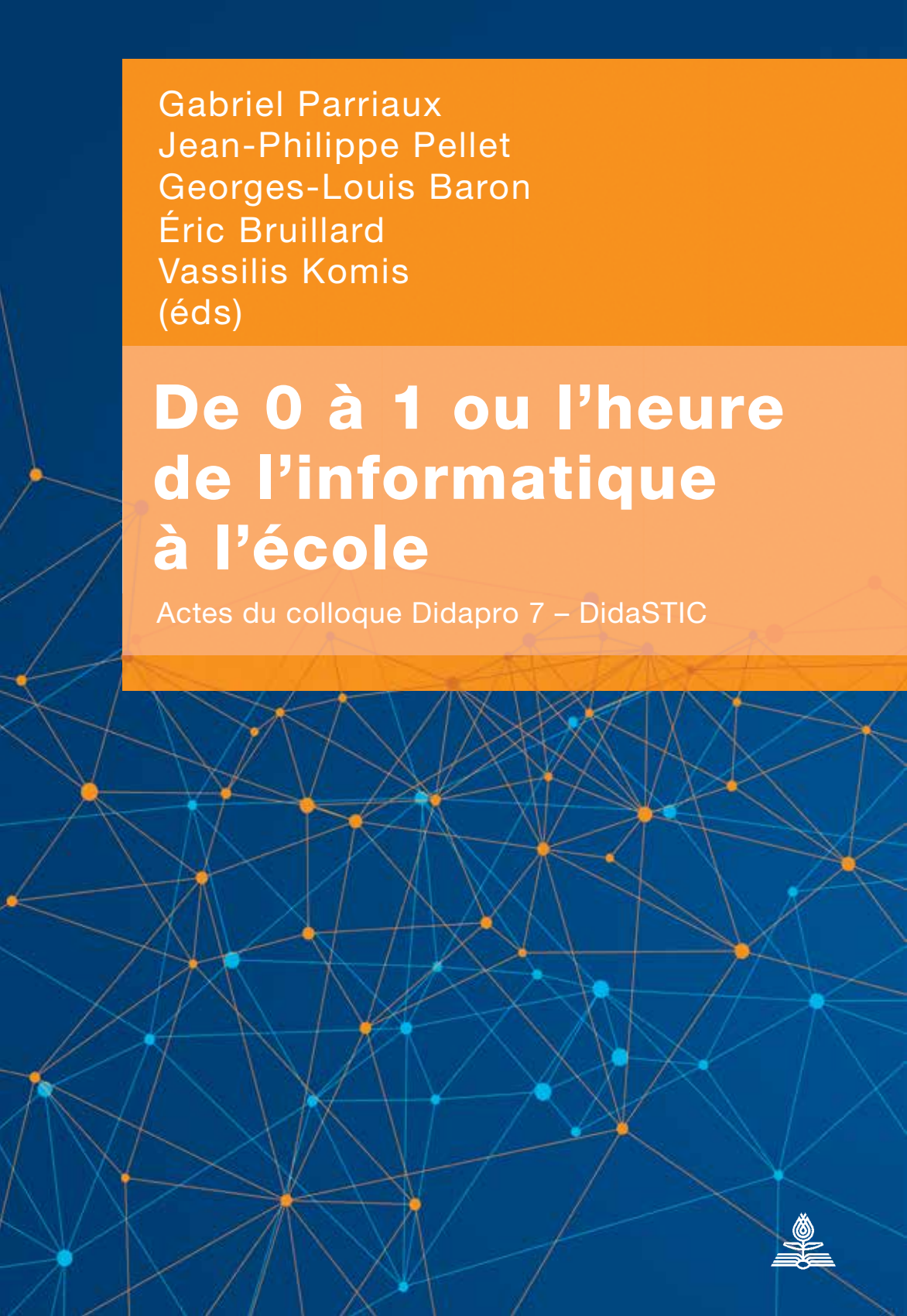
Gabriel Parriaux
Jean-Philippe Pellet
Georges-Louis Baron
Éric Bruillard
Vassilis Komis
(éds)

De 0 à 1 ou l'heure de l'informatique à l'école

Actes du colloque Didapro 7 – DidaSTIC



Depuis que l'informatique est un objet d'enseignement–apprentissage, les acteurs de la recherche se rencontrent régulièrement autour des dilemmes que suscite cet objet. C'est à ce titre que, depuis 1988, les colloques de didactique de l'informatique, puis le colloque Didapro – DidaSTIC, explorent ce domaine.

Cet ouvrage constitue les actes de l'édition 2018 tenue à la Haute école pédagogique du canton de Vaud à Lausanne et présente les recherches les plus récentes dans le domaine, classifiées en trois grandes thématiques.

Au cœur de la discipline Informatique – rassemble les articles qui portent une réflexion sur les fondements de la discipline informatique, ce qui lui donne sens, la délimite ainsi que sur l'organisation de ses concepts.

Formation des enseignant·e·s et enjeux institutionnels – réunit les recherches centrées sur les questions cruciales de formation des enseignant·e·s à l'enseignement de l'informatique ou présentant la situation propre à un pays.

Quels enseignements, avec quels outils ? – fédère les études de dispositifs d'enseignement, d'outils et d'usages innovants pour l'apprentissage de l'informatique aux différents niveaux de la scolarité.

Gabriel Parriaux est professeur-formateur à la Haute école pédagogique du canton de Vaud. D'une formation en langues orientales (chinois, sanscrit), il mène des recherches et enseigne dans le domaine de la didactique de l'informatique en Suisse romande. Il est également vice-président de la Société suisse pour l'informatique dans l'enseignement.

Jean-Philippe Pellet enseigne la programmation à l'École Polytechnique Fédérale de Lausanne, en Suisse, et est impliqué dans la formation des enseignants d'informatique à la Haute école pédagogique du canton de Vaud. Il est titulaire d'un doctorat en informatique de l'ETH Zurich.

Georges-Louis Baron est professeur émérite de sciences de l'éducation à l'Université Paris Descartes, laboratoire EDA. Il mène et dirige des recherches depuis une trentaine d'années sur la didactique de l'informatique et l'appropriation des instruments informatiques par les apprenants et les enseignants.

Éric Bruillard est professeur des universités à Paris Descartes au laboratoire EDA. Il travaille depuis plus de 35 ans sur les questions de conception et d'usage des technologies issues de l'informatique en éducation. Concepteur et coordinateur de la série de Mooc eFAN (enseigner et former avec le numérique), ancien rédacteur chef de STICEF, il est vice-président de l'IARTEM.

Vassilis Komis est professeur au Département des Sciences de l'Éducation à l'Université de Patras (Grèce), où il enseigne la didactique de l'informatique et les Technologies de l'Information et de la Communication en Éducation. Sa recherche concerne l'enseignement de l'informatique à l'école, la conception des environnements numérique d'apprentissage et la formation des enseignants aux TICE.

www.peterlang.com

De 0 à 1 ou l'heure de l'informatique à l'école

Gabriel Parriaux, Jean-Philippe Pellet,
Georges-Louis Baron, Éric Bruillard,
Vassilis Komis (éds)

De 0 à 1 ou l'heure de l'informatique à l'école

Actes du colloque Didapro 7 – DidaSTIC

PETER LANG
Bern • Berlin • Bruxelles • New York • Oxford • Warszawa • Wien

Information bibliographique publiée par «Die Deutsche Bibliothek»
«Die Deutsche Bibliothek» répertorie cette publication dans la «Deutsche Nationalbibliografie»; les données bibliographiques détaillées sont disponibles sur Internet sous ‹http://dnb.ddb.de›.

ISBN 978-3-0343-3307-8
ISBN 978-3-0343-3326-9 MOBI
DOI 10.3726/b13411
ISBN 978-3-0343-3324-5 eBook
ISBN 978-3-0343-3325-2 EPUB

PETER LANG
Open

Open Access: Cette œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0. Pour consulter une copie de cette licence, visitez le site internet https://creativecommons.org/licenses/by-nc-nd/4.0/

Cette publication a fait l'objet d'une évaluation par les pairs.

© Gabriel Parriaux, Jean-Philippe Pellet, Georges-Louis Baron, Éric Bruillard, Vassilis Komis (éds), 2018
Peter Lang AG
International Academic Publishers
Bern

Préface

Chacun·e s'accorde aujourd'hui sur la place prééminente qu'occupent l'informatique et les réalisations qui en découlent dans nos sociétés. Pratiquement tous les domaines de l'activité humaine ont été profondément impactés, transformés, remodelés par cette science et ses développements. Pour autant, l'informatique en tant que discipline d'enseignement reste jeune. Sa place dans les curricula, son statut, ses contenus, la formation de ses enseignant·e·s – tous ces éléments varient grandement d'un pays à l'autre, d'une région à l'autre, en fonction de choix politiques, sociaux ou économiques.

Le colloque Didapro s'intéresse depuis ses débuts à l'enseignement et à l'apprentissage de l'informatique et on peut mettre en lumière un certain parallélisme entre l'évolution du statut de l'informatique dans le domaine de l'éducation et l'évolution de ce colloque.

Dans les écoles, les premières expériences avec des ordinateurs sont réalisées dès les années 70. Ceux-ci ne disposent pas encore d'interface graphique et leur usage est réservé à des spécialistes qui ont acquis des bases de langages de programmation. Durant cette période, les premiers colloques de didactique de l'informatique, réalisées alors sous l'égide de l'Association Francophone pour la Didactique de l'Informatique (AFDI), concentrent leurs réflexions sur l'enseignement de la programmation.

Les années 80 ont vu les ordinateurs personnels arriver en masse sur le marché, avec une baisse très importante de prix. Dans les années 90, les interfaces graphiques se sont généralisées, les rendant accessibles au plus grand nombre tandis que les outils bureautiques étaient utilisés en dehors du monde professionnel. En milieu scolaire, l'informatique est devenue un outil de production et de communication, un support aux apprentissages des élèves dans les diverses disciplines. Ensuite, les réflexions sur l'apprentissage des logiciels sont passées au premier plan. Les colloques ont alors pris, au début des années 2000, le nom de « Didapro », pour « didactique des progiciels ». Le colloque de Patras en 2011 a élargi la thématique à la didactique des Sciences et Technologies de l'Information et de la Communication (STIC), d'où le nom de « Didapro – DidaSTIC ».

Plus récemment, l'approche qui voudrait résumer l'informatique scolaire à l'usage d'outils a été mise en question. Cet usage d'outils, dénué des fondements conceptuels qui les régissent, est insuffisant pour amener les enfants à mieux comprendre le monde qui les entoure. Un mouvement vers l'introduction d'enseignement de l'informatique à tous les niveaux, en tant que discipline à part entière, a vu le jour dans de nombreux pays. Nos deux derniers colloques ont assez largement consacré ce mouvement en faisant porter une partie importante de leurs réflexions sur ces problématiques.

Cette édition de Didapro en 2018 a eu pour objectif d'analyser les évolutions intervenues depuis les années 80 tout en présentant et discutant les résultats des recherches les plus récentes dans le domaine de la didactique de l'informatique. Pour la septième édition de ce colloque, le comité a reçu 41 résumés initiaux d'articles et 33 articles complets, dont 18 ont été acceptés pour présentation et pour publication dans les actes.

Nous avons regroupé les articles publiés en trois grandes thématiques.

Au cœur de la discipline Informatique – rassemble les articles qui portent une réflexion sur les fondements de la discipline informatique, ce qui lui donne sens, la délimite ainsi que sur l'organisation de ses concepts.

Formation des enseignant·e·s et enjeux institutionnels – réunit les recherches centrées sur les questions cruciales de formation des enseignant·e·s à l'enseignement de l'informatique ou présentant la situation propre à un pays.

Quels enseignements, avec quels outils ? – fédère les études de dispositifs d'enseignement, d'outils et d'usages innovants pour l'apprentissage de l'informatique aux différents niveaux de la scolarité.

L'intégralité des articles publiés ici est également disponible en *open access* en ligne dans l'archive ouverte pluridisciplinaire HAL[1].

Le colloque a été enrichi par les conférences invitées de trois expert·e·s de renommée internationale : Pierre Dillenbourg, Marina Umaschi Bers et Ivan Kalaš. Les résumés de leurs interventions font partie de ces actes.

Les présentations et conférences invitées ont été complétées par des tables rondes et des sessions de présentation de posters et la première journée du colloque, organisée comme journée des enseignant·e·s, a donné l'occasion de mettre sur pied des ateliers où enseignant·e·s et chercheur·e·s ont présenté des démarches innovantes d'enseignement. Une partie des documents relatifs reste disponible en ligne[2].

1 <https://hal.archives-ouvertes.fr/DIDAPRO7>

2 <https://www.didapro.org>

Remerciements

Nous remercions Bernard Baumberger, responsable de l'unité d'enseignement et de recherche Médias et TIC dans l'enseignement et la formation à la Haute école pédagogique du canton de Vaud, qui a apporté son soutien à toutes nos démarches organisationnelles. Nous remercions aussi tous les membres du comité d'organisation local, parmi lesquels Isabelle Grosjean pour son implication administrative sans égal et Morgane Chevalier pour son regard affuté.

L'appui constant des présidents historiques du colloque, Georges-Louis Baron et Éric Bruillard, ainsi que celui de Vassilis Komis, ont été sans faille. Nous leur sommes grandement reconnaissants de la générosité avec laquelle ils nous ont fait bénéficier de leur expérience. Nous remercions également les membres du comité scientifique qui, par leurs lectures et analyses, nous ont permis de sélectionner les meilleures productions pour Didapro.

Le cœur du colloque est bien sûr constitué de ses auteurs et de sa communauté propre. Nous remercions tous les auteurs qui ont soumis des contributions !

Nous souhaitons également remercier le personnel des unités de service de la Haute école pédagogique, sans qui l'organisation au sein de ces locaux n'aurait pas été possible : les unités Communication, Informatique et Infrastructure. Un merci particulier à Pierre Ramelot, du Centre de soutien à la recherche, pour son appui efficace dans la recherche de financement.

Nous n'oublions pas nos partenaires, dont le Fonds national suisse de la recherche scientifique (FNS), Google, la Fondation Hasler, le Centre de Compétences romand de Didactique Disciplinaire (2Cr2D), l'Agence universitaire de la Francophonie (AUF) et Mémoire Vive – Apple Solution Expert Éducation, qui ont rendu possible la mise sur pied du colloque dans d'excellentes conditions, et notre éditeur, Peter Lang, qui a fait preuve d'intérêt dès nos premiers contacts et s'est montré flexible et efficace. À tous, un grand merci !

Gabriel Parriaux & Jean-Philippe Pellet,
novembre 2017

Table des matières

Organisation du colloque 13

Conférences invitées

Pierre Dillenbourg
Pensée computationnelle : Pour un néopapertisme durable car sceptique 17

Marina Umaschi Bers
La programmation en tant que place de jeu développementale : la pensée informatique et la robotique dans la petite enfance 21

Ivan Kalaš
La programmation à l'école primaire : De Papert à la nouvelle informatique 23

Articles

I. Au cœur de la discipline Informatique

Yannis Delmas-Rigoutsos
Proposition de structuration historique des concepts de la pensée informatique fondamentale 31

Julie Henry, Alyson Hernalesteen, Bruno Dumas & Anne-Sophie Collard
Que signifie éduquer au numérique ? Pour une approche interdisciplinaire 61

Frédéric Drouillon
Regard littéraire sur la programmation comprise comme une écriture 83

II. Formation des enseignant·e·s et enjeux institutionnels

Isabelle Debled-Rennesson, Philippe Févotte,
Monique Grandbastien, David Langlois & Hélène Tanoh
La formation des professeurs de la spécialité ISN dans l'Académie
de Nancy-Metz : Récit et analyse de six ans d'expérience 99

Frédérique Chessel Lazzarotto
Former à la programmation en primaire, une form'action :
Robots d'Evian 2015–2018 117

Julie Henry & Anne Smal
« Et si demain je devais enseigner l'informatique ? »
Le cas des enseignants de Belgique francophone 129

Florent Tasso & Monique Ouassa Kouaro
Intégration de l'enseignement de l'informatique dans
les établissements d'enseignement secondaire du Bénin 151

III. Quels enseignements, avec quels outils ?

Sylvie Tissot & Mireille Bétrancourt
La formation des nouveaux étudiants à l'usage des TIC :
méthode des invariants ou apprentissage des procédures ? 167

Christelle Pauty-Combemorel
Utilisation d'un jeu vidéo dans le cadre de l'enseignement
des SVT : le cas de Minetest 187

Béatrice Drot-Delange & Françoise Tort
Concours Castor, ressource pédagogique pour l'enseignement de l'informatique ? Étude exploratoire auprès d'enseignants 199

Dimitri Racordon & Didier Buchs
Démystifier les concepts informatiques par l'expérimentation 219

Sandra Nogry
Comment apprennent les élèves au cours d'une séquence de robotique éducative en classe de CP ? 235

Thibault Desprez, Stéphanie Noirpoudre, Théo Segonds, Damien Caselli, Didier Roy & Pierre-Yves Oudeyer
Poppy Ergo Jr : un kit robotique au cœur du dispositif Poppy Éducation 245

Thierry Karsenti & Julien Bugmann
Un robot humanoïde pour enseigner la programmation : une recherche exploratoire auprès d'élèves ayant des difficultés d'apprentissage 257

Patrice Frison, Moncef Daoud & Michel Adam
Transition didactique de l'activité débranchée à la programmation avec AlgoTouch 273

Chryssa Tsourapi, Vassilis Komis & Georges-Louis Baron
L'étayage des enseignants de l'école maternelle au cours des activités de programmation avec le logiciel ScratchJr 291

Sévastiani Touloupaki, Georges-Louis Baron & Vassilis Komis
Un apprentissage de la programmation dès l'école primaire : le concept de message sur ScratchJr 303

Étienne Vandeput & Julie Henry
Apprendre à programmer Comment les enseignants justifient-ils le choix d'un outil didactique ? 325

Index des auteur·e·s 343

Organisation du colloque

Co-présidents

Gabriel Parriaux	*HEP Vaud*
Georges-Louis Baron	*Sorbonne Paris Cité, Université Paris Descartes*
Éric Bruillard	*ENS Paris-Saclay*

Comité d'organisation

Gabriel Parriaux	*HEP Vaud*
Jean-Philippe Pellet	*HEP Vaud, EPFL*
Isabelle Grosjean	*HEP Vaud*
Vassilis Komis	*Université de Patras*
Georges-Louis Baron	*Sorbonne Paris Cité, Université Paris Descartes*
Éric Bruillard	*ENS Paris-Saclay*
Morgane Chevalier	*HEP Vaud*

Comité scientifique

Georges-Louis Baron	*Sorbonne Paris Cité, Université Paris Descartes*
Monique Baron	*UPMC Sorbonne Universités*
Bernard Baumberger	*HEP Vaud*
Lætitia Boulc'h	*Université Paris Descartes*
Éric Bruillard	*ENS Paris-Saclay*
Didier Buchs	*Université de Genève*
Morgane Chevalier	*HEP Vaud*
Vassilios Dagdilelis	*Université de Macédoine*
Pierre Dillenbourg	*EPFL*
Gilles Dowek	*ENS Saclay*
Béatrice Drot-Delange	*Université Blaise Pascal*
Monique Grandbastien	*Université Poincaré, Nancy*
Julie Henry	*Université de Namur, Université de Liège*
Rolf Ingold	*Université de Fribourg*
David Janisczek	*Paris 5*
Thierry Karsenti	*Université de Montréal*
Vassilis Komis	*Université de Patras*
Caroline Ladage	*Aix-Marseille Université*

Mona Laroussi	*Institut francophone d'ingénierie de la connaissance, Tunis*
Pascal Leroux	*Université du Maine*
Olivier Lévêque	*EPFL*
Malika More	*Université d'Auvergne*
Aude Nguyen	*ex-Université de Namur*
Claver Nijimbere	*ENS Bujumbura*
Gabriel Parriaux	*HEP Vaud*
Jacques Pasquier-Rocha	*Université de Fribourg*
Jean-Philippe Pellet	*HEP Vaud, EPFL*
Florence Quinche	*HEP Vaud*
Martin Quinson	*ENS Rennes*
Konstantinos Ravanis	*Université de Patras*
Margarida Romero	*ÉSPÉ de Nice*
Didier Roy	*INRIA*
André Schiper	*EPFL*
Daniel Schneider	*Université de Genève*
Françoise Tort	*ENS Saclay*
Éric Tortochot	*Aix-Marseille Université*
Étienne Vandeput	*IUFE Genève, Université de Namur*
François Villemonteix	*Université de Cergy-Pontoise*
Emmanuelle Voulgre	*Sorbonne Paris Cité, Université Paris Descartes*

CONFÉRENCES INVITÉES

Pierre Dillenbourg

École Polytechnique Fédérale de Lausanne, Suisse
pierre.dillenbourg@epfl.ch

Pensée computationnelle : Pour un néopapertisme durable car sceptique

« *On a vu souvent rejaillir le feu d'un ancien volcan qu'on croyait trop vieux* » chantait le Grand Jacques. L'intérêt actuel pour la pensée computationnelle n'a-t-il pas un goût de déjà vu pour ceux qui ont connu les années Logo ? L'histoire de l'éducation est peuplée d'exemples dans lesquels une idée ne s'impose pas à sa première apparition mais à sa deuxième, ou à sa troisième… Le contexte actuel explique la résurgence de ces idées brillantes. On peut décrire le contexte en termes d'offre et de demande. Du côté de la demande, l'explosion des performances de l'intelligence artificielle et de la robotique a fait prendre conscience de l'urgence de considérer la pensée computationnelle comme compétence fondamentale de tout citoyen confronté aux algorithmes qui nous influencent. Le climat est cependant différent : alors que les tentatives des années 80 reposaient sur un formidable optimisme, les efforts actuels baignent dans un certain climat d'inquiétude, notamment quant à l'emploi. Du côté de l'offre, il n'existe plus une approche unique de la question, comme Logo, mais une multitude de plateformes logicielles et robotiques pour l'apprentissage de la programmation et/ou de la pensée computationnelle. Je vous parlerai évidemment de Thymio dont mon collègue Francesco Mondada a déjà vendu plus de 30 000 exemplaires et pour lequel il a formé 900 enseignants. Je vous présenterai également les projets de robotique développés dans mon laboratoire, tels que Cellulo[1]. Mais, bien que l'offre et la demande convergent magiquement, la situation n'est pas sans embûches. Il est bon de regarder à la fois en arrière, apprendre des erreurs commises avec Logo, et vers l'avant, pour anticiper les écueils que l'on voit poindre à l'horizon.

1 <https://chili.epfl.ch/cellulo>

Dans le rétroviseur, on s'aperçoit que le potentiel pédagogique de Logo fut en quelque sorte victime du niveau d'attentes suscité par le discours de Papert, discours certes brillant et charismatique, mais promettant des effets qu'aucune approche pédagogique ne pouvait atteindre. Si on vous promet de gagner 1 million, vous serez déçu d'en recevoir un demi. Aujourd'hui, soyons modestes dans nos promesses de résultats. Du côté futur, anticipons que l'engouement actuel, le *hype* autour de ce thème, s'affaiblira à terme parmi les décideurs de nos systèmes éducatifs. Il convient dès lors d'inscrire nos plans d'actions dans le long terme, de conduire des projets qui continueront à fonctionner quand les médias auront détourné leur attention.

Dans le rétroviseur, on peut dire qu'il régnait un certain abus de confiance dans les propriétés éducatives intrinsèques dont un outil technologique serait pourvu. Un élève peut passer des heures avec un langage ou un robot sans rien apprendre. Ce qu'il apprend dépend de l'activité qu'il fait avec ce robot : copie-t-il du code fourni par un (mauvais) enseignant ou explore-t-il, va-t-il utiliser des variables ou la récursion ? Aucun outil n'a des propriétés pédagogiques intrinsèques ; un outil possède certes un certain potentiel (« affordance ») mais ce potentiel ne devient effet réel qu'en fonction des activités que l'élève réalisera. Or, qui prépare ces activités ? L'enseignant ! Ne commettons donc pas pour la seconde fois l'erreur de négliger le rôle des enseignants dans l'effet des technologies utilisées. Dès lors, du côté futur, préparons les enseignants à la pensée computationnelle et, en toute urgence, formons les formateurs d'enseignants.

Dans le rétroviseur, l'approche Logo postulait le développement des capacités de résolution de problèmes qui soient indépendantes du domaine d'application. Or, l'existence même de ces compétences a été remise ensuite en cause par les travaux sur la cognition située. Pourquoi suis-je capable de décomposer un problème complexe en problèmes simples lorsqu'il s'agit de programmation et non pas lorsque j'essaie vainement de réparer la plomberie de ma salle de bains ? Parce que notre système cognitif n'est pas composé de strates, certaines génériques, d'autres spécifiques. Le transfert a toujours été le talon d'Achille de l'éducation. Pour qu'un élève soit capable de transférer une compétence d'un domaine à l'autre, les cours doivent inclure des activités de transfert entre plusieurs domaines. C'est pourquoi il ne faut pas dédier un nouveau cours à la pensée computationnelle, probablement assigné aux professeurs de maths, mais au contraire parler de ces principes dans les cours d'histoire, d'allemand, d'économie, etc. Le défi en termes de formation des enseignants n'en est que décuplé.

Dans le rétroviseur, certains ont confondu la pensée algorithmique avec l'histoire ou la sociologie des médias. Dans le futur, il faut certes parler des *fake news* et des réseaux sociaux, mais une vraie compréhension des médias actuels ne limite pas à identifier les mauvaises intentions d'une inéluctable partie de notre société, mais à identifier par exemple les propriétés des graphes sous-jacents aux réseaux, propriétés qui expliquent certaines failles et les biais du système.

Dans le rétroviseur, on revit les combats idéologiques autour de Logo, qui ressemblent assez à ceux que j'observe aujourd'hui sur la relation entre pensée computationnelle et programmation. Certains considèrent la première comme un pas intermédiaire et la seconde comme l'objectif principal. D'autres craignent que les activités de codage nuisent à l'acquisition de la pensée computationnelle, en exagérant les contraintes formelles, notamment la syntaxe. Mais dans ce cas, on peut craindre que la pensée computationnelle soit enseignée comme des maths. Aujourd'hui, les outils de programmation permettent de libérer les élèves des contraintes syntaxiques et donc aux enseignants de leur faire découvrir les algorithmes de manière exploratoire, grâce à la programmation. La programmation permet de rendre un problème croustillant ; l'élève peut commettre autant d'erreurs qu'il veut et recommencer souvent sans s'exposer au feedback d'un enseignant. Je considère que la programmation est au service de la pensée computationnelle et non l'inverse.

Papert était un scientifique très inventif et un communicateur de génie, et les efforts actuels ne sont pas dénués d'un certain militantisme sympathique. Il n'y a certes pas d'éducation sans système de valeurs, mais gardons un regard relativement objectif et scientifique sur la conception des programmes de formation à la pensée computationnelle.

Marina Umaschi Bers

Tufts University, Massachusetts, États-Unis
marina.bers@tufts.edu

La programmation en tant que place de jeu développementale : la pensée informatique et la robotique dans la petite enfance

Marina Umaschi Bers est professeure au département Eliot-Pearson de l'étude de l'enfant et du développement humain et au département d'informatique de la Tufts University. Elle dirige le groupe de recherche interdisciplinaire des technologies développementales. Ses recherches comprennent la conception et l'étude de technologies d'apprentissage qui favorisent le développement positif des enfants. Elle a créé l'application de programmation bien connue ScratchJr et le kit robotique KIBO, destinés aux enfants de 4 à 7 ans.

Dans ma conférence, je présenterai un survol de mon programme de recherche interdisciplinaire en utilisant la métaphore de l'opposition entre place de jeu et parc pour enfants pour comprendre le rôle de la programmation dans l'apprentissage de l'informatique que font les enfants. Les places de jeu sont des espaces prisés des jeunes enfants pour jouer et apprendre. Les enfants peuvent y être autonomes tout en développant différentes séries de compétences. Les parcs pour enfants, en revanche, « parquent » les enfants dans un espace confiné et sûr. Même si les risques y sont limités, il ne s'y passe que peu d'exploration, de résolution de problèmes ou de jeu imaginatif.

Ma présentation utilisera la métaphore « place de jeu/parc pour enfants » pour investiguer le rôle de la programmation et de la pensée informatique chez les jeunes enfants. Je présenterai un cadre de réflexion autour des idées fortes de l'informatique et de l'ingénierie adaptées à de jeunes enfants et montrerai des exemples qui mettent en œuvre les deux environnements que j'ai créés, le langage de programmation ScratchJr et le kit robotique KIBO.

La conférence reprendra des idées de mon récent livre *Coding as a Playground*, où je propose que la programmation soit considérée non

seulement comme une compétence technique, mais comme une nouvelle littératie – un nouveau moyen pour les enfants d'exprimer et de partager leurs idées. En tant que littératie, la programmation a le pouvoir de changer le monde. Je présenterai enfin mes nouvelles recherches, qui s'intéressent aux bases cognitives et neuronales de l'apprentissage de l'informatique dans la petite enfance.

Ivan Kalaš

Université Comenius de Bratislava, Slovaquie
UCL Institute of Education, Londres, Royaume-Uni
ivan.kalas@fmph.uniba.sk

La programmation à l'école primaire : De Papert à la nouvelle informatique

Conférence originale en anglais

Ces dernières années, nous avons été témoins d'un regain d'intérêt sans précédent pour la programmation éducative, non seulement à l'école secondaire, mais aussi à l'école primaire et même à des degrés préscolaires. Si la tendance actuelle n'est pas la première de l'« ère numérique », c'est certainement la plus forte, la plus complexe et la plus substantielle de toutes les précédentes. Les « pères fondateurs » qui, dans les années 70 et 80, ont reconnu le potentiel des ordinateurs pour les apprentissages chez les enfants par l'exploration, l'expression et la création (voir Papert, 1980, ainsi que les travaux précurseurs de Feurzeig, Kay, Clayson, Goldenberg et d'autres) ont été récemment propulsés sur le devant de la scène et retravaillés au travers de nouvelles publications d'influence comme, entre autres, Wing (2006), le rapport révolutionnaire de la British Royal Society (2012) et la transformation qui s'en est suivie en septembre 2014 de la traditionnelle « éducation aux TIC » en une nouvelle discipline « Informatique » (*Computing*) au Royaume-Uni[1], soutenue par un ambitieux plan d'études national (voir DfE, 2013).

Actif dans le domaine de la programmation et de l'informatique éducatives depuis le milieu des années 80, j'essaierai dans ma conférence de tirer profit de mes expériences passées et présentes de développement d'environnements logiciels pour enseigner la programmation au sein de mon groupe à l'Université Comenius, de mon récent rôle dans la création de contenus pour l'apprentissage de la programmation dans les écoles

1 Dès l'âge de 5 ans et jusqu'au baccalauréat.

primaires du Royaume-Uni (*UCL ScratchMaths*[2]) et de mon implication à l'UNESCO, à l'IFIP (*International Federation for Information Processing*) et dans le programme *Partners in Learning* de Microsoft. En partant de cette perspective multiple, je poserai la question de savoir si la poussée d'intérêt actuel envers la programmation éducative diffère des précédentes initiatives et, dans l'affirmative, j'interrogerai nos chances de réussir une implantation durable.

Comme nous travaillons avec des concepts parfois peu clairs comme TIC, *computing*, littératie numérique ou informatique[3], je commencerai par définir la programmation éducative en me concentrant exclusivement sur le milieu scolaire, principalement au primaire[4], où toutes les matières ou presque sont traitées par un seul enseignant généraliste. Si cela est parfois vu comme un obstacle à l'enseignement de l'informatique à ces degrés, je me ferai l'avocat du contraire : l'atmosphère et la culture d'apprentissage de l'école primaire offrent un potentiel extraordinaire pour tirer parti de l'informatique et de la programmation au bénéfice des jeunes apprenants – en tant que nouvel et puissant instrument d'exploration, d'apprentissage, de création et de communication.

