux enseignants de travailler sur ce scénario. Ce scénario utilise comme artefact un robot pédagogique (Ozobot). Nous avions une ligne directrice pour l'animation de ces deux ateliers : identifier des éléments de structure et de contenu pour documenter un scénario d'apprentissage en informatique. L'idée étant d'abstraire le scénario-test avec Ozobot pour définir un modèle général de scénarios.

5.1 Présentation du scénario avec Ozobot

Nous avons établi une première définition d'un scénario-test, intitulé « Alphabet de l'informatique avec Ozobot », qui vise l'appropriation de connaissances dans le domaine de l'informatique et le travail de compétences associées à la pensée informatique (abstraction, pensée algorithmique, décomposition, évaluation). Il s'appuie sur l'usage d'un robot nommé Ozobot que les élèves observent (construction d'une représentation de son fonctionnement interne) et programment (résolution de défis). Ce scénario peut être mené dès le début du cycle 3 et vise à travailler les concepts en informatique suivants :

- Le rôle de capteurs, actionneurs, contrôleur/processeur et mémoire (morte et vive) ;

- Le principe du codage d'une information (ici une instruction codée sous la forme d'une combinaison de trois couleurs) ;
- Les notions de mémoire et d'exécution ;
- La notion d'aléatoire ;
- Et la notion de variable (compteur).
- Le rôle de capteurs, actionneurs, contrôleur/processeur et mémoire (morte et vive) ;
- Le principe du codage et de langage de programmation ;
- La notion d'instruction, du transfert d'une instruction en mémoire et de l'exécution d'une instruction ;
- L'exécution de plusieurs instructions en parallèle ;
- Et la notion de variable (compteur).

5.2 Déroulement et objectifs des ateliers

Atelier 1. Le premier atelier a eu lieu en avril 2019 et a duré 2 heures. Les participants sont 8 professeurs des écoles n'ayant jamais mené de séances d'apprentissage de l'informatique. Le protocole de travail avec les enseignants s'est déroulé en trois temps :

1. La présentation de ce que recouvrent les 4 domaines de l'informatique selon la typologie de G. Dowek ;
2. La réalisation par les enseignants pendant une heure des quatre premières tâches du scénario « Alphabet de l'informatique » ;
3. La confrontation des enseignants à des questionnaires individuels pour (1) évaluer leur motivation et le degré d'effort estimé pour qu'ils puissent proposer ces tâches dans leur classe et (2) identifier ce qui pourrait diminuer cet effort (formation des enseignants, scénario pédagogique ou autres).

Atelier 2. Le deuxième atelier a eu lieu en mai 2019 et a duré 2 heures. Les participants sont 22 professeurs des écoles inscrits au M2-MEEF - Mention premier degré. Le protocole de travail avec les enseignants s'est déroulé comme suit :

4. La présentation de ce que recouvrent les 4 domaines de l'informatique selon la typologie de G. Dowek ;
5. Les enseignants sont répartis en 5 groupes pour réaliser les 4 premières tâches du scénario « Alphabet de l'informatique ». Dans chaque groupe, un participant jouait le rôle de l'enseignant et les autres le rôle des élèves. Le participant incarnant le rôle d'enseignant avait 15 minutes pour découvrir le scénario test avant de le dérouler avec les autres membres de son groupe. Chaque groupe disposait de son propre robot ;
6. Les groupes déroulent le scénario avec la possibilité de poser des questions au concepteur (informaticien faisant partie de l'équipe de recherche) ;
7. Des questionnaires individuels ont été soumis aux enseignants pour (1) déterminer leur motivation et le degré d'effort estimé par eux pour s'approprier et mettre en œuvre le scénario test en classe, (2) évaluer leur intérêt de disposer de scénarios pédagogiques en général et pour l'apprentissage de l'informatique en particulier et (3) identifier les caractéristiques d'un « scénario idéal » en explicitant clairement les

informations qui manqueraient au scénario-test pour le rendre utilisable par des enseignants de primaire.

5.3 Retours des ateliers de conception

Les enseignants ont exprimé clairement l'intérêt de réutiliser des scénarios pédagogiques faits par d'autres en les adaptant à leur contexte. Ils estiment que disposer de scénarios pédagogiques pour l'informatique est particulièrement utile pour mener une première séquence pédagogique avec leurs élèves et pour combler chez eux un manque de formation en sciences informatiques. Ils souhaitent d'une part, des scénarios qui combinent manipulation, questionnement, observation, hypothèses et validation de la part des élèves et d'autres part, des scénarios en pédagogie de projet avec des objectifs explicites et une identification des compétences travaillées. Majoritairement, ils ont estimé que le scénario-test est accessible et que l'effort d'appropriation de ce scénario est modéré. Néanmoins, ils nous ont fait part de leur inquiétude quant à la mise à disposition et la fiabilité de l'instrumentation, notamment à cause des problèmes techniques qui pourraient survenir en classe et pour lesquels ils pourraient être désarmés.

A cause de l'échantillon restreint, ces deux ateliers n'ont qu'une portée limitée, mais ils montrent qu'il est possible à des enseignants de primaire qui ne sont pas formés à la pensée informatique et qui ne l'ont jamais enseignée de s'approprier un scénario comportant des conceptualisations simples du domaine. Par ailleurs, si l'approche pédagogique (pédagogie active intégrant des situations-problèmes) est plébiscitée, les enseignants pointent une lacune du scénario-test : il manque un objectif explicite de réalisation qui permet de donner sens à l'acquisition des connaissances. Cette observation a été prise en compte pour l'élaboration du modèle de scénarisation pédagogique.

Dans la section suivante, nous présentons le modèle de scénarisation qui a émergé des deux itérations de conception avec des enseignants de primaire. La démarche de conception est incrémentale et un certain nombre d'itérations va permettre d'améliorer, d'enrichir le modèle de scénarisation et d'en valider les principes.

6 Modèle de scénarisation pédagogique

Dans le cadre de ce travail de recherche et à l'issue des deux ateliers menés avec des enseignants, les éléments d'un modèle de scénarios pédagogiques pour l'apprentissage de l'informatique à l'école primaire ont émergé. Ce modèle a pour objectif d'être adapté au contexte de l'école primaire, utilisable par des enseignants et instanciable pour construire des scénarios pédagogiques.

6.1 Besoins et contraintes liées au contexte de l'école primaire

Le déficit de formation chez les enseignants de primaire en sciences informatiques, l'organisation des classes de primaire avec la possibilité de mener des projets transversaux et l'hétérogénéité des classes et des élèves ont fait émerger plusieurs questions

quant aux éléments constitutifs du modèle de scénarios : quel est le niveau de granularité du scénario ? dans quelle mesure peut-on l'adapter au contexte d'usage ? quels éléments de contenu et de forme (représentation visuelle) sont adaptés pour décrire un tel scénario ?

Les éléments de réponse apportés, relatives à la granularité du scénario, au langage de description et à la formalisation des scénarios et leur propriété d'adaptation, ont été inspirés des modèles existants et de l'état de l'art, en particulier les travaux de Dillenbourg sur les graphes d'orchestration [11] et ceux de Komis sur la scénarisation [12] :

1. la séquence didactique est définie comme un ensemble organisé de tâches réalisées par les élèves avec le concours de l'enseignant pour atteindre un objectif spécifique (détecter une représentation, motiver les élèves, introduire une nouvelle notion, consolider un concept, évaluer une compétence, etc.),
2. un ensemble de descripteurs a émergé à l'issue de deux itérations de conception avec des enseignants (cf. Fig. 1). Un descripteur est une métadonnée ou un attribut décrivant un aspect d'une tâche composant un scénario (type de la tâche, son objectif, sa modalité sociale, l'artefact utilisé, etc.). Cet ensemble de descripteurs est susceptible d'évoluer à l'issue d'autres itérations planifiées en janvier 2020,
3. la propriété d'adaptation du scénario est cruciale pour permettre sa robustesse dans différents contextes. Cette propriété est mise en place via un descripteur spécifique (cf. section 6.3) nommé « Typologie des interventions extrinsèques ». Concrètement, ce descripteur est utile lors de l'activité des élèves réalisant les tâches du scénario, lorsque par exemple, un groupe d'élève est bloqué ou qu'il avance beaucoup plus vite que les autres. Dans les deux cas, les élèves peuvent attendre pendant un temps indéterminé l'intervention de l'enseignant. Pour répondre à cette situation, fréquente dans les classes, le modèle intègre des propositions d'interventions pour soutenir l'activité des élèves en fonction de la situation. Ces interventions doivent pouvoir être mises en œuvre aisément dans un contexte de classe et être en lien avec l'activité de l'élève. À titre d'exemples, il peut s'agir d'une complexification ou d'une simplification de la tâche, d'une reformulation ou de l'apport d'un exemple pour éclairer un concept différemment, etc.
4. Le formalisme utilisé pour communiquer et expliquer les descripteurs et les tâches aux enseignants combine une fiche pédagogique organisée en rubriques (correspondant aux descripteurs) et une représentation graphique reprenant le modèle des graphes d'orchestration [11].

6.2 Cadre d'usage et éléments constitutifs du modèle

Le modèle que nous proposons est prévu pour servir de cadre à la production de scénarios d'apprentissage de l'informatique à l'école. Il est le produit de deux itérations de conception avec des enseignants de primaire. Ce modèle comprend 5 éléments :

- Une présentation de l'approche pédagogique ;
- Une architecture globale du scénario ;

- Un langage de description qui est composé de descripteurs relatifs aux séquences pédagogiques (objectifs didactiques, stratégies pédagogiques, mode d'organisation de la classe…) ;
- Une méthodologie de conception, décrivant un ensemble d'étapes pour aboutir à un scénario validé ;
- Et un formalisme pour le représenter qui consiste en une représentation graphique du scénario accessible aux non informaticiens.

6.3 Description détaillée du modèle de scénarisation

Nous détaillons ici les cinq éléments qui composent le modèle.

Approche pédagogique globale. Un scénario d'apprentissage de l'informatique à l'école primaire devrait vérifier les trois principes suivants, exprimés par les enseignants participants aux deux ateliers, dans le but de favoriser son usage au sein des classes :

- Mettre en place une pédagogie par projet ;
- Émettre clairement un objectif de réalisation explicite à l'issue du scénario (résoudre un défi, fabriquer un objet, etc.) ;
- Permettre un apport de la connaissance « juste à temps » (apprentissage au moment opportun) et donc éviter de surcharger cognitivement les différents acteurs (enseignant et élèves).

Architecture globale du scénario. Un scénario comprend deux phases. Ces deux phases ont émergé fortement lors des ateliers organisés avec des enseignants de primaire et visent également à favoriser les deux types de transformation en œuvre lors de la genèse instrumentale.

D'abord, la réalisation d'un projet guidé ayant pour objectif l'acquisition de connaissances/compétences relatives à la science informatique et centré sur la découverte d'un artefact. La transformation dominante, visée ici, est celle de type instrumentation pour développer chez l'élève les schèmes induits par l'artefact (affordance). Cette phase vise l'acquisition de connaissances/compétences par les apprenants et leur mise en réussite à travers un guidage fort et une granularité fine des tâches à réaliser suivant deux principes :

- Une tâche fait travailler au plus une seule nouvelle connaissance/compétence ;
- Les compétences/connaissances travaillées sont réinvesties au fil du scénario dans un esprit de progression.

La seconde phase propose aux apprenants de réaliser un projet parmi un ensemble de propositions prévues par le scénario. Ces projets permettent de réinvestir des connaissances/compétences travaillées lors de la première phase. La transformation dominante, visée ici, est celle de type instrumentalisation où l'élève doit exploiter l'artefact à bon escient pour répondre à ses besoins. Contrairement à la première phase, où les élèves

sont fortement guidés, dans la seconde phase, ils doivent opérer des choix et les justifier au moment de la présentation de leur projet.

Langage de description d'un scénario. Le langage de description que nous proposons est composé d'un ensemble de descripteurs permettant de caractériser une tâche composant un scénario. Tous les descripteurs ne sont pas nécessaires pour décrire une tâche. Le choix d'utiliser ou non un descripteur appartient au concepteur du scénario. La figure 1 est un diagramme entité-association explicitant l'ensemble des descripteurs d'une tâche et les liens qui les relient.

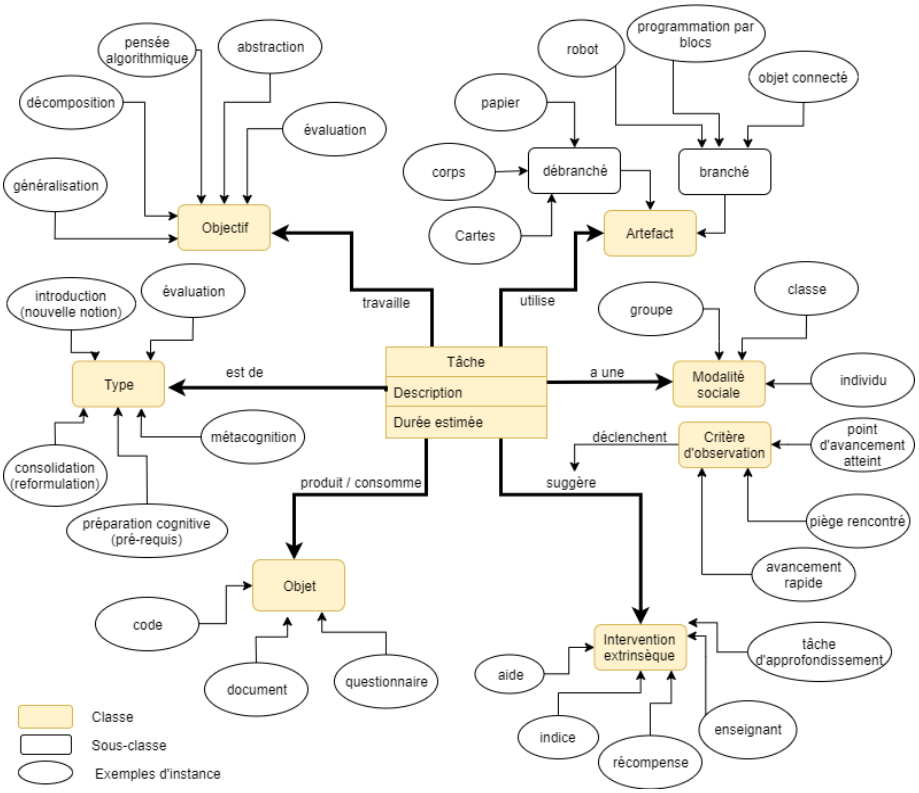


Fig. 1. Modèle de tâche (les rectangles arrondis représentent les descripteurs d'une tâche et les ellipses des valeurs possibles pour ces descripteurs)

Méthodologie d'instanciation (élaboration) d'un scénario à partir du modèle et sa représentation graphique. La méthodologie proposée consiste à partir de tâches existantes et de les articuler dans un scénario en identifiant ce qu'elles mettent en jeu dans les domaines de la science informatique et de la pensée informatique. Elle s'articule suivant le processus itératif suivant :

1. Identifier des tâches existantes ;
2. Identifier les concepts/connaissances mis en jeu dans les tâches en lien avec les artefacts utilisés ;
3. Identifier les compétences relatives à la pensée informatique mises en jeu dans les objectifs visés de chaque tâche ;
4. Construire un scénario pédagogique en articulant les tâches reprises, adaptées et celles nouvellement conçues ;
5. Formaliser le scénario dans un graphe d'orchestration en décrivant les tâches (durée, objectif, artefact...) et les liens entre les tâches (précédence, composition, etc.) ;
6. Ajouter des descripteurs permettant d'analyser les tâches et leurs dépendances ;
7. Expérimenter le scénario avec des élèves ;
8. Observer les activités des élèves (enregistrements vidéo, traces numériques produites par les artefacts…) ;
9. Analyser : identifier les risques d'échecs et les causes ;
10. Réitérer à l'étape 4 en fonction des résultats de l'étape 9 d'analyse.