J'essaierai aussi de formuler un cadre conceptuel pour la programmation éducative au primaire ainsi que d'identifier certains principes qui, je pense, devraient être respectés si l'on souhaite concevoir et mettre en œuvre de manière pérenne des contenus pour l'apprentissage de la programmation ainsi que la pédagogie qui les accompagne. Je mentionnerai enfin les défis et les risques potentiels dans le futur proche. J'illustrerai ceci par trois projets que nous avons développés récemment ou qui sont en cours : les débuts de la programmation avec Thomas le Clown en première primaire, notre dernier projet de construction systématique d'éléments de programmation pour les degrés 3 et 4, et notre cursus ScratchMaths pour les degrés 5 et 6 (de 9 à 11 ans).

Mots clés : programmation au primaire, programmation, informatique, computing, pensée informatique, pertinence de développement

2 Voir <http://ucl.ac.uk/scratchmaths>.

3 En tant que domaine d'études aux degrés primaires et secondaires.

4 Au sens plus large selon le schéma ISCED de l'UNESCO, avec des durées typiques de 4 à 6 voire 7 ans dès l'âge de 5 ou 6 ans.

Références

Benton, L, Saunders, P., Kalaš, I., Hoyles, C., Noss, R. (2017). Designing for learning mathematics through programming : a case study of pupils engaging with place value. To appear in *The International Journal of Child-Computer Interaction*.

Blackwell, A. F. (2002). What is Programming ? In *14th Workshop of the Psychology of Programming Interest Group*. 204–218.

DfE. (2013). *Computing Programmes of Study : Key Stages 1 and 2*, DfE. Available at : <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>.

Gujberova, M., & Kalaš, I. (2013). Designing productive gradations of tasks in primary programming education. In *Proceedings of WiPSCE 2013*, 108–117. ACM Digital Library DOI 10.1145/2532748.2532750.

Kabatova, M., Kalaš, I., Tomcsanyiova, M. (2016). Programming in Primary Slovak Schools. *Olympiads in Informatics*, Vol 10 (2016), 125–158.

Papert, S. (1980). Mindstorms : Children, Computers, and Powerful Ideas. New York : Basic Books, Inc.

The Royal Society. (2012). *Shut down or restart ? The way forward for computing in UK schools.* Available at : <https://royalsociety.org/topics-policy/projects/computing-in-schools/report/>.

Wing, J. (2006). Computational Thinking. *Communication of the ACM*, Vol 49, No. 3, 33–35.

ARTICLES

I.
Au cœur de la discipline Informatique

Yannis Delmas-Rigoutsos

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

Proposition de structuration historique des concepts de la pensée informatique fondamentale[1]

Résumé

Les programmes de 2015 de l'enseignement obligatoire français (MENESR, 2015b) placent celui-ci dans le cadre du Socle commun de connaissances, de compétences et de culture *(MENESR, 2015a), définissant ainsi un objectif pédagogique large, relevant de la culture commune de base du pays. Pour l'informatique, comme pour de nombreuses autres matières, l'approche disciplinaire, savante, n'est guère adaptée à cet objectif, pas plus qu'une approche qui serait seulement opératoire ou instrumentale, limitée, comme par le passé, à des usages des technologies usuelles d'information et de communication (TUIC).*

Qu'est-ce que l'informatique ? *Des auteurs précédents (Papert, 1996 ; Denning, 2003 ; Wing, 2006, 2008 ; Dowek, 2011 ; Hartmann et al., 2012) ont suggéré plusieurs approches et structurations du champ, visant des objectifs différents, toutefois. Pour définir les besoins de l'honnête Homme, il convient de reprendre cette question sur de nouvelles bases.*

En adaptant une définition de Schwill (1993), nous proposons une cartographie de ce que pourraient être les concepts les plus fondamentaux de la pensée informatique, susceptibles de contribuer à la formation générale de l'esprit humain, au même titre que les mathématiques ou les humanités. Pour cela, nous nous appuyons sur l'histoire de l'informatique et du numérique et soulignons également les liens que ces concepts peuvent entretenir avec d'autres savoirs. Ces liens ne sont pas seulement culturels, ils permettent aussi d'entrevoir les liens qui peuvent exister entre disciplines et placer ainsi l'informatique dans un cadre scientifique élargi.

Mots clés : didactique de l'informatique, enseignement scolaire, histoire de l'informatique, pensée informatique

1 Recherche conduite dans le cadre du programme « didactique et apprentissage de l'informatique à l'école » (DALIE) soutenu par l'ANR (14-CE24-0012–01).

1 Introduction

Les technologies numériques sont partout, ambiantes, remodèlent les rapports humains et le rapport aux connaissances. Comme Lawrence Lessig l'a abondamment défendu avec son célèbre « Code is law » (2000), les artefacts numériques constituent des lois pour la société informatisée. Puisqu'il s'agit de vecteurs de contrôle social, de sédimentation de choix, de stratégies et d'idéologies, le Citoyen est face à une alternative simple : « coder ou être codé », autrement dit être éclairé sur le cadre technique qui l'entoure ou se rendre, renoncer à son autorité pour devenir leur sujet. Mais que signifie ce « coder » pour la formation de l'esprit du Citoyen ?

Avec le développement considérable de la science informatique, d'une part, et l'extension de l'enseignement de « l'informatique » en deçà de l'enseignement supérieur, d'autre part, se pose la question pratique de la définition de *curricula*. Comme l'argumente Peter Denning (2003), ancien président de l'Association for computing machinery (ACM)[2], l'approche par domaines de spécialisation universitaire n'est plus adéquate : ils sont une trentaine, chacun avec ses apports fondamentaux, ses idées-forces, etc. À l'extrême inverse, celui-ci ne se montre pas plus satisfait d'approches se limitant aux grandes idées (*great ideas*), trop restrictives, par construction. Il propose donc de poser différemment la question curriculaire, en distinguant quatre niveaux d'approche : 1) les domaines d'application, l'informatique étant largement une discipline ancillaire, 2) les domaines de l'informatique, *core technologies*, qui s'attachent à la conception des différents types de traitements, 3) les principes d'organisation humaine de l'acte de conception et 4) les grands principes du corpus théorique de l'informatique, qu'il nomme « *computing mechanics* » par parallélisme avec la physique. Son objet est bien de définir les contours de ce qu'est épistémologiquement le traitement de l'information (*computation*) – quelle est sa mécanique interne. C'est cette même question que nous posons également ici.

Chaque réponse à cette question, chaque découpage, chaque nomenclature, chaque cartographie est, en soi, légitime ; il reste à en déterminer la pertinence. Or, dire pertinence, c'est souligner que toute

2 L'ACM est la principale société savante de l'informatique, couvrant à la fois ses aspects scientifiques et pédagogiques.

catégorisation relève d'un arbitrage, d'une finalité. Peter Denning (2003) visait essentiellement à structurer les concepts de l'informatique savante. Gilles Dowek (2011), à l'occasion du colloque Didapro de 2011, fit, lui, une proposition visant le secondaire supérieur (en France, le lycée). Pour ce qui nous concerne ici, nous souhaitons viser ce que nous considérons comme le socle fondamental de formation de l'esprit et de culture que devrait maîtriser tout honnête Homme contemporain. Aux termes des instructions officielles pour l'enseignement obligatoire français, ce cadre est défini par le *Socle commun de connaissances, de compétences et de culture* (MENESR, 2015a) et règle les enseignements élémentaire et secondaire inférieur (collège) ; c'est donc ce que nous viserons. Cet horizon, beaucoup plus large, est celui de la culture commune de base du pays. Comme ces programmes, nous nous plaçons en rupture avec leurs prédécesseurs récents, qui ne concevaient le numérique qu'instrumental et opératoire, dans le champ restreint des technologies usuelles d'information et de communication (TUIC).

2 Méthode et travaux antérieurs

2.1 Propositions antérieures

Avant de nous attacher à notre méthode, rappelons quelques propositions antérieures récentes[3].

Jeannette Wing s'attache, avant tout, à défendre l'intégration de la pensée computationnelle (*computational thinking*) au socle intellectuel fondamental (2006) et son insertion curriculaire systématique dans l'enseignement scolaire (2008). Par ce terme de « pensée computationnelle », elle désigne les outils intellectuels qui relèvent d'une compétence fondamentale, au-delà des questions d'artefacts et/ou de routines techniques. Ses principales caractéristiques sont les suivantes :

3 Nous laissons ici de côté l'ouvrage de Seymour Papert (1996), qui demanderait un développement plus conséquent.

- « *The essence of computational thinking is abstraction*. » Ceci inclut : reformuler un problème, p. ex. par réduction, plongement, transformation ou simulation ; utiliser l'abstraction et la décomposition ; s'attacher à l'encapsulation (*separation of concerns*) ; choisir la représentation la plus adaptée pour modéliser un problème ; utiliser des invariants pour décrire le comportement d'un système ; modulariser, pour anticiper différents usages possibles ; précharger et mettre en cache des données en vue d'une utilisation future.
- Ceci inclut également de grandes méthodes de conception de programmes : penser récursivement ; traiter en parallèle ; interpréter le code comme de la donnée ou la donnée comme du code ; penser la vérification de type de données comme une généralisation de l'analyse dimensionnelle [en physique] ; saisir les avantages et inconvénients de donner plusieurs noms à une entité ; mesurer les coûts et bénéfices de l'adressage indirect et de l'appel de procédures. C'est également être capable de percevoir l'esthétique d'un programme.
- « *Because our abstractions are ultimately implemented to work within the constraints of the physical world, we have to worry about edge cases and failure cases*. » Notamment : raisonner en termes de prévention, protection et récupération des pires scénarios au moyen de la redondance, du confinement des dégâts et de la correction d'erreurs ; « *call gridlock deadlock* »[4] ; définir des contrats d'interfaces ; apprendre à éviter les situations de concurrence (*race conditions*) en cas de synchronisation.
- Enfin, c'est utiliser un raisonnement heuristique pour parvenir à une solution ; planifier, apprendre et programmer (*schedule*) face à l'incertitude ; utiliser des techniques de recherche de données ; utiliser de grandes quantités de données pour accélérer un traitement ; arbitrer entre temps et espace, entre puissance de calcul et capacité de stockage.

Peter Denning (2003) distingue au sein de l'informatique (*computing mechanics*) cinq registres (*windows*), chacun centré sur un objectif (*concern*) et s'appuyant sur des histoires (*stories*, ce qu'on peut raconter sur la discipline et que l'on retrouve dans les domaines technologiques). Ces registres de l'informatique savante sont :

4 Cette formulation est intraduisible en français : un *gridlock* est un interblocage dans le contexte de la circulation routière et un *deadlock* un interblocage de plusieurs processus informatiques.

- Le calcul ou traitement, en anglais *computation*, centré sur la question de ce qui peut être calculé, dont : algorithme, structures de contrôle, structures de données, automates, langages, machines de Turing, machines universelles, complexités de Turing et de Chaitin, autoréférence, logique des prédicats, approximations, heuristiques, calculabilité, traductions, réalisations physiques ;
- La communication, transmission de messages d'un point à un autre, dont : transmission de données, entropie de Shannon, encodage sur un médium, capacité d'un canal, réduction du bruit, compression de fichier, cryptographie, réseaux à commutation de paquets reconfigurables, vérification d'erreur de bout en bout ;
- La coordination, coopération de multiples agents, dont : de personne à personne (boucles d'action, *workflows* […]), personne-machine (interface, entrée, sortie, temps de réponse), machine-machine (synchronisation, concurrence, interblocage, sérialisabilité, actions atomiques ;
- Automatisation (*automation*), dont : simulation de tâches cognitives, problématiques philosophiques liées à l'automatisation, ingénierie cognitive (*expertise*) et systèmes experts, amélioration (*enhancement*) de l'intelligence, tests de Turing, apprentissage automatique et récognition, bionique ;
- Mémorisation (*recollection*), dont : hiérarchies de stockage, localité de la référence, mise en cache, espace d'adressage, nommage, partage, emballement, recherche, accès par nom, accès par contenu.

Gilles Dowek (2011), qui s'intéresse au niveau secondaire supérieur (lycée), insiste sur les notions fondamentales suivantes, souvent utilisées de conserve dans toute conception informatique :

- Algorithme, dont : non exclusivité de l'informatique ; algorithmes déterministes ou non ; nécessité de la représentation symbolique ;
- Machines, pour exécuter les algorithmes, dont : leurs limites physiques ; ordinateur, réseau, machines parallèles et systèmes embarqués ; concept de machine universelle ;
- Langage, dont : langages pour écrire des algorithmes ; langages de programmation ; antériorité historique par rapport à l'ordinateur ; langage au sens général : langues naturelles et formelles ;
- Information, dont : représentation symbolique ; « numérisation » qui serait mieux désignée comme une mise en symboles ; codage, compression et chiffrement ; archivage et restitution.

Gilles Dowek « insiste sur l'importance de respecter l'équilibre entre ces différents concepts dans la conception d'un programme d'enseignement de l'informatique au lycée ».

En ne citant que ces seuls auteurs, nous constatons déjà une grande diversité de choix. Nous constatons également combien certaines notions semblent tout à fait inaccessibles à un niveau d'enseignement élémentaire ou secondaire inférieur, faute d'un arrière-plan intellectuel, et d'une capacité d'abstraction, suffisants. Nous allons maintenant nous efforcer de structurer un ensemble d'idées fondamentales, accessibles dès l'enseignement primaire ou secondaire inférieur (collège) et suffisamment articulées entre elles pour permettre aux élèves de se forger une représentation globale. Nous ne prétendons pas à l'exhaustivité, mais le cadre que nous proposons demandera déjà un effort de formation des enseignants concernés.

2.2 Ce qu'est une idée fondamentale

Jerome Bruner propose dans son précis d'éducation (1960) une notion très générale de thèse fondamentale. Cette notion est systématisée et adaptée à l'informatique par Andreas Schwill (1993), pour qui une idée est fondamentale, si elle respecte quatre grands critères, d'horizontalité, de verticalité, de pérennité et de sens. Hartmann *et al*. (2012) ajoutent un critère pédagogique que nous intégrerons aux critères de verticalité et de sens, que nous élargirons. Nous définissons ici une *idée fondamentale* par trois critères :

- *Extension* : elle apparaît sous différentes formes dans au moins deux domaines (critère d'horizontalité) et peut être décrite à un niveau élémentaire comme avancé (critère de verticalité) ;
- *Généralité* : elle est pertinente pour s'appliquer ou se transférer à des situations réelles, matérielles ou intellectuelles hors du seul champ de l'informatique (élargissement du critère de sens) ;
- *Influence* : son histoire montre une influence durable sur les sciences du numérique (critère de pérennité) ; elle a été féconde dans un autre domaine du savoir ou présente un enjeu sociétal particulier.

Une idée fondamentale doit se retrouver, à un degré ou à un autre, dans la plupart des dispositifs informatiques. Aucune situation écologique n'est

donc pure. Ceci n'est pas nécessairement le cas des situations pédagogiques, qui peuvent aussi bien être pures, ou réalistes, ou relier telle idée avec tel autre champ du savoir, ou encore transférer une idée issue de l'informatique à une situation qui n'en relève pas.

2.3 Organisation historique

Ces idées fondamentales ne sont pas toujours marquées historiquement. Elles prolongent parfois des réflexions, voire des théories, antérieures à l'informatique elle-même. Nous organiserons cependant notre propos en suivant trois grandes périodes de l'histoire de l'informatique en reprenant la périodisation de David Fayon (2010), dans le champ économique, que nous étendons à d'autres aspects (*cf.* Delmas-Rigoutsos, 2014). Nous souhaitons ainsi seulement faciliter la compréhension de leurs relations.

Tableau 1 : Grandes périodes de l'histoire de l'informatique.

	Années 1930–1940	**Années 1940–1950**	**1947–1980**	**1974–2005**	**1999–**
Mouvement dominant	logique mathématique	pionniers	première industrie	loi de Moore	numérique ambiant
Artefacts dominants	modèles de calcul (dont la machine de Turing)	quelques *computers*	dizaines de milliers d'ordinateurs centralisés	dizaines de millions d'ordinateurs personnels	dizaine(s) de milliards d'objets connectés
Usages cibles	démonstration de théorèmes	calcul militaire	recherche, grands fichiers admin.	TIC : bureautique et communication	numérique embarqué, pervasif
Principal développement	théorie	ingénierie	machines	logiciels (dont IHM)	contenus (données)
Leader	universités	armée états-unienne	IBM	Microsoft	Alphabet (Google)

Évoquons donc successivement trois idées fondamentales liées aux machines (section 3), trois liées aux algorithmes et programmes (section 4) et trois liées aux données (section 5).

3 Les machines autonomes et leurs réseaux

3.1 Autonomie, automatique et automatisation

Généralité On caractérise souvent l'Homme comme l'espèce qui utilise extensivement des outils. Les anthropologues, notamment André Leroy-Gourhan, ont montré combien l'Homme était lui-même modelé par ses outils, concevant généralement ceux-ci comme des prolongements de la main, d'abord au sens propre, puis au figuré. Avec le numérique, une nouvelle classe de machines se détache de la main pour être *autonome*, c'est-à-dire *fonctionner de façon réactive, mais non contrôlée* (parfois supervisée, cependant). Ce concept fondamental de fonctionnement autonome s'applique à de nombreux registres techniques : certains pièges de chasse préhistoriques, automates à eau de la civilisation arabe, automates à mécanisme d'horlogerie du 18e s. européen. Son application fut aussi non technique, ainsi Jacques de Vaucanson utilisa-t-il, au 18e s., des automates mécaniques pour faire progresser l'anatomie. En philosophie, l'influence de l'automatique fut considérable : « *À la vision médiévale d'un ordre hiérarchique d'êtres vivants créés et gouvernés par Dieu, la pensée du 17e s. opposait celle d'un ‹monde machine›, dénué de fin et de volonté, dont toutes les composantes, animaux et humains compris, étaient dirigées par les lois de la physique* » (Heudin, 2008).

Cette idée fondamentale n'est pas évoquée directement par les programmes (MENESR, 2015b). Elle mériterait pourtant notre attention, tant les représentations premières des élèves peuvent être erronées. Ainsi le fonctionnement autonome peut être confondu avec la vie (dont il partage quelques attributs), d'une façon qui peut rappeler les mots de Voltaire dans son cinquième discours en vers sur l'Homme : « *Le hardi Vaucanson, rival de Prométhée, | Semblait, de la nature imitant les ressorts, | Prendre le feu des cieux pour animer les corps* ». Il convient, de même, de bien différencier l'autonomie des notions de volonté propre ou d'intelligence.

Influence et extension horizontale Par définition, le fonctionnement automatique couvre toute l'informatique, puisque celle-ci se définit comme la science du traitement automatique de l'information. Cela fait partie de l'esprit même de la discipline, technique comme théorique, que de s'efforcer, dans une conception, de faire le moins possible appel à un contrôle

extérieur. Cette culture n'est pas sans alimenter d'ailleurs les peurs sociales du remplacement de l'Homme par « la machine » du fait de l'automatisation, qui prennent la suite des mêmes peurs de remplacement du fait de la mécanisation. Les œuvres de fiction d'anticipation abondent en réflexions sur ce thème. Dans le champ prospectif, le débat est actuellement intense de savoir si nos sociétés vont faire le choix de l'automatisation ou de l'instrumentation (*to automate or to informate*). Les avancées importantes actuelles de l'Intelligence artificielle, réelles ou publicitaires, posent cette question avec toujours plus d'acuité.

Extension L'idée fondamentale du fonctionnement autonome recouvre, en informatique, de nombreuses notions. Au sens strict, il peut s'agir d'un automate informatique, qui peut effectuer un traitement lui-même ou le faire effectuer par différents organes, comme dans une machine de Turing ou au cœur d'un processeur conçu selon l'architecture de von Neumann. À un haut niveau conceptuel, on peut aller jusqu'aux réflexions sur la nature même de ce qui constitue un traitement, un calcul, réflexions importantes en logique et informatique fondamentale depuis environ un siècle, dont les résultats les plus emblématiques étayent la thèse de Church et la correspondance de Curry-Howard (*cf.* ci-après, § 5.3).

À un niveau élémentaire, il nous semble important, au minimum, de faire comprendre les principes d'organisation des machines automatiques les plus courantes ainsi que leur dépendance à un programme de commande, afin d'écarter les interprétations magiques ou vitalistes et les lectures en termes de volonté ou d'intelligence. Les programmes de 2015 le prévoient explicitement pour le cycle 4 (C4[5]), mais ceci est accessible et pertinent dès la confrontation à des robots pédagogiques (C2/C3). Il est intéressant, à ce propos, d'observer la disjonction entre l'imaginaire, dans lequel les robots sont souvent anthropomorphes, images en miroir de l'Homme, de l'ouvrier ou de l'esclave, parfois de l'animal (*Electric dog*, 1912 ; *RUR*, 1922 ; *Metropolis*, 1927 ; etc.), et les réalisations techniques effectives. *Unimate* (1961), le premier robot industriel est seulement un bras : le corps se limite à l'instanciation de sa fonction utile (pensons de même aux automates téléphoniques, de paiement, etc.).

5 Nous abrégeons ici « C2 » le cycle des apprentissages fondamentaux (cycle 2 français : CP, CE1, CE2), « C3 » le cycle de consolidation (cycle 3 : CM1, CM2, 6e) et « C4 » le cycle des approfondissements (cycle 4 : 5e, 4e, 3e).

3.2 Cybernétique et interface : les flux d'information

Généralité et influence On conçoit souvent le structuralisme comme un mouvement intellectuel des sciences humaines et sociales. Il a, en réalité, été également fécond en sciences avec, notamment, les mathématiques de Bourbaki et la cybernétique de Norbert Wiener, qui aura eu une influence sur plusieurs sciences, dont l'informatique, les sciences de l'information et de la communication et les sciences cognitives. L'idée fondamentale de la cybernétique est que le *comportement de communication*, c'est-à-dire son comportement en termes de *flux d'information* (C4), est plus utile pour comprendre un système que sa nature physique, qu'il s'agisse de sociétés, d'êtres vivants ou de machines. Cette théorie a permis de repenser complètement les relations de commande et de contrôle (C4), en particulier dans le cas des robots (C2/C3).

Extension Dans le domaine informatique, l'idée du comportement de communication se traduit principalement par la notion centrale d'interface : entrées/sorties d'un traitement, interfaces de programmation et, surtout, interfaces Homme-machine (IHM, C4). Présente dès la première industrie, la question des IHM, a pris une ampleur considérable depuis que les artefacts appelés commercialement « ordinateurs » ne constituent plus qu'une petite fraction des objets numériques. La plupart de ceux-ci sont embarqués et connectés ; leurs interfaces, d'une part avec l'Homme, d'autre part avec le réseau, sont donc primordiales pour leur fonction. L'ordinateur, *lato sensu*, est une machine très générique de contrôle, il a donc pris, avec les années, une place considérable dans le contrôle de très nombreux dispositifs, y compris non numériques, favorisant, de ce fait, un mouvement d'intégration des dispositifs de commande. Les questions sociétales deviennent prégnantes : faut-il penser les dispositifs portés comme des prothèses, des prolongements ou des augmentations de l'Homme (au sens de Douglas Engelbart) ? Quelle trajectoire technologique choisissons-nous pour notre société : celui d'interfaces « faciles » qui prennent de nombreuses décisions pour l'utilisateur sans l'en informer, ou celui d'interfaces riches qui demandent à l'Homme des efforts et le font coévoluer avec elles ?

3.3 Les systèmes distribués intégrateurs

Généralité Norbert Wiener prédit très tôt que les ordinateurs allaient devenir des machines à communiquer et qu'ils joueraient un rôle important dans la société. Depuis cette année (2017), on estime qu'il y a plus d'objets connectés à Internet que d'être humains sur Terre. Ceux-ci confèrent désormais à de nombreux aspects de notre vie quotidienne une épaisseur numérique. Dans certains domaines, en particulier la communication, la médiation numérique est même devenue largement dominante : qui n'a pas constaté combien les adolescents pouvaient échanger par minimessages plutôt qu'oralement, y compris à quelques mètres de distance ? Le phénomène est également très présent dans certaines professions pourtant sédentaires. Certains analysent l'évolution de notre société en « société de communication ». Cette omniprésence de l'intercommunication soulève de nombreuses questions ; parmi celles-ci, retenons ici la notion fondamentale de *système distribué : un ensemble composé de nombreux agents*, pas nécessairement localisés au même endroit, *qui bénéficie d'un fonctionnement intégré*, c'est-à-dire que l'on peut percevoir comme celui d'un agent indivis. De nombreux systèmes non informatiques, tels que des entreprises, ou plus généralement des personnes morales, ont les caractéristiques de systèmes distribués. Nous nous intéresserons ici surtout aux agents informatiques en réseau, tels que les principaux outils en ligne (C3/C4) ou les objets connectés (C4).

Extension Dès le début les ordinateurs ont été conçus comme un concours de plusieurs éléments : l'architecture de von Neumann fait ainsi collaborer une unité de contrôle, une unité de mémoire et une unité arithmétique et logique. Les grands calculateurs qui ont précédé les ordinateurs, y compris la machine de Babbage, utilisaient aussi des transports de données entre sous-parties. En 1940, un congrès de l'American mathematical society à Hanover (NH, USA) connecta même plusieurs terminaux à un calculateur arithmétique localisé à New York (NY, USA). Dès le début, l'ordinateur fut donc un assemblage d'éléments centraux et d'éléments périphériques qui communiquaient entre eux. Au cours de la période de la loi de Moore, le nombre de processeurs centraux augmenta : le processeur principal se vit épaulé d'un coprocesseur, qui intégra ensuite la même puce, de processeurs de sons et de traitement de signaux analogiques, de processeurs spécialisés dans l'affichage vidéo, dans la gestion de mémoires de masse, etc.. Puis, les

machines deviennent parallèles ; le nombre de processeurs « principaux » augmenta, « le » microprocesseur intégrant plusieurs files de calcul, puis plusieurs cœurs. La période actuelle voit, elle, l'ordinateur se développer à travers le réseau, réseau local, puis Internet : d'abord certains périphériques, puis certains services de stockage puis de traitement. Aujourd'hui, à l'heure du « cloud », il n'est pas rare de voir certains services d'application fournis par une architecture N-tiers incluants parfois des dizaines de machines serveuses. Beaucoup de ces services (SaaS, PaaS, IaaS) sont conçus pour que l'infrastructure soit invisible à l'utilisateur, permettant ainsi une grande souplesse de mise en œuvre et, en particulier, l'extensibilité (*scalability*). Le principe de fonctionnement des systèmes distribués nécessite une forme de transparence au réseau ; pour le web, par exemple, ceci s'appuie sur un système d'adressage universel (*uniform*).

Influence Il nous semble essentiel de ne pas se limiter, comme les programmes actuels, au web et outils associés (C2), aux systèmes d'information distribués (C3) ou à la compréhension des arborescences (C4). Il nous semble important de comprendre qu'un système distribué est constitué de plusieurs agents logiciels qui interagissent, qu'il occasionne donc de nombreuses communications entre machines : des centaines d'échanges de données entre plusieurs dizaines de machines pour une simple consultation du web. En effet, les systèmes distribués ont vocation à prendre une ampleur dépassant de beaucoup le cas du web dans le cadre de l'*informatique pervasive*. Ce paradigme a été conçu par Mark Weiser, d'abord comme *informatique ubiquitaire* (1991), imaginant de nouveaux types d'appareils omniprésents mais invisibles, « disparaissant dans l'arrière-plan », s'intégrant de façon « transparente » à notre environnement. L'ordinateur serait alors remplacé par une multitude d'objets interconnectés. Cette informatique ambiante, ubiquitaire, qui est un tissu de systèmes distribués, est en train d'advenir, avec, comme interface principale le « téléphone mobile » individuel. Comme tout système distribué, comme le web avant elle, elle est, par nature, par conception, un *système intégrateur*, c'est-à-dire qui a vocation à incorporer de nombreuses fonctions auparavant assurées par d'autres dispositifs. C'est ce qui a fait la faveur du web, intégrant l'accès à de nombreux contenus ainsi qu'à quantité d'applications. Pour beaucoup, les systèmes qui sont ainsi intégrés s'étiolent, voire disparaissent, dans leur version non intégrée. Dans le cas du web, ceci a profondément transformé notre rapport à l'information d'actualité et au savoir. Il faut s'attendre à ce

que l'*Internet des objets* renouvelle aussi profondément le rapport à notre environnement, ou en tout cas à la part de celui-ci qui sera intégrée ou médiatisée par l'informatique pervasive.

4 De la conception des algorithmes à la rédaction des programmes

4.1 L'algorithme, procédure systématique, abstraite et finalisée

Influence Les inventions les plus fondamentales sont invisibles, seule la confrontation à leur absence nous les rend présentes à l'esprit. Là est la difficulté de conception des enseignements les plus fondamentaux : produire une évolution dans l'esprit des élèves sur une notion que l'enseignant ne perçoit pas sans effort conscient. Difficulté supplémentaire pour nombre d'outils numériques : il s'agit de technologies cognitives, comme l'écriture ou l'imprimerie avant elles, et donc complètement intriquées dans la plupart des activités humaines. L'objet intellectuel algorithme en est certainement le plus parfait exemple, tant il est présent, silencieusement, dans nombre d'activités. Son histoire remonte au moins à l'Antiquité grecque. Sciences et philosophies y étaient alors fondées sur le discours, le logos (*λόγος*, toujours très présent dans notre héritage, comme l'atteste le nom en « -logie » de nombre de disciplines scientifiques ou prétendant l'être) ainsi que sur l'enquête descriptive, l'historia (*ἱστορία*, histoire politique, histoire naturelle, etc.) ; pour autant cette culture conçut également des procédés systématiques permettant de résoudre certains problèmes, tels que l'algorithme d'Euclide pour la détermination du PGCD. Les procédés de ce type se développèrent surtout, pour l'aire occidentale, dans le cadre de la science arabe médiévale. Le mot « algorithme » dérive d'ailleurs du nom d'Al-Khawârizmî, savant perse du XVIII^{e}–IX^{e} siècle, qui rédigea plusieurs traités, dont des recueils de procédés de calcul. À cette époque, on est encore dans le domaine des outils intellectuels ; la confrontation systématique au monde, l'expérimentation, la sortie de la Caverne, ne se fera que plus tard, sous l'impulsion des Galilée, Newton et Bacon. L'*algorithme*, que nous pouvons définir comme une *procédure formalisée et systématique résolvant un problème posé dans un cadre formel*, outille l'Homme bien

avant le premier ordinateur. Il est présent aussi bien en mathématiques, en chimie ou en ingénierie, par exemple.

Généralité Pour ce qui est de l'adéquation aux niveaux élémentaires, adoptons une définition scolaire plus simple que la définition savante : *procédure systématique, abstraite et finalisée* (c. à d. visant un objectif donné). Sous cette formulation, c'est déjà une compétence à développer au C2 : l'élève doit apprendre à ne pas présupposer que l'autre sait déjà. Il doit ainsi être capable d'indiquer son chemin à un passant ; il doit pouvoir décrire une scène ou un dessin à un camarade de façon à ce que celui-ci puisse la reproduire ; etc. Cette compétence peut s'instancier dans de nombreuses situations de jeu de rôle pédagogique ou de la vie courante.

Moyennant notre définition adaptée, la recette de cuisine n'est plus seulement une métaphore d'algorithme, mais un algorithme à proprement parler. Sa mise en œuvre permet de faire le lien entre la procédure écrite, abstraite, et le processus effectif qui l'actualise. Cet exemple permet d'ailleurs de voir que l'algorithme de l'école primaire n'est pas nécessairement linéaire ou circulaire, à l'image des orgues de barbarie ou des métiers Jacquard, il peut également introduire des conditionnels, des options, des paramètres, des itérations. Ces aspects pourront être ensuite formalisés au C4.

Extension Pour compléter, notons avec Baron & Bruillard (2001) combien l'algorithmique, la science des algorithmes, est essentielle dans la construction de la science informatique.

4.2 Le programme, implémentation concrète d'un algorithme

L'algorithme est la compréhension d'un processus, tandis que le codage sous forme d'un programme est la rédaction d'un texte qui permet de lever certaines ambiguïtés de la conception (toutes, à l'idéal) et de faire exécuter les commandes par un agent (humain ou artefact). Cette rédaction comporte de nombreuses dimensions ; quelle(s) notion(s) de programmation retenir comme fondamentale(s) ?

Influence Le premier concept, qui nous semble le plus fondamental après celui de l'algorithme, est celui de *programme enregistré*. Pour la plupart des historiens de l'informatique, c'est souvent la présence d'un programme enregistré qui sert à poser l'acte de naissance de l'ordinateur – tous ne

s'accordent pas, d'ailleurs, sur les autres critères[6]. On peut faire remonter cette idée, au moins, aux automaticiens Jaquet-Droz du 18e s. (*cf.* Heudin, 2008). Ceux-ci, parmi les plus remarquables fabricants d'automates à mécanismes d'horlogerie de leur époque, créèrent un automate écrivain permettant de produire « n'importe quel texte comportant un maximum de 40 lettres ou signes », de même qu'un dessinateur et une musicienne. Ces trois automates étaient programmables au sens où les mouvements d'horlogerie étaient codés par des ergots sur un cylindre amovible. Cela peut sembler anecdotique rétrospectivement, mais le fait que ce cylindre soit amovible constitue une rupture majeure. Cela change toute la perspective : toutes les lettres, y compris celles qui ne sont pas dans le premier texte, doivent être codées. La programmation par cylindre et ergots, qui auparavant activait directement des actionneurs, à la façon d'un langage-machine, devenait un langage de plus haut niveau, un véritable langage de communication. Le célèbre métier à tisser Jacquard, au début du 19e s., utilisa la même idée pour enregistrer un programme de tissage sur une bande de carton. Ce métier inspira plus tard les machines à calculer à mécanisme d'horlogerie de Charles Babbage, dont l'analyse par Ada Lovelace conduisit à la première formalisation du concept d'algorithme et au premier programme de calcul jamais écrit.

Extension Du point de vue pédagogique, l'idée fondamentale de programme enregistré permet de développer plusieurs idées, à partir du cycle 3. Tout d'abord, cela permet de bien distinguer entre ce qu'est un algorithme et ses *implémentations* sous la forme de programmes. Les programmes eux-mêmes, qui sont des textes, sont, comme tous les textes, dans tous les langages, encore des virtualités qui attendent d'être actualisées sous la forme d'une interprétation, d'une exécution. Ceci permet d'ancrer les notions d'*instruction* et de *programmation impérative*, très tôt, montrant par l'exemple que « la mécanisation est possible du fait de […] la précision de la notation » (Wing, 2008). À un niveau plus avancé, C4, cela peut être l'occasion de faire toucher du doigt les dimensions documentaire et effective des langages informatiques ; citons (Lévy, 1992) :

> Comme le dit Daniel Bougnoux : « Le poète ne se contente pas de donner à entendre ou à voir, il fait exister. Le poème ne signifie pas, il est. » […] De là une confusion

6 Certains voient ainsi le *Zuse 3* comme premier ordinateur, tandis que d'autres attribuent ce mérite aux premiers ordinateurs utilisant l'architecture de von Neumann.

> féconde entre le signe et la chose : « En poésie, les mots fonctionnent comme choses parmi les choses plutôt que comme signes. » | On voit ce qui rapproche le poème du logiciel. Au fur et à mesure que l'on s'éloigne des interfaces et langages tournés vers l'utilisateur humain et qu'on s'approche des rouages de la machine, le logiciel abandonne le caractère fantomatique et subordonné des signes pour acquérir la persistance et la densité des choses. | Plus on descend vers le langage-machine et plus la fonction référentielle du langage semble s'effacer […] | Telle est l'ambivalence fondamentale du logiciel : il peut être tour à tour symbole et agent, effectif et descriptif.

Généralité Autour du concept fondamental de programme enregistré, nous avons choisi de décliner la séparation entre code et mécanisme, la notion de langage de programmation impératif, la démarche d'implémentation et une première approche de la dimension documentaire[7]. Ce choix permet de développer une réflexion plus générale sur le codage, des langues naturelles comme des langages informatiques, tant pour ses dimensions lexicales, syntaxiques que, dans une certaine mesure, conatives, notions essentielles des cycles 2 à 4, indépendamment de l'informatique. La dimension sémantique semble, en revanche, difficile à aborder dans le cadre de l'enseignement obligatoire, à ce jour.

Cette dimension linguistique est très ancrée dans l'esprit humain, ou dans la culture contemporaine ; c'est peut-être une des sources d'un phénomène qui ne laisse pas d'émerveiller tout enseignant d'informatique : les cris de joie très spontanés des élèves quand s'exécute, et fonctionne, leur premier programme. Ce phénomène est tellement courant qu'il a conduit à une forme de rite d'initiation : écrire un premier programme affichant « Bonjour, tout le monde ! » (« Hello, world », en anglais).

4.3 La modularité, élégance conceptuelle et pragmatique des programmes

Généralité et influence Terminons ce second volet par une troisième notion fondamentale, la *modularité*, qui n'apparaît dans les programmes de 2015 que sous la forme de la décomposition d'un problème en sous-problèmes (C4). Elle est pourtant, à nos yeux, la plus importante, car elle apporte à

7 D'autres formes de programmation existent, au-delà de la programmation impérative ; nous les écartons ici (cependant, *cf.* § 5.3), dans la mesure où elles font appel à des conceptions, mathématiques notamment, qui ne sont pas en place au niveau auquel nous nous plaçons dans cet article.

la fois une dimension esthétique (certes peu accessible aux profanes) et un outillage intellectuel très général, dont l'intérêt dépasse très largement l'informatique : philosophie, ingénierie, gestion de projets, mathématiques, etc.

Extension Historiquement, le premier grand pas dans la théorisation de la modularité, en 1968, est l'article « Go To Statement Considered Harmful », où Edsger Dijkstra défend la nécessité de la programmation structurée. Le principe de modularité comportera par la suite de nombreux autres avatars, notamment l'encapsulation (*separation of concerns*) et la programmation orientée objet. C'est encore la modularité qui fait que dans une page web on s'efforce de sépare le contenu (HTML), sa mise en forme (CSS) et l'interaction (ECMAScript). À un niveau plus élémentaire, c'est le fait d'utiliser systématiquement des feuilles de style dans un traitement de texte. La *modularité* c'est une *démarche d'abstraction de situations concrètes afin d'en percevoir la généricité et*, celle-ci identifiée, *d'affecter des traitements spécifiques à des procédures spécifiques*.

On le voit, l'enseignement de démarches relevant de la modularité a tout à fait sa place dans l'enseignement obligatoire. Il est même possible de l'aborder dès le C3 à l'aide de logiciels comme *Scratch*, ou même *ScratchJr*, qui permettent de travailler avec plusieurs agents simultanés (les lutins), attribuant à chacun un programme spécifique réagissant à certains sémaphores ou messages. Cela peut également être un moyen de montrer l'importance de la dimension documentaire que nous évoquions précédemment. Comme pour les langues naturelles, plus la maîtrise de la langue est limitée, ou plus le sujet est complexe, et plus il convient de bien structurer un texte, de bien l'expliciter.

5 L'information et son codage

La troisième grande époque du développement de l'informatique, en cours, est celle des contenus, des données, de l'information. L'informatique est la science du « traitement rationnel de l'information, notamment au moyen de l'ordinateur » ; le bit est l'« unité fondamentale de quantité d'information » ; notre époque serait celle de la « société de l'information » ; mais qu'est-ce que l'*information* ? En science informatique ou en sciences de

l'information et de la communication, le sens est clair, défini par de nombreux théoriciens à la suite de Claude Shannon, depuis le milieu du 20e s. En revanche, la scolarité obligatoire n'introduit aucune de ces théories, nous laissant, au mieux, avec une notion protéiforme dont les différents aspects ne sont pas conceptuellement reliés et, au pire, avec un mot très polysémique. Qui plus est, l'« information », au sens de l'informatique, serait certainement occultée par l'« information » au sens de l'éducation aux médias et à l'information (EMI), qui, elle, occupe une place importante dans les programmes. Nous proposons de ne pas conserver ce mot pour l'enseignement obligatoire, et de lui préférer la notion plus accessible de « donnée » : on parlera ainsi de « traitement de données » et de « quantité de données ». Sur le fond, il nous semble essentiel d'aborder les trois idées fondamentales suivantes, du codage, de la structuration et du typage.

5.1 Principe de codage : « un bit est un bit »

Généralité Notre première idée fondamentale de ce domaine est ce que nous appellerons le *principe de codage : une donnée est un signe plus un code* (au sens d'encodage). Son principe, l'arbitraire du signe, remonte aux origines de l'écriture alphabétique elle-même : le signe ou l'assemblage de signes (le signifiant) qui désigne un sens (le signifié) n'a pas de nécessité intrinsèque ; il relève, pour l'essentiel de choix et de déterminismes historiques. Même si ce principe n'a été étudié théoriquement qu'à la suite des travaux de Ferdinand de Saussure (19e s.), le phénomène est connu depuis l'Antiquité, celui-ci permettant de composer des codes secrets, dits de substitution, un signe remplaçant une lettre, un groupe de lettre ou un mot. Francis Bacon (16e–17e s.) inventa, sur cette base, le premier code binaire en représentant chaque lettre par une suite de ce que nous appelons aujourd'hui des bits. Le grand pas suivant est réalisé en 1936 par Alan Turing, qui conçoit, pour résoudre une question mathématique, un modèle de calcul, que nous appelons aujourd'hui machine de Turing, en codant tout calcul par une suite de symboles tirés d'un alphabet fini.

Influence Ce modèle se traduira en artefacts matériels grâce à l'architecture de von Neumann, dont relèvent encore tous les ordinateurs aujourd'hui (certes sous une forme très évoluée). L'idée de Turing, reprise par von Neumann, et développée ensuite, est qu'une mémoire informatique stocke

toutes natures de données dans les mêmes bits. Une même série de cases mémoires peut coder une partie de programme, une partie de texte, une partie de vidéo ou toute autre donnée. Conséquence conceptuelle, les bits ne portent pas d'information en eux-mêmes, il est nécessaire de spécifier un codage pour que les données prennent sens. Conséquence pratique : une même mémoire informatique, un même fichier, un même canal de transmission, un même logiciel peuvent stocker, contenir, véhiculer et traiter de façon indifférente tous types de données. Dès le début, le multimédia, les documents interactifs et la convergence numérique étaient déjà en germe. Nicholas Negroponte résume cela par sa célèbre formule « un bit est un bit ».

Extension L'extension horizontale ne faisant guère de doute, évoquons seulement l'extension verticale. Le programme de 2015 (MENESR, 2015b) couvre bien les éléments pratiques du codage, évoquant la notion de signal (C3/C4), la représentation binaire (C3) et le bit (C4), les mémoires informatiques (C2) et leurs ordres de grandeur de capacité (C3/C4), la quantité de données (« d'information », C3/C4), les documents multimédias (C3). Les aspects conceptuels eux, relèvent plutôt de l'enseignement de français : arbitraire du signe linguistique (C2), chiffrement par substitution (C4). Il nous semble important de pouvoir faire le lien entre les deux. Le principe de codage, qui est fondamentalement un principe logique, dit que *toute information enregistrée numériquement est symbolique*, qu'il s'agisse de micro-symboles, comme des pixels, ou de macro-symboles, comme des mots. Ceci permet également de mettre en perspective les technologies cognitives, des tablettes argiles sumériennes aux fichiers numériques actuels. C'est pour cette raison que nous préférons dire « numérique », même si les signes ne sont pas seulement des nombres, plutôt qu'« informatique » ou, pire, « *computer science* ».

5.2 « Algorithmes + structures de données = programmes »

Extension Deuxième avancée majeure : la compréhension de l'importance de la représentation des données dans la conception des logiciels. Dans ce domaine, le célèbre ouvrage de Niklaus Wirth, *Algorithms + Data Structures = Programs*, publié en 1976, fait figure de *credo*. Il ne s'agit pas là seulement d'une question pratique de codage des contenus

(C2) ou de fichiers (C2, C4), dans le prolongement du principe de codage (§ 5.1), mais d'une seconde idée fondamentale : *toute donnée nécessite un référentiel* (techniquement un *type*). Il s'agit d'un cadre conceptuel qui permet de lui donner une signification, opératoire notamment. Prenons un exemple, le plus courant : l'immense majorité des données (sons, images fixes, vidéos, etc.) sont acquises par numérisation, en combinant un *échantillonnage* et une *quantification* qui produisent des données brutes, qui seront ensuite structurées de façon pertinente en données élaborées, lesquelles, enfin, seront codées à proprement parler pour être stockées. Échantillonner demande une représentation de l'espace et/ou du temps et la quantification une représentation de la grandeur à enregistrer, notions qui peuvent être reliées au champ du corps et de l'espace (position relative, latéralisation, etc.), avec les mathématiques (repère cartésien), avec les sciences de la nature (données expérimentales). De même, l'élaboration des données sonores ou visuelles, par exemple, s'appuie sur une représentation de l'ouïe ou la vue humaines. Cette réflexion, omniprésente en informatique fondamentale, a créé, dans le domaine appliqué, nombre de professions : spécialistes de bases de données, de systèmes experts, d'ontologies, *data scientists*, architectes de l'information, etc., toutes spécialités dont l'activité essentielle consiste à trouver la représentation la plus pertinente d'un corpus de données. La place considérable prise par les données et leur exploitation dans la période actuelle fait de cette question un enjeu économique considérable.

Influence Pour mesurer l'influence de cette idée fondamentale, évoquons seulement le cas des sciences humaines et sociales, dont certaines approches ont été profondément renouvelées depuis les années 1990 par l'emploi de bases de données et de traitement de corpus étendus, construisant peu à peu le champ maintenant désigné par le terme « humanités numériques ». Il ne s'agit pas que d'outils technologiques : il s'agit aussi d'outils conceptuels fondés précisément sur une réflexion sur la structuration des données. Laissons la parole à Peter Denley, cofondateur et premier Secrétaire général de l'Association for history and computing, qui explique (1994, p. 34) que comprendre le principe d'organisation d'une base de données relationnelle serait, pour tous les historiens, « *a considerable service to their understanding of structured systems of any kind* ».

Généralité Un tel niveau informatique n'est pas aujourd'hui accessible à l'enseignement obligatoire. En revanche, ce principe est loin de se limiter à l'informatique. Citons, en mathématiques, le repère cartésien et l'organisation des données (C3), ou, plus généralement, la question du repérage du corps (latéralisation, p.ex.) ou dans l'espace (C2). En physique, la question de la relativité galiléenne peut être abordée dès le C4. En arts graphiques, la pratique du *pixel art* ou de la mosaïque à la façon de l'artiste français Invader, outre leur intérêt artistique, permettent de faire le lien entre histoire de l'art et pratiques artistiques explicitement numériques. Elle permet, de plus, d'aborder conceptuellement la différence entre un document nativement numérique et un document issu d'une numérisation.

5.3 Types de données : démontrer, c'est programmer

Extension La troisième idée fondamentale du domaine des données est, selon nous, la *correspondance de Curry-Howard*, ainsi que ses développements dans les années 1980 et suivantes et leurs applications dans le domaine de la preuve de programmes en génie logiciel. Cette correspondance fait le lien entre démonstrations logiques et exécution de programmes informatiques, *via* les types de données, et permet de les considérer, d'une certaine manière, comme deux faces d'une même pièce, deux façons de présenter le même phénomène de calcul. Cette correspondance permet aussi de fonder théoriquement la dimension documentaire d'un programme, le fait qu'un programme ne doive pas être considéré seulement comme une suite d'instructions, mais aussi comme un texte qui a des lecteurs humains. Il doit donc avoir un sens. Un programme correctement rédigé doit comporter, outre les instructions, un commentaire décrivant, voire expliquant, les algorithmes et structures de données utilisés ainsi qu'une spécification, un typage, pour chaque traitement de ses entrées, sorties et variables, et, enfin, un guide ou manuel d'usage du logiciel. L'exemple le plus abouti, en la matière, est probablement le programme *Tεχ* de Donald Knuth, qui intègre à la fois des instructions permettant de produire un logiciel exécutable et la documentation de ce même logiciel. Qui plus est, cette documentation peut être mise en forme par le logiciel *Tεχ* lui-même. Une tradition des années 1990, de façon peut-être un peu désobligeante, prétendait que ce programme serait le seul dénué de bugs.

Généralité et influence À un niveau plus élémentaire encore, l'idée fondamentale de la correspondance de Curry-Howard nous enseigne que tout calcul devrait se concevoir comme typé. Ne voyons pas là seulement une norme opératoire ; ce principe a également un intérêt didactique et heuristique. Dans le domaine de la programmation, il est aisé de le mettre en œuvre de façon élémentaire. Ainsi le langage visuel *Scratch* code-t-il les (nombres) par des ovales, les [chaînes de caractères] par des rectangles et les <booléens> par des hexagones. En physique, ce même principe correspond à l'analyse dimensionnelle (équations aux dimensions), très féconde historiquement. À l'école élémentaire, sans même parler du secondaire (voire de certains étudiants du supérieur), nous sommes toujours frappés de constater combien d'élèves s'évertuent à combiner des grandeurs incommensurables telles que cargaison et âge du capitaine : ils se lancent dans des calculs sans réflexion sur leur sens, et exécutent le calcul avant d'avoir pris la peine de l'écrire. En mathématiques comme en informatique, ce principe fondamental montre qu'il est important de faire précéder l'exécution par une spécification, et de bien distinguer les deux. Faut-il enseigner une arithmétique non typée au C2 ? Nous ne le croyons pas ; les enfants découvriront la beauté des corps quand ils seront plus grands ! À cet âge, il convient plutôt de poser les bases d'un raisonnement structuré. Nous sommes ici au croisement de la logique, de la physique, de la philosophie et de l'informatique. Ainsi, au C2, il conviendrait d'expliquer que 8 lignes-de-chocolat × 4 carrés-par-ligne = 32 carrés-de-chocolat ; que 60 billes ÷ 4 personnes = 15 billes-par-personne ; que l'on ne peut pas additionner 3 pommes et 4 bananes, mais que 3 fruits + 4 fruits est possible ; etc. Plus tard, au C3 ou C4, on pourra aller plus loin et considérer que 3 pommes + 4 bananes donnent (mais pas « égalent ») 7 fruits.

6 Conclusion

À titre de conclusion, listons ici les principales idées relevées ainsi que quelques repères suggérés pour chacune (pour une suggestion de développement curriculaire de ces idées, *cf.* l'annexe en fin d'article) :

- Fonctionnement autonome : définition : « fonctionnement réactif mais non contrôlé » ; différence avec la vie, la volonté propre et l'intelligence ; automatique ; peur du remplacement ; débat entre automatisation et instrumentation ; programme de commande ; robot.
- Comportement de communication : flux d'information pour comprendre un système (société, être vivant, machine) ; interface ; interface Homme-machine ; débat prothèse/prolongement/augmentation.
- Système distribué : définition : « ensemble composé de nombreux agents qui bénéficie d'un fonctionnement intégré » ; web ; système intégrateur ; transparence au réseau ; interactions entre logiciels, entre machines ; informatique ubiquitaire, pervasive.
- Algorithme : définition : « procédure formalisée et systématique résolvant un problème posé dans un cadre formel » ; définition simplifiée : « procédure systématique, abstraite et finalisée » ; apprendre à ne pas présupposer qu'un interlocuteur sait déjà ; limites des algorithmes linéaires (ou circulaires).
- Programme enregistré : distinction entre langage de programmation et langage-machine ; distinction entre algorithme et implémentation ; exécution d'un programme ; instruction ; programmation impérative ; importance de la précision de la notation ; codage et langages informatiques.
- Modularité : définition : « démarche d'abstraction de situations concrètes afin d'en percevoir la généricité et d'affecter des traitements spécifiques à des procédures spécifiques » ; décomposition d'un problème en sous-problèmes ; utiliser systématiquement des feuilles de style dans un traitement de texte.
- Principe de codage : une donnée est un signe plus un encodage ; les bits ne portent pas d'information en eux-mêmes : « un bit est un bit » ; une mémoire, un fichier, un canal de transmission, un logiciel peuvent stocker, contenir, véhiculer et traiter de façon indifférente tous types de données ; toute information enregistrée numériquement est symbolique.
- « Algorithmes + structures de données = programmes » : toute donnée nécessite un référentiel conceptuel ; échantillonnage ; quantification ; différence entre document numérisé et document nativement numérique.
- Correspondance preuve-programme : dimension documentaire d'un programme ; typer tout calcul ; analyse dimensionnelle.

Références

Baron, G., & Bruillard, E. (2001). Une didactique de l'informatique ?, *Revue fr. de pédagogie*, 163–172.

Bruner, J. S. (1960), *The Process of Education*, Harvard University Press.

Delmas-Rigoutsos, Y. (2014), *Histoire de l'informatique, d'Internet et du web*, notes de cours, Université de Poitiers, < https://delmas-rigoutsos.nom.fr/documents/YDelmas-histoire_informatique.pdf>.

Denley, P. (1994), Models, Sources and Users : Historical Database Design in the 1990s, *History and Computing* 6 (1), 33–43.

Denning, P. J. (2003), Great principles of computing, *Communications of the ACM* 46 (11), 15–20.

Dowek, G. (2011), Les quatre concepts de l'informatique. In *Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques*, actes Didapro 4, 24–26 oct. 2011, Université de Patras, < https://edutice.archives-ouvertes.fr/edutice-00676169/fr/>.

Fayon, D. (2010), *Web 2.0 et au-delà*, Economica, 2e éd.

Heudin, J.-Cl. (2008), *Les créatures artificielles, des automates aux mondes virtuels*, Odile Jacob, Paris.

Lessig, L. (2000), Code is Law—On Liberty in Cyberspace, *Harvard Magazine*, jan. 2000, trad. française : < https://framablog.org/2010/05/22/code-is-law-lessig/>. Ces idées sont développées dans : *Code and Other Laws of Cyberspace*, Basic Books, 1999, et *Code : Version 2.0*, < http://codev2.cc/>, 2006.

Lévy, P. (1992), *De la programmation considérée comme un des beaux-arts*, éd. La découverte.

Ministère de l'éducation nationale de l'ens. sup. et de la recherche (2015a), Socle commun de connaissances, de compétences et de culture, *Bull. off. éducation nationale*, décret n° 2015–372 du 31/03/2015, France.

Ministère de l'éducation nationale de l'ens. sup. et de la recherche (2015b), Programmes d'enseignement du cycle des apprentissages fondamentaux (cycle 2), du cycle de consolidation (cycle 3) et du cycle des approfondissements (cycle 4), *Bull. off. éducation nationale*, bulletin officiel spécial n°10 du 19/11/2015.

Hartmann, W., Näf, M., Reichert, R. (2012), *Enseigner l'informatique*, Springer, éd. orig. 2006.
Papert, S. (1996), An Exploration in the Space of Mathematics Educations, *International Journal of Computers for Mathematical Learning*, 95–123.
Schwill, A. (1993), Fundamentale Ideen der Informatik, *Zentralblatt für Didaktik der Mathematik*, 20–31.
Weiser, M. (1991), The Computer for the 21st Century, *Scientific American* 265 (3), sept. 1991 ; repris en français : Les réseaux informatiques de l'an 2000, *Pour la science* 169, nov. 1991.
Wing, J. (2006), Computational thinking, *Communications of the ACM*, 33–35.
Wing, J. (2008), Computational thinking and thinking about computing, *Philosophical Transactions of the Royal Society* A 366, 3717–3725.

A Annexe : trame de curriculum

Nous synthétisons ici les éléments curriculaires proposés ci-dessus. Ces éléments ne visent que la pensée informatique et ne concernent donc ni les usages du numérique ni l'éducation aux médias et à l'information, sauf dans les quelques cas où il y a convergence entre ces objectifs. Par ailleurs, ces éléments se réfèrent à des notions scolaires qui transposent des concepts savants, et non à ces derniers. Nous suivons le découpage français utilisé dans l'article : C2, le cycle 2 des apprentissages fondamentaux (CP, CE1, CE2, 6–9 ans) ; C3, le cycle 3 de consolidation (CM1, CM2, 6e, 9–12 ans) ; C4, le cycle 4 des approfondissements (5e, 4e, 3e, 12–15 ans).

A.1 Les machines autonomes et leurs réseaux

A.1.1 Autonomie, automatique et automatisation

Notion centrale. Machine autonome ou automatique : fonctionnant de façon réactive, mais non contrôlée.

Points de repère. C2 : Structure des machines autonomes courantes, en particulier l'ordinateur, désignation de ses parties usuelles. C2 : L'autonomie n'est pas la vie (même s'il y a des attributs communs). C2/C3 : Dès leur première utilisation, structure des robots pédagogiques ; notion de programme de commande. C2 : L'autonomie n'implique pas forcément de volonté propre ni d'intelligence.
Question transversale possible. C4 : Question de société : le remplacement de l'Homme par la machine, qui peut être abordée sous l'angle de l'opposition automatisation/instrumentation.

A.1.2 Les flux d'information

Notion centrale. Comportement de communication, c'est-à-dire comportement en termes de flux d'information, d'une machine, d'un être vivant, d'une personne ou d'une société[8].
Points de repère. C2/C3 : Comportement de communication de robots pédagogiques. C4 : Notion de flux d'information. C4 : Interface personne-machine. C4 : Communications entre machines.
Question transversale possible. C4 : Les machines comme prothèses (réparation), prolongements (outillage) ou augmentations (potentialisation) de l'Homme.

A.1.3 Les systèmes distribués intégrateurs

Notion centrale. Système distribué : ensemble composé de nombreux agents, qui bénéficie d'un fonctionnement intégré bien que les agents ne soient pas nécessairement localisés au même endroit.
Points de repère. C2 : Structure du web, adresse universelle. C3/C4 : Systèmes d'information distribués, structure des outils en ligne usuels, arborescence. C4 : Fonctionnement des objets connectés ; transparence au réseau.
Question transversale possible. C4 : Informatique ambiante/ubiquitaire/pervasive, Internet des objets.

8 La notion d'information, au sens de l'informatique ou au sens des sciences de l'information et de la communication, est à réserver pour la suite de la scolarité. Le « flux d'information » est ici essentiellement un échange de signaux, de données ou de messages.

A.2 Les algorithmes et les programmes

A.2.1 L'algorithme

Notion centrale. Algorithme : procédure systématique, abstraite et finalisée.
Points de repère. C2 : Dans une transmission d'instructions, apprendre à ne pas présupposer qu'un interlocuteur sait déjà. C2/C3 : Exemples courants d'algorithmes (calcul posé, recette de cuisine, etc.), notamment qui ne soient ni linéaires ni circulaires. C2/C3 : Des activités de programmation simple pourront aborder des exemples de conditions, de paramètres et d'itérations. Ces trois notions seront formalisées au C4. C4 : Limites des algorithmes linéaires et circulaires.
Questions transversales possibles. Les programmes offrent déjà de nombreux points de contact entre la notion d'algorithme et l'enseignement des procédures (C2–C3) et des mathématiques (C3–C4).

A.2.2 Le programme

Notion centrale. Programme enregistré : texte composé d'une suite organisée d'instructions.
Points de repère. C3 : Implémentation d'un algorithme simple ; faire dire « Bonjour, tout le monde ! » ; importance de la précision de la notation. C3 : Distinction entre programme et exécution, notion de programmation impérative. C4 : Distinction entre langage de programmation et langage-machine.
Question transversale possible. C2–C4 : Codage des langues naturelles et des langages informatiques, dimensions lexicales, syntaxiques et, dans une certaine mesure, conatives.

A.2.3 La modularité

Notion centrale. Modularité : soucis d'affecter des traitements spécifiques à des procédures spécifiques.
Points de repère. C3 : Décomposition d'un problème en sous-problèmes, par exemple en faisant agir plusieurs agents simultanément (p. ex. lutins de *Scratch*). C3/C4 : Structuration des programmes. C4 : Structures de contrôles simples. C4 : Démarche d'abstraction de situations concrètes afin d'en percevoir la généricité ; cas pratique : utiliser systématiquement des feuilles de style dans un traitement de texte.

Question transversale possible. La démarche de recherche de généricité peut être conduite dans de nombreuses situations, pas nécessairement reliées explicitement à l'informatique.

A.3 Les données et leur codage

A.3.1 Principe du codage

Notion centrale. Donnée : un signe, à interpréter à l'aide d'un codage.
Points de repère. C2 : Les ordinateurs traitent des données, qui sont stockées dans des mémoires informatiques ; mémoires de masse usuelles. C3 : Toute information enregistrée numériquement est symbolique ; un même signe peut représenter de nombreuses données ; ex. de la représentation binaire. C3/C4 : Notion de signal. C4 : Quantité de données, le bit, l'octet, mesures usuelles. C4 : Notions générales sur le chiffrement des communications.
Question transversale possible. C2–C4 : Arbitraire du signe linguistique, p. ex. avec des jeux de chiffrement.

A.3.2 Structures de données

Notion centrale. Toute donnée nécessite un référentiel pour l'interpréter.
Points de repère. C2 : Exemples de données rapportées à un référentiel, p. ex. latéralisation de mouvements relatifs d'un robot. C3 : La numérisation, comme combinaison d'un échantillonnage et d'une quantification, ex. des pixels. C3 : Différence entre document numérisé et document nativement numérique. C3 : Exemples de référentiel de données, p. ex. mouvement d'un lutin ou d'un robot codé en coordonnées cartésiennes ou en déplacement relatif. C4 : Formats usuels de fichiers.
Questions transversales possibles. C2 : Repérage du corps ou dans l'espace. C3 : Repères cartésiens, organisation des données. C4 : Données expérimentales. C4 : Relativité galiléenne. C2–C4 : *pixel art* en arts plastiques.

A.3.3 Typage des calculs et des traitements de données

Notion centrale. Toute donnée a un type, qui détermine les traitements qu'elle peut subir.

Points de repère. C2 : Tout calcul, en mathématiques, en sciences et en informatique, doit être typé. C3 : Analyse dimensionnelle élémentaire. C4 : Tout programme doit être commenté et documenté : type des entrées, type des sorties, effets.
Question transversale possible. C2 : L'apprentissage du calcul est au carrefour des mathématiques, des sciences et de cette notion fondamentale du typage de données.

Julie Henry[a], Alyson Hernalesteen[b], Bruno Dumas[a] &
Anne-Sophie Collard[b]

a. Centre de recherche PReCISE – NAmur Digital Institute (NADI) – Université de Namur (Belgique)
julie.henry@unamur.be
b. Centre de recherche CRIDS – NADI – Université de Namur (Belgique)
anne-sophie.collard@unamur.be

Que signifie éduquer au numérique ? Pour une approche interdisciplinaire

Résumé
Qu'est-ce que l'éducation au numérique ? Quelles en sont les finalités ? Quelles sont les compétences nécessaires au développement d'une culture numérique chez les jeunes ? Partant des deux approches disciplinaires distinctes mais complémentaires que sont l'éducation à l'informatique et l'éducation aux médias numériques, des éléments de réponse sont discutés, issus d'une réflexion menée autour de trois axes identifiés par les auteurs comme des finalités d'une éducation au numérique : l'informatique comme discipline fondamentale, les représentations du numérique chez les jeunes (dans un but de recrutement) et l'éducation citoyenne à travers les médias. Une rencontre de ces deux approches et de ces trois axes est proposée à travers le glissement d'un modèle de la littératie médiatique et numérique, la matrice des compétences médiatiques de Fastrez et De Smedt, vers les outils numériques qui ne seraient a priori pas définis comme des médias. Il s'agit ici de poser les premières briques d'un travail de réflexion qui se doit d'être interdisciplinaire.