Représentation d'un scénario concret « Alphabet de l'informatique avec Ozobot ». La figure 2 présente un exemple de formalisation du scénario « Alphabet de l'informatique avec le robot Ozobot » en s'inspirant des graphes d'orchestration proposés par Pierre Dillenbourg [11]. Sur cet exemple, les tâches sont représentées par les nœuds du graphe et sont décrits à l'aide du modèle (type de tâche, objectif, artefact utilisé, critères d'évaluation…). Le graphe permet de visualiser le déroulement du scénario et les possibilités d'adaptation d'une tâche en fonction de l'activité des élèves.

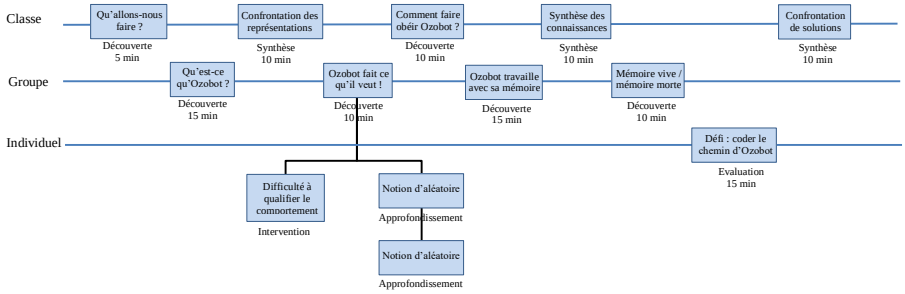


Fig. 2. Exemple de formalisation du scénario dans un graphe d'orchestration

7 Conclusion

Le travail présenté dans cet article a pour objectif de favoriser les conditions d'exploitation de ressources pédagogiques visant l'apprentissage de l'informatique à l'école primaire et issues des multiples initiatives existantes en structurant et harmonisant l'approche.

Pour répondre à cet objectif, nous avons proposé un modèle de scénarisation reposant sur les théories de l'activité et les graphes d'orchestration. Ce modèle est le résultat

de deux itérations de conception avec des enseignants de primaire dans le but de favoriser l'appropriation du modèle obtenu par ces mêmes enseignants.

Il constitue à la fois une méthodologie et un cadre de réflexion pour la production de scénarios et un formalisme de description en vue d'une réutilisation des scénarios dans d'autres contextes que ceux dans lesquels ils ont été conçus.

Cette conception en laboratoire (car seulement en partie « écologique ») mérite maintenant d'être raffinée pour une utilisation à plus grande échelle en contexte écologique. Nous envisageons donc d'itérer sur ce modèle dans une démarche de co-conception avec des enseignants de terrain pouvant concevoir leurs propres scénarios, adapter des scénarios construits par d'autres et les expérimenter avec leurs propres élèves. Ainsi, plusieurs séances de co-conceptions sont prévues en 2020.

References

1. Baron, G.-L., Drot-Delange, B. (2017). L'informatique comme objet d'enseignement à l'école primaire Française ? Mise en perspective historique. *Revue Française de Pédagogie*, 195.
2. Institut de France, Académie des sciences : L'enseignement de l'informatique en France – Il est urgent de ne plus attendre. [Consulté le 30 octobre 2019]. https://www.academie-sciences.fr/fr/Rapports-ouvrages-avis-et-recommandations-de-l-Academie/l-enseignement-de-l-informatique-en-france-il-est-urgent-de-ne-plus-attendre.html (2019).
3. Wing, J.: Computational thinking. In: Communication of the ACM, 49(3), 33-35 (2006).
4. Dowek, G. : Les quatre concepts de l'informatique. Dans : Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques, Patras, Grèce. pp. 21-29 (2011).
5. Kradolfer, S., Dubois, S., Riedo, F., Mondada, F., Fassa, F.: A sociological contribution to understanding the use of robots in schools: the Thymio robot - ICSR 2014, LNAI 8755, 217-228 (2014).
6. Daniellou, F., Rabardel, P.: Activity-oriented approaches to ergonomics: some traditions and communities. In: Theoretical Issues in Ergonomics Science, 6(5), 353-357 (2005).
7. Nogry, S., Decortis, F., Sort, C., Heurtier, S. : Apports de la théorie instrumentale à l'étude des usages et de l'appropriation des artefacts mobiles tactiles à l'école. Dans : Revue STICEF, Volume 20 (2013).
8. Ouari, S.: Adaptation à la volée de situations d'apprentissage modélisées conformément à un langage de modélisation pédagogique. Thèse. Université Grenoble Alpes, (2011).
9. Rabardel, P. : Les hommes et les technologies, approche cognitive des instruments contemporains. Paris : Armand Colin (1995).
10. Dillenbourg, P.: Design for classroom orchestration. In: Computers & Education, 69, 485–492 (2013).
11. Dillenbourg, P.: Orchestration Graphs : Modeling Scalable Education. Lausanne, EPFL Press (2015).
12. Komis, V., Tzavara, A., Karsenti, T., Collin, S., Simard, S.: Educational scenarios with ICT: an operational design and implementation framework. In R. McBride & M. Searson (Eds.), Proceedings of SITE 2013--Society for Information Technology & Teacher Education International Conference (pp. 3244-3251). New Orleans, Louisiana, United States: Association for the Advancement of Computing in Education (AACE), (2013).

Une approche didactique pour l'introduction de la Programmation Orientée-Objet en classe

Fahima Djelil, Maria Teresa Segarra Montesinos et Jean-Marie Gilliot

Lab-STICC, IMT Atlantique, 29 238 Brest, France
Fahima.djelil@imt-atlantique.fr
mt.segarra@imt-atlantique.fr
jm.gilliot@imt-atlantique.fr

Résumé. Ce papier s'intéresse à l'ingénierie didactique pour l'enseignement de la Programmation Orientée-Objets aux débutants. Il propose de définir une nouvelle approche didactique, dérivant d'une pratique d'enseignement répandue dans la communauté anglo-saxonne, l'approche « objet en premier ». Cette nouvelle approche, dite « par emboitement », décrit un processus à trois étapes permettant l'introduction de trois niveaux de concepts de manière progressive : utilisation d'objets, création de classes et conception. Nous nous focalisons sur le principe de l'emboitement, qui permet à l'apprenant de se concentrer sur un niveau de concepts en lui occultant les concepts du niveau suivant, dans le but de réduire les difficultés d'apprentissage. Nous montrons dans un premier temps, comment nous avons implémenté cette approche dans un nouveau micromonde de programmation. Ensuite, nous montrons la transposition de cette approche dans un environnement de classe en école d'Ingénieur. Enfin, nous examinons cette approche selon une dimension temporelle afin de couvrir l'ensemble d'un programme d'enseignement, en présentant une vision itérative et en spirale.

Mots-clés : Didactique de l'Informatique, Enseignement de la Programmation, Paradigme Orienté-Objet.

1 Introduction

L'enseignement de la Programmation Orientée-Objet (POO) aux débutants peut s'avérer difficile. En particulier, le changement de paradigme de programmation dans les enseignements introductifs est une source de difficulté souvent perçue chez les débutants [1]. Cela est confirmé par de nombreuses recherches en didactique de l'informatique [2] [3] [4], où l'enseignement de la programmation constitue l'un des défis majeurs [5].

Il existe plusieurs tendances pour l'introduction de la programmation et du paradigme Orienté-Objet (OO), qui ont été mises en pratique à grande échelle et qui se différencient par le type du premier paradigme introduit dans les enseignements [6]. L'approche « Objets en Premier » se distingue par un grand nombre de travaux dans la communauté anglo-saxonne, montrant l'intérêt développé pour cette approche en vue de réduire les difficultés d'apprentissage chez les débutants [3].

L'objectif de ce papier, est de décrire une nouvelle approche didactique pour l'introduction des concepts fondamentaux de l'OO, dite « par emboitement », et qui dérive de l'approche « Objets en Premier ». Nous montrons, par ailleurs, comment cette nouvelle approche est implémentée dans le micromonde de programmation PrOgO [7]. Nous montrons, ensuite comment elle peut se transposer pour être utilisée plus largement en classe, dans l'organisation des activités d'enseignement.

Par ailleurs, ce travail ne s'inscrit pas dans un paradigme méthodologique comparatiste. Notre objectif est de caractériser cette approche didactique, afin de montrer comment elle peut aider à structurer un programme d'enseignement introductif de la POO en classe, plutôt que de la comparer à une autre approche didactique.

Ce papier s'organise comme suit. La section 2 décrit comment l'approche « Objet en Premier » se met en pratique dans les curricula et comment elle est décrite dans la littérature. Nous décrivons ensuite notre nouvelle approche didactique qui en dérive (section 3), son implémentation dans l'environnement PrOgO (section 4), puis son élargissement en vue d'un usage en classe (section 5). Enfin, nous concluons par nos principales contributions (section 6).

2 Approche « Objet en Premier »

2.1 L'approche « Objet en Premier » dans les curricula

L'approche « Objet en Premier » (*Object-First*) est décrite dans le rapport CC2001(Computing Curricula 2001) [6] publié par l'ACM (*Association for Computing Machinery*) et l'IEEE Computing Society, sur la base des pratiques d'enseignement les plus répandues dans l'enseignement supérieur à l'échelle mondiale. Cette approche accorde une grande importance aux fondamentaux de la conception et de programmation OO dès le début des enseignements. En pratique, les premiers enseignements abordent rapidement les notions « d'objets » et « d'héritage », afin de confronter de manière précoce les apprenants à ces concepts. Après l'introduction de ces notions dans des programmes interactifs simples, les enseignements passent aux notions de programmation classique comme les structures de contrôle, tout en restant focalisés sur la conception OO. Le déroulement des enseignements permet ensuite d'introduire les algorithmes, les structures de données fondamentales et la conception logicielle avec plus de détails.

Le principal avantage de cette approche est la confrontation précoce des débutants à la POO [6]. Elle peut néanmoins présenter des inconvénients liés à la complexité des langages de programmation utilisés pouvant surcharger les débutants. Les spécificités des langages de POO tels que C++ et Java peuvent accentuer les difficultés d'apprentissage avec cette approche. C'est justement pour remédier à cela que l'usage des environnements visuels et interactifs comme les micromondes de programmation s'est beaucoup développé. La littérature montre que l'approche « Objet en Premier » a été portée par l'usage de ce type d'environnements [2]. L'objectif étant d'aider les débutants à construire une compréhension fine et rapide des concepts abstraits de l'OO, en leur offrant des activités attrayantes et signifiantes.

2.2 L'approche « Objet en Premier » dans la littérature

Dès le milieu des années 1990, l'approche « Objet en Premier » s'est beaucoup répandue en pratique et a émergé comme une réelle approche didactique dans l'enseignement introductif de la POO. À titre indicatif, l'interrogation de la base de données ACM Digital Library sur l'expression « *Object-First approach* » (approche Objet en Premier) retourne 208.739 résultats [recherche réalisée le 16 septembre 2019].

Une analyse de plus de 200 contenus a permis d'identifier trois étapes introduisant trois catégories de concepts par lesquelles cette approche se met en application [2]:

- Utilisation d'objets : l'apprenant utilise des « objets » préalablement implémentés. Il passe à la définition de « classes », une fois le concept d'objet maîtrisé. On se concentre alors sur l'usage de l'objet avant son implémentation.
- Création de classes : l'apprenant définit et implémente des classes et crée des instances de classes préalablement définies. On aborde alors cet aspect de programmation de manière concrète et créative.
- Concepts : l'apprenant apprend des principes généraux du paradigme OO, par la création de modèles OO. On se concentre sur les aspects conceptuels de l'OO.

De nombreuses expériences d'enseignements décrites dans la littérature peuvent se ramener à cette approche. Ainsi, Woodworth et Dann [8] décrivent une approche par laquelle l'apprenant est guidé dans la conception et l'utilisation d'abstractions, en modélisant des propriétés d'objets manipulés dans une précédente phase de résolution de problèmes.

Buck et Stucki [9] proposent de structurer un cours introductif de la POO de façon à permettre aux apprenants de débuter en modifiant un code préalablement implémenté, puis de concevoir des parties du système OO. Ils soutiennent une approche où les détails d'un langage de programmation doivent être introduits de façon progressive selon les besoins.

Kölling et Rosenberg [1] ont mis en application une démarche qui consiste à commencer par les objets et non par la programmation impérative. L'apprenant est amené à créer et à manipuler des objets à partir de classes existantes, puis à modifier des programmes de code existants au lieu de les créer par lui-même. Cette démarche incite à la lecture de code bien construit, afin de le reproduire et d'acquérir des bonnes pratiques de programmation. Ces programmes sont modélisés en OO et montrent des relations hiérarchiques entre classes. Cette approche est implémentée dans l'environnement Greenfoot [10].

Becker [11] adopte une approche qui se consacre en premier à la création et à l'utilisation d'objets, avant d'introduire les notions conceptuelles de l'OO par l'extension de classes existantes. Les notions de la programmation impérative (structures de contrôle conditionnelles et itératives, expressions booléennes, …) sont introduites en dernier de manière progressive. Cette approche a été implémentée et mise en expérience dans les environnements Karel et ObjectKarel [12].

Joel et Frens [13] présentent une méthodologie en trois phases. La première (objets en premier) consiste à apprendre à résoudre un problème en identifiant les objets à

déclarer dans un programme. La seconde (classes et méthodes) se consacre à l'introduction de méthodes par leur déclaration et définition dans une classe. Cette phase permet d'introduire des notions de programmation impérative. La troisième phase (héritage) permet d'introduire les concepts d'héritage et de polymorphisme.

Cooper, Dann et Pausch [14] ont adopté l'approche « Objet en Premier » en utilisant l'environnement Alice. Leur démarche consiste à introduire les objets dans un contexte visuel et significatif. Les notions de programmation impérative sont introduites de manière progressive en se focalisant sur le comportement des objets.

On peut observer que l'intérêt développé pour cette approche est induit par le changement de paradigme de programmation en enseignement de l'informatique, source de difficultés de compréhension des fondamentaux de l'OO. Cette approche préconise l'OO dès le début des enseignements, et une découverte progressive de la programmation impérative avec une focalisation sur l'OO. Or, en pratique le choix du paradigme dans l'introduction des concepts de programmation reste très varié. À titre indicatif, le rapport CC2001 liste six approches différentes d'enseignement de la programmation de l'informatique aux débutants. Chacune présentant ses avantages et ses inconvénients.

Nous ne focalisons pas notre attention sur le paradigme utilisé en premier dans l'approche « Objet en premier », mais sur la manière dont cette approche introduit les concepts fondamentaux de l'OO. Nous ressortons une approche didactique par « emboîtement » des concepts OO.