Mots clés : éducation au numérique, éducation aux médias numériques, éducation à l'informatique, pensée informatique, approche interdisciplinaire

1 Introduction

Les initiatives d'éducation au numérique se développent aujourd'hui dans diverses directions, touchant toutes les tranches d'âge, dans des cadres

éducatifs autant formels qu'informels. En Belgique francophone, elles s'inscrivent dans une évolution de l'enseignement qui cherche à intégrer une culture numérique afin de, notamment, préparer les élèves au « monde numérique » qui les entoure et dans lequel la technologie est appelée à jouer un rôle croissant. Dans le but de susciter et de soutenir ces initiatives, des financements impulsionnels, tels que ceux proposés lors des appels à projets « École Numérique » dans le cadre de la stratégie numérique « Digital Wallonia »[1], visent de plus en plus à équiper les écoles et les lieux éducatifs parascolaires sensibilisés à l'intérêt d'une éducation au numérique. Les enjeux politiques sont à la fois sociaux, citoyens et économiques, les perspectives de déploiement du numérique touchant tous les secteurs de la société (éducation, santé, industrie, administration, etc.).

Dans ce foisonnement, il est difficile d'identifier clairement les finalités et l'objet de l'éducation au numérique. Que recouvre la culture numérique ? Utiliser les réseaux sociaux, mobiliser les technologies de l'information et de la communication (TIC) pour travailler en équipe, manipuler des objets connectés dans des pratiques sportives, « customiser » un jeu vidéo (via des mods) ou encore construire son propre « chat bot », la palette des activités impliquant le numérique est très large. Quelles sont dès lors les compétences auxquelles les lieux éducatifs doivent former pour développer une culture numérique ? Que l'on se centre sur les aspects informatiques, industriels, communicationnels ou encore sociétaux des technologies numériques, plusieurs approches peuvent répondre à cette question. Une articulation entre ces différentes approches de l'éducation au numérique est-elle dès lors envisageable ? En quoi une telle articulation est-elle intéressante ou difficile ?

Notre communication propose de réfléchir aux contours de ces questions en croisant deux approches disciplinaires qui abordent l'éducation au numérique sous des angles différents mais qui nous semblent complémentaires : l'éducation à l'informatique et l'éducation aux médias. Nous développerons cette réflexion autour de trois axes, identifiés comme des finalités de l'éducation au numérique à travers les différents projets de recherche que nous menons. Le premier axe concerne l'intégration dans les cursus scolaires d'une formation à l'informatique en tant que formation fondamentale. Le deuxième axe recouvre les activités d'initiation à la technologie et à l'informatique en vue d'attirer des étudiants dans ces filières. Le dernier axe s'inscrit dans une approche d'éducation citoyenne et d'éveil

1 <http://www.ecolenumerique.be/qa/>

au numérique en vue de comprendre le fonctionnement et les enjeux du numérique dans notre société. Nous commencerons par développer chacun de ces axes, les deux premiers étant envisagés selon une éducation à l'informatique, le dernier étant abordé au travers de l'éducation aux médias. Cette première étape permet d'en préciser les spécificités, les évolutions et les enjeux. Il nous semble en effet important de pouvoir les distinguer lorsque des initiatives d'éducation au numérique sont mises en place. Nous proposons enfin de les confronter et de réfléchir à de possibles rencontres, voire à des articulations, qui permettraient de les nourrir mutuellement.

2 Apprendre l'informatique : jusqu'à quel point ?

En 2014, le Conseil Scientifique de la Société Informatique de France (SIF, 2014) propose une définition de l'informatique mettant en évidence ses deux facettes que sont « la science et la technique » : science, parce qu'elle permet d'expliquer le monde, à l'instar des mathématiques, de la physique et de la chimie, entre autres ; technique, parce qu'elle permet de créer. Si la frontière entre informatique et numérique est distincte (bien que mouvante), la SIF considère que l'informatique est « la science fondamentale de notre monde numérique », ce qui rend le numérique possible.

Si de nombreux pays évoquent une éducation au numérique plutôt qu'à l'informatique, celle-ci n'en reste pas moins un élément incontournable : « le numérique s'accompagne d'un changement de paradigme, de façon de penser ; l'éducation à l'informatique permet d'accompagner ce changement » (De la Higuera, 2015). En effet, les démarches d'initiation à la pensée informatique, à l'algorithmique, au codage, à la programmation, et à toutes autres appellations similaires utilisées dans les médias et instances éducatives, sont une partie de cette informatique garantissant une compréhension des outils numériques.

Axe 1 L'informatique pour se développer

Comme il est fondamental de former les élèves à la démarche scientifique, l'éducation au numérique met bien souvent l'accent sur cet enjeu qu'est la

formation à une pensée informatique (Wing, 2006). Mais cette pensée est-elle spécifique à la discipline informatique ou déjà prise en charge, parfois sous une autre appellation, par d'autres disciplines, comme les mathématiques, par exemple ?

On peut, en effet, se demander comment s'organisent entre elles les pensées informatique, algorithmique et mathématique. Selon Modeste (2012), si la pensée algorithmique peut être regardée comme une pensée des mathématiques, la « voir comme pensée majeure de l'informatique permet un réel enrichissement de son analyse ». Il se pose alors la question de la place de la pensée mathématique dans la pensée informatique. Selon lui, la proximité entre informatique et mathématique laisse à penser qu'un enseignement de la pensée informatique pourrait être l'occasion de développer une certaine pensée mathématique, là où cette dernière ne développerait que partiellement la pensée informatique. Celle-ci aurait donc un rôle à jouer et une place à occuper dans l'éducation.

Mais que vise-t-on en développant la pensée informatique ? De nombreux problèmes se résolvent aujourd'hui en passant par trois étapes (De la Higuera, 2015) :

- transformation des données du problème en information
- traitement algorithmique de cette information
- restitution de la solution sous une forme utile, acceptable, agréable

Cette façon de résoudre un problème, en passant par la transformation en information et sa résolution informatique repose sur la pensée informatique. Celle-ci est notamment définie par Barefoot[2] à travers six concepts : la logique, les algorithmes, la décomposition (d'un problème complexe en sous-problèmes), les patterns (similarités et autres caractéristiques que les sous-problèmes partagent), l'abstraction et l'évaluation.

Quelle place occupe dès lors la programmation ? L'algorithmique abordée par la pensée informatique est-elle suffisante, ou la pratique de la programmation doit-elle être additionnée/favorisée ? De nombreuses activités réalisées pour développer cette pensée informatique le sont de façon débranchée, minimisant le questionnement sur l'obligation d'aller jusqu'à programmer. Car oui, la pensée informatique peut avoir du sens même sans ordinateur ! La démarche procédurale, l'abstraction, la capacité de créer des algorithmes sont autant de qualités à apprendre puis à maîtriser pour

2 <https://barefootcas.org.uk/>

résoudre des problèmes de toutes natures. Le programme est principalement là pour exécuter cette résolution et mettre en pratique les langages informatiques. Ce rôle secondaire ne reflète pas vraiment l'importance qu'on a donnée à l'activité de programmation ces dernières années dans le monde de l'éducation. Rushkoff et Purvis (2011) considèrent même l'apprentissage de la programmation comme aussi important que celui de la lecture et de l'écriture. Toutefois, dans le contexte actuel belge où aucune formation didactique en informatique n'est organisée et compte tenu du manque de compétences perçu chez les enseignants (Henry & Joris, 2013)[3], il serait peut-être opportun de se limiter, dans un premier temps, à un enseignement de la pensée informatique. Il faut cependant souligner l'intérêt de la mise en pratique de la pensée informatique via la programmation, et la nécessité de l'intégrer par la suite. Pour revenir aux trois étapes de De la Higuera (2015) présentées plus haut, la troisième étape de « restitution de la solution sous une forme utile, acceptable, agréable » implique une création qui puisse être confrontée à des utilisateurs. On notera au passage que la seule pensée informatique ne permet pas de remplir cette étape, et que des compétences en ergonomie sont supposées.

Outre la programmation, l'informatique comporte ainsi d'autres volets qui s'avèreraient utiles dans le cadre d'une éducation au numérique. Duchâteau (1992) parle de « culture informatique », comme d'un

> [. . .] ensemble, aussi stable et adapté que possible, de savoirs et de savoir-faire qui permettent d'être à l'aise face à l'ordinateur et aux outils informatiques, de comprendre et de juger ce que permet l'informatique et ce qui est hors de sa portée… Évidemment ces connaissances et ces pratiques vont lentement colorer la vision que nous aurons du réel et de nous-mêmes pour s'intégrer à ce qui fait notre culture globale.

Selon lui,

> [. . .] l'informatique est une quête incessante pour débusquer le sens sous la forme, c'est une entreprise d'enfermement dans la forme de ce que nous appelons le sens. Plus personne ne devrait demain sortir de l'enseignement obligatoire sans au moins avoir perçu cela à propos de l'informatique.

Dès lors, des sujets tels que le codage de l'information, le traitement formel de celle-ci, le schéma fonctionnel d'un système informatique (SI) et ses composants, les objets d'interaction possibles et le fonctionnement du SI

3 Voir également l'article de Henry et Smal publié dans ce même ouvrage.

en interaction avec ceux-ci, entre autres, devraient idéalement composer une éducation au numérique. Sass et Vandeput (1993) ont tenté, dans une démarche inspirée par Duchateau, de dégager les notions fondamentales de l'informatique et de ses outils (traitement de texte et tableur, notamment), sans se perdre dans les détails ou les fonctionnalités jugées non indispensables dans une première approche. En plus de replacer l'ordinateur, et de façon plus générale le SI, dans son rôle d'exécutant, Sass et Vandeput (1992) insistent sur un apprentissage raisonné des outils (et donc, de leur usage).

> En quoi l'utilisation d'un logiciel peut-elle être méthodique ? Y a-t-il dans ce type d'activité des démarches analytiques ? Y a-t-il des concepts importants à cerner ? En quoi certains traitements différés rapprochent-ils l'utilisation des (outils) de l'analyse et de la programmation ?

Dix ans plus tard, Vandeput et Colinet (2004) persistent en affirmant que les utilisateurs sont bien souvent surpris par les comportements inattendus d'un outil s'ils ne possèdent pas des connaissances solides à propos de ses principes organisateurs. Selon Vandeput,

> L'importance du fossé d'évaluation entre les buts de l'utilisateur traduits en action et les effets de cette action […] tel qu'observé depuis de nombreuses années en formation et en enseignement sont donc un bon point de départ pour […] un classement en importance des concepts fondamentaux que l'on peut se permettre d'appeler les invariants.

Mais, si la « méthodologie » des invariants (Vandeput, 2011) prône l'importance des concepts pérennes et généralisables (indépendants d'une marque d'outil ou d'un SI), que doit-on penser des sujets d'informatique considérés avancés par les praticiens mais auxquels le grand public se retrouve régulièrement confronté tels que le big data, la cybersécurité, l'internet des objets ou l'intelligence artificielle ? Dans quel cadre les aborder (s'ils doivent l'être) ?

Certains sujets paraissent plus fondamentaux que d'autres et pourraient constituer dès lors des matières à enseigner. Ainsi l'intelligence artificielle (IA), par la solide base en statistiques et mathématiques requise pour aborder un tant soit peu sérieusement la compréhension de ses algorithmes classiques, nécessite un bagage préalable qui rend le sujet difficile comme simple illustration. L'intérêt que lui porte actuellement la France avec le collectif « France is IA » va d'ailleurs dans ce sens. Les autre sujets évoqués, pour leur part, devraient donner avant tout, dans le cadre d'une

éducation au numérique, du sens aux apprentissages en les ancrant dans la réalité. En effet, ces sujets, souvent abordés par la presse, attirent et interrogent, tout en nécessitant pour leur bonne compréhension des bases en informatique. Posséder une « culture informatique », telle que précédemment décrite, assurerait dans une certaine mesure cette compréhension (tout en restant à un niveau simpliste, voire une vulgarisation) et éviterait beaucoup de questionnement. Cependant, est-ce suffisant de comprendre lorsqu'il s'agit de sujets plus « sensibles » ? Les aspects relevés jusqu'à présent se concentrent effectivement sur une vision essentiellement centrée sur la compréhension et la production, laissant de côté la réflexion critique sur les outils numériques… une réflexion qui a pourtant sa place dans une éducation au numérique et qui sera discutée ultérieurement.

Axe 2 L'informatique pour recruter

Il s'agit ici de considérer les enjeux liés au recrutement d'étudiants dans les filières de formation en informatique et à ses implications au niveau du développement des secteurs technologiques. La digitalisation de la société requiert des experts capables de porter les innovations technologiques. La carence continue de tels experts est par ailleurs régulièrement soulignée (Albessart, Calay, Guyot, Marfouk & Verschueren, 2017). Selon le rapport sur l'Attractivité des études et des métiers scientifiques et techniques publié par le Conseil Wallon de la Politique Scientifique (CPWS, 2014), le nombre de jeunes s'orientant vers les études scientifiques et techniques, et particulièrement vers l'informatique, en Fédération Wallonie-Bruxelles est insuffisant pour couvrir les besoins de nos entreprises. Les initiatives d'éducation au numérique qui s'inscrivent dans cette perspective visent dès lors à donner le goût aux études en informatique et à en construire une représentation qui correspond à leur but, à leurs exigences et à leur contenu.

Sur ces dix dernières années, de nombreuses études ont été menées autour des représentations qu'ont les jeunes (de 12 à 18 ans) de l'informatique. Si de nombreux facteurs sont susceptibles d'influencer leur manque d'intérêt envers ce domaine, la représentation qu'ils en ont reste un facteur majeur. Cassel, Guzdial et Roberts (2007) mentionnent, par exemple, certaines craintes/préjugés (« fear of becoming isolated in jobs perceived to involve little human contact ») et une méconnaissance générale du domaine

(« little public understanding of the broader dimensions of the computing field »).

À la fin des années 90, Greening (1998) met déjà en cause les représentations (« *students might also not be enrolling in computer science courses due to misconceptions* ») et propose de faire évoluer l'enseignement « *in such a way that it increases the likelihood that students become excited about their learning* ». Près de dix ans plus tard, Carter (2006) conclut qu'un manque d'éducation peut être tenu responsable des mauvaises représentations que possèdent les jeunes. Ce dernier recommande alors de contextualiser l'enseignement (« *educating students on how computing is really used in the real world – for special effects in the movies, to improve the quality of life for people with missing limbs, and for allowing communication for people with speech impediments* »). Yardi et Bruckman (2007) suggèrent un curriculum « *to prepare and motivate teenagers for careers in today's expanding, Internet-based, global economy* ». Ils décrivent l'informatique comme « *an innovative, creative and challenging field with authentic, real-world applications* » et pensent qu'il existe une possibilité d'augmenter l'intérêt des jeunes envers l'informatique en reliant leurs représentations aux opportunités offertes par le domaine. Dans la même lignée, Grover, Pea et Cooper (2014) présentent les résultats d'une intervention au sein d'un curriculum ayant pour but de montrer l'informatique « *in a new light – in real world contexts and as a creative and problem-solving discipline; as something bigger and broader than the computer-centric view that many students are known to harbor* ».

Éduquer à l'informatique pour faire changer les représentations (et potentiellement intéresser les jeunes aux études associées) ? Oui, à condition de rester attentif à ne pas proposer une vision « restrictive » de l'informatique. Il s'agit de montrer, à travers le contenu d'apprentissage proposé et sa contextualisation (ce qui nous renvoie ici notamment aux sujets avancés en informatique précédemment cités – cybersécurité, IA, etc. – et à leurs enjeux – économiques, sociaux, sociétaux, politiques), toute la diversité du domaine et ce afin de ne pas produire l'effet inverse de celui attendu.

Une autre problématique qui traverse cet axe est celle du genre, fortement présente dans l'actualité scientifique depuis quelques années. Les attitudes envers l'informatique seraient genrées (Collet, 2004, 2007, 2011) et l'exposition croissante aux technologies ne comble pas la différence existant (Cai, Fan & Du, 2017 ; Colley & Comber, 2003 ; Durndell, Glissov & Siann, 1995). En Belgique, comme ailleurs, la faible proportion de filles

entamant des études en informatique est un constat récurrent : les filles ne représentent que 8 % des inscriptions en Faculté d'informatique au sein de notre université pour l'année académique 2017–2018, elles étaient moins de 7 % l'année précédente. Selon Cox, les stéréotypes n'aident pas à faire monter les chiffres : les femmes sont encore souvent montrées dans des situations à problème avec l'ordinateur (entre autres) (cité par Drot-Delange, 2011). Or, pour Valenduc (cité par Drot-Delange, 2011), ces images ont une influence négative sur l'orientation des filles. Certaines solutions ont été apportées, à divers niveaux. Ainsi, en France, les manuels scolaires de la technologie au collège, discipline dans laquelle est enseignée l'informatique dans le second degré, ont évolué vers des représentations moins stéréotypées des femmes (Baron, Drot-Delange, Khaneboubi & Sedooka, 2010). De notre point de vue, sans cours et sans manuel de référence, le travail sur la représentation du domaine pourrait s'avérer ici aussi pertinent. Selon Barker et Aspray (2006), la méconnaissance des métiers est un facteur (parmi d'autres) expliquant l'inégalité des femmes et des hommes en informatique.

Enfin, en plus d'une contextualisation de l'enseignement, il semble que la période à laquelle s'organise cet enseignement ait également une importance. En effet, Hewner et Guzdial (2008) montrent qu'une éducation à l'informatique organisée postérieurement à l'enseignement obligatoire ne présente pas un impact important sur l'attitude des jeunes envers ce domaine. D'où l'importance d'une éducation organisée au plus tôt dans le cursus des jeunes.

3 Devenir citoyen du monde numérique

À côté d'activités davantage tournées vers l'éducation à l'informatique et à la technologie et orientées vers les deux finalités que nous venons d'exposer, nous observons des initiatives qui visent à « éduquer au numérique » dans le cadre d'une éducation citoyenne. L'objectif est d'éveiller au numérique (entendu ici dans un sens large que nous préciserons par la suite) et de développer une posture critique chez les apprenants, afin de les rendre autonomes face aux choix qui se présentent à eux et de les faire participer à la société dans laquelle ils vivent. Nous entendons la participation dans le sens où Carpentier (2011) la définit dans le contexte médiatique,

c'est-à-dire en considérant que chaque citoyen doit avoir les moyens, en termes d'accès à la technologie et de compétences, pour être à la fois usager et producteur médiatique. L'éducation au numérique doit dès lors chercher à faire acquérir un certain nombre de compétences nécessaires aux citoyens dans une société où le numérique occupe de plus en plus de place, à des fins d'usage mais aussi de production d'outils numériques.

Quelles sont ces compétences nécessaires ? Quels en sont les enjeux ? Leur définition apparaît comme un exercice complexe mobilisant plusieurs cadres théoriques relatifs à différents types de littératies : technologique, numérique, informationnelle, médiatique, etc. La question de l'éducation citoyenne a d'ailleurs déjà été en partie abordée dans le premier axe de notre présentation, davantage dans le sens d'une culture informatique ou technique qui serait, selon De Noblet, « nécessaire pour que tout individu puisse agir avec et sur son environnement, gage de sa liberté et de son indépendance. La culture numérique se définirait alors par la possession d'un minimum de connaissances et de savoir-faire » (cité par Drot-Delange & Bruillard, 2012, p. 2).

Pour le troisième axe, notre point de départ est le cadre de l'éducation aux médias numériques, à partir duquel nous tâchons modestement de préciser les contours des compétences numériques citoyennes. Il nous semble en effet intéressant de mobiliser ce type d'approche car, d'une part, l'éducation aux médias numériques est un pan important de l'éducation à la citoyenneté (citons, par exemple, les travaux du Conseil Supérieur de l'Education aux Médias et les travaux de recherche, en cours, accompagnant la mise en oeuvre du Pacte pour un Enseignement d'Excellence en Belgique francophone[4]). D'autre part, notre réflexion cherche à montrer comment ce cadre pourrait être élargi à d'autres technologies numériques pour une éducation à des outils numériques qui ne seraient *a priori* pas définis comme des médias ou des TIC.

Axe 3 Vers une éducation aux médias numériques

Les compétences numériques ont souvent d'abord été définies comme les compétences nécessaires pour comprendre la technologie et le contexte social dans lequel elle s'inscrit, et pour pouvoir l'utiliser de manière efficace,

4 <http://www.pactedexcellence.be/>

sûre et critique (Dobson & Willinsky, 2009 ; Mabrito & Medley, 2008 ; Ng, 2016). Cette approche, qui est plutôt celle de la littératie numérique (*Digital Literacy*), met en avant l'usage de la technologie en général et les compétences fonctionnelles liées à cet usage (savoir réaliser un certain nombre d'opérations comme, par exemple, rechercher une information sur Internet ou savoir utiliser un ordinateur), ainsi que la réflexion critique sur la technologie. L'éducation au numérique met en évidence essentiellement ici l'acquisition de compétences liées à la dimension technique des outils numériques.

D'autres définitions de la littératie numérique se centrent plus précisément sur les médias et les TIC, ce qui implique de prendre en compte d'autres dimensions :

> *From this perspective, a digitally literate individual is one who can search efficiently, who compares a range of sources, and sorts authoritative from non-authoritative, and relevant from irrelevant, documents* (Livingstone, Van Couvering & Thumim, 2005, p. 31). *There is little recognition here of the symbolic or persuasive aspects of digital media, of the emotional dimensions of our uses and interpretations of these media, or indeed of aspects of digital media that exceed mere « information »*. (Buckingham, 2015, p. 266)

Afin de dépasser une approche plus fonctionnelle et instrumentale des médias, Buckingham (2015) les considère non en tant que technologies mais en tant que formes culturelles. Ce changement de point de vue élargit le spectre des compétences à développer :

> *The skills that children need in relation to digital media are not confined to those of information retrieval. As with print, they also need to be able to evaluate and use information critically if they are to transform it into knowledge. This means asking questions about the sources of that information, the interests of its producers, and the ways in which it represents the world ; and understanding how these technological developments are related to broader social, political and economic forces.* (Buckingham, 2015, p. 267)

Il propose de développer une éducation aux médias numériques élaborée autour de quatre concepts : représentation (les représentations du monde, les valeurs et idéologies véhiculées par les médias, la question de l'auteur), langage (utiliser et comprendre le langage, les formes de communication), production (repérer qui communique avec qui et pourquoi, les influences, commerciales notamment) et audience (être conscient de sa position d'audience, ses usages, etc. et de comment le média est ciblé vers une audience).

Par ailleurs, dans une société où les outils numériques ont considérablement augmenté les moyens d'accès à et de production de l'information, un des enjeux est de former les citoyens, non seulement aux usages de l'information et à leur analyse, mais aussi à des compétences de production de contenus médiatiques :

> *[…] we define digital and media literacy as a constellation of life skills that are necessary for full participation in our media-saturated, information-rich society. These include the ability to do the following : […] Create content in a variety of forms, making use of language, images, sound, and new digital tools and technologies.* (Hobbs, 2010, pp.vi–vii)

Ces contenus étant souvent envisagés dans une perspective de partage avec différents publics, la littératie médiatique doit donc s'entendre, citant le travail réalisé par l'Ofcom, comme « *the ability to access, understand and create communications in a variety of contexts* » (cité par Livingstone *et al.*, 2008, p. 104). Plus largement, dans une culture de la participation où chacun crée, diffuse et se connecte aux autres, Jenkins, Purushotma, Weigel, Clinton et Robison (2009) insistent sur la dimension sociale des compétences médiatiques, qui ne seraient pas uniquement orientées vers une performance mais vers les relations sociales : s'intégrer dans un groupe, coopérer, prendre sa place dans une communauté, occuper un rôle, etc.

Les compétences médiatiques évoquées jusqu'ici concernent essentiellement les usages des médias, que ce soit pour consulter ou pour produire de l'information, prenant en compte leurs finalités sociales. En s'éloignant d'une approche plus fonctionnelle, l'éducation aux médias semble avoir mis entre parenthèse la dimension technique des technologies numériques et s'est moins intéressée à la question des compétences relatives à la compréhension et à la production des outils numériques eux-mêmes. Dans leur travail de synthèse sur les compétences du XXIe siècle dans une société transformée par les TIC, Voogt et Roblin (2012) introduisent la nécessité de comprendre la technologie, ses principes et ses stratégies afin de développer des solutions et de réaliser des objectifs. Allant un peu plus loin dans une approche de la littératie médiatique et numérique qui introduit la dimension technologique, les travaux de Vuorikari, Punie, Gomez, Van Den Brande (2016, p. 11) ajoutent le traitement de données (data) et la programmation dans les compétences de création de contenu (« *to plan and develop a sequence of understandable instructions for a computing system to solve a given problem or perform a specific task* »). On relèvera un début

de recoupement avec les trois étapes de résolution de problèmes liés à la pensée informatique de De la Higuera (2015) présentés dans l'axe 1 de cet article, même si la pensée informatique proprement dite n'est considérée que via la programmation.

La matrice de compétences médiatiques élaborée par Fastrez et De Smedt (Fastrez, 2011 ; Fastrez & De Smedt, 2012) (*cf.* Figure 1) permet de faire la synthèse des différentes compétences mentionnées jusqu'ici dans l'axe 3 et considérées comme nécessaires pour « évoluer de façon critique et créative, autonome et socialisée dans l'environnement médiatique contemporain » (Fastrez, 2011, p. 36) qui, aujourd'hui, est largement numérique. Elle situe l'ensemble de ces compétences au croisement des trois dimensions des objets médiatiques (il est question ici d'objet, et non d'outil afin d'éviter un point de vue utilitariste) avec les différents types d'activités médiatiques des usagers. La première dimension est informationnelle et concerne la signification des médias et les représentations qu'ils véhiculent (les aspects sémiotiques). La deuxième dimension est relative à leur fonctionnement technique, leur compréhension et leur utilisation. La troisième dimension est sociale, elle met en évidence le contexte relationnel et socioculturel dans lequel les médias s'inscrivent (les aspects pragmatiques). Au niveau des activités, la lecture et la navigation correspondent, dans le sens le plus large, à des activités de réception, qu'elles soient orientées vers la consultation d'un objet médiatique ou de plusieurs. Les activités d'écriture et d'organisation consistent, elles, en des activités de production, c'est-à-dire de création et de diffusion centrées sur un objet médiatique ou sur plusieurs.

Un des apports de ce modèle réside dans la présence à la fois d'activités de consultation et d'activités de production. Orientées vers la dimension informationnelle, il s'agit de mobiliser des compétences en vue de la compréhension ou de la production de contenus. Pour la dimension sociale, elle reprend les compétences liées à la compréhension du contexte pragmatique des médias ou au développement de relations sociales au moyen de médias (pour le dire brièvement). Au niveau technique, la matrice met en évidence les compétences nécessaires à l'utilisation des technologies médiatiques ou à leur production. Un autre apport de la matrice est d'envisager un modèle intégré où les trois dimensions des objets médiatiques sont présentes. Les compétences médiatiques numériques évoquées peuvent s'inscrire davantage dans une des dimensions. Mais chaque activité médiatique peut être analysée au regard des trois. Cependant cette matrice,

orientée vers l'éducation aux médias, ne semble pas recouvrir directement les dimensions évoquées pour l'éducation au numérique dans les axes 1 et 2 de notre article.

Dimension / Activité	Informationnelle	Technique	Sociale
Lire			
Ecrire			
Naviguer			
Organiser			

Figure 1 : Matrice des compétences de Fastrez et De Smedt.

4 Confrontation et rencontre des axes

Après avoir identifié les trois finalités de l'éducation au numérique telles qu'elles nous apparaissent et après les avoir envisagées à partir de la littératie informatique et de la littératie médiatique, pouvons-nous imaginer des croisements ou des rencontres possibles ? Ou le cloisonnement est-il inévitable, voire nécessaire ? Une éducation à l'informatique peut-elle se passer d'une réflexion sur le contexte social et culturel des technologies (leurs dimensions sociale et informationnelle/symbolique) ? Comment la technique imprègne-elle nos vies ? Les compétences critiques et la contextualisation sociétale des outils numériques permettent, il nous semble, de donner du sens au travail de l'informaticien. Ce travail s'inscrit en effet dans un contexte de représentations sur la technologie, que la production alimente à son tour. À l'inverse, la compréhension des médias numériques peut-elle se passer d'une certaine profondeur dans la compréhension des mécanismes technologiques ? Quel est le rôle de la capacité à produire techniquement un outil numérique dans la compréhension de son fonctionnement et de la manière dont il influence la communication dans ses dimensions informationnelle et sociale ? Ou, pour le dire autrement, à l'instar de

Brotcorne (2017), quel est le niveau de compétences techniques nécessaires pour développer une compréhension critique des outils numériques ?

Sans répondre à toutes ces questions, la matrice de Fastrez et De Smedt (Fastrez, 2011 ; Fastrez & De Smedt, 2012) apporte un cadrage pour identifier les compétences médiatiques et numériques dans leurs différentes dimensions, en consultation comme en production. Il nous semble que cette réflexion menée dans le champ médiatique pourrait interroger la manière de définir les compétences numériques liées de manière plus large aux outils numériques et aux différentes technologies qui apparaissent dans notre société (ordinateurs, objets connectés, robots, intelligence artificielle, big data, etc.). Ce glissement permettrait d'envisager, de manière intégrée, des compétences d'utilisation et de production de ces outils relativement aux dimensions techniques, informationnelles et sociales. Le modèle reprend donc, d'une part, les trois axes présentés : des compétences informatiques fondamentales en utilisation d'outils et en production de ceux-ci (axe 1), une capacité à se représenter le numérique sous ses multiples formes en fonction de ses différentes applications (axe 2) et une réflexion critique sur les usages des outils numériques et leur place dans la société (axe 3). D'autre part, il dépasse le cloisonnement initial en permettant d'envisager des activités impliquant des compétences croisées entre ces différents axes.

Pour la dimension technique, le modèle prend en compte des compétences d'utilisation, impliquant la mobilisation d'outils numériques en vue de l'accomplissement d'une tâche : par exemple, utiliser un tableur pour effectuer des opérations comptables, utiliser une montre connectée pour mesurer son effort physique, ou utiliser une plateforme collaborative pour réserver un séjour. En production, il s'agit d'être capable de créer soi-même, et donc de comprendre, un programme, une base de données, un algorithme, un robot, etc. La finalité éducative, pour la dimension technique, est d'identifier et de former aux compétences informatiques relatives à ces activités. L'intérêt de réfléchir à la manière dont la matrice des compétences médiatiques peut être mobilisée pour les technologies numériques en général est de montrer qu'une éducation au numérique vise non seulement ces compétences techniques, mais aussi des compétences de types « informationnel » et « social » qui permettent d'en avoir une compréhension et un usage critiques. Cette approche est soutenue par les récents travaux de Collard et Jacques (2017) sur l'enseignement de la robotique en vue de développer les compétences critiques des apprenants envers les technologies. Ils définissent la technologie en tant que « système sémiotique qui soutient

les interactions », définition qui met en lumière les dimensions informationnelle et sociale. Les outils numériques sont dès lors considérés comme des objets techniques qu'il faut pouvoir manipuler voire produire, mais aussi comme des systèmes de représentation socialement construits (dimension informationnelle, ou sémiotique), porteurs des intentions des acteurs impliqués (dimension sociale, ou pragmatique) (Collard & Jacques, 2017).

Un exemple qui nous semble illustrer l'application des dimensions technique, informationnelle et sociale à des technologies numériques est le projet TeSLA[5]. Il s'agit d'un projet de recherche européen qui vise actuellement à développer un système permettant d'authentifier des apprenants en e-learning à l'aide d'une série d'outils de biométrie et d'authoring pour certifier l'absence de triche. Cette technologie rassemble plusieurs composants informatiques basés sur différents algorithmes permettant la reconnaissance faciale, la reconnaissance vocale, la reconnaissance de frappe au clavier et la reconnaissance de texte. Portée notamment par des ingénieurs, elle apparaît *a priori* comme « une interface neutre, flexible, pouvant être déployée sur n'importe quelle pratique pédagogique pré-existante, sans en affecter les modalités et les relations » (De Vos, 2018). Toutefois, comme l'analyse De Vos (2018), chercheur sur ce projet, elle embarque plusieurs normes qui traduisent une certaine conception de l'éducation à distance. Une des intentions des acteurs du projet, bien que non explicitée, est de fournir une technologie qui rendrait les apprenants « égalitaires » face à l'évaluation : l'apprenant travaille seul devant l'écran (égalité par la non-ingérence externe), il est identifié objectivement par son corps (égalité des corps), l'accès au savoir est démocratisé grâce aux facilités d'évaluation en ligne (égalité de diffusion) et le savoir est restitué selon des modalités standardisées (égalité de restitution). Les algorithmes de reconnaissance sont élaborés de façon à soutenir cette conception « égalitaire » particulière sur une base définie (paradoxalement) comme objective. De Vos (2018) raccroche cette conception de la formation à la notion de mérite dans le sens où les normes embarquées dans TeSLA peuvent « être comparées avec ce principe de justice basé sur la théorique égalité des chances des étudiants à l'entame de leur cursus et la suppression des déterminismes externes à l'individu ».

Au terme de son développement, cette technologie est sans doute amenée à se répandre dans les dispositifs d'e-learning. Relativement à

5 <http://tesla-project.eu/>

cet exemple précis, une éducation au numérique devrait permettre, à un premier niveau, de pouvoir l'utiliser dans une activité d'apprentissage à distance (configurer les éléments, se connecter au système, paramétrer un compte, etc.). Toutefois, l'apprenant devrait pouvoir identifier les normes embarquées par la technologie, provenant d'une conception particulière des acteurs et des intentions qu'ils ont traduites dans le système. L'usager devrait pouvoir en comprendre les principes, les limites et la manière dont elles influencent le système de formation dans lequel il est partie prenante, permettant ainsi de se positionner par rapport au modèle de l'éducation sous-jacent. À ce niveau, une connaissance technique du fonctionnement algorithmique permettrait de comprendre que cette technologie n'est pas neutre et est soutenue par des choix de conception (notamment les choix des échantillons de comportements humains sur lesquels se basent les algorithmes de reconnaissance). Une certaine connaissance technique permettrait aussi à l'utilisateur de juger objectivement du caractère plus ou moins invasif des technologies de reconnaissance biométrique déployées, et de les accepter – ou remettre en question – en connaissance de cause. Nous envisageons ici le point de vue de l'utilisateur (l'apprenant en e-learning), mais ce type d'analyse critique des intentions des acteurs impliqués et des représentations véhiculées permettrait également d'enrichir la réflexion des concepteurs « techniques » (les ingénieurs) et d'expliciter le sens donné à leur travail.

5 Conclusion

En partant du constat d'une grande diversité d'activités éducatives centrées sur les (des) compétences numériques et parfois d'un flou concernant l'objet réel de ces initiatives, nous avons commencé par identifier et distinguer trois finalités possibles de l'éducation au numérique : une éducation à l'informatique pour se développer, une éducation à l'informatique pour recruter et une éducation aux médias. Un des apports de ce travail est d'en avoir cerné les principaux enjeux et les spécificités en termes de compétences à développer. Ces trois axes sont différents et chacun important à prendre en compte, notamment lorsqu'il est question de repenser les programmes d'enseignement.

Nous les avons définis au regard de deux approches disciplinaires, celle de l'éducation à l'informatique et celle de l'éducation aux médias (numériques). Leurs racines sont différentes mais leurs différences résident parfois plus dans le point de vue qui est adopté que dans l'objet décrit. Nous avons notamment évoqué certains auteurs de chaque approche abordant des questions similaires, mais avec un point de vue propre. Il nous semble en tout cas qu'une complémentarité peut être trouvée et qu'une approche interdisciplinaire permet d'envisager de nouvelles dimensions et de poser de nouvelles questions.

Pour terminer, nous avons proposé une rencontre (ou un dialogue ?) de ces deux approches et des trois axes identifiés à travers le glissement d'un modèle de la littératie médiatique et numérique vers les outils numériques et technologiques. Il s'agit d'une réflexion qui est toujours en cours et qui nécessite de préciser, d'adapter, voire de définir encore davantage les différentes dimensions de ces objets et les types d'activités qui y sont associées. Son intérêt réside essentiellement dans l'intégration dans un modèle unique de compétences techniques largement documentées (mais où beaucoup reste à faire, notamment au niveau didactique) avec des compétences informationnelles et sociales qui semblent peu explicitées aujourd'hui. La terminologie utilisée devra sans doute être revue pour s'adapter aux outils numériques qui ne sont pas nécessairement définis comme des médias. Approfondir cette réflexion amènera certainement aussi quelques nuances dans l'application du modèle original.

Références

Albessart, C., Calay, V., Guyot, J., Marfouk, A. & Verschueren, F. (2017). *La digitalisation de l'économie wallonne : une lecture prospective et stratégique.*

Barker, L. J., & Aspray, W. (2006). *The state of research on girls and it*. na.

Baron, G.-L., Drot-Delange, B., Khaneboubi, M. & Sedooka, A. (2010). Genre et informatique : compte rendu d'une enquête récente par questionnaire sur les opinions d'élèves de lycée. *Revue de l'EPI*.

Brotcorne, P. (2017). L'éducation au numérique, à l'informatique à l'école : termes et enjeux d'un débat houleux. In J. Henry, A. Nguyen &

E. Vandeput (Eds.), *L'informatique et le numérique dans la classe. qui, quoi, comment ?* Namur : Presses Universitaires de Namur.

Buckingham, D. (2015). Defining digital literacy-what do young people need to know about digital media ? *Nordic Journal of Digital Literacy*, *10*(Jubileumsnummer), 21–35.

Cai, Z., Fan, X. & Du, J. (2017). Gender and attitudes toward technology use : A meta-analysis. *Computers & Education*, *105*, 1–13.

Carpentier, N. (2011). The concept of participation : If they have access and interact, do they really participate ? *CM-časopis za upravljanje komuniciranjem*, *6*(21), 13–36.

Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. *ACM SIGCSE Bulletin*, *38*(1), 27–31.

Cassel, A., L. B.and McGettrick, Guzdial, M. & Roberts, E. (2007). The current crisis in computing : what are the real issues ? *ACM SIGCSE Bulletin*, *39*(1), 329–330.

Collard, A.-S., & Jacques, J. (2017, juin). *Enseigner la robotique pour développer les compétences critiques des apprenants.* Communication au 4ème Colloque international en éducation : enjeux actuels et futurs de la formation et de la profession enseignante (Montréal).

Collet, I. (2004). *L'informatique a-t-elle un sexe ?* Consulté le 2017–09-17, sur <http://blog.mondediplo.net/2007-05-29-L-informatique-a-t-elle-un-sexe>.

Collet, I. (2007). *La disparition des filles dans les études d'informatique : les conséquences d'un changement de représentation.* Consulté le 2017–09-17, sur <https://www.cairn.info/revue-carrefours-de-l-education-2004-1-page-42.htm>.

Collet, I. (2011). Effet de genre : le paradoxe des études d'informatique. *tic&société*, *5*(1).

Colley, A., & Comber, C. (2003). Age and gender differences in computer use and attitudes among secondary school students : what has changed ? *Educational Research*, *45*(2), 155–165.

CPWS. (2014). *Attractivité des études et métiers scientifiques et techniques.* Consulté le 2017–09-17, sur <http://www.cesw.be/uploads/publications/CPS_Rapport_janvier2014.pdf>.

De la Higuera, C. (2015). *À l'école, doit-on enseigner l'informatique ou le coding ?* Consulté le 2017–09-17, sur <http://www.slate.fr/story/110897/ecole-enseigner-informatique-coding>.

De Vos, N. (2018, avril). *Système d'e-learning et norme de mérite : le cas de tesla.* Communication soumise au 143ème congrès du CTHS « La transmission des savoirs », Paris.

Dobson, T., & Willinsky, J. (2009). Digital literacy. *The Cambridge handbook of literacy*, 286–312.

Drot-Delange, B. (2011). Informatique et web : quelle place pour les filles ?. le cas d'étudiants d'une formation hybride. *Questions Vives. Recherches en éducation*, *8*(15).

Drot-Delange, B., & Bruillard, E. (2012). Éducation aux tic, cultures informatique et du numérique : quelques repères historiques. *Études de communication*(1), 69–80.

Duchâteau, C. (1992). *Peut-on définir une « culture informatique » ?* Facultés universitaires Notre-Dame de la Paix. Durndell, A., Glissov, P. & Siann, G. (1995). Gender and computing : Persisting differences. *Educational research*, *37*(3), 219–227.

Fastrez, P. (2011). Quelles compétences le concept de littératie médiatique englobe-t-il ? une proposition de définition matricielle. *Recherches en communication*, *33*(33), 35–52.

Fastrez, P., & De Smedt, T. (2012). Une description matricielle des compétences en littératie médiatique. *La littératie médiatique multimodale. De nouvelles approches en lecture-écriture à l'école et hors de l'école*, 45–60.

Greening, T. (1998). Computer science : through the eyes of potential students. In *Proceedings of the 3rd acse* (pp. 145–154).

Grover, S., Pea, R. & Cooper, S. (2014). Remedying misperceptions of computer science among middle school students. In *Proceedings of the 45th acm technical symposium on computer science education* (pp. 343–348). Henry, J., & Joris, N. (2013). Maîtrise et usage des tic : la situation des enseignants en belgique francophone. In *Sciences et technologies de l'information et de la communication (stic) en milieu éducatif.*

Hewner, M., & Guzdial, M. (2008). Attitudes about computing in post-secondary graduates. In *Proceedings of the fourth icer* (pp. 71–78).

Hobbs, R. (2010). Digital and media literacy : A plan of action. *Washington, DC : The Aspen Institute*.

Jenkins, H., Purushotma, R., Weigel, M., Clinton, K. & Robison, A. J. (2009). *Confronting the challenges of participatory culture : Media education for the 21st century*. Mit Press.

Livingstone, S., Van Couvering, E. & Thumim, N. (2005). Adult media literacy : A review of the research literature. Livingstone, S., Van Couvering, E., Thumin, N., Coiro, J., Knobel, M., Lankshear, C. & Leu, D. (2008). Converging traditions of research on media and information literacies. *Handbook of research on new literacies*, 103–132.

Mabrito, M., & Medley, R. (2008). Why professor johnny can't read : Understanding the net generation's texts. *Innovate : Journal of Online Education*, *4*(6), 2.

Modeste, S. (2012). La pensée algorithmique : apports d'un point de vue extérieur aux mathématiques. In *Actes du colloque espace mathématique francophone.* Ng, W. (2016). *New digital technology in education.* Springer.

Rushkoff, D., & Purvis, L. (2011). *Program or be programmed : Ten commands for a digital age (berkeley, ca : Counterpoint)*.

Sass, F., & Vandeput, E. (1992). Découvrir le travail avec un ordinateur. In *Troisième rencontre francophone de didactique de l'informatique* (pp. 201–208).

Sass, F., & Vandeput, E. (1993). Informatique utile et raisonnée. *Van In*. SIF. (2014). *L'informatique : la science au cœur du numérique.* Consulté le 2017–09-17, sur <http://binaire.blog.lemonde.fr/files/2014/01/14.Informatique.pdf>.

Vandeput, E. (2011). Méthodologie d'identification des invariants du traitement de l'information numérique. In *Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques.* (pp. 93–107).

Vandeput, E., & Colinet, M. (2004). Les invariants du tableur.

Voogt, J., & Roblin, N. P. (2012). A comparative analysis of international frameworks for 21st century competences : Implications for national curriculum policies. *Journal of curriculum studies*, *44*(3), 299–321.

Vuorikari, R., Punie, Y., Gomez, S. C., Van Den Brande, G. *et al.* (2016). *Digcomp 2.0 : The digital competence framework for citizens. update*

phase 1 : The conceptual reference model (Rapport technique). Joint Research Centre (Seville site).

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Yardi, S., & Bruckman, A. (2007). What is computing ? : bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the third international workshop on Computing education research* (pp. 39–50). ACM.

Frédéric Drouillon

Doctorant au CREAD, EA n°3875
fdr@free.fr

Regard littéraire sur la programmation comprise comme une écriture

Résumé
Nous appréhendons la pratique de la programmation en accentuant des questions d'écriture, de langage, de culture et d'expression qui ne requièrent pas nécessairement de prérequis scientifiques. L'objectif n'est pas de traiter des sujets fondamentaux de la recherche en informatique mais simplement de voir, dans une perspective didactique, comment la programmation peut être abordée différemment selon la diversité des publics actuellement concernés par elle : enfants, collégiens, lycéens, étudiants de culture littéraire, artistique et scientifique ainsi qu'un large éventail de profils dans le domaine de la formation professionnelle. Notre premier propos consiste à extraire de la programmation ce qui peut la rapprocher d'un texte. Ensuite d'envisager une analyse de ce texte selon la notion théorique de « figures de pensée » empruntée à l'anthropologue Jack Goody. L'écriture du texte informatique, à l'image de celle du texte littéraire, se révèle être une activité d'auteur qui engage des aspects affectifs et émotionnels pertinents à prendre en compte lors des apprentissages de la programmation. Extraite de son contexte théorique, technique et scientifique, la pratique de la programmation, sans pour autant être facile, peut être proposée avec pragmatisme à de plus nombreux publics.

Mots clés : programmation, langage, code, texte, écriture, technologies de pensée, expression, art

1 Au commencement un texte

Bernard Chazelle (2011) présente la machine de Turing comme un petit train qui circule sur une voie ferrée illimitée et divisée en cases avec un chiffre (une donnée) par case. Selon les instructions qu'il reçoit le train peut

se déplacer d'une case à la fois, d'un côté ou de l'autre, et lire ou écrire. Alan Turing montre qu'il est possible de faire exécuter par cet automate tous les calculs possibles à partir de listes d'instructions élémentaires simples. Cette liste d'instructions qui permet l'accomplissement d'une tâche constitue un algorithme. L'idée vient alors d'écrire également cette liste sur la voie comme le sont les données, le texte final obtenu composant un programme. Ce modèle, qui inaugure un système textuel hors du commun, se trouve complété avec les concepts de fonction et d'application (lambda-calcul) donnés par Alonzo Church[1]. Ces concepts amorcent le développement de langages spécifiques : les langages de programmation.

1.1 L'algorithme comme scénario d'actions

Un algorithme se définit par une suite finie d'opérations organisée en vue d'accomplir une tâche. Trouver un algorithme revient à décomposer la réalisation finale de l'action en une succession d'étapes et de moments intermédiaires et nécessaires. Maurice Nivat, un des pionniers français de la recherche en informatique et algorithmique[2], le résume dans une définition concise : « un algorithme c'est une façon de faire ». Écrire un algorithme revient selon lui à scénariser une manière de s'y prendre pour arriver à un but. Dans cette optique l'algorithme apparaît dédoublé avec d'une part l'aspect descriptif d'une suite d'opérations et d'autre part son accomplissement effectif. En tant que description il exprime un déroulement de l'action ce qui le rapproche d'une narration et, en extrapolant un peu, ce qui semble lui donner la possibilité de raconter quelque chose.

Un algorithme peut traduire et représenter des actions prises dans la plupart des activités humaines et plus largement les algorithmes peuvent fournir des descriptions pour toutes sortes de « processus ». La définition générale d'un processus[3] est presque similaire à celle de l'algorithme : « Suite continue de faits, de phénomènes présentant une certaine unité ou

1 Pour une présentation rapide du lamba-calcul : <https://fr.wikipedia.org/wiki/Lambda-calcul>.

2 Maurice Nivat : <https://fr.wikipedia.org/wiki/Maurice_Nivat>.

3 Celle donnée ici provient du dictionnaire TLFI (Trésor de la langue française informatisée) sur le site CNRTL (Centre national de ressources textuelles et lexicales) : <http://www.cnrtl.fr/definition/processus>.

une certaine régularité dans leur déroulement […] Ensemble d'opérations successives, organisées en vue d'un résultat déterminé. »

Qu'un phénomène puisse s'exprimer sous la forme d'un algorithme ne signifie pas pour autant qu'il puisse toujours donner lieu à un programme. L'algorithme cependant, grâce très largement à l'informatique, ouvre des possibilités d'interpréter, de simuler, de se représenter et par là de comprendre, d'expliquer, d'interroger, de rendre lisibles et intelligibles nombre de phénomènes. Il permet également d'imaginer et d'agir sur le réel. Le système textuel initié par la machine de Turing a ouvert l'espace inouï d'une écriture renouvelée et comme Bernard Chazelle (2013) le souligne « l'algorithme n'est pas tant un objet utile, tel qu'une voiture, un parapluie, ou un cure-dents, qu'une façon différente de penser ».

1.2 Expression et jeu dans des langages spécialisés

Pour extraire un algorithme d'un processus ou d'une action il faut pouvoir les décrire et les analyser. On a donc besoin d'un langage. Le langage à utiliser dépend du domaine : un parent qui apprend à son enfant à s'habiller le fera dans sa langue naturelle. Action ou processus peuvent être décrits dans un langage naturel, ils peuvent être aussi décrits en utilisant un système formel de notation appartenant à une discipline scientifique ou artistique et, afin d'être interfacés avec des machines, ils peuvent être décrits dans des langages de programmation.

Du point de vue d'une culture artistique et littéraire un langage de programmation fusionne des figures de pensée (ce concept est précisé ci-dessous en 2.) avec des contraintes techniques d'une machine pour rendre possible l'écriture de programmes capables de la piloter. Le programmeur se retrouve un petit peu dans la situation d'un marionnettiste : via un langage de programmation il articule et établit un fonctionnement que la machine exécute en différé comme s'il s'agissait d'une marionnette. L'image est assez nette en robotique ; elle l'est un peu moins dans les domaines des réseaux.

Les langages de programmation définissent en général deux catégories de vocabulaire. En nous fondant sur le langage C nous trouvons d'une part un vocabulaire d'instructions primitives (les sauts, les branchements, les boucles) sur la base desquelles le programmeur va construire des instructions plus évoluées, notamment sous la forme de fonctions. D'autre part un

vocabulaire pour le stockage des informations c'est-à-dire pour les données. Il s'agit d'une multiplicité de types de variables qui déterminent des espaces mémoires afin d'enregistrer des valeurs sur lesquelles sont opérées les instructions. La dualité théorique initiale instructions-données du modèle de Alan Turing est palpable dans la plupart des langages de programmation.

Pour chacun de ces langages une grammaire définit et ordonne les possibilités d'écriture comme dans tout langage et ces vocabulaires primitifs sont régis par des syntaxes plus ou moins différentes d'un langage à l'autre. Ces syntaxes sont en général assez simples sans pour autant être aisées à manipuler. En particulier, les rigidités d'un langage de programmation dues à la machine sous-jacente diffèrent des difficultés que l'on peut rencontrer avec des langues naturelles et c'est souvent un aspect déstabilisant pour les apprenants des disciplines littéraires. Cependant tout programmeur apprend à jouer et à maîtriser les formalismes de ses langages de programmation jusqu'à finalement les intégrer, au point de les « oublier » lorsqu'il pense avec eux.

2 Des figures de pensée libres d'interprétation

2.1 Héritage technologique du texte littéraire

La syntaxe d'un texte littéraire supporte et incorpore des figures conceptuelles comme des listes, des tableaux ou des arborescences. Ces figures constituent des moyens de penser, une manière graphique de raisonner et de connaître, une technologie de l'intellect. Jack Goody (1979) fait apparaître que le rôle de ces figures à l'œuvre dans l'élaboration des pensées permet la distinction et la comparaison entre cultures orales et cultures écrites. Écrire ne consiste pas à « enregistrer la parole », écrire c'est aussi « se donner le moyen d'en découper et d'en abstraire les éléments, de classer les mots en listes et combiner les listes en tableaux ».

Les technologies du texte littéraire étudiées par Jack Goody, tableaux, classement, liste et recette, sont toutes présentes dans le texte informatique et dans des versions très augmentées. On y trouve en plus des algorithmes de tri et de nombreuses variations sur les listes. Tout cet ensemble riche et diversifié de figures s'intègre dans ce que l'on appelle « les structures de données ».

Le principe de la recette s'utilise souvent comme métaphore pour illustrer ce qu'est un algorithme : une suite d'instructions à effectuer les unes après les autres comme dans une recette de cuisine. Mais le plus stupéfiant dans ce rapprochement tient à la manière de penser à partir d'une recette qui ressemble à la manière dont on peut jouer avec un algorithme : « [les recettes] sont des textes qui prescrivent une action, qui formulent un programme ; ils conduisent à une extension des connaissances dont disposent le spécialiste et même l'amateur ; ils invitent à ‹mettre à l'épreuve› les recettes (à les ‹tester›, à les ‹expérimenter›) et à procéder à une évaluation comparative des résultats. » (Goody, 1979).

C'est exactement ce que peut être amené à faire un programmeur avec un algorithme pour l'étudier, l'augmenter, le détourner ou l'adapter.

Dans les figures conceptuelles étudiées par Jack Goody il manque l'arborescence, manifeste cependant au VIIe siècle dans les travaux d'Isidore de Séville (devenu d'ailleurs pour cette raison saint patron des informaticiens en 2002). Des arborescences très sophistiquées, avec toutes sortes de formes et d'implémentations, sont intégrées à de nombreux textes informatiques.

2.2 Apport technologique du texte informatique

Sur le même registre littéraire et sur les pas de Jack Goody notre idée consiste à repérer dans le texte informatique des figures intellectuelles à partir desquelles peuvent s'élaborer et s'exprimer des pensées. En nous référant au langage C et à ses dérivés (C++, C#, Objective-C, Perl, Python, Java, php, Ruby, JavaScript, Groovy, Scala, etc.) nous suggérons, afin d'inaugurer une recherche sur des approches littéraires possibles de ces figures, cinq exemples du texte informatique assez caractéristiques et largement employés : le si-sinon, la notion de variables et de nombres, la boucle, la fonction et pour la programmation orientée objet (POO) le principe des classes et des objets. Très brièvement en voici quelques interprétations.

Si-Sinon

Ce formalisme d'apparence logique est solidaire du fonctionnement de la machine et mis au service de ce que l'on souhaite voir faire à la machine, mais il n'enferme pas les raisonnements à l'origine des scénarios projetés sur la machine. « Si le personnage qui entre est vert alors le joueur

s'envole… » aucune logique, juste une invention scénaristique, le si-sinon devient un instrument dont on joue. Il intervient dans le raisonnement mais il n'emprisonne pas le raisonnement.

Variables et nombres

Sous la forme de variables les nombres acquièrent des propriétés remarquables. Les variables traduisent des particularités, des marques, des signes, des attributs, des émotions, etc. toutes sortes de caractéristiques qui permettent par exemple, dans un scénario, de qualifier un personnage, une chose ou une action, les nombres en donnent les valeurs et les intensités. Peur, anxiété, courage, loyauté, jalousie, magie… tout se retrouve traduit numériquement dans des fourchettes de valeurs. Autant de paramètres que l'on peut ensuite apprécier, comparer, mixer et confronter selon les relations souhaitées dans la trame du programme : « s'il a suffisamment de courage, il peut avancer et engager le combat sinon il doit continuer sa quête par ailleurs ». Le nombre devient une matière première, la gouache initiale à laquelle on s'habitue pour imaginer et mettre en scène, il peut représenter quasiment n'importe quoi et se traduire en ce que l'on veut. Il peut être envisagé comme l'équivalent d'un adjectif ou d'un adverbe (« il frappe doucement/fort, il parle lentement/vite »).

Boucle

Un texte algorithmique s'exécute de façon linéaire du début à la fin. La boucle permet de répéter une séquence d'instructions et donc de rompre la linéarité d'un texte en localisant le saut en arrière sur une portion de code facile ensuite à identifier. En général un programme obéit à une boucle principale qui prend fin lorsque l'utilisateur le demande. Elle permet de capturer toutes ses entrées (souris clavier, etc.) et constitue l'essence de l'interactivité.

Cette acquisition intellectuelle de la boucle dans la syntaxe même de l'écriture prend certainement part à la généralisation d'une pensée qui fonctionne par itérations successives et tâtonnements expérimentaux. Son principe se retrouve dans l'organisation circulaire et rétroactive de nombreuses situations de création, de conception et de réalisation des programmes (Drouillon, 2003).

Fonction

Une fonction permet de rassembler en une seule instruction, plusieurs instructions nécessaires à la réalisation d'une tâche. C'est un algorithme complet ou une partie d'un algorithme. Elle apparaît un peu comme un verbe. Elle

peut recevoir des valeurs en entrée et éventuellement retourner des valeurs. Dans un langage non objet une fonction peut être partagée par des données différentes. Pour que Blanche Neige croque une pomme nous écrirons une fonction « croquer » avec par exemple en entrée deux paramètres : une princesse et une pomme. Au moment où la fonction « croquer » est appelée pour être exécutée nous lui fournissons une princesse et une pomme spécifiques. Nous choisirons Blanche Neige ou n'importe quelle autre princesse, avec soit une pomme empoisonnée, soit une pomme enrichie de vitamines, ça dépend des circonstances et du rôle attribué à la pomme dans la fonction.

Classe et objets

Une classe réunit des variables et des fonctions afin de définir une entité complète dans le programme. Cette entité peut-être une chose, un personnage, une opération ou une action. À partir de la classe on décline des « objets » de la classe. La classe agit comme un moule qui permet de déterminer sur le même modèle plusieurs éléments semblables, mais dont les propriétés prennent des valeurs différentes. Blanche Neige serait probablement un objet d'une classe « Princesse » à partir de laquelle nous pourrions également obtenir une méchante reine et différentes autres personnalités, autant d'objets différents de la même classe. Banche Neige aurait une jalousie inoffensive de zéro, la méchante reine une jalousie féroce de cent sur cent.

Un grand intérêt de l'objet est qu'il suppose des mises en relation des objets entre eux. Elles s'effectuent sous forme d'échanges, avec des envois de messages pratiqués grâce aux paramètres en entrée des fonctions membres d'une classe, ou de connaissances mutuelles, avec des possibilités de partage d'adresses mémoire réalisées avec des pointeurs.

Le principe des classes et des objets fournit un autre moteur essentiel pour la pensée que l'on retrouve aussi à d'autres niveaux de l'écriture des programmes. Un objet peut être compris comme un petit programme autonome, sujet à des relations avec d'autres objets, sous-programmes comme lui, et le tout organisé dans un scénario général constitué par le programme dans son ensemble. Le programme général apparaît comme une sorte de réseau de programmes connectables les uns aux autres. Nous trouverions certainement une corrélation historique entre l'apparition des réseaux informatiques et l'évolution des langages vers la programmation orientée objet. Au départ ces propriétés d'échange et de partage entre des objets informatiques ne sont pas explicites dans un langage, elles naissent de l'appropriation des possibilités fournies par la syntaxe de l'écriture et s'imposent avec l'expérience.

Nous pourrions formuler l'hypothèse que progressivement ces échanges et partages entre objets, inhérents à la conception de nombreux programmes, se retrouvent aussi comme figures intellectuelles présentes dans la réflexion à d'autres niveaux : lorsqu'il est question d'échanges et de partages entre les acteurs d'un projet, dans son organisation, mais également dans l'imagination et la formulation de programmes qui interviennent dans les domaines de la communication, des échanges et des partages.

Les langages de programmation informatique, qui se comptent maintenant par milliers, sont à l'origine de nombreuses autres formes et figures qui interviennent dans la pensée. Il semblerait intéressant de repérer leurs influences non seulement sur la manière de programmer mais aussi plus largement sur les manières de penser ainsi que sur des comportements et des éléments de culture qui pourraient en découler.

3 Une activité d'auteur

Dans la pratique programmer n'apparaît pas comme un acte subalterne ou anodin : programmer c'est vivre quelque chose et s'y engager. Des rapprochements entre programmation et activité d'auteur au sens artistique et littéraire sont intéressants pour aborder l'expérience sensible de la pratique du programmeur et la comprendre. La programmation est depuis longtemps présente en art[4] mais l'émotionnel dans la programmation reste toujours très marginal comme objet d'étude ou de recherche. Le rapport de la personne à cette activité et son investissement demeurent *a priori* largement inexplorés. Nous pouvons cependant citer l'exemple du séminaire « codes sources » dont les organisateurs[5] se donnent pour but « de décrire ces œuvres de l'esprit [les programmes] comme des textes à part entière ».

Les personnes qui actuellement rendent compte des possibilités expressives et émotionnelles dans la programmation sont le plus souvent des personnes qui connaissent à la fois une pratique artistique et une pratique

4 Pour un historique précis des rapports entre art et programmation se référer à Lartigaud, 2011.

5 Mélès B. (CNRS, Archives Henri-Poincaré), Fournier-S'niehotta R. (CNAM), Tabourier L. (LIP6, UPMC/CNRS), séminaire code source depuis 2015 : <http://codesource.hypotheses.org/>.

de la programmation. L'intelligence des deux pratiques leur permet d'opérer des rapprochements et des comparaisons.

Vikram Chandra (2014) est un romancier qui a financé ses études littéraires en travaillant comme informaticien. Dans un ouvrage étonnant, *Geek sublime*, il montre comment ses expériences d'informaticien et de romancier s'enchevêtrent et s'entrecroisent. Il témoigne de connexions sensibles entre la programmation et des arts : « parce qu'ils [les programmeurs] créent des objets complexes, et ne se soucient pas seulement de leurs fonctions, mais aussi de leur beauté, ils agissent exactement comme les peintres et les sculpteurs ».

Paul Graham, un informaticien de formation littéraire ayant de surcroît étudié la peinture à l'académie des beaux-arts de Florence, met en évidence dans son livre (2004) des points communs entre ses expériences de la programmation et sa connaissance de la peinture. Dans cet ouvrage il s'intéresse aux « hackers » c'est-à-dire, selon la terminologie anglo-saxonne, les amateurs au sens fort, ceux qui aiment l'informatique et font de la programmation une pratique inventive. Il affirme : « en fait, de tous les différents types d'individus que j'ai rencontrés, les hackers et les peintres sont ceux qui se ressemblent le plus. Les uns et les autres ont ceci en commun : ce sont tous des créateurs. Comme les compositeurs, les architectes et les écrivains, ce qu'ils tentent de faire, c'est de créer des objets qui soient réussis ».

De nombreux aspects émotionnels propres à la musique se transposent également assez facilement ou se retrouvent en programmation informatique. Dans notre expérience de musicien compositeur et de développeur les deux pratiques s'éclairent mutuellement et c'est parfois inattendu. Par exemple un frisson similaire à celui que procure parfois la musique peut être provoqué en programmation. On peut être touché par une expérience très spécifique de beauté et de compréhension, ce qui témoigne d'aspects émotionnels et psychologiques qui entrent dans le rapport à la programmation. On peut lire Mihaly Csikszentmihalyi (2006) pour une réflexion sur le flow et la créativité qui intéresse la programmation.

David-Olivier Lartigaud constate l'apparition depuis les années 2000 d'une génération d'artistes qui opèrent un retour aux codes sources des programmes informatiques. Des artistes s'approprient un ou plusieurs langages de programmation pour en faire le fondement d'une recherche et d'une écriture. Ils s'engagent dans une expérience artistique de pratique de la programmation. La programmation pour eux ne relève pas de l'utilitaire, elle relève de l'écriture et de la composition. Cette intimité avec la

programmation suscite de leur part un regard décalé sur elle ainsi que sur les cultures scientifique et technique qui lui sont en général associées.

4 Une entrée pragmatique dans la programmation

L'expérience montre que la programmation informatique peut être ouverte à toutes sortes de personnes quels que soient leurs âges et leurs formations : des enfants, des collégiens, des lycéens ou des adultes de culture littéraires, scientifiques ou artistiques[6]. Leurs approches, leurs objectifs ainsi que le niveau des performances diffèrent mais cette diversité de publics révèle néanmoins un espace de la programmation qui n'est pas monolithique. On peut observer également que l'entrée dans la programmation n'est pas nécessairement assujettie à des prérequis scientifiques.