3 L'approche didactique par emboîtement des concepts fondamentaux de l'OO

L'approche « Objet en Premier » nous a permis de définir une nouvelle approche didactique pour l'introduction des concepts de POO, par un principe « d'emboîtement » [7]. Nous retenons dans l'approche « Objet en Premier » l'ingénierie didactique par laquelle les concepts fondamentaux de l'OO sont introduits aux débutants. Celle-ci se caractérise par trois étapes qui se suivent par emboîtement :

1) Utilisation d'objets : l'apprenant apprend à se familiariser avec le concept de l'objet et ses propriétés. À ce stade, le concept de classe est volontairement occulté. On montre brièvement la relation qui lie l'objet à la classe (l'objet étant une instance de classe), sans se focaliser sur la classe. L'apprenant se concentre dans cette phase sur comment les objets sont utilisés dans la résolution d'un problème donné.
2) Création de classes : l'apprenant apprend à créer des abstractions ou des modèles à partir de propriétés d'objets, une fois ce concept maîtrisé. À ce stade les relations et associations entre différentes classes ne sont pas apparentes.
3) Conception : nous proposons de retenir la notion de conception, plutôt que le terme concept, dans le sens où les concepts introduits le sont dans une logique de conception d'application. L'apprenant apprend à construire des modèles OO et à organiser son programme en classes liées par des relations spécifiques.

Ces trois étapes emboitent les concepts OO de façon à amener l'apprenant, à garder à l'esprit qu'une classe permet de définir les propriétés d'objets et que la modélisation a pour objectif de spécifier la solution à un problème, que l'on peut concrètement résoudre au moyen d'objets. C'est un moyen d'aider les apprenants débutants à mieux mentaliser le paradigme OO.

Nous voyons une approche didactique se caractérisant par un emboîtement de trois catégories de fondamentaux de l'OO (Fig. 1). Il s'agit à chaque étape, de permettre à l'apprenant, d'interagir avec les propriétés d'une catégorie tout en lui occultant les propriétés de la catégorie suivante. L'objectif étant de s'affranchir de la complexité de la modélisation OO lors de la résolution de problèmes par les débutants, pouvant constituer un frein à leur apprentissage.

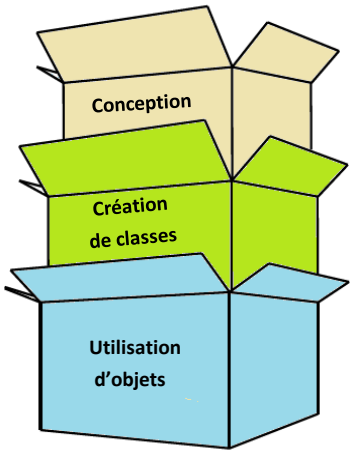


Fig. 1. Approche didactique par emboitement des concepts OO [7].

Cette approche didactique, nous a servi pour la conception de l'environnement PrOgO [7], qui implémente les deux premières phases, utilisation d'objets et création de classes. La troisième phase n'étant pas implémentée, l'enseignant peut l'introduire sous forme de débriefing avec ses apprenants après l'utilisation de l'environnement PrOgO.

4 Implémentation de l'approche didactique dans l'environnement PrOgO

4.1 L'environnement PrOgO

PrOgO[1] est un micromonde de programmation OO, conçu dans le but d'aider les débutants à apprendre par le jeu les fondamentaux de l'OO. Il repose sur une métaphore

[1] PrOgO est le résultat d'une thèse (prix AUF 2018) financée dans le cadre du Programme d'Investissement d'Avenir par le projet E-Education Tactileo (2013-2016). http://progo.iut-lepuy.fr

de jeu de construction et d'animation en 3D. Son interface graphique est essentiellement constituée d'une scène 3D dédiée à la création et à l'animation de constructions graphiques (Fig.2), et d'un éditeur à auto-complétion qui permet de visualiser ou de saisir des instructions de code en langage C++ entièrement synchronisé avec la scène 3D. Les constructions graphiques sont des représentations visuelles de systèmes OO.

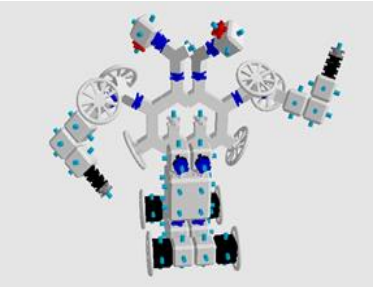


Fig.2. Une construction 3D dans PrOgO [7].

4.2 Modèle OO sous-jacent aux constructions 3D dans PrOgO

L'interface de PrOgO fournit des modèles graphiques 3D représentant des classes qui donnent lieu à des objets directement utilisables par l'apprenant. Chaque construction 3D réalisée par l'apprenant constitue une instance d'une nouvelle classe et consiste en un système OO qui peut être décrit par un diagramme de classes UML « *Unified Modeling Language* » (Fig.3). Deux types de classes permettent de modéliser les composants graphiques servant au jeu de construction [7]:

- La classe « `ComposantStructurel` » modélise l'ensemble des composants graphiques servant à la construction de la réalisation 3D. Cette classe possède un attribut « `couleur` » et deux méthodes « `connecter()` » et « `colorierPendant()` », qui permettent respectivement de connecter un objet à un autre et de colorier un objet pendant un temps donné.
- La classe « `ComposantActif` » modélise l'ensemble des composants structurels qui sont capables d'exécuter un mouvement (une action de rotation pendant un temps donné). Cette classe hérite de la classe `ComposantStructurel`, et possède deux autres caractéristiques, l'attribut « `angleDeRotation` » et la méthode « `tournerPendant()` ».

Les classes visualisées à l'interface dérivent soit de la classe « `ComposantStructurel` », c'est le cas de « `Base` », « `Cube` », « `BifurcationY` » et « `Ressort` », soit de la classe « `ComposantActif` », c'est le cas de « `Engrenage` » et « `Roue` ». La classe « `Base` » possède une unique instance et représente la base de toute construction dans la scène 3D.

Une construction 3D possède également deux méthodes : la méthode « `animer()` » regroupant les comportements d'animation programmés par l'apprenant lors de la réalisation de sa construction ; la méthode

« `réinitialiser()` » qui permet de réinitialiser une construction 3D à son état de création.

On voit apparaître dans ce modèle OO trois relations de conception :

- L'héritage : les classes « `Engrenage` » et « `Roue` » héritent de la classe « `ComposantActif` ». Les classes « `Base` », « `Cube` », « `BifurcationY` » et « Ressort » héritent de la classe « `ComposantStructurel` ». Les classes dérivées partagent les mêmes propriétés que la classe de laquelle elles dérivent.
- La composition : la classe « `Construction3D` » se compose à la fois de la classe « `Base` » et de la classe « `ComposantStructurel` ». La suppression de la classe composée induit la suppression des classes qui la composent.
- L'agrégation : s'applique sur la classe « `ComposantStructurel` ». Tout composant structurel se trouve lié à un autre composant structurel, mais la suppression de l'un n'implique pas la suppression de l'autre.

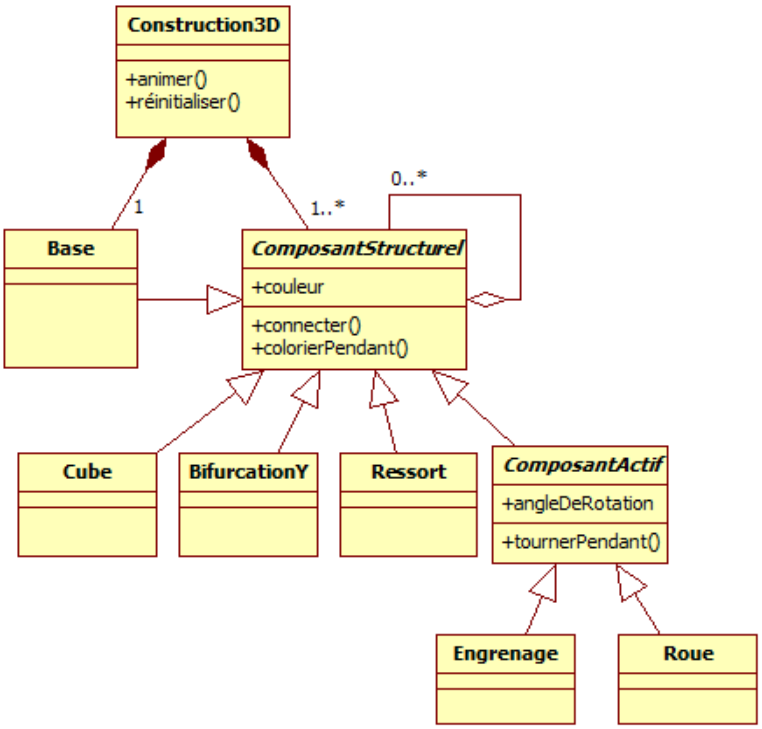


Fig.3. Diagramme de classes UML modélisant une construction 3D dans PrOgO [7].

4.3 Implémentation de l'approche didactique

Lors du jeu de construction, l'apprenant est amené à imaginer et à créer la structure qu'il souhaite, tout en accédant aux concepts OO sous-jacents. L'apprentissage se

déroule en deux phases selon l'approche didactique par emboîtement des concepts OO : utilisation d'objets et création de classes (Tableau 1). La troisième phase n'est pas implémentée dans la version finale de PrOgO, mais l'enseignant peut l'introduire en classe en expliquant le modèle OO sous-jacent aux constructions 3D réalisées par l'apprenant.

Tableau 1. Concepts OO introduits par l'approche didactique dans l'environnement PrOgO.

Étape didactique	Concept
(1) Utilisation d'objets	1.1 Lien existant entre l'objet et la classe : un objet est une instance de classe. 1.2 Caractéristiques d'objet : un objet se caractérise par des attributs et des méthodes. 1.3 Modification de l'état d'un objet : modifier la valeur d'un attribut d'objet ou réaliser un appel de méthode sur l'objet.
(2) Création de classes	2.1. Rôle d'une classe : une classe sert à décrire des propriétés d'objets. C'est un nouveau type de données. 2.2. Encapsulation dans une classe : une classe encapsule ses données et fonctions membres, afin de contrôler leur accès depuis l'extérieur. 2.3 Constructeur de classe : un constructeur de classes permet d'initialiser son instance, l'objet à sa création.

Utilisation d'objets. Dans PrOgO, chaque graphique élémentaire 3D est une représentation visuelle d'un objet informatique. Les objets s'obtiennent par instanciation des modèles graphiques fournis à l'interface. Chaque objet possède la même apparence visuelle que sa classe à la création. L'apparence d'un objet est définie par sa position (son emplacement par rapport à un autre objet), sa couleur ou sa rotation. L'apprenant peut ensuite changer l'apparence de l'objet en modifiant ses attributs et en exécutant des appels de méthodes. Le comportement d'un objet est défini par deux fonctionnalités, la possibilité de changer de couleur pendant un temps donné et/ou la possibilité de réaliser une rotation pendant une certaine durée. Enfin, chaque objet peut être assemblé à un autre objet afin de construire une structure plus complexe. L'ensemble de ces opérations sont possibles à la fois dans la scène 3D et dans l'éditeur de code en langage C++. Dès que l'apprenant a terminé sa construction, il est invité à créer une nouvelle classe avec sa réalisation dans une seconde phase.

Création de classes. Toute nouvelle construction dans PrOgO constitue une nouvelle classe que l'apprenant peut créer, nommer et ajouter à sa liste de classes. Cette nouvelle classe peut alors être instanciée et visualisée dans la scène 3D. C'est un nouveau modèle qui regroupe les propriétés choisies par l'apprenant lors de sa réalisation. La création d'une nouvelle classe entraîne l'apparition d'un nouveau programme principal contenant une fonction « `main()` » vide invitant l'apprenant à instancier sa nouvelle

classe. Deux nouveaux fichiers apparaissent également, l'un contient la déclaration de la nouvelle classe « `*.hpp` », et le second contient sa définition « `*.cpp` ».

L'apprenant est amené à manipuler ces fichiers afin de découvrir le concept d'encapsulation. Certaines données et fonctions membres de la classe nouvellement créée sont privées (précédées par le mot clé « `private` ») et d'autres sont publiques (précédées par le mot clé « `public` »). À titre d'exemple, l'apprenant accède depuis la fonction « `main()` » aux fonctions publiques de sa nouvelle classe, mais n'accède pas aux membres privés qui ne sont visibles que depuis le fichier de définition de classe.

La classe nouvellement créée possède également un constructeur qui permet d'initialiser un objet lors de sa création. L'apprenant peut également observer le code du constructeur. À titre d'exemple, il peut remarquer que le constructeur est publique, porte le même nom que sa classe, et n'a pas de type de retour.

L'environnement PrOgO montre l'implémentation de l'approche didactique par emboîtement de concepts OO sur un cas d'utilisation unique (le modèle OO sous-jacent aux constructions 3D dans PrOgO). En pratique, dans un programme d'enseignement en classe, plusieurs principes de conception sont couverts. La section suivante montre comment cette approche peut se transposer en classe pour couvrir un large spectre de concepts OO.

Par ailleurs, l'environnement PrOgO a été utilisé avec des publics de différents niveaux, à savoir, le secondaire et le supérieur. Cela montre le potentiel de transférabilité de cette approche didactique à des publics différents.

5 Extension de l'approche didactique en vue d'un usage en classe

Au sein de IMT Atlantique, dans le programme de formation d'Ingénieurs, l'organisation de l'enseignement introductif de la POO s'effectue également selon l'approche « par emboîtement » en trois étapes. Elle a été mise en œuvre dans le cadre du module « Modèles et Programmation Orientée-Objet » en Tronc Commun. Ce module comprend 40 heures en face à face pédagogique, dont 22 heures de cours respectant cette approche didactique et 18 heures de projets. Les activités de programmation s'effectuent avec le langage Java. Les élèves possèdent des prérequis en programmation procédurale en langage Python. Leur effectif varie entre 40 et 60 élèves. Le Tableau 2 montre les concepts introduits aux élèves dans ce module en les associant à chacune des étapes didactiques : utilisation d'objets, création de classes et conception.

Tableau 2. Concepts introduits dans l'enseignement introductif de la POO au sein de IMT Atlantique.

Étape didactique	Concept
(1) Utilisation d'objets	1.1 Objet : abstraction d'une entité du monde réel participant à l'exécution d'un programme.

	1.2 Attribut, attribut d'instance : variable d'un objet. Sa valeur représente une partie de l'état de l'objet. 1.3 Méthode, méthode d'instance : traitement associé à un objet.
(2) Création de classes	2.1 Classe : construction d'un langage OO qui définit la structure d'une famille d'objets et son comportement. 2.2 Instanciation : action de créer un objet à partir d'une classe. 2.3 Constructeur : méthode d'une classe permettant de créer des objets de la classe. Elle initialise l'état de ces objets. 2.4 Méthode de classe : traitement associé à une classe, qui ne peut pas manipuler des attributs d'instance. 2.5 Encapsulation : principe par lequel l'état d'un objet ne peut être accédé directement que par l'objet lui-même.
(3) Conception	3.1 Association : lien qui dure dans le temps entre deux objets. L'identité d'un objet est mémorisée dans un autre objet. 3.2 Agrégation : association non symétrique, qui exprime une relation de type « ensemble/élément ». Une instance d'élément agrégé peut exister sans agrégat (et inversement). 3.3 Composition : association non symétrique, qui exprime un couplage fort et une relation de subordination. Détruire l'agrégat détruit les éléments agrégés. 3.4 Héritage : concept propre aux langages de programmation pour la réalisation d'une relation de type classe mère/classe fille où les classes filles partagent des membres avec la classe mère. Il s'agit d'une relation de type « est-un ». Etc.

L'utilisation de cette approche didactique en classe montre qu'elle nécessite d'être pensée sur le long terme, selon une dimension temporelle, afin de couvrir l'ensemble du programme pédagogique d'enseignement.

En pratique, un retour de l'étape « conception » vers l'étape « utilisation d'objets » est nécessaire, afin de permettre aux apprenants de mieux visualiser les éléments conceptuels sur le comportement des objets. Les classes étant le moyen de modéliser les objets, l'étape « création de classe » intervient de nouveau dans les activités. Cela dit, pour chaque nouveau concept OO, les trois étapes didactiques s'enchainent successivement. Ainsi, au sein d'IMT Atlantique, l'exemple de la problématique de gestion de stocks en magasin est pris comme cas d'utilisation pour introduire successivement des concepts OO comme « l'association », « la composition », « l'agréation » « l'héritage », etc.