Cette large ouverture possible, similaire à celle que l'on retrouve par exemple en musique avec l'apprentissage des instruments de musique, se relie probablement au fait que la programmation s'acquiert essentiellement par la pratique. Effectivement, apprendre à programmer c'est un peu comme apprendre à jouer d'un instrument de musique. Dans ces domaines le savoir théorique est indissociable de l'expérimentation répétée à travers des réalisations diverses et variées et de différents niveaux. En programmation, avec le temps, la pratique seule permet d'acquérir une intuition spécifique qui permet plus facilement de repérer ou de poser des problèmes et de trouver des solutions ensuite.

4.1 Distinction entre langage et discours

Pour les apprentissages de la programmation il apparaît très utile de distinguer d'une part le langage comme technologie du discours et d'autre part le discours comme une production intentionnelle associée à l'exercice du langage. La partie langage donne des possibilités d'architecture et de

6 Nous nous référons ici à différents ateliers de programmation que nous avons animés en bibliothèques auprès d'enfants, de collégiens et de lycéens du CM2 à la terminale, auprès d'adultes amateurs de programmation, dans le cadre de cours à l'université d'Arras en licence de lettres modernes et également dans l'école supérieure d'informatique cs2i Bourgogne de Nevers, en licence et mastère d'informatique.

construction, la partie discours fait appel à des connaissances externes. L'élaboration et la réalisation d'un programme associent les deux.

Lors de l'apprentissage de la programmation les premières expérimentations d'un langage portent sur des discours simples. Moment clé pendant lequel l'apprenant se confronte à l'acquisition d'un langage doté d'une syntaxe pour lui assez rude et surtout porteuse de figures de pensées inusitées dans les langues naturelles. Ces nouveautés de syntaxes et de figures (quelques-unes décrites en 2.2) lui rendent inaccessible une trop haute complexité de discours. Quel que soit son âge il se trouve dans la situation d'un élève de CP et CE1 lors des premiers apprentissages de la lecture et de l'écriture : toute son énergie intellectuelle passe à intégrer codage et décodage ce qui rend l'accès au sens extrêmement laborieux. La complexité des discours s'ajoute progressivement, élaborée la plupart du temps avec des connaissances totalement extérieures au langage de programmation lui-même.

Ces autres connaissances nécessitent souvent des apprentissages extrinsèques. Par exemple maîtriser des modèles mathématiques pour des traitements de l'espace ou de l'image, des modèles linguistiques dans le cadre du traitement automatique de la langue, des problématiques de fouilles de données, de big data, de recherches ou d'activités professionnelles variées… Ces modèles pour être portés par un langage de programmation nécessitent des dispositions appropriées, ils doivent pouvoir *se traduire* dans un langage de programmation. Cette reconfiguration et ce portage d'un modèle externe en un discours opérationnel sur machine exigent de confronter connaissance et compréhension du sujet traité d'une part et maîtrise du langage de programmation d'autre part.

4.2 Comprendre n'est pas savoir faire

Pour les débutants confrontés à un langage de programmation, en général comprendre ne suffit pas. Il faut aussi savoir faire et dans ce domaine comprendre n'est pas savoir-faire (Drouillon, 2014).

Pour prendre une image, c'est un peu la même situation pour apprendre à jouer d'un instrument de musique. Si un professeur ou un tutoriel vidéo explique comment jouer de la guitare, vous pouvez comprendre les explications, mais vous n'êtes pas devenu guitariste ni compositeur de musique. Devenir guitariste suppose de nombreuses heures de travail sur

son instrument. Devenir compositeur et avoir des idées de morceaux de musique ou d'arrangements musicaux supposent également beaucoup de questionnements et de travail. C'est la même chose en programmation et conception informatique.

Écrire un programme nécessite un minimum d'expertise des outils fondamentaux, c'est-à-dire un minimum d'expérience et de temps consacré à l'activité. De notre point de vue, l'acquisition progressive d'une maîtrise des fondamentaux, même incomplète, permet seule de faire des hypothèses de réalisation parce qu'elle produit petit à petit cette compétence d'identifier et de poser les problèmes avec pertinence, ce qui permet d'inventer des solutions. En effet, trouver une solution pour résoudre un problème revient souvent à créer cette solution. Il s'agit d'un processus d'élaboration qui nécessite au plan de l'écriture de l'intuition, de la recherche et de la créativité.

4.3 Un apprentissage non linéaire

D'une façon générale, l'apprentissage n'est pas réductible à une mécanique simple. Pour l'apprenant la clarté se fait progressivement un peu comme un brouillard qui se lève. Au départ il ne distingue rien et petit à petit, au fur et à mesure que le brouillard se dissipe, des formes apparaissent de plus en plus précises et claires. Il arrive que certains se sentent perdus au départ comme devant un mur lisse sur lequel ils n'ont aucune prise, peut-être un peu la même impression que dans un pays étranger en entendant une langue que l'on ne maîtrise pas bien et que l'on s'efforce de comprendre.

Ces phénomènes s'expliquent probablement par l'absence initiale de référent à la pratique d'un langage de programmation. Tout se passe comme si le débutant était totalement étranger à l'espace de pensée ouvert par le langage de programmation. Pour lui le langage ne se réfère à rien et du coup il ne voit pas comment le « parler », il n'arrive pas à le relier avec du sens, avec ce qu'il souhaite faire. De ce fait l'apprentissage n'est pas frontal, direct, facile à planifier. Il ne s'agit pas pour l'apprenant d'apprendre des solutions, il s'agit pour lui d'apprendre à les trouver par lui-même. Pour l'apprenant, qui se trouve obligé d'entrer dans une nouvelle activité de penser, il est question d'une véritable conquête. Avec le temps, des expérimentations et de l'entraînement, des pôles de clarté, de compréhension et de savoir-faire s'affirment et se relient doucement entre eux.

5 Conclusion

La programmation informatique passe par des langages qui répondent à différentes sortes de grammaires (syntaxes) à partir desquelles nous pouvons élaborer du sens et exprimer des points de vue sur toutes sortes de sujets. La programmation, matière scientifique et technique, n'apparaît pas seulement sous le prisme de langages formels et scientifiques, elle apparaît également comme une écriture qui permet de placer l'imaginaire et la sensibilité émotionnelle au cœur de l'élaboration de multiples réalisations techniques. Cette écriture qui n'a rien de mécanique permet de s'exprimer et de produire des récits sous forme de scénarios d'actions algorithmiques aux prises avec l'imaginaire plus général des actions possibles. Elle peut être abordée de multiples façons. Elle peut notamment être appréhendée de façon littéraire, artistique et philosophique du point de vue de la maîtrise technique comme du sens parce qu'elle se travaille comme un art, comme un instrument de musique par exemple. Sa plasticité conceptuelle est considérable et elle autorise de nombreuses et originales associations entre arts, philosophie, sciences, technologie.

Références

Chazelle, B. (2013). *L'algorithmique et les sciences.* Fayard et Collège de France.

Csikszentmihalyi, M. (2005). *La créativité, psychologie de la découverte et de l'invention*. Robert Laffont.

Drouillon, F. (2014). *Du C au C++, de la programmation procédurale à l'objet.* ENI.

Drouillon, F. (2003). *Chimères et gargouilles informatiques.* Thèse de doctorat Paris 8, diffusion ANRT.

Goody, J. (1979). *La raison graphique.* Les éditions de minuit.

Graham, P. (2004). *Hackers & painters, big ideas from the computer age*. O'Reilly.

Chandra, V. (2013). *Geek sublime*, *Une vision esthétique, littéraire, mathématique et pleine d'autodérision du codage*. Robert Laffont.

Lartigaud, D. (dir.) (2011). *Art++*. Éditions HYX.

II.
Formation des enseignant·e·s et enjeux institutionnels

Isabelle Debled-Rennesson[a], Philippe Févotte[b],
Monique Grandbastien[a], David Langlois[a] & Hélène Tanoh[b]

a. Université de Lorraine, LORIA, Vandœuvre-lès-Nancy, F-54506, France
prenom.nom@univ-lorraine.fr, prenom.nom@loria.fr
b. Académie de Nancy-Metz
prenom.nom@ac-nancy-metz.fr

La formation des professeurs de la spécialité ISN dans l'Académie de Nancy-Metz – Récit et analyse de six ans d'expérience

Résumé

L'informatique et plus généralement les sciences du numérique entraînent des mutations profondes de nos sociétés sur les plans personnel, professionnel, culturel. Les choix économiques et politiques qui en découlent supposent des citoyens qui maîtrisent les mécanismes fondamentaux qui régissent ces mutations. Cette nécessité commence à être perçue, en témoignent notamment des changements à grande échelle dans les curriculums des enseignements primaire et secondaire au Royaume-Uni et en France depuis la rentrée 2016.

Le présent document a pour objectif de tirer parti de l'expérience acquise au cours de six années de formation complémentaire continue d'enseignants de lycée pour la spécialité Informatique et sciences du numérique (ISN) créée pour la rentrée 2012 en France et de formuler quelques préconisations pour la formation continue des professeurs appelés à enseigner des éléments d'informatique en école, collège et lycée (6 à 18 ans en France). Cette réflexion peut être enrichie par les approches mises en œuvre dans d'autres pays et déboucher vers une véritable didactique de l'informatique au niveau scolaire.

L'article décrit successivement le contexte local d'organisation de la formation, ses contenus pédagogiques, son public, puis l'organisation de l'habilitation à enseigner ISN et l'implantation de cette spécialité dans les établissements du secondaire. Nous montrons enfin les évolutions de la formation, sa continuation dans le cadre de la formation continue et proposons des perspectives visant à élargir la formation à une plus grande diversité des publics.

Mots clés : formation des maîtres en informatique, spécialité ISN (Informatique et sciences du numérique), pensée informatique

1 Introduction

À une période où la formation de tous les futurs citoyens aux bases de l'informatique, la discipline qui constitue le socle du monde numérique, s'impose, la question de la formation d'enseignants compétents est plus que jamais cruciale. Après des décennies de tentatives diverses (Baron *et al.*, 2014), l'enseignement de l'informatique a fait à nouveau une apparition limitée en France dans les classes terminales des lycées généraux et technologiques sous la forme d'une spécialité[1] intitulée Informatique et sciences du numérique (ISN) à la rentrée 2012.

Selon les textes officiels[2], ISN propose à raison de 2 heures par semaine une introduction à la Science Informatique : information numérique, algorithmes, langages, architectures. Les objectifs sont de développer des compétences de base dans le domaine de l'informatique, donner le goût des sciences du numérique lors d'activités variées : travaux pratiques, projets, exposés et débats, développer la rigueur en apprenant les bases de la programmation, clé de la maîtrise des ordinateurs, s'interroger sur la qualité, la sûreté, la fiabilité et la sécurité des données numériques, identifier et s'interroger sur les progrès, les avantages et les risques que génère la société numérique. L'apprentissage par la réalisation de projets est privilégié.

Cet enseignement a été confié à des professeurs volontaires, de différentes disciplines scientifiques (mathématiques, sciences physiques et chimiques) et technologiques (sciences et techniques industrielles, etc.), habilités à l'issue d'une formation complémentaire organisée à l'initiative des académies[3].

Le présent article a pour objectif de compléter les descriptions déjà proposées pour d'autres centres en France, notamment (Cogis et al., 2015) à Montpellier, et de contribuer à la réflexion commune en présentant la formation complémentaire mise en place à Nancy depuis la rentrée 2011 pour

1 Les élèves de terminale S choisissent entre plusieurs spécialités (Mathématiques, Physique-chimie, Sciences de la Vie et de la Terre, ISN), les terminales STI prennent ISN en option.

2 Site du ministère de l'Éducation à propos d'ISN : <http://eduscol.education.fr/cid59678/presentation.html>.

3 En France, on appelle académies les divisions administratives du territoire pour l'Éducation Nationale. Il y 30 académies, y compris celles d'outre-mer, chacune d'elles est administrée par un Recteur représentant le Ministre.

des enseignants déjà en poste. Nous décrivons successivement le contexte local d'organisation (collaboration université-rectorat, contenus, mise en œuvre, évolutions, facteurs favorables et défavorables), le contenu de la formation initiale, les données académiques (nombres d'enseignants formés, nombres de classes ouvertes), les actions de formation continue organisées chaque année pour prolonger et consolider la formation de base, ainsi que quelques réflexions et préconisations à propos de telles formations. Sur le plan des perspectives, ces différentes remarques, ajoutées à l'étude d'expériences d'autres pays (Yadav *et al.*, 2014 ; Brodnik, 2016), pourraient servir également à la définition de formations continues au-delà du cas particulier de la spécialité ISN, ainsi qu'à la conception de formations initiales, par exemple pour le CAPES Mathématiques-Informatique dont la première session s'est achevée en juin 2017.

2 Organisation de la formation des enseignants ISN dans l'académie de Nancy-Metz

Les actions de formation et certification d'enseignants nécessaires à la mise en place d'ISN dans les académies sont définies dans la note de service n° 2011–178 (BO 2011).

Dans l'académie de Nancy-Metz, la formation des enseignants a été construite en étroite collaboration entre les responsables académiques et l'Université dès la rentrée 2011, avec l'appui du LORIA[4] et de l'IUFM/ESPE[5] de Lorraine dans la perspective de constituer un premier vivier de professeurs compétents.

Une enseignante informaticienne a pris en charge la définition du contenu de la formation et a assuré avec plusieurs de ses collègues la totalité de la formation, l'IUFM/ESPE et l'Université de Lorraine prenant en charge le défraiement des formateurs.

L'implication forte des différents partenaires (Université de Lorraine, via l'ESPE, le LORIA) a permis de proposer une formation complète sur

4 LORIA : Laboratoire lorrain de recherche en informatique et ses applications ; un des laboratoires d'informatique de l'université de Lorraine.

5 IUFM : Institut universitaire de formation des maîtres, réorganisé ensuite en ESPE : École supérieure du professorat et de l'éducation.

deux ans. À cela s'ajoute un suivi *via* l'organisation de journées annuelles de séminaires et d'ateliers venant compléter la formation en l'ouvrant aux problématiques de recherche et sociétales de l'informatique.

Deux niveaux de formation ont été proposés sur deux ans. Le niveau 2 peut aborder des notions plus complexes ou avancées que le niveau 1, mais le niveau 2 permet aussi d'étudier des notions importantes (comme les bases de données) non vues en année 1. Au cas par cas, certains professeurs, déjà compétents sur certaines notions, ont pu être exemptés en partie de la formation. Cette exemption de formation a pu être totale et remplacée par un protocole de validation des acquis. La formation a lieu principalement en présentiel avec un travail personnel conséquent (ce point sera développé par la suite). Nous avons aussi expérimenté une modalité de formation à distance pendant plusieurs années afin de rationaliser les déplacements, à la charge des professeurs formés. Pour cela, nous avons adapté nos supports en les commentant afin de pallier l'absence de l'enseignant. Nous avons aussi organisé avec les groupes formés à distance deux journées par an de rassemblement pour faire un bilan sur les modules passés et préparer les modules à venir. Malgré cet investissement, nous n'avons pu que constater une grande difficulté pour les professeurs à suivre le rythme de la formation. Une autre difficulté a été le peu d'interactions (questions) entre enseignants et professeurs formés du fait du temps lourd à rédiger une question écrite par rapport au fait de la poser oralement devant un écran. Nous pensons que pour mieux accompagner les professeurs, il aurait fallu une refonte globale des supports de formation pour les adapter à un apprentissage en autonomie (ce qui aurait demandé un investissement trop lourd par rapport aux ressources disponibles) ainsi que la banalisation de moments de formation hebdomadaires synchronisés à distance.

La formation est prise en charge principalement par deux enseignants de l'ESPE de Lorraine, tous deux ayant le même degré de connaissance sur l'organisation de la formation. Cela permet de renforcer la cohérence de la formation puisque les professeurs formés se retrouvent souvent devant les mêmes interlocuteurs. Toutefois, nous faisons aussi appel à des « experts » externes (enseignants-chercheurs et chercheurs) sur des problématiques ciblées : réseau, cryptographie, robotique. Ce dernier point favorise une ouverture vers d'autres points de vue et vers la recherche (ouverture amplifiée par les journées ISN/EPI, voir Section 8).

3 Le contenu de la formation

Le contenu a été construit à partir de l'expérience des formateurs et d'une première édition de Dowek *et al.* (2013) avec l'objectif de donner aux professeurs formés les compétences nécessaires pour enseigner le programme d'ISN et évaluer les élèves pour une épreuve au baccalauréat (oral individuel à partir du dossier de projet).

Les intitulés des Unités d'enseignement (UE) des années 1 et 2 sont listés respectivement dans les tableaux 1 et 2. La répartition du volume horaire entre les unités d'enseignement est restée relativement stable même si des ajustements ont pu être faits suite à l'apparition de besoins, ou le désistement d'intervenants extérieurs. Nous décrivons et commentons ci-dessous certains aspects de cette formation.

Tableau 1 : Intitulé des Unités d'Enseignement en année 1.

	Nb heures
UE1 : Algorithmique et programmation	
Bases de l'algorithmique et de la programmation	31,5
Image et Géométrie	10,5
Cryptographie	7
Total UE1	49
UE2 : Architecture et représentation de l'information	
Représentation de l'information	10,5
Architecture	7
Total UE2	17,5
UE3: Initiation au réseau et à la programmation web	
Réseau 1	10,5
Programmation web 1 : HTML, CSS et JavaScript	21
Total UE3	31,5
Total Année 1 (présentiel)	**98**
Projet année 1 : 100 heures de travail personnel	

Tableau 2 : Intitulé des Unités d'Enseignement en année 2.

	Nb heures
UE1 : Algorithmique et programmation objet	
Programmation et conception objet	31,5
Structure de données et algorithmique avancée	14
Total UE1	45,5
UE2 : Conception de modèles	
Programmation web 2 : bases de données et PHP	21
Langages	7
Total UE2	28
UE 3: Système, réseau, robotique	
Réseau 2	7
Système	7
Robotique	7
Total UE3	21
Total Année 2 (présentiel)	**94,5**
Projet année 2 : 100 heures de travail personnel	

La formation débute par l'algorithmique en année 1. Elle aborde de manière classique les notions de base en algorithmique impérative : séquences d'instructions, variables, expressions booléennes, instructions conditionnelles et répétitives. La notion de sous-programme est aussi abordée. Le langage support choisi est Python. Ce choix repose sur deux raisons principales. D'abord, il s'agissait d'une préconisation nationale. Ensuite, nous avons considéré que Python permettait aux apprenants d'obtenir des résultats plus rapidement à l'écran du fait que le langage est associé à une ligne de commandes interprétée. Toutefois, ce choix a prêté à discussion car Python s'éloigne des habitudes algorithmiques et peut donc déstabiliser. En effet, Python est très souple et permet d'obtenir des résultats de plusieurs manières dont certaines sont très compactes et peu lisibles (nous pensons notamment aux listes définies en intention). Par ailleurs, le typage du langage est faible, ce qui cache en partie les problématiques du typage en informatique. Enfin, la notion de passage par valeur (copie) ou référence (adresse) n'est pas explicite, et dépend du type de données passé en paramètre d'une fonction (type mutable ou non). Sur le plan pédagogique,

Python pose donc quelques difficultés. En parallèle de la découverte de l'algorithmique et de la programmation en Python, nous abordons la programmation d'interfaces afin que les professeurs puissent rapidement créer des applications finalisées et attrayantes (donc motivantes, à notre avis). Cette formation algorithmique se poursuit en année 2 avec la découverte de la programmation objet, les patrons de conception et d'architecture (patron Modèle-Vue-Contrôleur), l'étude de structures de données classiques (arbres, graphes) ainsi que les différents paradigmes de programmation : récursivité, algorithmes gloutons, programmation dynamique, etc.

La formation algorithmique est articulée avec une découverte de l'architecture des machines et de la représentation numérique de l'information. Cette partie permet de comprendre pourquoi le binaire est utilisé en informatique (en partant du transistor), et comment les données de différentes natures (valeurs numériques, textes, images, sons) sont codées numériquement. Au-delà des aspects « techniques », nous initions une réflexion sur le rôle d'un code, d'une norme. Les théories de la quantité d'information ainsi que les méthodes de compression sont aussi abordées. L'ensemble donne lieu à des exercices de programmation.

Nous avons aussi développé un module « Images et Géométrie » permettant d'aborder le codage numérique des images (matriciel et vectoriel), le codage des couleurs, ainsi que les différents traitements classiques sur les images (échantillonnage, histogrammes, filtrage, compression, opérations morphologiques). Ce module aborde aussi l'extraction d'information (segmentation, composantes connexes, contours, présence de droite discrète).

La formation au réseau comprend deux parties. La première partie, qu'on pourrait qualifier de « bas niveau », aborde les couches OSI et la communication de l'information entre machines. La deuxième partie aborde Internet, le Web et surtout la programmation Web : HTML, feuilles de style CSS, JavaScript en première année, puis bases de données et PHP en deuxième année.

4 La formation et son public

L'ensemble de la formation atteint un volume de 200 heures approximativement (hors projet). Elle aborde de nombreux sujets. Le programme vise la même ambition que des formations informatiques de niveau Licence.

C'est donc une formation exigeante pour les professeurs. Ceux-ci doivent suivre une séance hebdomadaire de 3 h 30, et doivent travailler entre les séances. À ceci s'ajoute un projet de programmation par an donnant lieu à une soutenance devant les formateurs et les pairs. Précisons par ailleurs que les professeurs n'ont pas de décharge de service pour cette formation, et doivent donc accomplir leur service sur le reste de la semaine. Malgré cela, l'investissement des professeurs est très fort. Leur niveau d'écoute et de participation est particulièrement élevé. Le temps de travail consacré aux projets étonne souvent. Enfin, les professeurs sont demandeurs au-delà de la formation sur les deux années : demande d'approfondissement, de découverte de nouveaux langages de programmation, forte participation aux journées ISN/EPI (voir Section 8).

Comme le montre le tableau 3, la formation renouvelle son public d'année en année. En présentiel, nous recevons entre 15 et 20 nouveaux candidats chaque année. Quand un niveau 1 n'a pas été ouvert, ce fut du fait d'un manque de moyens, et non d'un manque de candidatures. D'ailleurs, cette année 2017–2018, nous avons ouvert deux groupes de niveau 1 sur deux sites éloignés, permettant ainsi de réduire les déplacements des professeurs à travers la région Lorraine, très étendue du Nord au Sud, et ainsi de pallier la disparition de la modalité de formation à distance.

Tableau 3 : Effectif du public selon les années, ayant suivi la formation ISN ou ayant été exempté de formation. Les inscrits ayant abandonné en cours d'année, ou ayant été trop absents (concerne la formation en présentiel), sont exclus du recensement.

	Niv1 présentiel	Niv1 distance	Niv2 présentiel	Niv2 distance	Effectif total niveau et toute modalité
2011–2012	20	Non ouvert	11	Non ouvert	31
2012–2013	14	16	15	Non ouvert	45
2013–2014	Non ouvert	15	10	14	39
2014–2015	16	Non ouvert	Non ouvert	12 (année 2 sur 2 ans)	16
2015–2016	14	Non ouvert	11		37
2016–2017	Non ouvert	Non ouvert	11	Non ouvert	11
2017–2018	31	Non ouvert	Non ouvert	Non ouvert	31

À la fin de l'année scolaire 2016–2017, 106 enseignants auront reçu une formation solide dans le domaine des sciences du numérique (sans comp-

ter les 31 qui entrent en formation cette année 2017–2018). Notons que d'autres professeurs ont été dispensés de formation, mais sont passés par un protocole de validation des acquis de l'expérience.

Les professeurs formés viennent de trois disciplines : Mathématiques (Maths), Sciences physiques et chimie (SPC), et Sciences et techniques de l'ingénieur (STI). Le tableau 4 indique la répartition dans chacune de ces 3 catégories des professeurs inscrits à la formation. On remarque que les disciplines sont scientifiques (au niveau national, on cherche maintenant à ouvrir l'informatique aux autres disciplines – voir Section 9), avec une prédominance des mathématiques et une contribution tout de même élevée (et équilibrée) des SPC et STI. En pratique, même si le public n'est pas toujours mathématicien, l'attrait pour un peu de théorie est suffisant pour aborder les grands principes informatiques. Dans la formation, nous ne cherchons pas nécessairement à théoriser les notions, la pratique reste centrale, mais quand des notions théoriques sont abordées (en complexité par exemple), il n'y a pas de rejet de la part du public.

Tableau 4 : Effectif des professeurs inscrits à la formation ISN selon leur discipline. Le total est de 127, ce qui est plus élevé que les 106 annoncés précédemment. Ce différentiel est dû dans ce présent tableau à la prise en compte des inscrits à la formation, comptant donc ceux qui ont démissionné en cours d'année.

Discipline	Effectif
Maths	55
SPC	38
STI	34

5 L'habilitation à enseigner cette spécialité

Une procédure en deux temps, définie dans la note déjà citée (BO 2011) prévoit : une autorisation provisoire à enseigner pour les enseignants formés ou en formation, délivrée par le groupe de pilotage constitué des trois IA IPR[6] (mathématiques, physique et chimie, STI) et les enseignants universitaires

6 IA IPR Inspecteur d'académie inspecteur pédagogique régional.

en charge de la formation, suivie d'une validation définitive qui intervient à l'issue de la seconde année de pratique de l'enseignement ISN ; elle s'appuie sur une visite d'inspection durant la période.

Tous les enseignants qui ont suivi la formation avec assiduité et qui ont validé leurs projets de fin d'année, ou qui sont actuellement en formation, ont obtenu cette autorisation à enseigner.

Une inspection par des IPR des disciplines mathématiques, physique–chimie ou STI a été effectuée depuis la rentrée 2012, parfois lorsque c'était possible par deux inspecteurs, pour 43 enseignants qui assurent actuellement l'enseignement de cette spécialité. Les inspections se poursuivent à l'heure actuelle.

Pour attester de cette habilitation et afin de valoriser le fait d'avoir suivi la formation, nous délivrons un document en deux volets. L'un atteste de la réussite du professeur formé (décision fondée sur le suivi de formation et la réussite des projets), l'autre liste les modules de formation suivis par le professeur. Ce document est complété par l'Université de Lorraine et est signé par les représentants du rectorat et par la direction de l'ESPE de Lorraine.

6 L'implantation de cette spécialité

L'inspection générale et le ministère ont souhaité pour la première année une « implantation contrôlée » dans environ un quart à un tiers des établissements d'une académie. Dans l'académie de Nancy-Metz, cette spécialité a été proposée initialement dans 21 établissements publics et 5 établissements privés. À la fin de l'année scolaire 2016–2017, cette spécialité était proposée dans 38 établissements sur 60 (public) et 4 sur 19 (privé) ; 900 élèves ont choisi cette spécialité. Soixante enseignants sont impliqués dans cet enseignement (certains établissements ont une ressource humaine ISN de 3 professeurs). Cinquante sont habilités à l'enseignement de l'ISN.

Les élèves de terminale S choisissent entre plusieurs spécialités : Mathématiques, Physique–Chimie, Sciences de la Vie et de la Terre, ou ISN. Le tableau 5 donne l'évolution de la répartition au cours des années 2011–2015 des élèves de Terminale dans ces spécialités. Ces statistiques portent sur les seuls établissements proposant les 4 spécialités.

Tableau 5 : Répartition au cours des années 2011–2015 des élèves de Terminale S dans les spécialités du Baccalauréat offertes en classe de Terminale S.

	Mathématiques		SVT		Physique–chimie		ISN	
Rentrée 2011	395	19 %	924	45 %	738	36 %	Non ouvert	
Rentrée 2012	367	17 %	882	41 %	576	27 %	329	15 %
Rentrée 2013	697	20 %	1403	40 %	800	23 %	571	16 %
Rentrée 2014	670	18 %	1445	40 %	892	24 %	657	18 %
Rentrée 2015	942	21 %	1772	41 %	908	20 %	781	18 %

Les implantations ont été validées par le Recteur ; les choix se sont faits en s'appuyant sur les critères suivants :

- repérage préalable des compétences dans les établissements ;
- recensement dans les établissements des intentions d'ouvertures et des projets envisagés ;
- analyse de la pertinence des projets présentés (conformité aux attentes des programmes, au moins deux disciplines représentées, au moins un professeur ayant suivi une formation) ;
- maillage académique des ouvertures dans les bassins de formation.

Il faut noter que l'ouverture de cette spécialité n'a pas d'influence sur la dotation attribuée aux établissements et qu'elle se fait donc à moyens constants.

7 Évolutions de la formation

La formation a déjà évolué face aux retours du public.

Premièrement, nous avons gommé les excès de théorisation quand cela a été nécessaire. Il ne faut pas oublier que les professeurs se retrouvent rapidement devant des lycéens (parfois ils ont déjà une classe en charge dès le début de la formation). Les professeurs ont besoin d'être opérationnels très rapidement. Leur première inquiétude est de se montrer compétents en programmation. C'est donc à ce besoin que nous répondons en favorisant l'algorithmique, la programmation et les exercices.

Deuxièmement, comme déjà précisé plus haut, nous avons voulu proposer une modalité de formation à distance. Cette modalité a été utile pour limiter les déplacements des professeurs formés (certains professeurs ont renoncé à s'inscrire soit du fait de la masse de travail, soit du fait du poids des déplacements, qui plus est, non remboursés). Mais elle demandait aux professeurs une motivation trop forte. De fait, beaucoup ont abandonné. De notre côté, nous ressentions fortement l'inadéquation de nos supports (malgré une adaptation) et l'impossibilité pour nous de proposer une formation de type MOOC ou SPOC[7] (un tel investissement n'avait pas été prévu). En revanche, pour l'année 2017–2018, nous avons décidé d'ouvrir un groupe de niveau 1 sur Nancy, et un autre sur Metz. Cela permet de diminuer les distances parcourues. De fait, beaucoup plus de professeurs se sont inscrits. Quant à un retour de la modalité à distance, cela reste envisageable, mais il faudra alors s'en donner les moyens en offrant une plateforme de formation *ad hoc* ainsi que des interventions synchrones à distance. Cela demande un investissement fort en infrastructure web et en suivi humain (équipe technique, et enseignants) et en formation des enseignants devant créer et animer le cours à distance.

Dans le futur, et à la demande des professeurs formés, nous voulons poursuivre l'offre de formation au-delà des deux premières années. Pour cela, nous voulons proposer des mini-modules de formation continue. Nous avons déjà pensé à un module centré sur la démarche de projets en programmation (outils de travail collaboratif, GIT, méthode agile…). Cette proposition a été accueillie favorablement par le public cible. Cette offre de formation viendrait compléter les journées ISN/EPI, dont le principe est décrit dans la section suivante.

8 Formation continue : journées ISN/EPI

Le programme de formation ne pouvait atteindre le niveau Licence d'informatique qui était souhaité par l'université ; de plus les champs de l'informatique et du numérique évoluent très vite, aussi nous avons considéré qu'il était nécessaire de proposer une actualisation régulière de ces formations sous la forme d'une journée annuelle. Cette journée est organisée et offerte

7 MOOC : Massive Open Online Course ; SPOC : Small Private Online Course.

depuis 2012 par le LORIA et est inscrite au plan de formation continue des enseignants de l'académie. Elle a été ouverte dès 2013 à des représentants d'académies voisines. L'objectif de ces journées a été d'apporter des compléments de formation et de permettre des échanges et réflexions pédagogiques entre les enseignants, elles comprennent des conférences en amphithéâtre et du travail par petits groupes en ateliers (2 dans la journée). Elles ont accueilli 50 participants dès 2012 pour culminer à 115 inscrits en 2017.