Les trois étapes didactiques sont renouvelées de manière itérative et continue (en spirale) pour couvrir un grand nombre d'aspects de conception enseigné au travers d'un cas d'utilisation (*Fig. 4*).

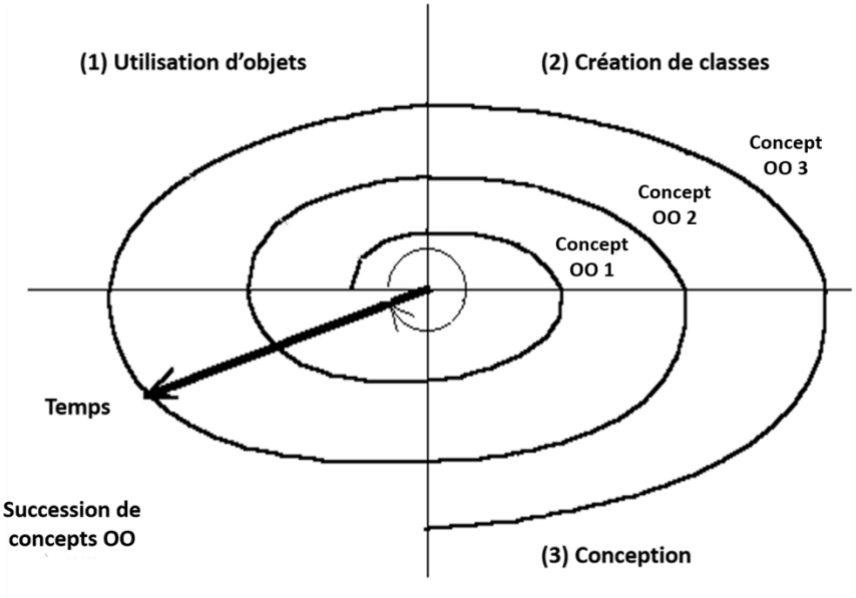


Fig. 4. Vision itérative en spirale de l'approche didactique pour l'enseignement de la POO en classe.

6 Conclusion

Ce papier propose d'étudier la question de l'introduction de la POO dans les programmes d'enseignements aux débutants, et cherche à établir une ingénierie didactique autour de l'enseignement des concepts OO en classe.

Pour ce faire, nous nous sommes intéressés à une approche très répandue dans la communauté anglo-saxonne, se focalisant sur l'enseignement du paradigme OO dès le début des enseignements, l'approche « Objets en Premier ». Cette approche nous a permis de ressortir trois étapes didactiques importantes dans le processus d'introduction des concepts OO aux débutants, associées à trois niveaux de concepts qui s'introduisent par emboîtement. Cette nouvelle approche didactique, ne se focalise pas sur le type de paradigme à introduire en premier comme dans l'approche « objet en premier », mais sur l'emboitement des trois niveaux de concepts OO, comme élément d'ingénierie didactique.

Cette approche, nous a servi dans la conception du micromonde de programmation PrOgO. Cela démontre la faisabilité de son implémentation dans un environnement numérique et ludique d'apprentissage. Par ailleurs, nous avons vérifié la transposition de cette approche en classe, en se basant sur le programme d'enseignement dispensé à l'école d'Ingénieurs IMT Atlantique. Nous représentons cette approche selon une dimension temporelle, afin de couvrir l'ensemble d'un programme d'enseignement en classe sur le long terme. Il en résulte une vision itérative en spirale, montrant l'application des trois étapes constitutives de cette approche à chaque itération, visant

à introduire de nouvelles notions de conception OO au travers d'un cas d'utilisation réel.

Enfin, cette approche contribue aux questions de didactique de l'informatique induites par la généralisation des compétences informatiques à l'école, comprenant l'initiation à la programmation et le paradigme OO. Cette approche, pose en effet un cadre pouvant servir à chaque enseignant s'interrogeant sur la façon d'introduire la POO en classe.

Références

1. M. Kölling et J. Rosenberg: Guidelines for teaching object orientation with Java.*ACM SIGCSE Bulletin,* vol. 33, n13, pp. 33-36 (2001).
2. J. Bennedsen et C. Schulte: What does Objects-First mean ? : An international study of teachers' perceptions of Objects-First. *Seventh Baltic Sea Conference on Computing Education Research* (2007).
3. J. Bennedsen : Teaching and learning introductory programming : a model based approach. Thèse de doctorat. University of Oslo. Faculty of Mathematics & Natural Sciences (2008).
4. A. Robins, J. Rountree et N. Rountree: Learning and teaching programming. A review and discussion. *Computer science education,* vol. 13, n° 12, pp. 137-172 (2003).
5. A. Mcgettrick, R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove et K. Mander: Grand challenges in computing: Education—a summary. *Computer Journal,* vol. 48, n° 11, pp. 42-48 (2005).
6. G. Engel et E. Roberts : Computing curricula 2001. Computer science, The Joint Task Force on Computing Curricula (2001).
7. F. Djelil : Conception et évaluation d'un micromonde de Programmation Orientée-Objet fondé sur un jeu de construction et d'animation 3D. Thèse de Doctorat. Université Blaise Pascal - Clermont II (2016).
8. P. Woodworth et W. Dann: Integrating console and Event-Driven models in CS1. *ACM SIGCSE Bulletin,* vol. 31, n°11, pp. 132-135 (1999).
9. D. Buck et D. J. Stucki: Design early considered harmful : Graduated exposure to complexity and structure based on levels of cognitive development. *ACM SIGCSE Bulletin,* pp. 75-79 (2000).
10. M. Kölling: Introduction to programming with Greenfoot, Pearson Education: Upper Saddle River, New Jersey, USA (2010).
11. B. W. Becker: Teaching CS1 with karel the robot in Java. *ACM SIGCSE Bulletin,* vol. 33, n°11, pp. 50-54 (2001).
12. S. Xinogalos, M. Satratzemi et V. Dagdilelis: An introduction to Object-Oriented Programming with a didactic microworld : objectKarel. *Computers & Education,* vol. 47, n°12, pp. 148-171 (2006).
13. J. Adams et F. Jeremy, «Object centered design for Java : teaching OOD in CS-1,» *ACM SIGCSE Bulletin,* vol. 35, n°11, pp. 273-277 (2003).
14. S. Cooper, W. Dann et R. Pausch: Alice : a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges,* vol. 15, n°15, pp. 107-116 (2000).

Promouvoir et soutenir la Pédagogie Par Projet Centré Humain dans le supérieur : le projet APACHES

Alexis Lebis[1 [0000−0003−2104−8671]], Estelle Prior[1,2 [0000−0002−2895−2690]], Nadine Mandran[2], Abir Karami[3], and Mathieu Vermeulen[1 [0000−0003−3646−1741]]

[1] IMT Lille Douai, Université de Lille, Lille, France
{alexis.lebis|estelle.prior|mathieu.vermeulen}@imt-lille-douai.fr
[2] Université Grenoble ALPES, Grenoble, France
nadine.mandran@univ-grenoble-alpes.fr
[3] FGES-Université Catholique de Lille, Lille, France
abir.karami@univ-catholille.fr

Résumé La pédagogie par projet a su montrer son efficacité et l'on constate un accroissement de sa popularité dans l'enseignement supérieur. Cette pratique met l'accent sur les compétences interdisciplinaires relatives à la gestion de projet que les étudiants sont amenés à mobiliser durant tout leur cursus. Nous nous intéressons plus particulièrement à la gestion de projet en informatique centrée humain, qui fait intervenir l'humain dans les processus de conception, d'utilisation et d'évaluation des projets. Dans cet article, nous abordons les questions de l'intégration et de l'évaluation de ces compétences interdisciplinaires chez les étudiants. Premièrement, nous traitons la question de la traçabilité de l'apprentissage de ces compétences dans le contexte de l'informatique centrée humain en adoptant une approche par compétences. Nous argumentons les avantages et les désavantages des approches existantes dans le contexte des méthodes Agiles dans la pédagogies par projet centrée humain. Deuxièmement, nous présentons des outils capables d'accompagner ce type de projet, et l'effet qu'ils ont sur l'enseignement, le cadre écologique des enseignants et la motivation des étudiants. Des entretiens semi-directifs ont été réalisés avec cinq enseignants concernant ces deux questions. Une des conclusions à ces entretiens est la nécessité de pouvoir tracer et analyser la progression des étudiants durant ces cours afin de pouvoir les aider de manière adéquat et de s'assurer de l'acquisition des compétences à la fois disciplinaires dont les enseignants sont experts, et celles interdisciplinaires de gestion de projet. Dans ce but nous proposons le projet *APACHES*, co-construit avec ces différents acteurs.

Keywords: Pédagogie par projet centré humain · Informatique centrée humain · Méthode Agile · Traçabilité · Approche par compétences · Learning Analytics.

1 Contexte

La pédagogie par projet est un type de pédagogie active où, par l'entremise de la résolution de questions complexes et de problèmes, les étudiants deviennent eux-mêmes acteurs de la production et de l'acquisition de leurs connaissances [17]. L'une des problématiques à ce type de pédagogie est la nature symplectique qui se manifeste entre les compétences disciplinaires (*e.g.* mathématiques) et interdisciplinaires (*e.g.* gestion de projet) acquises et à mobiliser par les étudiants[1]. De fait, les compétences interdisciplinaires ne sont pas uniquement transmises par des cours théoriques de gestion de projet : elles s'acquièrent par la mobilisation de connaissances dans des situations adéquates [16]. La difficulté est de pouvoir évaluer ces compétences [11].

Actuellement, bien que l'on puisse remarquer dans les *Learning Analytics* l'émergence de méthodes d'analyses de données [5,20], à notre connaissance l'évaluation de ces compétences interdisciplinaires de projet s'avère être une tâche complexe pour les enseignants. On remarque aussi un manque d'outils adaptés pour aider ces enseignants [7,9]. Par conséquent, il est courant de noter chez les étudiants des lacunes concernant les compétences et les méthodes en lien avec la pédagogie par projet (*e.g.* compétences de gestion Agile). Il est fréquent d'observer de la part des étudiants que l'accent est mis sur les besoins techniques de la gestion de projet (*e.g.* devoir établir un calendrier, définir des rôles), occultant les compétences à mobiliser pour le mener à bien et l'objectif pédagogique sous-jacent. De plus, on constate l'inclination de la pédagogie par projet à faire intervenir l'humain à différents niveaux. À ce titre, le domaine de l'informatique centrée humain se démarque ; son besoin étant d'explorer l'activité humaine pour concevoir des applications, et d'évaluer ces dernières avec des humains [1] [12](*e.g.* concevoir un simulateur de vol pour des pilotes, une plate-forme pour l'apprentissage des mathématiques). En outre, et bien que cela rende plus difficile le travail en mode projet, la collaboration et l'adaptation au contexte [1] y sont considérées comme capitales. L'informatique centrée humain repose sur une approche préconisant des méthodes itératives, qui ont aussi la possibilité d'accompagner l'activité humaine [2]. Actuellement, les étudiants en informatique sont formés aux méthodes Agiles qui préconisent l'intégration de l'humain dans le développement des applications. Mais ils sont rarement formés sur les fondamentaux des méthodes de production des données en sciences humaines et sociales (*e.g.* entretien, *focus-group*, observation) ou encore aux méthodes d'analyses des données (*e.g.* analyse thématique, statistiques) en situation écologique. Par conséquent, la question se pose sur les méthodes et les outils à mettre en place pour intégrer correctement l'humain dans la pédagogie par projet. En effet, centrer l'informatique sur l'humain fait intervenir des interactions fortes entre différents éléments (*e.g.* d'autres acteurs, des systèmes, des interfaces utilisateurs) dans des contextes riches et complexes : cela nécessite, en plus des compétences

1. D'après Tardiff, la notion de compétence se définit comme "un savoir-agir complexe reposant sur la mobilisation et la combinaison efficaces d'une variété de ressources internes et externes à l'intérieur d'une famille de situations" [16].

informatiques, de mobiliser celles des sciences humaines et sociales (SHS) [12]. Il devient donc nécessaire d'accompagner les enseignants dans l'élaboration de tels cours. Cette approche par projet en informatique centrée humain doit se faire sur le long terme. Les contenus disciplinaires informatique et SHS doivent être pensés de manière conjointe et évolutive. Les attentes des étudiants dans leurs premières années de cursus (*i.e.* L1 à L3) ne sont pas les mêmes que celles de doctorants dans ce même domaine. De fait, l'accompagnement des enseignants et des étudiants, ainsi que l'évaluation des compétences doit être graduelle tout au long du cursus.

Ainsi, le projet APACHES (Apprentissage des conduites de Projets Agiles et Centrés Humain dans l'Enseignement Supérieur) que nous présentons dans ce papier tente de répondre à la problématique : *comment assister les différents acteurs (e.g. enseignant, étudiant, décideur) dans leur démarche de pédagogie par projet centré humain en informatique, et transformer le cadre pédagogique de ces acteurs pour le rendre adéquat à ce type d'enseignement ? Comment identifier et évaluer les compétences interdisciplinaires de projets mobilisées et acquises par les étudiants ? Dans ce contexte, comment et pourquoi mobiliser les outils des Learning Analytics ?*

2 Travaux relatifs & définition

La littérature nous apporte des réponses quant à la problématique mentionnée. Nous notons par exemple qu'une approche d'enseignement reposant sur des principes Agiles a été formalisée par ALPES (Approches agiLes Pour l'Enseignement Supérieur) [19]. L'objectif est de fournir aux étudiants des concepts de gestion de projet par l'intermédiaire des approches Agiles qui sont intégrées dans différents cours, tout au long de leur cursus. ALPES s'inspire des notions socio-constructivistes pour définir la transmission de ces concepts, et promeut ainsi la co-construction des compétences et des connaissances plutôt qu'une transmission passive de ces dernières [8]. Ainsi en adéquation avec le manifeste Agile [1], ALPES favorise les interactions humaines [19].Une autre approche, nommée THEDRE pour *Traceable Human Experiment Desing* [12], propose une méthode pour conduire la Recherche en Informatique Centrée Humain (RICH). L'objectif de la RICH est la production de connaissances scientifiques et la construction des outils par et pour l'humain. Ce type de recherche requiert une approche basée sur l'expérimentation afin de générer des données spécifiques au domaine d'étude à analyser. De plus, l'humain est inclus à la fois dans l'élaboration et l'évaluation, et le contexte dans lequel il évolue est pris en compte. La RICH fait aussi intervenir la notion d'étapes incrémentales pour élaborer et évaluer les outils ainsi produits. La méthode THEDRE est un modèle de méthode pour accompagner les doctorants et les chercheurs dans le domaine de la RICH. Cette méthode fournit un ensemble de guides pour faciliter le travail des doctorants et améliorer les échanges entre acteurs de la recherche. Néanmoins, à notre connaissance, il n'existe pas de définition formelle de ce qu'est la Pédagogie Par Projet Centrée Humain (P^3CH), ce qui limite la manière dont nous pouvons

répondre aux différents problèmes soulevés. C'est pourquoi, en s'inspirant des travaux de la littérature [4,15,16,17], nous la définissons comme étant :

> *une démarche pédagogique socio-constructiviste ancrée dans un continuum de compétences à enseigner dans le temps, et dans laquelle l'acquisition de ces compétences par les étudiants s'effectue par la réalisation d'un projet impliquant des utilisateurs dans une situation écologique en accord avec les attentes pédagogiques, capable de les motiver par le questionnement et les problèmes intrinsèques à ce projet. Elle fait aussi intervenir les différents acteurs humains dans les processus de conception, d'utilisation, d'évaluation du projet et de ses livrables – notamment les utilisateurs des livrables.*

Nous présentons dans ce papier les besoins actuels des enseignants que nous avons pu recenser concernant la P^3CH en informatique. Ensuite, nous présentons le projet APACHES et la façon dont ce projet escompte répondre à la problématique de la P^3CH et aux besoins des enseignants. L'objectif de ce projet est de proposer un cadre théorique, un modèle de méthode d'enseignement et des outils afin de former et d'assister les différents acteurs académiques (*e.g.* étudiants de premier cycle, diplômés, doctorants, enseignants, superviseurs) à la P^3CH. Ce projet a aussi pour ambition de proposer une approche par compétences tout au long du cursus universitaire. Basé sur ALPES et THEDRE, APACHES a pour objectif d'aider la transformation pédagogique via l'introduction de la P^3CH et d'assister les différents acteurs avant, pendant et après leurs cours.

3 Etude exploratoire pour co-construire APACHES

Pour dimensionner le projet APACHES, nous avons impliqué des enseignants qui utilisent déjà cette pédagogie par projet dans leur enseignement en informatique et des chercheurs dont la problématique portent sur des méthodes de conception de dispositifs pédagogiques, sur des modèles et outils pour les *Learning Analytics* ou sur des méthodes de conduite de la recherche en informatique centrée humain. Comme préconisé dans le *Design Based-Research* [21], l'implication des chercheurs et des enseignants offre un cadre collaboratif et intégratif pour concevoir un dispositif d'enseignement en contexte réel. Ce travail collaboratif permet de faire émerger des questions de recherche du terrain et de proposer des outils issus de la recherche aux enseignants.

Pour l'instant, nous avons travaillé à trois niveaux. Premièrement, nous avons défini la problématique du projet APACHES, par différents échanges et réunions. Ce travail a fait émerger des questions de recherche sur l'évaluation des compétences dans ce type de pédagogie, sur les méthodes d'enseignements par projets et sur les indicateurs de suivi du travail des étudiants. Ce travail d'explicitation a été nécessaire pour préciser la problématique.

Parallèlement, nous avons choisi d'impliquer les enseignants qui ont déjà mis en place l'approche ALPES. Comme nous avions peu d'informations sur leur expérience, nous avons utilisé une méthode de production de données ancrée

dans une démarche qualitative, l'entretien. Cette démarche permet d'explorer en profondeur un champ non connu et de faire émerger les pratiques et besoins d'utilisateurs dans un domaine précis [6,13]. Ainsi, nous avons interrogé cinq enseignants [2] qui mettent en place depuis au moins quatre ans les principes de l'approche ALPES dans leurs enseignements en informatique *[E1, E2, E3, E4, E5]*, en intelligence artificielle *[E3]* et en gestion de projets *[E5]*. Le panel des participants est composé d'une femme et quatre hommes et comprend le concepteur de l'approche ALPES. Des entretiens semi-directifs ont été menés pour évaluer l'état actuel de l'utilisation de cette pédagogie par projet Agile en informatique mais aussi pour identifier les points à améliorer.

Ensuite, sur la base des premiers résultats issus de l'analyse des entretiens, nous avons animé un *focus-group* avec quatre personnes pour avoir une première version d'une méthode d'enseignement par projet pour l'informatique centrée humain. L'objectif de ce travail était de lister les tâches pour mettre en place cette pédagogie. Les résultats obtenus sont une première étape pour construire notre méthode, nous les avons confronté aux travaux de Raucent et ses collaborateurs [14]. Dans les sections suivantes, nous présentons les résultats des entretiens menés auprès des enseignants.

Le guide d'entretien abordait, d'une part, l'expérience des enseignants avec l'approche ALPES et ses outils et leur façon d'encadrer des thèses. D'autre part, il traitait de la numérisation des outils présents dans ALPES et des outils permettant de faciliter l'encadrement d'une thèse qui, *in fine*, seront à implémenter dans APACHES.

3.1 De l'émergence d'une transformation des pratiques pédagogiques

Les enseignants ont parlé de leur expérience d'enseignement avec l'approche ALPES. Ils ont abordé leur propre changement de pratiques ainsi que ceux remarqués chez leurs étudiants ($N = 4$), l'apport de nouvelles perspectives ($N = 5$) et les impacts de cette démarche Agile ($N = 4$). Pour le premier point, trois d'entre eux notent chez eux et leurs étudiants une amélioration de la motivation, des interactions et des conditions de travail. Concernant le second point, les principaux avis ($N = 3$) portent sur les progrès remarqués chez les

2. Les caractéristiques de ces enseignants sont présentés dans le Tableau 1.

Numéro du participant	Statut	Année d'enseignement	Année d'enseignement avec l'approche ALPES	Thèses encadrées
E1	Ingénieur/enseignant	8	5	0
E2	Ingénieur/enseignant	10	4	0
E3	Maître assistant	10	5	3
E4	Maître assistant HDR	14	5	5
E5	Maître de conférence	11	5	4

Table 1. Tableau présentant les caractéristiques des enseignants interrogés.

étudiants au niveau des notes, de la productivité et de l'autonomie. Un enseignant précise toutefois qu'il n'a pas perçu d'amélioration dans l'apprentissage des notions disciplinaires : *"Je ne vois pas forcément que l'utilisation des outils ALPES aide mieux les étudiants à apprendre le système de gestion de base de données, mais plutôt sur le développement et la gestion de projet informatique."(E2)* Les avis sont plus hétérogènes sur le dernier point : deux enseignants évoquent l'avantage d'une démarche itérative ; un autre celle de la flexibilité et de la personnalisation du cours ; et un dernier met en garde sur la possibilité de passer à côté des notions à apprendre à cause de la concentration que requiert la mise en place de cette approche.

Les enseignants ont donné leur avis sur les outils proposés dans ALPES. Seuls les outils utilisés par l'ensemble des enseignants sont cités : la *timebox* (affichage d'une minuterie), le *task board* (tableau de suivi de progression des tâches), les *user stories*[3] (scénario utilisateur), le *planning board* (tableau de suivi de progression des *user stories* à travers les séances), le *tweetback board* (tableau dans lequel les étudiants donnent leurs avis) et le *dojo* (résolution collective de problème) [19].

3.2 Vers une numérisation et un apport des Learning Analytics

Au niveau de la numérisation des outils, Un enseignant aborde l'importance de celle-ci pour tracer les activités des étudiants et pour avoir à disposition des indicateurs. Il souhaite avoir une possibilité de *gamification* tout en prenant garde au risque de compétitivité. Elle peut aussi être *"un levier de réussite"* pour les étudiants et non comme *"un outil remplaçant l'interaction enseignant-étudiant" (E5)*. Deux enseignants relatent les difficultés d'utilisation et le risque de perte d'information. L'un d'eux déplore aussi la perte de manipulation d'objets tangibles liée à la numérisation. Au sujet des fonctionnalités à intégrer au futur outil, les enseignants ont abordé des des outils dédiés à la gestion de projets ($N = 5$), de la facilitation du travail ($N = 5$), des indicateurs ($N = 3$) et du suivi de l'activité ($N = 3$).

Concernant les outils ALPES, tous les enseignants souhaite une intégration de ceux qu'ils utilisent le plus et l'ajout de nouvelles fonctionnalités. Pour les *boards*, ils désirent intégrer trois fonctionnalités : une qui amène les étudiants à être plus assidus dans leur utilisation, une seconde qui leur permet de modifier les *boards* et les personnaliser, et une troisième pour permettant,gérer les étudiants, les alerter sur leurs besoins ainsi que de synchroniser les modifications souhaitées sur les *boards* étudiant. Deux enseignants souhaitent que les *user stories* soient inscrites dans un tableau, avec des options en fonction de l'utilisateur, comme offrir la possibilité d'une meilleure organisation, apporter davantage de richesses dans les niveaux et pouvoir ajouter des éléments pour ralentir les étudiants les plus avancés. *"Avoir un tableau prof qui représente le projet. Le prof va pouvoir*

3. Il s'agit de la description des besoins et des attentes utilisateurs – décrits de manière indépendante les uns des autres – vis-à-vis de la production finale et conditionnant ses fonctionnalités [3].

modifier les user stories et ça se répercutera dans les tableaux des étudiants. Sachant que les étudiants ont leur propre tableau connecté à celui du prof et qu'ils peuvent faire des modifications propres à eux. Donc, il y a une forme de suivi de versions des tableaux." (E3) Concernant le *dojo*, un enseignant propose de les faire sous forme de vidéo pour les mutualiser à l'ensemble des utilisateurs de l'outil, de manière à diminuer les répétitions et les arrêts des étudiants dans leur travail. Pour la *timebox*, les enseignants ($N = 5$) soulignent la pertinence d'avoir des options favorisant l'adaptation à leurs étudiants et au contexte. *"Les étudiants sont tellement concentrés sur l'affichage de la timebox qu'ils oublient de travailler (...), à l'IUT je n'affiche pas la timebox, j'utilise une sonnerie (...). La fin d'une timebox peut être surprenante, donc ça nécessite de la souplesse pour finir la mission en cours." (E5)* Les enseignants ($N = 4$) évoquent aussi de nouvelles fonctionnalités, comme une aide à la construction du cours et une liste de vérification des notions disciplinaires à connaître qui pourra guider les *dojo* ; l'aide à l'évaluation Agile et à la validation continue, sans en faire un outil d'évaluation ; une aide à l'autonomie dans la recherche d'informations ; et enfin un système de conseils et d'assistances pour les étudiants – en veillant à ne pas remplacer le travail de l'enseignant : *"Le système de conseils aux élèves ne doit pas remplacer le fait qu'ils retournent vers l'enseignant pour des questions." (E4)*

Pour la facilitation du travail du point de vue étudiant, les enseignants proposent une aide à la communication entre eux et une accessibilité au cours ($N = 2$). Les enseignants expriment l'intérêt d'avoir une aide pour exporter et importer facilement les cours ; pour retrouver les tableaux des groupes et reconnaître rapidement les étudiants ; pour identifier, définir et valider les notions à acquérir ; et pour valider des travaux à distance. Ces points permettront d'améliorer l'autonomie de l'étudiant pour qu'il puisse avoir la maîtrise de son projet. Ils fourniront aussi à l'enseignant une fluidification de son travail, des interactions avec les étudiants et réduiront sa concentration sur la méthode et le risque d'oublis des notions de cours à aborder. *"Ca permettrait d'éviter de se perdre complètement dans la méthode et oublier les notions de cours. C'est une complémentarité pour valider les notions de cours apprises." (E3)*

Au niveau des *Learning Analytics*, les enseignants proposent différents indicateurs pour suivre les étudiants. Nous pouvons citer : 1) la "vélocité" pour rendre compte de la vitesse de réalisation du travail ; 2) "le temps passé à réaliser les tâches et répondre aux *user stories* ainsi que l'avancement du projet" , d'être informé en cas de retard mais aussi d'améliorer la gestion de leur travail ; 3) "le nombre de questions posées par groupe" afin d'apporter une aide dans l'évaluation de leur participation. *"un retour sur l'avancée des projets et la compréhension, avoir des stats et des indicateurs qui permettent de voir où en sont les étudiants, des indicateurs aussi sur le pourcentage d'avancement, le nombre de questions posées pour un groupe, (...), le temps passé sur une user-story pour mieux organiser les découpages (...), des indicateurs pour donner une note de participation (...), un retour aux étudiants sur leurs avancements par rapport aux autres." (E1) ; "Ça serait peut être intéressant de voir si un groupe a passé un certain*

temps sur une tâche en particulier. Ça veut dire qu'il faut préciser un temps maximum à ne pas dépasser pour réaliser la tâche." (E2).

Enfin, ils souhaitent recueillir des traces sur les versions et modifications des *boards*, la réalisation des actions effectuées à chaque étape du projet et les connaissances acquises. L'objectif principal est de pouvoir faire un retour aux étudiants sur les notions disciplinaires à connaître et de valoriser, partager et communiquer sur les différentes étapes du projet. Un des enseignants exprime l'importance de pouvoir accéder aux traces. *"Il ne faut pas enfermer les données dans l'outil, pour ceux qui veulent accéder à plus, il faut qu'ils puissent." (E4).*

3.3 Le doctorat en informatique centrée humain : un besoin de méthode et d'outils

Au niveau du suivi des thèses, trois enseignants se sont exprimés sur les outils qu'ils utilisent dans le suivi de leurs doctorants et sur les fonctionnalités dont ils souhaiteraient disposer pour ce suivi. Pour les outils utilisés, ils citent des outils d'organisation qui favorisent la visibilité sur trois ans, l'utilisation d'un retroplanning avec des objectifs définis ou la mise en place d'un *task board* et d'un *planning board*. Ils évoquent des moments de bilan comme les réunions périodiques, l'encouragement à les solliciter en cas de besoin, l'envoi d'un "résumé hebdomadaire de leur semaine", sont autant d'outils qu'ils instaurent pour favoriser une régularité dans leur suivi. Ils mettent aussi à disposition des outils pour communiquer et faciliter le partage de documents. L'un d'eux proposent à ses doctorants des outils d'aide à la réflexion et à la structuration. *"une méthode de conduite de la recherche avec une visibilité sur 3 ans, un rétroplanning, (...). On fait un point régulier toutes les une ou deux semaines ou autre selon le besoin" (E5).* Concernant les fonctionnalités demandées par les enseignants, deux d'entre eux ont répondu. Ils accentuent leurs discours sur la nécessité d'améliorer l'accompagnement du doctorant en lui apportant un cadre et d'être alerté en cas de retard sur les étapes de sa thèse. L'un ajoute le besoin de méthodes et fonctionnalités adaptatives. Le second évoque la nécessité d'outils de communication entre doctorants. *"C'est délicat pour l'investissement dans l'encadrement d'une thèse. Un outil numérique (...) pour le doctorant, ça peut lui donner un cadre, un prototype de thèse, d'état de l'art et un cadre temporel pour aider à ne pas prendre du retard. Des alertes à l'équipe encadrant s'il y a du retard sur certaines étapes (...). Ça peut aider s'il y a des outils, comme des sortes de réseau social de doctorants." (E3)*

4 Discussion des résultats, définition et intégration de APACHES dans l'enseignement supérieur

4.1 Discussion des résultats

Au niveau du retour d'expérience de l'approche ALPES, les enseignants ont remarqué une amélioration de leurs pratiques de travail mais aussi de celles des

étudiants ; avec entre autres une progression de l'autonomie chez ces derniers ; et des avantages rencontrés au niveau de l'itération, de la flexibilité et de la personnalisation possible de leurs enseignements. Concernant la numérisation des outils, les participants ont évoqué un besoin de traces et d'indicateurs, grâce à la facilitation du travail, l'intégration des outils ALPES utilisés et le suivi possible des modifications effectuées par les étudiants. Ils ont cependant mis en garde à propos des éventuelles pertes d'information, d'absence d'objets tangibles et de difficultés d'utilisation. Enfin, les enseignants se sont exprimés sur l'utilisation d'outils d'organisation, de bilan et de communication pour encadrer leur doctorant. Ils ajoutent un besoin d'améliorer cet accompagnement, d'avoir des fonctionnalités adaptatives, qui permettent de faciliter la visibilité des doctorants et la communication entre pairs.