Le programme est élaboré avec des enseignants et chercheurs en informatique ainsi que des enseignants d'ISN, les principaux besoins évoqués par ces derniers ont évolué au fil des années. En effet, l'évaluation des élèves était la préoccupation les premières années puis, avec l'expérience, des ateliers d'échanges de pratiques, pilotés par les enseignants d'ISN, ont aussi été proposés. Diverses activités débranchées introduisant des concepts liés à l'informatique (algorithmique, bases de données, compression, cryptage, etc.) ont été proposées dans le cadre des ateliers (par exemple : *Introduction à l'algorithmique avec le jeu Cargo-Bot*[8], *Résoudre une enquête à l'aide de bases de données*, *Des dés pour coder*, *Turing : sa machine version papier*, etc.). Des collègues universitaires ont présenté pendant les ateliers les logiciels qu'ils avaient développés et qu'ils utilisaient dans leurs pratiques pédagogiques : *ArtEoz*[9] (Gautier & Wrobel-Dautcourt, 2016), permettant de visualiser l'exécution d'un programme écrit en Java ou en Python – *PLM*[10] (Quinson & Oster, 2015), *un exerciseur pour Java et Python* utilisé pour les premiers apprentissages de la programmation. Les aspects sociétaux, comme la sécurité, le cryptage des données, Internet et les réseaux sociaux, l'évolution de la pensée informatique sont aussi abordés pendant ces journées dans le cadre des conférences. La présentation d'activités pédagogiques utilisant de nouvelles technologies comme les drones, les robots, les smartphones et tablettes remporte aussi beaucoup de succès et permet d'ouvrir les enseignants présents vers de nouvelles pratiques. Enfin, de nouveaux concepts informatiques (comme par exemple l'intelligence artificielle) non abordés pendant la formation ISN sont présentés pendant ces journées à travers des exposés ou des activités *clés en main* (par exemple *Résolution de jeu de stratégie abstrait*, *Moteurs physiques*, *Jeux vidéo et simulateurs de comportement*, etc.) permettant aux enseignants d'enrichir leur culture informatique et les propositions de projets donnés aux élèves.

8 <http://www-verimag.imag.fr/~wack/CargoBot/>

9 <http://arteoz.loria.fr/>

10 <http://people.irisa.fr/Martin.Quinson/Teaching/PLM/>

Des questionnaires bilans sont remplis par les participants et nous faisons évoluer les contenus des journées en fonction des retours obtenus.

Des documents relatifs à ces journées ainsi que leurs programmes sont accessibles en ligne[11].

9 Perspectives

Les perspectives sont évidemment nombreuses, nous nous limitons à trois directions qui correspondent à des besoins immédiats :

- développer une réflexion didactique pour améliorer l'efficacité de la formation et l'expérience des élèves. Ce thème est d'ailleurs bien plus fréquent dans les colloques et revues comme ACM-TOCE que celui de la formation initiale des enseignants, qui en est pourtant un préalable. Pour travailler sur la didactique d'une discipline, il faut déjà en avoir bien assimilé les fondements, la pratiquer et avoir l'expérience de son enseignement. On peut espérer qu'avec le recul et l'expérience pédagogique acquise, les enseignants formés constituent des groupes de travail autour de questions didactiques. Certains ateliers des journées ISN/EPI en sont des prémisses. E. Vandeput rappelle dans l'entretien accordé à la revue 1024 (Vandeput, 2016) qu'un enseignant doit être à son niveau à la fois un pédagogue et un didacticien, mais qu'il ne peut évidemment pas se consacrer à de la recherche dans ces domaines ;
- définir, à partir de l'expérience acquise les contenus et pratiques des formations initiales d'enseignants d'informatique appelées à se mettre en place, par exemple le CAPES Mathématiques Informatique en France. Des pays, par exemple certains états allemands, Israël, la Grèce ont dans leurs universités des cursus pour de futurs professeurs d'informatique, mono ou bivalents selon les cas, analogues à ceux d'autres disciplines. On y prépare donc des mémoires consacrés à des questions de didactique de l'informatique ;

11 <http://idees.loria.fr/index.php?n=Main.ProgrammeJourneeISN-EPI>, <http://idees.loria.fr/index.php?n=Main.ProgrammeJourneeISN-EPI2014>, <http://idees.loria.fr/index.php?n=Main.ProgrammeJourneeISN-EPI2015>, <http://idees.loria.fr/index.php?n=Main.ProgrammeJourneeISN-EPI2016>, <http://idees.loria.fr/index.php?n=Main.ProgrammeJourneeISN-EPI2017>

- tirer parti de l'expérience des enseignants pionniers et de leur rôle de contact dans les établissements (ils n'enseignent pas que la spécialité ISN) pour proposer des formations adaptées aux besoins de tous les enseignants qui doivent accompagner des élèves dans leur découverte et leur compréhension du monde numérique en général et de certains aspects de l'informatique en particulier. Or, en France, une initiation ou formation à l'informatique et aux sciences du numérique est maintenant proposée aux autres sections : collège et primaire. Par ailleurs, l'initiative ISN est étendue à d'autres disciplines au lycée. Ces généralisations reposent sur des programmes différents. Il n'est plus possible alors de se reposer uniquement sur le programme à enseigner aux élèves trop différents d'un cas à l'autre, il faut aussi prendre compte les activités diverses des enseignants comme dans Parriaux & Chevalier (2016). Cela concerne par exemple en France les professeurs de mathématiques ou de technologie qui ont à enseigner des concepts informatiques dans leurs programmes ainsi que tous les maîtres de l'école primaire. Ces besoins ont été souvent décrits sous le label de « pensée informatique » à cause du papier fondateur de Wing (2006) ; cependant il faut bien remarquer que si le cadre général de réflexion est intéressant, la mise en pratique de cette approche en formation des maîtres reste à préciser (Yadav et al., 2014) et à mettre en œuvre.

10 Conclusion

Dans cet article, nous avons décrit la formation ISN de la région Lorraine en France. Nous avons décrit le contexte ayant permis d'initier cette formation, le contenu de la formation, comment le public des professeurs formés a reçu cette formation, l'articulation de la formation avec la certification à enseigner ISN, pilotée par le rectorat. Nous avons aussi décrit l'implantation de l'enseignement ISN dans les lycées de la région.

Nous avons souligné particulièrement l'importance des contacts qui perdurent entre enseignants et formateurs, non seulement au cours des journées annuelles ISN/EPI dont le programme est élaboré en commun, mais aussi par la participation des responsables universitaires aux commissions d'harmonisation des notes au baccalauréat et plus généralement sur la liste globale de diffusion.

Nous avons aussi mis en avant que la formation et les journées ISN/EPI n'ont pu exister sous cette forme en Lorraine que grâce au soutien de structures comme le LORIA et l'ESPE, en partenariat fort avec le rectorat. Cette convergence de bonnes volontés a été une chance. De telles initiatives nécessaires doivent-elles reposer sur la chance ?

Enfin, nous avons esquissé des perspectives possibles visant à élargir ISN au-delà du lycée.

Cette expérience de 6 ans nous permet d'avancer les préconisations suivantes pour favoriser la réussite d'une mise en œuvre d'une formation informatique :

- l'informatique, entrant dans les programmes à tous les niveaux (primaire, collège, lycée) sous différentes appellations et avec des programmes différents, il faut étendre la formation aux professeurs concernés, ce qui implique de définir un tronc commun indépendant des programmes ;
- il faut de la formation continue après la formation initiale pour les personnes formant les élèves ;
- la formation doit dépasser le cadre de l'informatique pure : formation aux enjeux sociétaux du numérique (problématique du *big data*, de la vie privée, potentiels, limites et risques de l'intelligence artificielle, etc.). En effet, au-delà des compétences informatiques, c'est vers une maîtrise du monde numérique qu'il faut aller. Notre réponse est l'organisation des journées ISN/EPI ;
- formation à la pédagogie par projet : les professeurs formés acquièrent des compétences informatiques. Il reste à encadrer des groupes de lycéens. Les seules compétences techniques ne suffisent pas alors, et il faut donc renforcer l'aspect pédagogique de la formation.

Références

Académie des Sciences, (2013) L'enseignement de l'informatique en France – Il est urgent de ne plus attendre, <www.academie-sciences.fr/pdf/rapport/rads_0513.pdf>.

ACM Europe : Informatics Education : Europe cannot afford to miss the boat, <http://europe.acm.org/iereport/>.

Baron, G.-L., Drot-Delange, B., Grandbastien, M., & Tort, F. (2014). Computer science education in French secondary schools : Historical and didactical perspectives. *ACM Trans. Comput. Educ*. 14, (2), Article 11 (Juin 2014), 27 pages.

BOEN. (2011). Prise en charge pédagogique de l'option de TL Droit et grands enjeux du monde contemporain et de l'enseignement de spécialité de TS Informatique et sciences du numérique, *Bulletin Officiel du Ministère de l'Education Nationale*, n° 36, 6 octobre 2011.

Brodnik, A., & Tort, F ; (eds.). (2016). Informatics in Schools : Improvement of Informatics Knowledge and Perception, ISSEP 2016 proceedings, Münster, Germany, Springer LNCS n° 9973.

Cogis, O., Palaysi, J. & Terrat, R. (2015). Rapport d'activité de la formation ISN dans l'académie de Montpellier de septembre 2011 à juin 2015, 1024, *Bulletin de la société informatique de France*, n° 7, nov. 2015, 65–79.

Dowek, G., Archambault, J. P., Baccelli, E., Cimellli, C., Cohen, A., Eisenbeis, C., Viéville T., Wack, B., Bersini, H., & Le Blanc, G. (2013). *Informatique et sciences du numérique* – Édition spéciale Python. Eyrolles, 342 pages.

Gautier, M. & Wrobel-Dautcourt, B. (2016) artEoz – Dynamic Program Visualization in Informatics in Schools : Improvement of Informatics Knowledge and Perception. 9th International Conference on Informatics in Schools : Situation, Evolution, and Perspectives, ISSEP 2016 proceedings, LNCS 9973, 2016.

Ministère français de l'Education, Programmes ISN : <http://eduscol.education.fr/cid59678/presentation.html>.

Parriaux, G., & Chevalier, M. (2016). Formation didactique des enseignants d'informatique : défis posés par une régionalisation, Actes du colloque Didapro 2016, 4 pages.

Quinson, M., & Oster, G. (2015) A Teaching System To Learn Programming : the Programmer's Learning Machine. 20th ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ItiCSE), July 2015.

Vandeput, E. (2016). La didactique de l'informatique, entretien réalisé par C. De la Higuera, 1024, *Bulletin de la Société Informatique de France,* numéro 8, avril 2016, 105–111.

Wing, J. M. (2006) Computational Thinking, Communications of the ACM, Mars 2006. Vol. 49, No. 3.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014), Computational Thinking in Elementary and Secondary Teacher Education, *ACM Trans. Comput. Educ*, 14, (1), 1–16.

Frédérique Chessel Lazzarotto

Conseillère pédagogique, circonscription d'Evian, France
frederique1875@gmail.com

Former à la programmation en primaire, une form'action : Robots d'Evian 2015–2018

Résumé
Depuis la rentrée de septembre 2016, l'enseignement de la programmation informatique a fait son entrée dans les programmes de l'école primaire en France. La circonscription d'Evian a anticipé cette introduction en proposant une formation interne à ses enseignants dans le cadre des heures de formation continue. Après 2 années scolaires, plus de 50 enseignants ont expérimenté la mise en œuvre d'activités de programmation auprès de leurs élèves. Leurs pratiques en classe et leurs retours permettent de cerner obstacles et enjeux de l'introduction de la programmation à l'école, en validant le format et le contenu retenu dans le cadre de la formation continue.

Mots clés : programmation, école primaire, formation continue, robotique pédagogique

1 Introduction

Depuis l'introduction en septembre 2016 des nouveaux programmes scolaires en France, les élèves du cycle 1 au cycle 4, bénéficient de l'enseignement de nouvelles compétences liées à la programmation. L'équipe de circonscription d'Evian a créé un parcours de formation de six heures destiné aux professeurs des écoles. Il a été déployé de manière expérimentale en 2015–2016 auprès de seize enseignants volontaires dans le cadre d'un appel à projets académiques, « les Heures numériques ». Depuis la rentrée de septembre 2016, il est proposé dans le plan de formation de la circonscription pour tous les enseignants désignés ou volontaires. Cet article s'appuie sur cette expérimentation locale, qui a concerné en deux ans 53 enseignants de tous niveaux. Le contenu de la formation propose d'aborder les concepts

de programmation informatique liés à des activités de programmation dite « branchée » et « débranchée » et de robotique pédagogique utilisant différents robots (Thymio, Bluebot et Cubetto). Les échanges et productions recueillis sont autant de signes et éléments tangibles permettant de cerner les difficultés rencontrées par les équipes ainsi que les impacts perçus tant au niveau des élèves que de leurs professeurs dans la mise en œuvre d'activités de programmation. Les retours collectés permettent d'apporter certaines réponses aux questions issues de la formation des enseignants quant à l'introduction de la programmation informatique :

— quel contenu pour quel enseignement de l'informatique à l'école primaire en formation continue ?
— comment les enseignants intègrent-ils les activités de programmation dans leur enseignement ?
— quelles sont les représentations des enseignants à l'égard de ce nouvel enseignement ?

2 Présentation du projet

Le projet *Robots d'Evian* s'est constitué au départ autour d'un événement pérenne organisé pour les écoles de la circonscription : le festival de sciences. La thématique de la programmation informatique a été choisie en mai 2015 au moment où les robots Thymio faisaient leur entrée dans des classes tests. Il a été alors décidé d'expérimenter un parcours de formation de six heures à la programmation informatique.

2.1 Une formation concentrée

À l'heure du choix du contenu de la formation, des objectifs réalisables et modestes ont été fixés. Il s'agissait en six heures de formation de présenter les concepts clés de la programmation informatique (Wilgenbus, Calmet et Hirtzig, 2016). La forme présentielle a été adoptée dans le but de pouvoir faire vivre des situations de recherche collectives en privilégiant la démarche d'investigation.

Des activités débranchées ont été choisies notamment dans le document *Unplugged*[1] pour apporter le lexique adapté dans un premier temps. Une présentation rapide de la pensée informatique inspirée de Vincent Englebert (Henry *et al.*, 2017) permet une ouverture sur l'intégration des activités dans les différents domaines et le développement de compétences transversales. Une deuxième partie consiste à découvrir en démarche d'investigation des robots pédagogiques, particulièrement Thymio. Une série de missions, inspirées de la démarche déployée par Morgane Chevalier[2] et le projet Flowers de l'INRIA[3], est proposée jusqu'à découvrir la programmation par langage visuel (VPL).

2.2 Des sollicitations auprès des écoles

Les classes peuvent participer à une journée de festival ou de rencontres : les élèves sont à la fois visiteurs et exposants. L'engagement de leur part est remarquable et s'évalue d'une part, dans les explications qu'ils donnent lors de l'accueil des élèves à leur stand, et d'autre part, dans l'intérêt et la curiosité dont ils font part lorsqu'ils visitent les ateliers proposés par des professionnels ou leurs camarades. L'équipe organise de plus le prêt de matériel. Des robots Thymio, Bluebot ou Cubetto circulent dans les classes demandeuses pour une durée de 3 semaines. Enfin, un appui de proximité est donné aux enseignants qui mettent en œuvre des projets particuliers de classe ou qui en font la demande. Cet accompagnement permet de prélever des informations *in situ* sur la mise en œuvre en classes, de les recenser et de les mettre en valeur.

3 Méthodologie

Pour apporter des éléments de réponses quant à l'efficacité de la formation proposée, nous avons utilisé les enquêtes suivantes.

1 <https://interstices.info/upload/csunplugged/CSUnplugged_fr.pdf>

2 <https://aseba.wdfiles.com/local--files/fr%3Athymiorobotsenclasse/exemple-cycle1.MC.pdf>

3 <http://www.dm1r.fr/_documents/IniRobot_missions_VPL14.pdf>

3.1 Des questionnaires en ligne

Plusieurs groupes ont été ciblés durant ces enquêtes, nous permettant d'affiner nos perceptions quant à la formation des enseignants dans le choix de son contenu et de la mise en œuvre des activités de programmation en classe.

Aux enseignants formés
Cinquante-trois enseignants ont bénéficié d'une formation de six heures. Une première volée s'est constituée en septembre 2015 avec 16 professeurs des écoles pionniers qui se sont engagés à participer avec leurs élèves à un festival de programmation sur le temps scolaire. Ils ont été sollicités directement par l'équipe pour faire partie d'un projet « Heures numériques ». Une deuxième volée de 27 professeurs s'est constituée à la rentrée 2016 en 2 groupes différenciés selon les cycles d'enseignement. Une troisième formation a été réalisée sur le temps scolaire au début de septembre 2016 pour 10 professeurs de cycle 3. L'enquête a été systématiquement envoyée quelques semaines après la formation afin de laisser le temps de la mise en œuvre aux enseignants. Sur ces 53 professeurs, 29 réponses ont été obtenues.

Un questionnaire envoyé aux enseignants non formés
Durant l'année, nous avons constaté lors de nos déplacements dans les écoles que les activités de programmation étaient aussi mises en œuvre dans les classes visitées par des enseignants qui n'étaient pas inscrits aux formations proposées. Huit enseignants ont été ainsi repérés et ils ont été destinataires d'un questionnaire permettant d'évaluer les ressources utilisées ainsi que leurs besoins en formation. Tous ont répondu en ligne.

Un questionnaire envoyé aux élèves
Les élèves de deux classes de cycle 3 de CM1 et CM2 ayant participé à une séquence d'activités de programmation ont été sollicités pour répondre à un questionnaire d'évaluation en ligne pour tester leurs compétences.

3.2 Des échanges collectifs

Un groupe « ressource » de circonscription regroupant neuf professeurs s'est réuni une fois par trimestre pour échanger sur les pratiques de classe. Ces moments réflexifs ont permis de recueillir des éléments importants

liés à l'organisation matérielle et pédagogique, à la gestion des élèves, aux difficultés ou réussites dans la conduite des séances ainsi qu'à l'efficacité de la formation.

4 Résultats

Les résultats obtenus au cours de l'expérimentation s'appuient sur les 29 réponses des enseignants formés à la programmation, les 8 réponses des enseignants non formés, les 42 retours des élèves de CM1 et CM2, étayés par les différents échanges formels et informels.

4.1 Origine des enseignants

La répartition hommes-femmes montre une légère surreprésentation des hommes (28 %) par rapport à la moyenne nationale du Ministère établie à 19 %. Seulement un quart des enseignants sont issus d'une filière scientifique. L'ancienneté n'est pas un critère retenu. L'entrée réalisée par les activités débranchées a lissé les appréhensions liées à la maîtrise des outils numériques pour certains enseignants. La démarche d'investigation proposée laissant émerger différentes hypothèses et mettant en valeur les stratégies de résolution par tâtonnement expérimental a pu estomper les origines des parcours et nourrir les échanges et les interactions.

4.2 Activités réalisées en classes

Tous les enseignants estiment pouvoir commencer des séances de programmation informatique sans matériel. Le temps consacré aux activités est lié à l'engagement ou non dans un projet de rencontres ou de festival, plus de dix séances de 30 à 45 minutes sont alors nécessaires. Dans ce cas, les élèves se mobilisent davantage en interdisciplinarité, chacun d'entre eux doit pouvoir présenter l'atelier choisi, le temps d'appropriation est important.

Le tableau 1 montre les situations proposées aux élèves ; elles sont classées par occurrence dans les réponses au questionnaire.

Tableau 1 : Les situations proposées aux élèves.

Débranchées	Jeu du robot, Machine à trier, jeu de Nim, système binaire, pixellisation, nombre pensé, Jeu de l'orange, Île au trésor, le crêpier psychorigide, les Marmottes
Branchées	Thymio, Bluebot et sa barre, Cubetto, Beebot, Inobot, Lightbot, Ozobot, Scratch, Scratch Jr, Code.org

La diversité des enseignants offre une appropriation des propositions en s'adaptant au profil ou aux projets des classes. Lors des rencontres, les classes inscrites réalisent un parcours avec les robots, tous sont différents : permis de conduire, mythe de Thésée, Vendée Globe, Guerre de Troie, la barrière Thymio, etc.

Ils s'insèrent spontanément dans les projets de classe et sont pluridisciplinaires conviant principalement la littérature, les mathématiques et les arts.

Figure 1 : Images collectées lors des rencontres,
une diversité de productions en interdisciplinarité.

4.3 La programmation dans les classes de la circonscription

Afin de mesurer l'impact de la formation, les réponses aux questionnaires ont permis d'évaluer le nombre d'élèves concernés par l'enseignement de la programmation ainsi que les perceptions des enseignants quant à la mise en œuvre des activités dans leurs classes après la formation.

Effectifs et perméabilité

En deux années, 1300 élèves, soit un quart de la totalité des élèves de la circonscription d'Evian, ont pu aborder la programmation informatique. À la question « combien de classes ont mené un travail sur la programmation,

avez-vous eu des échanges dans l'équipe à ce sujet », 26 enseignants y répondent positivement. On compterait alors environ 35 classes supplémentaires soit près de 900 élèves : 40 % de l'effectif des élèves seraient ainsi concernés par l'expérimentation. Cette diffusion s'explique par le format particulier des écoles de la circonscription, de petites écoles en grande majorité, où les doubles niveaux sont fréquents et induisent un travail d'équipe conséquent. Les activités sont souvent menées dans des lieux collectifs de passage, aiguisant la curiosité de tous. Les robots prêtés pour 3 semaines dans l'école poussent l'équipe pédagogique à s'en emparer rapidement du fait que leur présence est limitée dans le temps, souvent sous forme de stage.

De la formation à la mise en œuvre en classe

Tous les enseignants ont accordé que la formation avait été satisfaisante voire très satisfaisante, notamment par le fait de pouvoir utiliser directement les situations vécues et les ressources fournies. La plupart sont capables de donner la définition de programmation, code et algorithme. Ils estiment que la mise en œuvre est possible voire facile en classe, aucun retour négatif n'a été enregistré. Quasiment tous les participants à la première cohorte en 2015 ont souhaité poursuivre l'expérimentation en se réinscrivant au groupe ressource de la circonscription afin de tester de nouvelles situations ou de nouveaux robots et de profiter des expériences de chacun. Ce positivisme apparent semble s'expliquer dans l'évolution de la définition que se font les enseignants de la programmation informatique après avoir vécu la première partie débranchée de la formation. En effet, tous s'accordent à l'associer à l'action de trouver les différentes étapes d'un problème, avant même la réalisation d'un programme ou la connaissance informatique d'un langage dédié. La pensée logique et la mise en œuvre des stratégies de résolutions de problème semblent prendre le devant de la maîtrise spécifique de la programmation. Les difficultés rencontrées apparaissent plutôt dans l'organisation matérielle des séances et la disponibilité des outils. Mener des séances de programmation suppose en effet une réorganisation des espaces, des groupes. Le choix a été fait de ne proposer que six robots aux classes, favorisant le travail de groupe induit par le matériel.

Les représentations des compétences développées chez les élèves

Les échanges donnent des éléments de réflexion qui souligne les bénéfices de ce nouvel enseignement. En dehors du langage oral très sollicité, le

développement des compétences des élèves se situe dans la découverte des concepts fondamentaux puis dans la maîtrise des outils informatiques. La résolution de problèmes vient ensuite dans les propositions ainsi que l'acquisition d'une culture numérique. La créativité est moins bien représentée. Les « autres » éléments ajoutés sont intéressants à remarquer même s'ils ne sont pas très fréquents dans les réponses : la collaboration, l'autonomie, la démarche d'investigation, la mise en avant des élèves en difficulté, l'organisation pédagogique différente. Des indices transmis au cours des rencontres du groupe et des visites dans les classes et qui se retrouvent dans la définition de la pensée informatique (Wing, 2006). Les difficultés de repérage spatial notamment allocentré ont été dévoilées grâce à la manipulation des robots pour plus d'un tiers des élèves de cycle 3, questionnant leurs professeurs sur les progressions établies en géométrie.

5 Discussions

Les résultats obtenus suite à la formation proposée dans la circonscription permettent de mettre en perspective l'expérimentation vers une formalisation de cette action.

5.1 Le contexte : un projet de circonscription

Certains éléments catalyseurs témoignent de la réussite de la formation qui a pris le visage d'une *form'action.* En comparant les éléments de l'action « aux conditions à créer pour un projet de formation viable » (Ria *et al.*, 2016), il apparaît que les rencontres multipliées par la constitution du groupe ressource engagé dans la réflexion et par les différentes sollicitations proposées ont pu permettre un « principe de collaboration dans une logique de projet négocié » : retour sur les pratiques de chacun, productions de séquences, élaboration de projets et évaluation anticipée affinant les observations quotidiennes. La localisation des actions sur un territoire défini de proximité avec un public de formés et formateurs réciproquement reconnus a facilité les échanges. Ceux-ci ont été aussi limités dans le temps car soumis aux échéances des actions du projet. De plus, la pérennité s'inscrit dans le projet

de circonscription dont les objectifs fixés et connus invitent à une « complicité et une sécurité » des engagements de chacun dans une continuité assurée.

Au-delà de la diffusion d'un contenu, la formation a proposé un accompagnement des enseignants. Le travail des classes est mis en valeur par des communications sur différents supports apportant reconnaissance et motivation des enseignants[4]. Cette *collaboration* est perçue comme un travail collectif dans lequel chacun apporte sa pièce, son identité aussi. Enfin, le projet s'articule autour d'une manifestation phare, un festival dédié à la programmation informatique. L'inscription, sur la base du volontariat, engage puissamment les enseignants et leurs élèves sur une année scolaire. Cette participation implique un investissement planifié et conséquent, nécessitant un travail pluridisciplinaire et élaboré, débordant souvent de sa propre classe et permettant une perméabilité de l'action à destination de l'équipe pédagogique et des autres classes.

5.2 Retours réflexifs sur le contenu et l'organisation de la formation

Bruno Hourst cite les éléments de motivation pour une formation réussie (Hourst, 2008). En premier lieu, la commande institutionnelle est un levier puissant pour expliquer l'accueil de la proposition de formation, cependant le format raisonnable, la mise en œuvre matérielle facilitée et l'accessibilité du contenu participent à la satisfaction déclarée des stagiaires. La pensée informatique évoquée rejoint dans sa partie logique le domaine des mathématiques, apprivoisant les enseignants timides face aux nouvelles technologies, en apportant des propositions de manipulation apparentées aux mathématiques qui sont, comme le souligne Perrine Brotcorne, « dissociées de la technologie informatique » (Henry *et al.*, 2017). Les sessions sont construites pour aborder les éléments progressivement avec des réussites accessibles au départ. Les échanges entre pairs sont privilégiés, les questions sont accueillies avec bienveillance pour rassurer et clarifier les enjeux et attendus de la programmation à l'école primaire. On constate que l'effectif du groupe et les situations actives avec des objets ludiques favorisent la prise de risque personnelle et désinhibent les échanges.

Les objectifs poursuivis visent des « bénéfices » immédiats avec des situations vécues qui, par isomorphisme, sont facilement être reprises en classes.

4 <http://www.ac-grenoble.fr/ien.evian/spip.php?article836>

D'autres « bénéfices indirects » concernent les changements de postures consentis lors de la pratique de la programmation : les enseignants prennent du recul pour observer leurs élèves, tâtonner et analyser leurs cheminements réflexifs, leurs échanges entre pairs. Chaque session est organisée de manière à présenter aussi aux enseignants le fonctionnement d'outils numériques participant aussi au développement de leur culture et compétences numériques. Les six heures consacrées sont vécues avec un rythme soutenu qui ne permet cependant pas l'approfondissement des concepts visés : des références et des ressources sont fournies aux stagiaires pour prolonger leur appropriation.

5.3 Un contenu adapté ?

Le contenu dispensé tant en informatique branchée et débranchée semble correspondre aux attentes des instructions officielles dont les concepts liés à la programmation sont répartis dans plusieurs domaines du socle sans jamais énoncer l'expression de « pensée informatique ». À ce jour ne sont diffusées officiellement, via les propositions d'accompagnement aux programmes sur le site Eduscol, que des activités liées à la géométrie et aux sciences[5]. Pourtant, en rassemblant les éléments informatiques présents dans les différents domaines du socle, on peut aller plus loin et proposer aux élèves un éventail de situations leur permettant de développer, tout en l'explicitant, leur pensée logique et leur abstraction pour résoudre des problèmes et apprendre à chercher, de construire leur identité citoyenne, d'acquérir une culture scientifique, numérique et informatique en découvrant le fonctionnement d'un ordinateur ou de robots et de leur programmation, de faire preuve de créativité, de coopération et de persévérance dans des projets personnels ou partagés.

6 Conclusion et perspectives

Deux années d'expérimentation dans un contexte particulier ne sauraient modéliser une quelconque formation *duplicable* à l'envi. Les formateurs œuvrent au quotidien et sur le terrain pour dynamiser cette introduction de

5 <http://cache.media.eduscol.education.fr>

l'informatique scolaire. La formation paraît répondre aux besoins de définition des premiers concepts informatiques abordables en classes primaires et à la connaissance de ressources dédiées tout en offrant un exemple de démarche active reproductible en classe. Des recherches plus approfondies et étayées devront explorer les axes suivants : interpeller le rôle de l'école maternelle dans l'appréhension des compétences spatiales et dans l'acquisition des liens logiques anticipant les modèles informatiques ; découvrir le pouvoir des objets tangibles sur les interfaces virtuelles notamment dans les phases manipulatoires propices à la construction progressive de l'abstraction ; appréhender le rôle des interactions langagières et sociales des élèves lors des séances de programmation, constater les changements perceptibles de rôle de chacun des acteurs dans la relation pédagogique et le regard des élèves ; enfin, évaluer l'implication de l'introduction de la pensée informatique dans le développement des compétences de résolutions de problèmes… Tout un programme !

Références

Hourst, B. (2008). Former sans ennuyer : concevoir et réaliser des projets de formation et d'enseignement Ed. d'Organisation.

Henry, J., Nguyen, A. et Vandeput, E. (2017). L'informatique et le numérique dans la classe. Qui, quoi, comment ? Presses Universitaires de Namur, 7–36.

Ministère de l'Éducation nationale (2015). Bulletin officiel spécial n°11 du 26 novembre.

Ria, L. (dir.) (2016*).* Former les enseignants au XXI[e] siècle. Professionnalité des enseignants et de leurs formateurs*, Recherche et formation.*

Wilgenbus, D, Calmet, C. et Hirtzig, M. (2016). 1, 2, 3… Codez. Éditions Le Pommier.

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, *49*(3), 33–35.

Julie Henry & Anne Smal

Centre de recherche PReCISE – NAmur Digital Institute (NADI)
Université de Namur (Belgique)
julie.henry@unamur.be, anne.smal@unamur.be

« Et si demain je devais enseigner l'informatique ? » Le cas des enseignants de Belgique francophone

Résumé
En Belgique francophone, le groupe de travail SI², constitué de représentants de toutes les Universités francophones et de certaines Hautes école des régions wallonne et bruxelloise, a été mandaté pour définir le référentiel de compétences d'un cours de sciences informatiques qui trouverait place en secondaire inférieur (élèves de 12 à 15 ans). Malgré l'absence de certitude que ce cours soit un jour réellement organisé, les auteures se sont intéressées aux enseignants susceptibles d'en être titulaire. Parmi eux, combien se sentent capables d'enseigner ce cours ? Combien sont prêts à le faire dans un futur proche ? Quel est leur profil ? Si une formation doit être dispensée à ces enseignants, quelles en seraient les modalités ? Une enquête à large échelle a été lancée. Les résultats de cette dernière constituent l'apport principal de cette contribution.

Mots clés : enseignement de l'informatique, formation des enseignants, compétences en informatique des ensei- gnants, auto-évaluation, enquête

1 Introduction

Quid de l'enseignement des sciences informatiques[1] en Belgique francophone ? Pour les élèves, le constat est sans appel : aujourd'hui encore, les jeunes sortent de l'enseignement obligatoire[2] en n'ayant eu presque aucune formation en informatique (Henry & Joris, 2016 ; Joris & Henry,

1 Incluant une « culture informatique » (Duchâteau, 1992).
2 Entre 6 et 18 ans.

2014). Pour expliquer ce constat, le manque de formation des enseignants (Henry & Joris, 2013) est souvent mis en cause. Pourtant, dans le contexte actuel belge, ceux-ci sont susceptibles d'être très prochainement appelés non seulement à intégrer le « numérique » au sein de leurs classes (si ce n'est déjà fait compte tenu des plans d'équipement « École Numérique »[3]), mais également à l'enseigner.