Utiliser la pédagogie par projet centrée humain pour l'informatique semble présenter de réels bénéfices pour les différents acteurs pédagogiques. Néanmoins, il apparaît nécessaire de disposer d'une méthode d'enseignement adaptable pour les différents enseignants. Pour soutenir cette pratique, un modèle de méthode d'enseignement doit être construit pour que chaque enseignant puisse s'approprier les différentes étapes incontournables de la méthode et l'adapter à son contexte de travail. Il est nécessaire de soutenir cette méthode par des outils informatiques spécifiques et par l'analyse des traces d'activités. Cette numérisation aura pour objectif d'accompagner l'enseignant dans la mise en place de ce type d'enseignement, d'offrir aux étudiants la possibilité de s'auto-organiser, de fournir de nouveaux outils spécifiques à la gestion de projets et de proposer des tableaux de bords pour suivre l'activité des étudiants. La présence d'une plate-forme support modulable permettra une meilleure inclusion et de meilleures interactions entre les différents acteurs lors de la réalisation du projet. Ces résultats exploratoires nous offre un point de départ pour formaliser le projet APACHES et commencer la réflexion sur la méthode d'enseignement, les outils et les supports numériques. APACHES vise à offrir un modèle de méthodes d'enseignement, des guides pour mettre en oeuvre cette méthode, des outils de suivi et d'évaluation des compétences et une plate-forme numérique support. Ce modèle s'adresse à des projets en informatique centrée humain, itératifs, incrémentaux et adaptatifs : ce sont là les principes que nous désirons recommander et transmettre [12]. Pour réaliser cela, nous nous inspirons des travaux réalisés dans l'approche ALPES, des approches Agiles, de la méthode THEDRE et de ceux utilisés en informatique centrée humain tel le *Design-Based Research* [21].

4.2 Perspectives

Il en résulte alors la définition d'un ensemble de modèles et de théories propres à la P^3CH afin de couvrir formellement ces différents aspects. Nous observons par exemple la nécessité de définir un modèle de traces propre à la P^3CH pour tracer l'activité des étudiants et servir de socle à des analyses pertinentes pour ce type de pédagogie. Nous notons le besoin de proposer une modélisation des compétences interdisciplinaires qui peuvent survenir au sein de cette pratique,

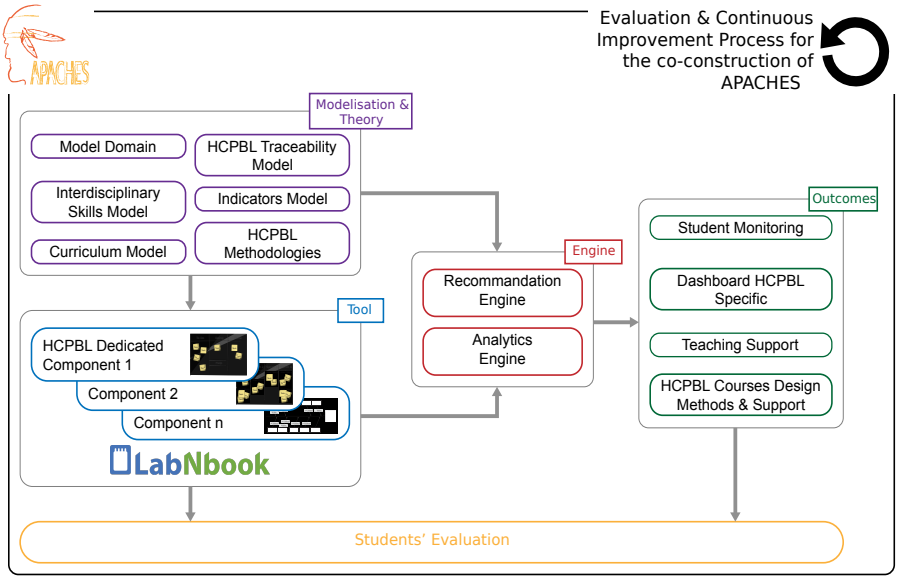


Figure 1. Diagramme des différents aspects du projet APACHES et de leur organisation. HCPBL est l'acronyme anglais de la P^3CH : *Human-Centered Project-Based Learning*. Les éléments pointés par les flèches utilisent les éléments les pointant.

par exemple, pour être en mesure d'évaluer le profil d'un apprenant et ensuite permettre une recommandation personnalisée concernant les cursus les plus adaptés pour ces objectifs. Tous les éléments identifiés comme formant le cadre théorique et formelle de la P^3CH sont présentés en haut à gauche de la Figure 1, dans l'encart *Modelisation & Theory*. Il en découle aussi la définition d'outils et de composants activables ancrés dans ce type de pédagogie (cf. l'encart *Tool* dans la Figure 1) et réifiant les modèles sus-mentionnés. Ces outils, instanciés dans une plate-forme support, seront utilisés par les différents acteurs lors de la P^3CH et fourniront les informations nécessaires pour les différentes analyses réalisables (*e.g.* comportementale, phénoménologique) et l'assistance. L'encart *Engine* de la Figure 1 illustrent ces deux éléments.

Tous ces éléments vont nous permettre d'obtenir des contributions importantes (cf. éléments en vert dans l'encart *Outcomes* de la Figure 1). Il devient possible d'assister et d'établir des recommandations pour les enseignants concernant l'élaboration de projets - afin par exemple de s'assurer que les compétences interdisciplinaires demandées soient conformes à celles mobilisables par les étudiants. Grâce à la modélisation d'indicateurs propres à la P^3CH, à la modélisation des compétences interdisciplinaires et à la possibilité de définir des profils d'apprenants, il devient alors possible de proposer des tableaux de bords dynamiques et en temps réels. Ils seront à la fois à destination des enseignants pour leur permettre de monitorer la progression des étudiants, et aussi à destination des étudiants

pour d'une part les assister durant leurs cours, et d'autre part pour les informer de l'état de leurs compétences liées à la P^3CH en informatique et les aider dans leur choix de cursus universitaire. Ces avancés nous permettent d'envisager de donner aux enseignants les outils adéquats pour concevoir des cours de P^3CH adaptés aux compétences des étudiants qu'ils rencontrent. De plus, ils pourront aussi avoir accès à un ensemble de méthodes et de bonnes pratiques à déployer dans ce type de pédagogie. De cette manière, l'acquisition des compétences – à la fois disciplinaire et interdisciplinaire – s'en trouvera renforcée, en accord avec les principes de la pédagogies par projet, et les interactions entre les différents acteurs humains pourront être mieux identifiées, et renforcées. L'ensemble des éléments présentés dans la Figure 1 ont été co-construits avec les différents acteurs pédagogiques et évalués depuis le début du projet en octobre 2018 - et continueront de l'être tout au long du projet. Les expérimentations concernant la première itération des outils et des modèles APACHES sont planifiées pour le premier trimestre 2020 et seront réalisées par des enseignants du supérieur volontaires. En plus de cela, nous prévoyons de choisir un panel d'étudiants, représentatif, afin d'identifier leurs besoins et leurs propositions, toujours dans l'objectif de mieux faire concorder APACHES aux attentes utilisateurs.

Une deuxième phase d'expérimentation aura pour but d'étudier les modalités de transmission des pratiques APACHES aux différents acteurs pédagogiques, et la façon dont les outils proposés peuvent les aider à transformer leurs cours et faire évoluer leurs méthodologies. Au fil de ces expérimentations, nous mèneront régulièrement des entretiens avec ces enseignants pour mieux comprendre l'appropriation qu'ils se font des outils et la manière de diffuser ces nouvelles méthodes. Aussi, nous tracerons de manière intensive les traces d'activités des apprenants dans les outils et les composants mis à leur disposition. Par l'entremise des techniques de *Learning Analytics*, ces données nous serviront dans l'élaboration des indicateurs pour monitorer les étudiants [10,18]. Ces derniers seront d'ailleurs définis en accord avec une approche centrée utilisateur, comme le prescrit THEDRE : nous impliquerons les différents acteurs pédagogiques dans l'exploration de leur besoin, dans la co-construction de ces indicateurs ainsi que dans leur évaluation.

Pour conclure, APACHES implique actuellement plus de 2000 étudiants, répartis dans trois établissement français, plus de 15 doctorants et 15 enseignants ; et nous prévoyons d'y inclure encore d'autres acteurs. Ce projet, prévu également sur plusieurs années, va nous permettre de transformer foncièrement les pratiques d'enseignement dans l'informatique centrée humain pour proposer à ces différents acteurs une pédagogie capable d'exploiter les outils numériques du 21^e siècle qui répondent au mieux à leur besoin.

Références

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. : Manifesto for agile software development (2001)
2. Cockburn, A., Highsmith, J. : Agile software development : The people factor. Computer (11), 131–133 (2001)

3. Cohn, M. : User stories applied : For agile software development. Addison-Wesley Professional (2004)
4. Condliffe, B., Visher, M.G., Bangser, M.R., Drohojowska, S., Saco, L. : Project-based learning : A literature review. New York, NY : MDRC (2016)
5. Cooper, A. : Learning analytics interoperability-a survey of current literature and candidate standards (2013)
6. Creswell, J. : Research Design : Qualitative, Quantitative, and Mixed Methods Approaches. SAGE Publications (2013)
7. Espinoza, A., Garbajosa, J. : A study to support agile methods more effectively through traceability. Innovations in Systems and Software Engr. **7**(1), 53–69 (2011)
8. Jonnaert, P. : Compétences et socioconstructivisme : un cadre théorique. Armando Editore (2009)
9. Kingston, S. : Project based learning & student achievement : What does the research tell us ? pbl evidence matters, volume 1, no. 1. Buck Institute for Education (2018)
10. Lebis, A., Lefevre, M., Luengo, V., Guin, N. : Capitalisation of Analysis Processes : Enabling Reproducibility, Openess and Adaptability thanks to Narration. In : LAK '18 - 8th International Conference on Learning Analytics and Knowledge. pp. 245–254. ACM, Sydney, Australia (Mar 2018)
11. Mandin, S., Guin, N. : Évaluation de savoir-faire au sein d'un environnement éducatif fondé sur un référentiel. In : Atelier Évaluation des Apprentissages et Environnements Informatiques, EIAH 2015. Agadir, Morocco (Jun 2015)
12. Mandran, N., Dupuy-Chessa, S. : THEDRE : A Traceable Process for High Quality in Human Centred Computer Science Research. In : 26th International Conference on Information Systems Development, ISD 2017. Larnaca, Cyprus (Sep 2017)
13. Mandran, N., Dupuy-Chessa, S. : Supporting experimental methods in Information System research. In : 12th IEEE International Conference on Research Challenges in Information Science RCIS'2018. Nantes, France (May 2018)
14. Raucent, B., Milgrom, E., Bourret, B., Hernandez, A., Romano, C., et al. : Guide pratique pour une pédagogie active : les APP. . . , Apprentissages par Problèmes et par Projet. Toulouse : Institut national des sciences appliquées (2010)
15. Simon, H.A. : The Sciences of the Artificial. MIT Press, Cambridge, MA, 3 edn. (1996)
16. Tardif, J., Fortier, G., Préfontaine, C. : L'évaluation des compétences : documenter le parcours de développement. Chenelière-éducation, Montréal (2006)
17. Thomas, J.W. : A review of research on project-based learning (2000)
18. Verbert, K., Govaerts, S., Duval, E., Santos, J.L., Assche, F., Parra, G., Klerkx, J. : Learning dashboards : an overview and future research opportunities. Personal and Ubiquitous Computing **18**(6), 1499–1514 (2014)
19. Vermeulen, M., Laval, J., Serpaggi, X., Pinot, R. : Soyez agiles dans les A.L.P.E.S. ! Une pedagogie en mode agile. In : 9ème Colloque Questions de Pédagogie dans l'Enseignement Supérieur (QPES 2017). Grenoble, France (Jun 2017)
20. Vie, J.J., Popineau, F., Bruillard, É., Bourda, Y. : A review of recent advances in adaptive assessment. In : Learning analytics : fundaments, applications, and trends, pp. 113–142. Springer (2017)
21. Wang, F., Hannafin, M.J. : Design-based research and technology-enhanced learning environments. Educational technology research and development **53**(4), 5–23 (2005)

Mise en œuvre d'approches pédagogiques fondées sur des pratiques de l'industrie du logiciel pour l'apprentissage de la programmation

Jean-Baptiste Raclet[1,2], Franck Silvestre[1,3], and Mika Pons[2]

[1] IRIT, Institut de Recherche en Informatique de Toulouse, France.
nom.prenom@irit.fr
[2] Université Paul Sabatier – Toulouse III, France.
[3] Université Toulouse Capitole – Toulouse I, France.

Résumé Cet article présente un nouveau protocole d'apprentissage de la programmation dans lequel les séquences de travail sont structurées en cycles : d'abord l'étudiant développe individuellement une partie d'un logiciel spécifiée par un ensemble exhaustif de tests automatisés fournis par l'enseignant. Ensuite, afin de favoriser les interactions entre pairs, une ou plusieurs phases de revue de code sont introduites. Ces phases de revue sont orchestrées avec la plate-forme elaastic.

L'orchestration du protocole est facilitée grâce à l'outil Git4School qui extrait des métriques d'apprentissage depuis les dépôts git des étudiants et les synthétise sous forme de graphiques permettant un suivi individualisé des étudiants.

Keywords: Ingénierie logiciel, conception guidée par les tests, revue de code, apprentissage de la programmation, évaluation par les pairs

1 Introduction

La formation de développeurs logiciel est un enjeu majeur comme l'indique la tension du marché de l'emploi. En 2017, La Direction de l'Animation de la Recherche, des Études et des Statistiques (DARES) rattachée au ministère français du travail rapporte un indicateur de tension de 30% concernant les ingénieurs en informatique [Bergeat, 2017], catégorie regroupant essentiellement *« les analystes programmeurs, chefs de projet et ingénieurs d'études ou de développement »* [Colin et al., 2015]. Dans le même temps, cet indicateur est de 4% pour l'ensemble du marché du travail.

En conséquence, une employabilité immédiate des jeunes diplômés est un défi qui consiste non seulement à former des étudiants à une discipline en terme de compétences techniques et méthodologiques mais aussi à transmettre les pratiques courantes de l'industrie du logiciel. Cela s'avère aussi essentiel pour la mise en œuvre de prestations de formation tout au long de la vie (FTLV) pour satisfaire les objectifs de mises-à-jour des connaissances ou reconversion professionnelle des participants.

208 Le SWEBOK [Bourque and Fairley, 2014], *Guide to the Software Engineering Body of Knowledge*, est un référentiel établi par l'IEEE qui structure les connaissances relatives au développement logiciel et propose une synthèse des pratiques, techniques et normes généralement acceptées dans l'ingénierie du logiciel. Il fait ainsi un inventaire méthodologique qui permet d'orienter le développeur. Dans le même esprit, le curriculum de l'ACM/IEEE en ingénierie logiciel [Sahami et al., 2013] est un ensemble de recommandations, établies par des professionnels conjointement avec des représentants du monde académique et régulièrement mis à jour, pour l'établissement de cursus d'apprentissage. Il identifie des corpus de connaissances, des dépendances entre eux et définit un degré d'importance pour chacun.

L'approche la plus courante dans l'enseignement supérieur est de considérer que la professionnalisation des pratiques s'effectue lors de projets longs, qui sont l'occasion d'expérimenter le travail en groupe, ou alors au cours d'immersion en entreprise lors de stages. Dans cet article, nous arguons que celles-ci peuvent être enseignées plus tôt en adaptant les pratiques pédagogiques utilisées. Pour cela, nous proposons en section 2 un nouveau protocole fondé sur le test et la confrontation de points de vue pour l'apprentissage de la programmation. Ce protocole inclut des outils et méthodes reconnues par les professionnels et mentionnées dans le SWEBOK ; il permet aussi de mieux couvrir les objectifs d'apprentissage définis dans le curriculum de l'ACM/IEEE. Ainsi, il autorise la mise en œuvre de nouveaux scénarios pédagogiques permettant aux élèves de consolider les connaissances et de se former aux pratiques industrielles de l'ingénierie du logiciel.

Dans le protocole proposé, le travail de l'étudiant comporte différentes tâches décomposées en plusieurs phases décrites ultérieurement dans cet article. Pour faciliter son orchestration, nous avons développé l'outil Git4School, présenté en section 3. Cet outil à destination des enseignants constitue un tableau de bord exposant différentes métriques d'apprentissage extraites des dépôts *git* contenant le travail individuel des étudiants permettant ainsi un suivi individualisé des apprenants.

Le protocole exposé ici est expérimenté depuis 2 ans dans le cadre d'enseignements introductifs et avancés dans le supérieur (en Licence professionnelles, en première et seconde année d'un Master spécialisé dans l'ingénierie logiciel) portant sur l'utilisation du *framework* Java de persistance des données JPA. Un retour sur ces expérimentations ainsi qu'un positionnement par rapport à des travaux connexes est effectué en section 4. Enfin, la conclusion présente les idées clés qui guideront nos travaux futurs.

2 Un protocole fondé sur le test et la revue par les pairs pour l'apprentissage de la programmation

Cette section décrit un nouveau protocole de travail à partir duquel des scénarios pédagogiques peuvent être établis. Le processus est gouverné par les principes suivants :

— Chaque étudiant travaille individuellement à développer une partie d'un logiciel dans lequel les fonctionnalités principales peuvent être réalisées avec des éléments de programmation ciblés.
— Pour chaque exercice, des consignes sont données de manière traditionnelle, avec une description en langage naturel.
— Les exercices demandant d'écrire du code viennent toujours avec un ensemble de tests automatisés qui doivent s'exécuter avec un verdict de succès pour considérer l'exercice comme effectué correctement.
— Lorsque l'étudiant a produit sa solution à un exercice et que tous les tests correspondants s'exécutent avec un verdict de succès, l'étudiant effectue un *commit* dans un dépôt *git* privé qui lui a été attribué.
— Chaque séquence de travail est structurée en une succession de cycles durant lesquels l'étudiant travaille d'abord individuellement, en autonomie (phase de travail guidé par les tests) puis s'en suivent des activités dans lesquelles les étudiants sont invités à partager leur solution, évaluer d'autres solutions et éventuellement ensuite adapter leur propre solution (phase de revue par les pairs).

Nous présentons maintenant en détails les phases mentionnées.

2.1 Phases de travail en solo guidée par les tests (TDD)

Dans l'ingénierie du logiciel, les tests représentent un dispositif incontournable pour vérifier et valider un système logiciel [Bourque and Fairley, 2014]. Le développement dirigé par les tests (TDD pour *Test Driven Development*) est une pratique d'ingénierie, formalisée initialement dans le cadre de la méthode XP, qui consiste à développer les fonctionnalités d'un logiciel dans une succession de cycles commençant par l'écriture d'un test, suivi de l'écriture du code faisant passer le test et terminant par l'amélioration du code principal (*refactoring*). Cette approche redéfinit complètement la place des tests en ingénierie logicielle : le TDD est une méthode d'analyse, de conception et de développement reposant sur l'écriture de tests automatisés tout au long du cycle de développement [Beck, 2003] et non plus uniquement à des fins de validation.

Le protocole que nous définissons requiert que chaque exercice soit équipé d'un ensemble de tests unitaires et de tests d'intégration. L'étudiant suit alors la philosophie du TDD et travaille par itérations successives sur son code jusqu'à ce que les tests s'exécutent avec un verdict de succès.

La validation des compétences et l'obtention de crédits ECTS étant individuels, les évaluations sommatives le sont aussi. Afin de conserver un alignement pédagogique tel que décrit dans [Biggs, 1996], il convient donc de proposer aux étudiant des activités individuelles. Les phases TDD durant lesquelles les étudiants travaillent seuls remplissent cette part du contrat pédagogique.

Le tableau 1 liste à travers l'étude d'un exemple les principales caractéristiques des tests fournis aux étudiants et leur traduction en terme de spécification sur ce que doit faire l'étudiant. On remarque que les tests peuvent être utilisés pour spécifier ce que le code principal doit faire (caractéristique 2) mais aussi comment il doit le faire (caractéristiques 1 et 3).

Id.	Caractéristique	Exemple

1	Un test ne compile que si les types de données manipulés dans le test sont créés correctement par l'étudiant dans le programme principal (idem pour les méthodes).	`// given: an enterprise with all`
		`// properties correctly set`
		`enterprise = new Enterprise();`
		`enterprise.setName("Company & Co");`
		`enterprise.setDescription("Comp desc");`

2	Les tests dits "fonctionnels" (qui testent le comportement attendu en mode boite noire) ne passent que si l'étudiant a codé les algorithmes qui permettent d'atteindre les états des objets spécifiés par les tests.	`// when we switch between, enterprise`
		`// 1 and 2 on project 1`
		`project.switchEnterprise(enterprise2);`
		`// and we save the project`
		`Project savedProject =`
		`enterpriseProjectService.save(project);`
		`// then the saved project is attached`
		`// to enterprise 2`
		`assertThat(savedProject.getEnterprise(),`
		`is(enterprise2));`
		`// and enterprise 1 has only one`
		`// associated project`
		`assertThat(enterprise1.getProjects().`
		`size(), is(1));`

3	Les tests dits "système" ou d'interactions permettent de forcer certains aspects de la solution : utilisation d'un composant précis ; collaboration entre objets, ...	`// when: trying to find the project by id`
		`enterpriseProjectService.`
		`findProjectById(anId);`
		`// then: the find operation is triggered`
		`// on the entity manager`
		`verify(entityManager).`
		`find(Project.class, anId);`

4	Les exécutions successives d'un test génèrent une succession de cycles essais-erreurs où les résultats des tests fournissent un feedback	La figure 1 présente un exemple de feedback obtenu suite à l'exécution d'un test en échec.

Table 1. Caractéristiques des tests

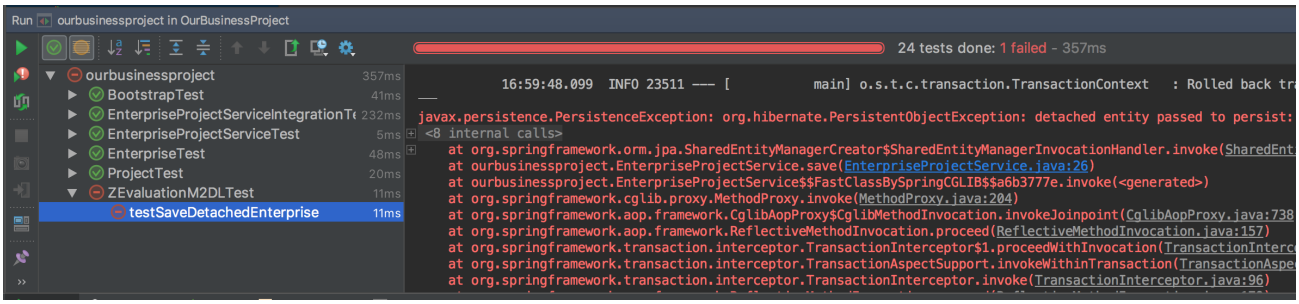


Figure 1. Un feedback obtenu suite à l'exécution d'un ensemble de tests

Un grand nombre d'objectifs d'apprentissage spécifiques à la programmation présentés dans [Sahami et al., 2013] sont couverts par les phases guidées par les tests ; le tableau 2 en mentionne quelques-uns. Néanmoins certains objectifs

d'apprentissage requièrent des interactions sociales comme cet attendu décrit dans [Sahami et al., 2013] : « Discuter des moyens de résoudre un problème avec différents algorithmes ayant chacun leurs propres propriétés ». Afin d'atteindre ces objectifs, le protocole inclut à la suite des phases TDD des phases de revue par les pairs que nous introduisons dans la partie suivante.

Id.	Objectif d'apprentissage	Phase
1	Analyser le comportement de programmes simples impliquant les éléments de base suivants : variables,expressions, affectations, E / S, structures de contrôle, fonctions, passage de paramètres et récursivité.	TDD
2	Concevoir, implémenter, tester et déboguer un programme qui utilise chacune des structures de programmation suivantes : calcul de base, E / S simples, structures conditionnelles et itératives, définition de fonctions et passage de paramètres.	
3	Écrire un programme utilisant des E / S de fichiers pour assurer la persistance entre plusieurs exécutions.	
4	Écrire des programmes qui utilisent chacune des structures de données suivantes : tableaux, enregistrements structures, chaînes, listes chaînées, piles, files d'attente, ensembles et cartes.	
5	Construire, exécuter et déboguer des programmes en utilisant les environnements de développement intégrés modernes	
6	Construire et déboguer des programmes en utilisant les bibliothèques standardisées disponibles pour un langage de programmation donné	
7	Discuter des moyens de résoudre un problème avec différents algorithmes ayant chacun leurs propres propriétés.	PR
8	Expliquer le comportement de programmes simples impliquant les éléments de base suivants : variables, expressions, affectations, E / S, structures de contrôle, fonctions, passage de paramètres et récursivité.	
9	Participer à une révision du code d'une petite équipe axée sur l'exactitude des composants.	
10	Analyser dans quelle mesure le code d'un autre programmeur respecte la documentation et les standards d'un style de programmation.	

Table 2. Extrait des objectifs d'apprentissage couverts par le protocole proposé

La phase TDD favorise un apprentissage où chaque étudiant avance à son rythme et, dans le meilleur des cas, un apprentissage auto-régulé. En effet, les étudiants travaillent sur des exercices en étant guidés par les *feedbacks* fournis par l'exécution des tests. Lors de nos expérimentations, cette approche a permis à certains étudiants d'appréhender et de comprendre seuls les notions abordées. En conséquence, l'enseignant dispose de davantage de temps à consacrer aux étudiants ayant des difficultés sur certains exercices.

2.2 Phases de revue de code par les pairs (PR)

A l'issue des phases de TDD, les étudiants arrivent avec une solution correcte vis-à-vis des tests fournis. L'objectif des phases de revue de code par les pairs (PR pour *peer review*) est de confronter les différentes solutions entre elles afin d'approfondir la compréhension par les étudiants des concepts sous-jacents aux exercices. Elles visent aussi à s'exercer aux activités sociales qui sont centrales dans des processus collaboratifs de développement ([Bourque and Fairley, 2014], chapitre 8).

Dans les expérimentations que nous avons menées, les phases PR ont été mises en œuvre en s'inspirant directement de l'approche d'Instruction par les Pairs (IP) introduite dans [Mazur, 1997]. Les études expérimentales présentées dans [Crouch and Mazur, 2001] montrent que l'IP est une approche d'évaluation formative tirant parti des interactions sociales ayant un impact positif sur l'engagement des étudiants et les résultats d'apprentissage.

Nos expérimentations de phase PR ont été orchestrées avec la plate-forme web elaastic [4] (anciennement Tsaap-Notes) proposant un protocole d'IP étendu [Silvestre et al., 2015]. La plate-forme elaastic permet à l'enseignant de poser des questions aux étudiants et de gérer un processus en 3 étapes :

— Pour chaque question, chaque étudiant, à l'aide d'un dispositif connecté (tablette, smartphone ou ordinateur portable), fournit une réponse (choix d'un item, argumentation au format texte libre) au système. L'enseignant dispose d'une interface l'informant en temps réel du nombre de réponses collectées.
— Sur la base de cette information, l'enseignant peut démarrer la deuxième étape du processus : le système demande à chaque apprenant d'étudier et d'évaluer jusqu'à cinq contributions apportées par les autres étudiants. Durant cette étape, les étudiants peuvent éventuellement modifier leur réponse initiale et la soumettre de nouveau à la plate-forme.
— L'enseignant peut décider de la publication du résultat en fonction du nombre d'étudiants ayant participé à la deuxième étape. Après publication, l'enseignant et les étudiants ont accès à la liste de toutes les contributions ordonnées de la mieux notée à la moins bien notée. L'enseignant et les étudiants ont alors l'opportunité d'échanger sur les résultats observés.

Nous avons utilisé elaastic pour demander aux étudiants de défendre leurs solutions. Ils devaient fournir des portions de code et expliquer leurs choix d'implantation. Ainsi les étudiants partagent donc leurs codes mais aussi leurs motivations argumentées par écrit pour choisir une solution plutôt qu'une autre et disposent de la possibilité d'améliorer leurs solutions à la lumière des confrontations de points de vue.

En terme de mise en œuvre, contrairement aux phases TDD qui demandent un taux d'encadrement raisonnable pour pouvoir assister les étudiants confrontés à des difficultés (typiquement, dans nos expérimentations, un enseignant pour 20 étudiants), les phases de PR sont légères en terme de logistique et peuvent être menées avec des groupes larges : nos expérimentations nous ont amené à effectuer des PR avec une cinquantaine d'étudiants en simultané.

4. `https://elaastic.irit.fr/elaastic-questions`

Alternativement à l'utilisation d'elastic, d'autres modalités de PR sans supervision de l'enseignant peuvent être pratiquées. Par exemple, les étudiants peuvent être organisés en petits groupes dans lesquels ils sont invités à se présenter entre eux leur solution et à discuter des différences aperçues. Une autre possibilité peut émerger du fait que les étudiants effectuent leur développement au sein d'un dépôt git individuel ; le mécanisme de *pull request* peut être exploité pour mener une activité de revue de code par les pairs.

La structuration de scénarios pédagogiques en une succession de cycles formés d'une phase TDD suivie d'une phase PR fait naître de nombreuses questions :

— Comment avoir une idée de la progression des étudiants ?
— Comment identifier des points de blocage communs à plusieurs étudiants ?
— Une phase PR n'a de sens que si les étudiants ont achevé la phase TDD associée et ont donc des solutions à comparer. A quel moment peut-on déclencher une phase de PR ?
— L'objectif des phases de PR est notamment de consolider les connaissances acquises lors des phases de TDD. Peut-on évaluer l'efficacité d'une telle phase avant l'évaluation sommative ?

L'étude de ces questions nous a mené à développer la plate-forme Git4School que nous présentons dans la section suivante.

3 Git4School : un tableau de bord pour le suivi des étudiants et l'orchestration de scénarios pédagogiques

Une hypothèse initiale de travail indique que chaque étudiant se voit attribuer un dépôt *git* dans lequel il réalise un *commit* à chaque fin d'exercice. L'obtention d'un tel dépôt est facilitée par l'initiative *Github Classroom* [5] qui automatise la création des dépôts *git* avec des droits d'accès paramétrables tout en y incluant un code de démarrage.

La plate-forme Git4School [6] est une application web disponible à toute personne disposant d'un compte Github. Son code source [7] est publié sous licence Apache 2.0. Elle exploite l'API REST de Github [8] pour extraire des informations sur les *commits* effectués par les étudiants au sein de leurs dépôts. Git4School est un tableau de bord, à destination des enseignants, permettant de visualiser la progression des étudiants et facilitant l'orchestration des différentes phases du protocole défini dans la section précédente.

Les deux seules hypothèses de travail conditionnant l'utilisation de Git4School sont d'une part, l'utilisation de Github par les étudiants et, d'autre part, le respect de la bonne pratique consistant à effectuer un *commit* lors de chaque achèvement majeur dans le travail tel que la terminaison d'un exercice. Git4School est donc très faiblement invasif d'un point de vue méthodologique et aussi en

5. `https://classroom.github.com/`
6. `https://git4school.firebaseapp.com/`
7. `https://github.com/git4school/git4school-visu`
8. `https://developer.github.com/v3/`

terme d'outillage : la plate-forme peut être utilisée quelque soit l'environnement de travail et le langage de programmation.

L'enseignant, une fois authentifié sur Git4School, fournit un fichier au format JSON contenant les informations suivantes :

— La liste des dépôts Github des étudiants ;
— Une liste d'identifiants pour les exercices qui sont ensuite recherchés dans les messages de *commits* des étudiants. La présence d'un identifiant est interprétée comme la terminaison de cet exercice par un étudiant lors d'une phase TDD ;
— La date et la durée des séances encadrées ;
— Pour chaque exercice, la date à laquelle a lieu la phase PR le concernant ainsi que la date de la publication de sa correction. Dans la suite, on parlera de jalon pour évoquer ces événements.

Ensuite, l'enseignant a à sa disposition trois graphiques :

— **Vue d'ensemble** (figure 2) : permet de visualiser pour chaque étudiant, en ordonnée, l'ensemble de ses *commits* au cours du temps, en abscisse. Sont repérés aussi les périodes correspondant à des séances encadrées, à la réalisation de phase de PR et à la publication d'une correction. Les *commits* sont modélisés par des formes de couleur différente selon s'il sont effectués avant ou après un jalon (phase PR ou publication de la correction).

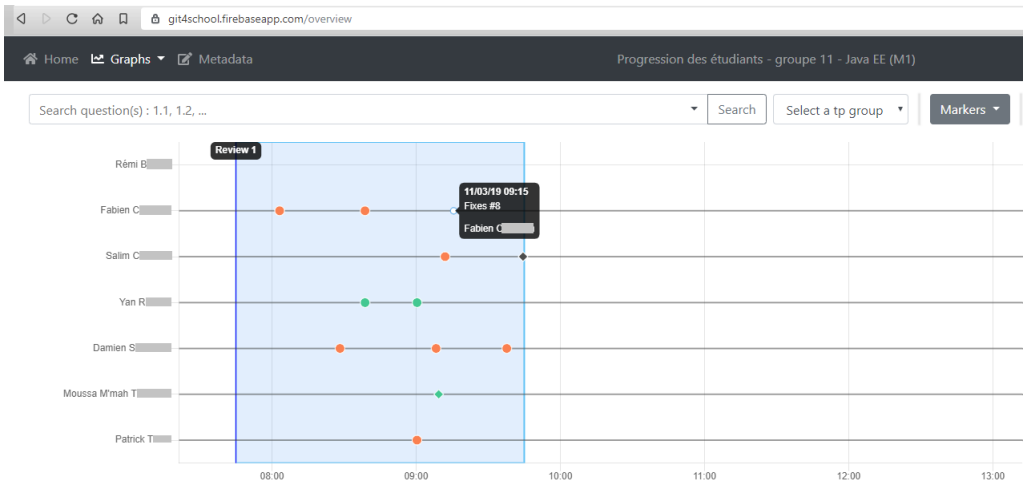


Figure 2. Visualisation de la progression de l'ensemble dans Git4School

— **Visualisation des *commits* typés** (figure 3) : permet de visualiser sous la forme d'un diagramme en bâton, pour chaque étudiant, le nombre de ses *commits* par rapport aux jalons (commits avant PR, avant publication de la correction, etc) ainsi que l'identifiant du dernier exercice traité.

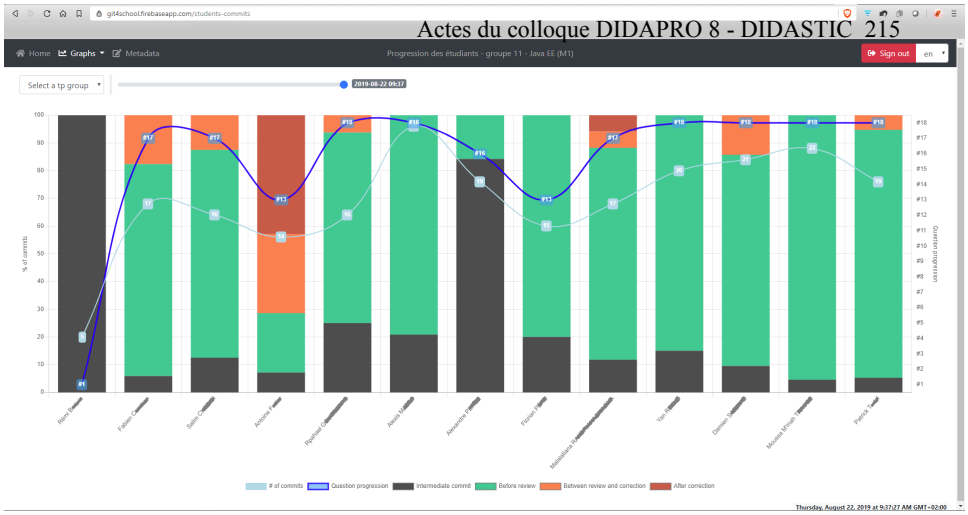


FIGURE 3. Visualisation du type des *commits* de chaque étudiant dans Git4School

— **Visualisation de l'accomplissement par exercice** (figure 4) : permet de visualiser sous la forme d'un diagramme en bâton, pour chaque exercice, le nombre d'étudiant l'ayant terminé et dans quelle condition (avant ou après la phase de PR, après la publication de la correction).

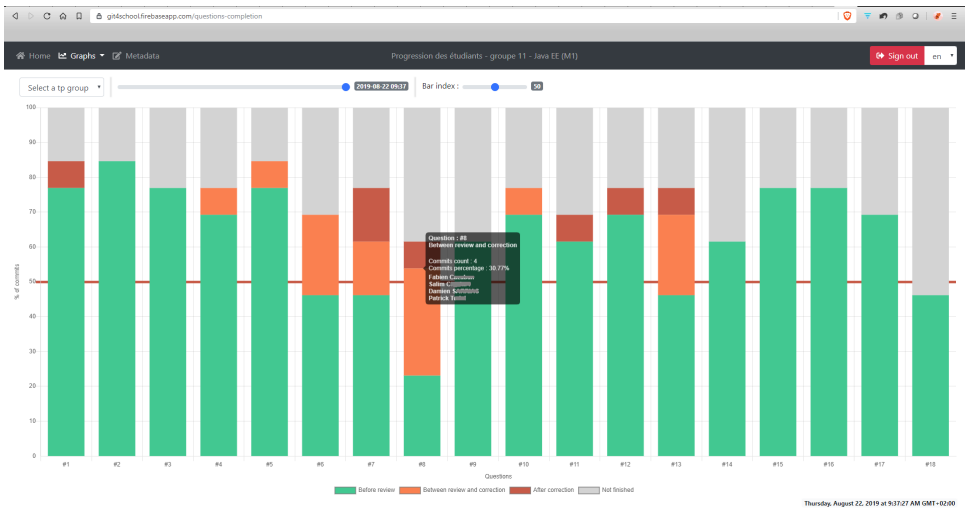


FIGURE 4. Visualisation de l'accomplissement par exercice dans Git4School

Les deux premiers graphiques permettent d'obtenir des informations sur la façon dont chaque étudiant travaille : comment se positionne t-il par rapport au reste de ses camarades ? s'il est en retard, essaie t-il de compenser en traitant des exercices en dehors des séances encadrées ? Tire-t-il profit des phases de PR ou de la publication de la correction pour achever les exercices sur lesquels il a des difficultés ?

Le troisième graphe est utile pour l'orchestration des phases du protocole : si suffisamment d'étudiants ont achevé l'exercice, il peut être pertinent de déclencher la phase de PR. Une autre utilisation possible de ce graphe concerne les exercices identifiés comme bloquants car terminés par peu ou pas d'étudiants ; une remédiation peut être effectuée au cours d'une phase de PR dont l'impact peut être visualisé par la couleur orange dans le diagramme en bâton qui indique le pourcentage d'étudiants ayant achevé un exercice après la phase de PR. L'enseignant dispose alors d'un retour sur sa tentative pédagogique de déblocage.

4 Discussions et travaux connexes

Alignements avec le SWEBOK et le curriculum de l'ACM/IEEE. Le protocole d'apprentissage introduit dans cet article permet d'initier précocement les étudiants à plusieurs pratiques professionnelles référencées dans le SWEBOK. Les phases TDD sont très formatrices pour le domaine de connaissance *« software construction tools »* ([Bourque and Fairley, 2014], chapitre 3) par l'utilisation intensive d'un environnement de développement, d'un *framework* de test, d'un gestionnaire de version *git* et d'un *debugger*.

Les aspects méthodologiques agiles ([Bourque and Fairley, 2014], chapitre 9) sont centraux dans le protocole avec l'idée de procéder par cycles courts de développement dans une démarche itérative centrée sur les tests ; de plus, les activités sociales des phases PR forment aux processus collaboratifs de développement ([Bourque and Fairley, 2014], chapitre 8 et 10).

Concernant le curriculum de l'ACM/IEEE, comme déjà mentionné lors de l'étude du tableau 2, le protocole grâce à l'ajout des phases PR offre une meilleure couverture des objectifs d'apprentissage du curriculum.

Engagement des étudiants. L'engagement des étudiants est un levier essentiel pour améliorer le processus d'apprentissage. Le cadre ICAP [Chi and Wylie, 2014] décrit quatre modes d'engagement des étudiants : passif, actif, constructif et interactif. En mode passif, les étudiants reçoivent directement les instructions de l'enseignant pendant les cours ou regardent une vidéo sans autre activité que l'attention à l'écran. Lorsque l'attention donnée est soutenue par une action motrice ou une manipulation physique, on dit que les étudiants sont en mode actif. Lorsque les élèves créent ou produisent du matériel supplémentaire, ils s'engagent de manière constructive. Enfin, lorsque les élèves interagissent avec d'autres (pairs, tuteurs, enseignants) pour construire de nouveaux supports, ils

sont ensuite engagés dans le mode interactif. D'après des mesures d'apprentissage, le mode interactif subsume le mode constructif qui subsume le mode actif subsumant le mode passif.

Une caractéristique majeure du protocole est qu'il propose un environnement dans lequel les étudiants basculent continuellement entre un mode constructif et un mode interactif tout au long des cycles répétés des phases TDD et PR. Ils sont donc toujours placés aux deux niveaux d'engagement les plus élevés, maximisant ainsi leurs chances d'améliorer leurs compétences en programmation.

Travaux connexes. L'apprentissage dirigé par les tests (TDL pour *Test Driven Learning*) a été introduit dans [Janzen and Saiedian, 2006] pour relever le défi d'apprendre à programmer en utilisant les tests afin d'expliciter l'usage et le comportement de portions de codes. Les résultats des expérimentations montrent que les étudiants gagnent en compréhension lorsqu'ils sont en situation d'apprentissage dirigée par les tests [Janzen and Saiedian, 2008] et qu'ils améliorent leurs performances aux évaluations [Hilton and Janzen, 2012].

Plusieurs travaux d'extraction de métriques à partir des logs d'un gestionnaire de version existent dans la littérature (voir [Mittal and Sureka, 2014] pour une vue d'ensemble) mais contrairement à notre approche qui analyse des messages de *commits* qui portent des informations sémantiques (la terminaison d'un exercice bien identifié), les travaux s'intéressent plutôt à des informations sur les *commits* de type volumétrie, fréquence, temporalité relative à une date limite ou analysent la collaboration entre membres d'un même dépôt.

Concernant Git4School, de nombreux tableaux de bord de suivi des apprentissages existent [Charleer et al., 2014,Putra et al., 2018]. Notre approche est originale par son aspect non-invasif et le fait qu'elle soit dédiée à la plate-forme Github.

5 Conclusion : pistes pour des travaux futurs

Le protocole rapporté ici a été raffiné au cours de deux années d'expérimentations. Arrivé maintenant à un état stable, les données d'apprentissage actuellement collectées sont fidèles à l'esprit du protocole. Le premier travail que nous mènerons à court terme va consister à analyser statistiquement ces données pour vérifier si les résultats aux examens se sont améliorés depuis l'établissement du protocole.

De même, peut-on corréler le comportement de l'étudiant au sein de son dépôt *git* et ses résultats lors des examens ? Le cas échéant, la mise en place d'un système d'alerte permettant d'anticiper un potentiel échec de l'étudiant et d'adapter en conséquence le contenu d'une phase PR constituerait une piste future de travail.

Références

[Beck, 2003] Beck, K. (2003). *Test-driven development : by example.* Addison-Wesley Professional.

[Bergeat, 2017] Bergeat, M. (2017). Les tensions sur le marché du travail en 2017. https://dares.travail-emploi.gouv.fr/IMG/pdf/2017-056.pdf. Rapport DARES Indicateurs, Ministère du travail.

[Biggs, 1996] Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher education*, 32(3) :347–364.

[Bourque and Fairley, 2014] Bourque, P. and Fairley, R. E., editors (2014). *SWEBOK : Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society.

[Charleer et al., 2014] Charleer, S., Santos, J., Klerkx, J., and Duval, E. (2014). Improving teacher awareness through activity, badge and content visualizations. In *New Horizons in Web Based Learning*, volume 8699, pages 143–152.

[Chi and Wylie, 2014] Chi, M. T. and Wylie, R. (2014). The icap framework : Linking cognitive engagement to active learning outcomes. *Educational Psychologist*, 49(4) :219–243.

[Colin et al., 2015] Colin, J.-F., Aboudara, S., Jolly, C., Lainé, F., Argouarc'h, J., and Bessière, S. (2015). Les métiers en 2022. https://www.strategie.gouv.fr/sites/strategie.gouv.fr/files/atoms/files/fs_rapport_metiers_en_2022_27042015_final.pdf. Rapport de la DARES, Ministère du travail.

[Crouch and Mazur, 2001] Crouch, C. H. and Mazur, E. (2001). Peer instruction : Ten years of experience and results. *American journal of physics*, 69(9) :970–977.

[Hilton and Janzen, 2012] Hilton, M. and Janzen, D. S. (2012). On teaching arrays with test-driven learning in webide. In *Conference on Innovation and technology in computer science education*, pages 93–98. ACM.

[Janzen and Saiedian, 2008] Janzen, D. and Saiedian, H. (2008). Test-driven learning in early programming courses. In *ACM SIGCSE Bulletin*, volume 40, pages 532–536. ACM.

[Janzen and Saiedian, 2006] Janzen, D. S. and Saiedian, H. (2006). Test-driven learning : intrinsic integration of testing into the cs/se curriculum. In *ACM SIGCSE Bulletin*, volume 38, pages 254–258. ACM.

[Mazur, 1997] Mazur, E. (1997). Peer instruction : getting students to think in class. In *AIP Conference Proceedings*, pages 981–988.

[Mittal and Sureka, 2014] Mittal, M. and Sureka, A. (2014). Process mining software repositories from student projects in an undergraduate software engineering course. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 344–353. ACM.

[Putra et al., 2018] Putra, F., Santoso, H., and Aji, R. (2018). Evaluation of learning analytics metrics and dashboard in a software engineering project course. *Global Journal of Engineering Education*, 20(3) :171–180.

[Sahami et al., 2013] Sahami, M., Roach, S., and ACM/IEEE-CS Joint Task Force on Computing Curricula (2013). Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press.

[Silvestre et al., 2015] Silvestre, F., Vidal, P., and Broisin, J. (2015). Reflexive learning, socio-cognitive conflict and peer-assessment to improve the quality of feedbacks in online tests. In *Design for Teaching and Learning in a Networked World*, pages 339–351. Springer.