En effet, l'enseignement de la Fédération Wallonie-Bruxelles (FWB) est en pleine réforme depuis 2015. Le Pacte pour un Enseignement d'Excellence[4], fruit d'un intense travail collaboratif autour de cette réforme, stipule que « dès le primaire, une initiation à la logique du numérique peut utilement être réalisée par la programmation de machines simples ». Ce rapport indique par ailleurs l'existence prochaine d'un tronc commun polytechnique dans l'enseignement obligatoire (élèves de 6 à 15 ans) sans pour autant donner d'informations précises quant à son contenu.

Si ce qu'évoque le terme « numérique » reste ici encore assez flou, les initiatives soutenues par la Région Wallonne laissent penser que les sciences informatiques ont leur place dans la révolution de l'enseignement qui est en marche en FWB. En effet, un groupe de travail, baptisé Sciences Informatiques au Secondaire Inférieur[5] (SI^2) et composé des cinq universités et d'une dizaine de Hautes Écoles[6] du paysage francophone belge, est actuellement (et ce depuis 2016) mandaté pour définir un référentiel de compétences en sciences informatiques destiné au premier degré du secondaire (élèves de 12 à 14 ans). Une première version de ce référentiel a été élaborée, proposant un enseignement de l'informatique autour de cinq thématiques : représentation des données, algorithmique, programmation, matériel et réseau et sécurité.

Étant donné qu'il n'existe actuellement en FWB aucune formation didactique (CAP, AESI, AESS)[7] en sciences informatiques à destination des enseignants du secondaire, vers qui devrait-on se tourner si un tel cours d'informatique faisait partie des programmes officiels ? Quels enseignants sont, en l'état, capables d'assurer ce cours ? Et si une formation

3 <http://www.ecolenumerique.be/qa/>

4 <http://www.pactedexcellence.be/wp-content/uploads/2015/01/pacte-pour-un-enseignement-d-excellence.pdf>

5 <https://sicarre.be/>

6 Enseignement supérieur non universitaire.

7 Certificat d'Aptitude Pédagogique, Agrégation de l'Enseignement Secondaire Inférieur et Agrégation de l'Enseignement Secondaire Supérieur.

était organisée, quels en seraient le contenu et les modalités d'organisation ? Quels enseignants auraient accès à cette formation et lesquels seraient prêts à se former ?

Le travail sur lequel repose cet article n'a pas l'ambition d'une recherche. Il s'agit avant tout d'apporter des éléments de réponses aux questions (précédemment citées) qui se posent à l'heure d'une importante réforme de l'enseignement. Pour l'éducation au numérique (incluant l'informatique), cette réforme est l'occasion de (re)faire son entrée dans le paysage éducatif obligatoire, davantage centré sur l'usage (éducation PAR le numérique) que sur la compréhension.

2 Problématique

Pourquoi (ré)introduire une éducation (au) numérique ? Les raisons citées sont nombreuses, bien que souvent non prouvées « scientifiquement » parlant. Une intuition, une impression. . . un constat aussi. Car oui, le niveau de compétences « numériques » des élèves en sortie de l'enseignement obligatoire est faible (Henry & Joris, 2016 ; Joris & Henry, 2014). Les élèves font ainsi preuve d'une connaissance souvent superficielle des concepts sous-jacents aux logiciels d'usage courant (logiciels de traitement de texte, tableurs, courrielleurs, navigateurs, etc.) et d'une pratique inefficace de ceux-ci. Leurs savoirs en « matériel et système » sont souvent limités à la simple identification du matériel de base que sont l'ordinateur et ses périphériques. Enfin, ils présentent une expérience quasi inexistante de la programmation et des notions « réseau et sécurité ».

2.1 Un bagage informatique nul (ou presque)

Durant leur cursus scolaire obligatoire, les élèves ont peu de réelles occasions d'être confrontés à un apprentissage de l'informatique, quelque soit son contenu (matériel, algorithmique, programmation, etc.). Dans le contexte propre à cet article, à savoir le premier degré de l'enseignement

secondaire tous réseaux confondus[8], un seul cours existe et prend la forme d'une activité complémentaire[9].

En ce qui concerne l'enseignement de la FWB, ce cours, intitulé Initiation à l'informatique, est clairement orienté « utilisation de l'ordinateur ». Les élèves, après avoir réalisé des activités « pensées uniquement pour que l'élève puisse utiliser de manière correcte les logiciels les plus courants », sont incités à présenter les épreuves du Passeport TIC[10]. Un module (parmi les quatre proposés) présente toutefois une réelle connotation « informatique » : « maîtriser les premières bases de l'outil informatique », incluant la découverte d'un ordinateur et de ses périphériques. Du côté de l'enseignement catholique, le même cours d'Initiation à l'informatique[11] est clairement mis en relation avec le cours d'Éducation par la technologie (EPT) organisé dans le tronc commun. S'il est précisé qu'il ne s'agit pas d'un cours de bureautique, les modules proposés tournent pourtant autour de l'utilisation des logiciels courants (« savoir utiliser un traitement de texte », « savoir utiliser l'Internet », « savoir utiliser le système d'exploitation »).

En résumé, dans la majorité des cas, un élève de 15 ans finissant son cycle secondaire inférieur aura au mieux eu l'opportunité d'aborder le concept de matériel en ce qui concerne l'informatique. Il aura également l'occasion d'utiliser[12] des outils faisant partie du quotidien professionnel (traitement de texte, tableur, navigateur Internet). Si aucune connaissance n'est requise pour utiliser ces outils, la compréhension des principes fondamentaux de l'informatique est indispensable pour en faire un usage efficace (Hartmann, Näf & Reichert, 2012 ; Vandeput & Colinet, 2004).

---

8 En Belgique, l'enseignement est organisé en réseaux. Seuls seront évoqués ici les réseaux pour lesquels il existe un cours (dit) d'informatique, à savoir le réseau de la FWB et le réseau catholique.

9 À savoir que l'enseignant en charge de ces heures (une à deux par semaine) est libre de choisir ou non de donner ce cours plutôt qu'un autre.

10 <http://www.enseignement.be/index.php?page=27182&navi=3683>, le référentiel du Passeport TIC donne une idée du contenu abordé dans le cadre du cours d'Initiation à l'informatique.

11 <http://admin.segec.be/documents/4861.pdf>

12 Nous parlerons ici d'une utilisation et non d'un apprentissage tant l'enseignement prodigué est centré sur un produit en particulier et non sur les concepts sous-jacents à ce produit, rendant non pérenne toute connaissance acquise.

2.2 Des enseignants non formés

Si les rares cours d'informatique présents dans les référentiels officiels de l'enseignement obligatoires sont sous-exploités, c'est notamment par manque d'enseignants formés. Si les enseignants en charge de ces cours sont majoritairement issus des sciences (chimie, physique, biologie), certains enseignants possédant une formation en sciences dites humaines se voient attribuer ces heures pour compléter leur horaire. Aucun de ces enseignants n'a reçu durant sa formation initiale un quelconque cours d'informatique. Seuls les enseignants en mathématiques pourraient, à ce titre, prétendre posséder des compétences suffisantes pour assurer un cours d'initiation à l'informatique. Malheureusement, leur fonction étant frappée de pénurie depuis quelques années, il est rare de rencontrer des mathématiciens aux commandes de tels cours.

2.3 Un futur incertain

La réforme du tronc commun (primaire et secondaire inférieur) nécessite logiquement de redéfinir les contours de ce tronc revisité. Dans son rapport d'avril 2017 (Groupe de travail du Pacte d'excellence, 2017), le groupe de travail en charge de cette tâche souligne qu'il est essentiel que

> les référentiels spécifient les objets techniques et technologiques d'apprentissage, de manière à se situer davantage sur le volet de la formation aux compétences manuelles, techniques et technologiques (alors qu'à ce jour, c'est le volet de la formation par les techniques et les technologies qui est privilégié).

Concernant le numérique, ce groupe opte pour

> un réel équilibre et une interaction féconde […] à trouver entre l'éducation par et au numérique, les deux se complétant. […] Il est important d'aller au-delà de la seule littératie numérique destinée aux élèves en tant qu'usagers et de les initier (à partir de la fin du primaire à tout le moins) aux sciences informatiques ou à la pensée informatique, notamment algorithmique […].

Un consensus se dégage sur le fait que

> l'éducation par le numérique se réaliserait principalement par la pratique active au sein des disciplines, tandis que l'éducation au numérique prendrait plutôt place au

> sein des périodes dédiées au développement des compétences manuelles, techniques et technologiques.

Il ressort enfin que « la place de l'éducation au numérique (doit être) très précisément identifiée (à la fois en termes de contenus, mais aussi en termes d'importance horaire) […]. »

Au jour d'écrire cet article, cette identification est toujours en discussion. Dans tous les cas, les sciences informatiques (ou la pensée informatique, dépendant du choix) ne seront qu'une thématique parmi d'autres à développer au sein de ce cours.

3 Ébauche d'un référentiel de compétences : Quoi enseigner et comment ?

Réfléchir au contenu d'un cours d'informatique à destination d'enfants âgés de 12 à 14 ans. . . L'idée n'étant pas révolutionnaire, il était inutile de vouloir réinventer la roue. Dès lors, la première version d'un référentiel de compétences a été élaborée par le groupe SI[2], largement inspirée par les expériences menées chez nos voisins proches (France, Royaume-Uni[13]) et plus lointains (Nouvelle Zélande[14], États-Unis[15]). Les compétences à développer chez les élèves y sont organisées en cinq thématiques : (1) représentation des données, (2) algorithmique, (3) programmation, (4) matériel et (5) réseau et sécurité (*cf.* Annexe A).

Si l'objet de cet article n'est pas la discussion de ce référentiel, il est intéressant d'en préciser les modalités de création. Établi par un groupe de travail constitué de professionnels en informatique issus de l'enseignement supérieur (universitaire et non universitaire), ce référentiel reste l'ébauche d'un travail qui va durer près de trois années. En effet, le groupe SI[2] doit, à travers la création de contenus (activités d'enseignement/apprentissage) et la mise en place de ceux-ci dans les classes (via des écoles-pilotes), valider chacun des items constituant le dit référentiel.

---

13 <https://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf>

14 <http://csfieldguide.org.nz/en/curriculum-guides/ncea/index.html>

15 <http://www.csteachers.org/page/standards>

Sur le papier, le projet prévoit le cours de sciences informatiques dans le secondaire inférieur. La possibilité d'un glissement vers l'enseignement primaire n'est toutefois pas complètement écartée, notamment parce que le tronc commun polytechnique dans lequel pourrait s'insérer le cours d'informatique couvrira les niveaux primaire et secondaire inférieur. Une inconnue subsiste cependant (et pas la moindre) : le volume horaire consacré à l'informatique, simple thématique d'un cours d'éducation numérique qui reste à ce jour un rêve.

4 Le profil « numérique » des enseignants en Belgique francophone : Qui pour enseigner ?

Il n'existe actuellement aucune formation préparant à l'enseignement des sciences informatiques, conséquence logique de l'absence de cette discipline dans les cursus scolaires. La réintroduction d'un cours d'éducation au numérique est évoquée, sans que l'on soit fixé sur le contenu d'un tel cours. L'hypothèse est faite qu'une des thématiques abordées dans ce cours porterait sur les sciences informatiques, visant dès lors le référentiel de compétences développés par le groupe SI[2]. Dès lors, il est intéressant d'identifier le profil « numérique » des enseignants (et futurs enseignants) pour déterminer lesquels, parmi eux, sont en l'état capables d'assurer un enseignement des sciences informatiques.

4.1 Récolte des données

Une enquête a été proposée à différents acteurs du monde de l'éducation : 355 directions d'établissements organisant un enseignement secondaire (issus des différentes régions composant la Belgique francophone), des Hautes Écoles (via les listes de diffusion du groupe SI[2] et du projet HETICE[16]), ainsi qu'aux sections didactiques (organisant l'AESS) scientifiques de l'Université de Namur. Il a été demandé aux destinataires du

16 <http://hetice.ulg.ac.be/>

courriel accompagnant l'enquête de diffuser celle-ci aux enseignants/étudiants les plus « concernés » par son sujet.

L'enquête est composée de quatre volets. Le premier volet permet de dresser le portait des participants : leur sexe, leur âge, s'ils enseignent ou sont en cours de formation, leur (futur) titre didactique (instituteur, AESI, AESS, CAP, CAPAES), le titre de leur formation initiale, la discipline enseignée et le niveau où elle est organisée (le cas échéant).

Le second volet les questionne sur leur relation avec l'informatique. En intègrent-ils dans leur cours ? Quels éléments en particulier ?

Dans un troisième volet, les enseignants sont invités à auto-évaluer leurs compétences en se positionnant, au moyen d'une échelle de Likert[17], pour chacun des items composant le référentiel.

Enfin, le quatrième volet demande aux enseignants de se positionner quant à l'enseignement de ce référentiel (intéressé à le faire, intéressant de le faire, prêt à faire une formation) et de décrire la formation idéale à destination des enseignants.

4.2 Analyse et discussion

L'enquête a récolté, à l'heure d'écrire cet article, 293 réponses venant d'enseignants de tous niveaux. Parmi eux, 99 enseignent (ou enseigneront) au niveau secondaire inférieur, niveau-cible où devrait s'intégrer le cours d'informatique faisant l'objet de ce travail. Les résultats qui seront discutés ici sont donc centrés sur ces 99 participants dont 73.7 % sont en cours de formation pour acquérir un titre pédagogique (en l'occurrence ici, l'AESI).

Le taux de participation moins élevé des enseignants en place laisse à penser qu'ils se sont sentis moins concernés par l'enquête, cependant il n'est pas possible de confirmer cette hypothèse. En effet, il s'agit peut-être tout simplement d'un manque de communication entre les directions ayant réceptionné le courriel et leur corps enseignant, là où il est facile pour un professeur en Haute École/Université de diffuser l'enquête auprès de ses étudiants « futurs enseignants ». Il faut également prendre en compte le fait que certains enseignants travaillent déjà au sein d'un établissement scolaire en parallèle de leur formation pédagogique. Dans tous les cas, il est intéressant d'obtenir les réponses d'enseignants « en devenir » qui constituent

17 tout-à-fait / plus ou moins / pas du tout / je ne sais pas / je ne comprends pas l'intitulé.

sans doute le public le plus susceptible[18] de se retrouver en charge d'un cours d'« éducation au numérique ».

Les participants ont été catégorisés sur base de leur formation initiale : informatique, mathématiques, formation à caractère scientifique et « autres ». Cette dernière catégorie reprend tous les diplômes qui, de prime abord, sont moins susceptibles d'avoir introduit des sciences informatiques (et donc d'avoir développé chez les enseignants les compétences associées) dans son programme. Parmi les participants, cinq sont issus du domaine informatique, 34 ont une formation de base en mathématiques, 30 sont plutôt scientifiques et 30 ont un diplôme majoritairement issu des sciences dites humaines.

Avant de découvrir les items issus du référentiel et couvrant le domaine des sciences informatiques tel qu'il serait enseigné en Belgique francophone, les enseignants ont eu à préciser s'ils intégraient eux-mêmes de l'informatique dans leurs cours. Si 61.6 % annoncent en intégrer, les exemples qu'ils donnent laissent très vite apparaître une confusion entre technologies de l'information et de la communication (TIC) et informatique. Ainsi, les citations d'outils du quotidien tels que « Word », « Excel » et « Powerpoint » sont nombreuses. Les outils spécifiques à un domaine de spécialité tel que « Geogebra » sont également souvent associés à l'informatique par les enseignants.

Seuls neuf enseignants dispensent le cours d'initiation à l'informatique (décrit en 2.1) : trois d'entre eux disposent d'une formation initiale en mathématiques, les autres sont soit informaticiens pour quatre d'entre eux, soit ont étudié dans un domaine lié de près ou de loin à l'informatique (infographie et écriture multimédia).

Il est intéressant de constater qu'à part les enseignants ayant une formation initiale en informatique, les autres témoignent des lacunes sur un nombre plus ou moins important d'items. Une première analyse globale des résultats montre, en effet, que, pour chacune des thématiques du référentiel, les informaticiens sont majoritaires à répondre « tout-à-fait » à la question « j'estime être capable d'enseigner… » (*cf.* Figures 1–5). Ils sont généralement suivis, mais de loin, par les mathématiciens. Excepté pour les thématiques Algorithmique et Matériel, les mathématiciens répondent majoritairement « pas du tout ». En ce qui concerne les algorithmes, il

18 Les jeunes enseignants se voient souvent attribuer les cours à peu d'heures par semaine et n'ayant pas de lien avec une discipline en particulier dont les anciens ne veulent pas.

n'est pas surprenant de voir cette matière maîtrisée par un certain nombre de mathématiciens, celle-ci faisant souvent partie de leur cursus de formation initiale. L'aspect matériel, quant à lui, présente une répartition plus homogène des enseignants (mis à part les informaticiens). Ceci peut sans doute être expliqué par le fait qu'il s'agit d'une des rares thématiques en informatique n'ayant jamais été retirée des référentiels composant l'enseignement obligatoire. Tout jeune enseignant en formation a eu ce cours lors de son passage dans l'enseignement secondaire.

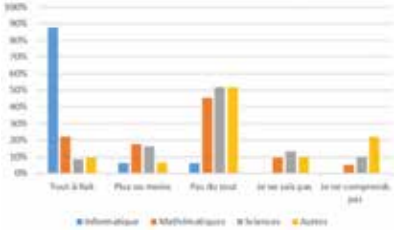


Figure 1 : Résultats pour la thématique Représentation des données.

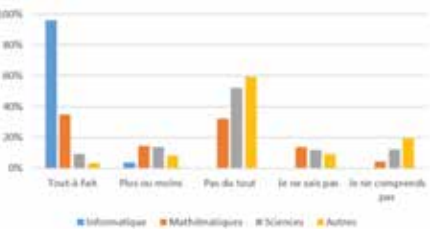


Figure 2 : Résultats pour la thématique Algorithmique.

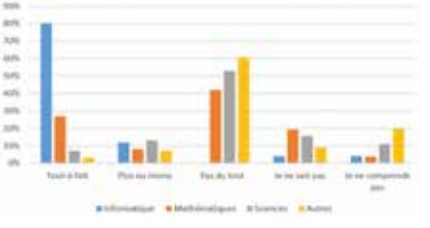


Figure 3 : Résultats pour la thématique Programmation.

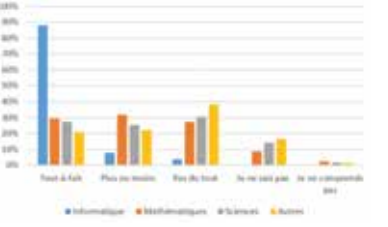


Figure 4 : Résultats pour la thématique Matériel.

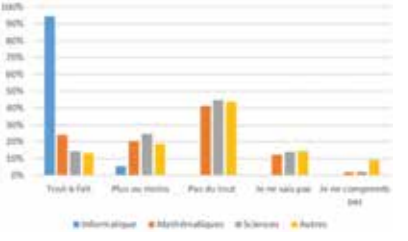


Figure 5 : Résultats pour la thématique Réseau et sécurité.

Ces résultats sont à nuancer de par le fait qu'il s'agit d'une auto-évaluation basée sur la seule compréhension d'un énoncé. Le participant pouvait d'ailleurs stipuler son incompréhension le cas échéant. En conclusion, outre les informaticiens (dont il est logique qu'ils possèdent les compétences nécessaires), il semble que ce soient les mathématiciens qui sont, en l'état actuel, les plus à même d'enseigner les sciences informatiques. Les résultats des scientifiques (chimie, physique, biologie, etc.) s'avèrent plus négatifs que prévu, dans le sens où il n'existe quasiment pas de différence entre eux et les enseignants possédant une formation initiale orientée sciences humaines (et supposés à priori moins à l'aise avec l'informatique).

Pour une analyse plus détaillée permettant de déterminer les items les plus problématiques, aucune distinction entre les catégories d'enseignants n'est faite (*cf.* Annexe B – Figures 8–12). Il apparaît ainsi que, mise à part la thématique Matériel, les enseignants se sentent peu capables d'enseigner ce qui constitue le référentiel de compétences élaboré par le groupe SI[2] (*cf.* Annexe A). En effet, ils répondent majoritairement (entre 40 et 50 %) « pas du tout » lorsqu'il s'agit de s'exprimer quant à leurs capacités d'enseigner la représentation des données, l'algorithmique, la programmation et le couple réseau-sécurité.

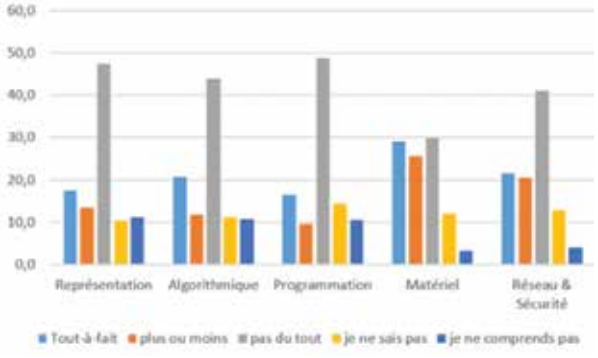


Figure 6 : Profils globaux pour les différentes thématiques.

Pour chaque thématique et pour l'ensemble des items la composant, des tendances globales ont été calculées (*cf.* Figure 6). Celles-ci font apparaitre des profils similaires en ce qui concerne les thématiques Représentation des données, Algorithmique et Programmation. Les items des thématiques Matériel et Réseau & Sécurité sont visiblement plus connus des enseignants,

probablement parce qu'il s'agit de concepts plus fréquemment rencontrés au quotidien.

À un item (R9) près, c'est plus d'un enseignant sur trois qui ne sent pas capable d'enseigner la thématique de Représentation des données (*cf.* Annexe B – Figure 8). Pour sept items sur treize, plus d'un enseignant sur deux avouent leurs lacunes. Pour les items R11 à R13, moins de 10 % se sentent « tout-à-fait » capables, environ 60 % ne s'en sentent pas capables et 15 % ne comprennent tout simplement pas de quoi il s'agit. Les items présentant la plus grosse différence entre les enseignants qui se disent « capables » et ceux qui pensent ne pas l'être sont principalement axés sur le traitement des images (R7, R8 et R10). C'est également le cas pour les items ayant à coeur de susciter la discussion entre l'enseignant et les élèves (« discuter des avantages et des inconvénients… », « expliquer l'intérêt… ») notamment à propos des standards d'encodage de caractères (R6) et des techniques de cryptographie (R11 à R13).

Les thématiques d'Algorithmique et de Programmation présentent des profils fortement similaires, y compris lorsqu'on y regarde plus en détail (*cf.* Annexe B – Figures 9 et 10), à savoir des items non maîtrisés pour plus de 40 % des enseignants (exception faite du A1). Il existe cependant une différence notable quant aux enseignants qui se sentent capables d'enseigner ces matières. Concernant l'Algorithmique, un enseignant sur cinq s'estime capable d'enseigner onze items sur les quinze. Au niveau de la Programmation, ça n'est le cas que pour un seul item sur les dix proposés.

Comme dit précédemment, la thématique Matériel semble être la mieux maîtrisée par les enseignants. Les items « de base » (M1 à M3) sont les plus connus (*cf.* Annexe B – Figure 11). Les deux autres posent problème à plus de 40 % des enseignants. Dans l'ensemble, les items composant cette thématique sont compris par la quasi totalité des enseignants, ce qui appuie la théorie d'une présence plus importante des concepts liés à cette thématique dans le quotidien de monsieur tout-le-monde.

Enfin, les réactions par rapport à la thématique Réseau & Sécurité sont plus mitigées : trois items étant maîtrisés par un tiers des enseignants (*cf.* Annexe B – Figure 12) tandis que sept autres posent problème à plus de 40 % d'entre eux. Ici aussi, les concepts semblent familiers à la plupart des enseignants.

L'enquête passée auprès de 99 enseignants du secondaire inférieur laisse apparaître un besoin de formation chez ceux-ci, dans l'hypothèse où

un cours incluant des sciences informatiques prendrait place dans le cursus obligatoire des élèves belges. Il serait intéressant d'analyser les résultats des enseignants du secondaire supérieur (104), des institutrices (30) et des enseignants du supérieur non universitaire (69). L'idée étant d'évaluer les conséquences d'un glissement de ce cours vers la fin du primaire (dépendant des compétences des institutrices) ou dans le secondaire supérieur. Les enseignants de Hautes Écoles, pour leur part, pourraient être amenés à inclure une préparation à ce cours dans la formation initiale des enseignants du secondaire inférieur. Dès lors, il reste important de mesurer leurs compétences dans le domaine de l'informatique. Ces différentes analyses n'ont pas été prévues dans le cadre de cet article mais feront l'objet de travaux futurs.

5 Quelle formation pour les enseignants ?

Sur les 99 participants à l'enquête, 38 se disent absolument intéressés à enseigner les sciences informatiques *à la rentrée prochaine*, 27 répondent « peut-être » et 32 ne le souhaitent pas. Deux enseignants n'émettent aucun avis. Lorsqu'il leur est demandé s'ils trouveraient ça intéressant pour leurs élèves, 58 enseignants en sont persuadés, 30 se posent la question et sept trouvent que ça n'a pas d'intérêt. Quatre restent sans avis. Enfin, 45 se disent prêts à suivre une formation pour pouvoir enseigner cette discipline, 36 sont mitigés et 17 ne le souhaitent pas. Un seul enseignant n'a pas d'avis.

Dès lors, pas loin d'un enseignant sur deux serait prêt à se former afin de pouvoir intégrer de l'informatique dans le cursus de ses élèves. Plus d'un sur deux trouve cela intéressant pour l'élève lui-même. Près de un sur trois serait déjà partant pour l'enseigner dès la rentrée prochaine, sachant que l'enquête a été passée dans le courant de l'année scolaire 2016–2017. Ces résultats s'avèrent particulièrement engageants.

Dans un objectif de coller au mieux aux desiderata des enseignants souvent frileux à l'idée de suivre des formations continues, la parole leur a été donnée quant aux modalités d'organisation d'une future formation. Du point de vue de la longueur de la formation, près de 2/3 des répondants optent pour une durée de 1 à 2 journées, formule actuellement mise en place en ce qui concerne la formation continue en Belgique francophone.

Concernant les modes d'enseignement, étant donné les avancées actuelles et le paysage toujours plus numérique de l'enseignement, il a été donné aux enseignants de faire un choix entre un enseignement entièrement en ligne (MOOC) et un enseignement mixte (blended learning). La mise en place d'un enseignement totalement ou partiellement à distance a été envisagé pour combler le manque de temps pour une formation en présentiel (confirmé par le choix des enseignants concernant la durée de celle-ci). Les enseignants préfèrent majoritairement un enseignement mixte.

Enfin, quant il s'agit de discuter du type de contenu qui devrait idéalement composer la formation (*cf.* Figure 7), une petite majorité d'enseignants (34 %) optent pour une formation proposant un apprentissage des concepts théoriques et une mise en pratique de ceux-ci, le tout agrémenté de réflexions didactiques. Les autres participants sont partagés entre des ateliers pratiques avec échanges d'expériences, des formations aidant à la création de leurs propres activités et à leur intégration, et des activités « clé-sur-porte » directement réutilisables en classe.

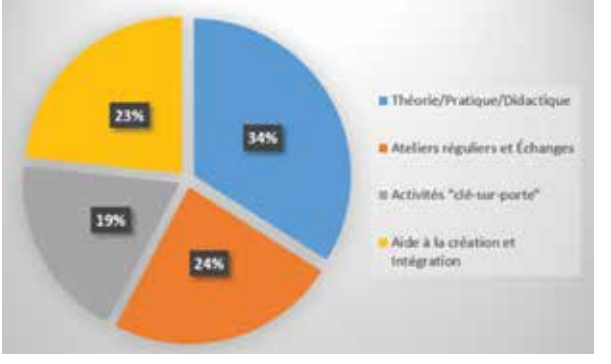


Figure 7 : De « clé-sur-porte » à « all by myself »…

6 Conclusion

Partant d'un hypothétique retour des sciences informatiques dans le tronc commun en Belgique francophone, une enquête a été passée auprès de différents acteurs de l'enseignement. Cette enquête, toujours en cours de

passation, a récolté à l'heure d'écrire cet article quelques 293 réponses d'enseignants de tous niveaux.

Les résultats présentés ici sont centrés exclusivement sur les 99 enseignants du secondaire inférieur qui représentent la population la plus susceptible de se retrouver première ligne si notre hypothèse se révélait vraie dans les années à venir.

Plus aucune formation didactique orientée vers la discipline qu'est l'informatique n'existe dans le paysage éducatif belge. Dès lors, la question posée ici est la suivante : dans quelle situation serions-nous si, du jour au lendemain, nos enseignants étaient contraints de dispenser un cours de sciences informatiques, que celui-ci soit un cours en lui-même ou une thématique abordée dans un cours plus large d'éducation au numérique ?

Les résultats obtenus sont plutôt engageants puisque certains enseignants seraient prêts (en terme d'engagement) à prendre en charge un tel cours dès que possible. Malheureusement, en termes de compétences, des apports doivent être faits. Les enseignants ont été amenés à auto-évaluer leur capacité à enseigner chacun des items composant un référentiel élaboré par le groupe de travail SI[2].

Il est donc clair qu'une formation à destination des enseignants titulaires doit être mise en place en parallèle, ou idéalement au préalable, de l'intégration de sciences informatiques dans le cursus scolaire. Concernant les modalités d'organisation de cette dernière, la parole est donnée aux enseignants : une formation continue courte, en enseignement hybride (blended learning).

Pour ce qui est du contenu, celui-ci fera l'objet d'une réflexion compte tenu des résultats obtenus via cette enquête et des desiderata des enseignants. Si les activités « clé-sur-porte » sont appréciées par les enseignants qui y voient la facilité de prendre en main une discipline qu'ils ne maîtrisent pas, ils sont bien conscients qu'ils n'échapperont pas à une formation poussée aux concepts de base et à une réflexion didactique autour de ceux-ci.

En attendant que le rêve devienne réalité…

Références

Duchâteau, C. (1992). *Peut-on définir une culture informatique ?* Facultés universitaires Notre-Dame de la Paix.

Groupe de travail du Pacte d'excellence. (2017, avril). *Rapport sur les éléments du plan d'action relatifs au nouveau tronc commun.* <http://www.pactedexcellence.be/>.

Hartmann, W., Näf, M. & Reichert, R. (2012). *Enseigner l'informatique*. Springer Science & Business Media. Henry, J., & Joris, N. (2013). Maîtrise et usage des tic : la situation des enseignants en belgique francophone. In *Sciences et technologies de l'information et de la communication (stic) en milieu éducatif.*

Henry, J., & Joris, N. (2016). Informatics at secondary schools in the french-speaking region of belgium : myth or reality ? *ISSEP 2016*, 13.

Joris, N., & Henry, J. (2014). L'enseignement de l'informatique en belgique francophone : état des lieux. *1024- Bulletin de la société informatique de France*(2), 107–116.

Vandeput, E., & Colinet, M. (2004). *Les invariants du tableur*. Facultés universitaires Notre-Dame de la Paix.

Annexe A : Référentiel de compétences en sciences informatiques

Représentation des données

En ce qui concerne la représentation des données, l'élève doit être capable de :

- Comprendre la représentation interne des données par une machine
  - Expliquer pourquoi une machine représente les données sous forme binaire (0 et 1) *(R1)*
- Comprendre la représentation binaire des nombres
  - Passer de la représentation décimale d'un nombre naturel vers sa représentation binaire *(R2)*

  - — Passer de la représentation binaire d'un nombre naturel vers sa représentation décimale *(R3)*
- Comprendre la représentation binaire des caractères
  - — Utiliser un code pour représenter un caractère sous forme binaire (coder) *(R4)*
  - — Utiliser un code pour retrouver un caractère à partir de son encodage binaire (décoder) *(R5)*
  - — Discuter des avantages et inconvénients de l'utilisation d'un standard d'encodage des caractères (p.e. ASCII) *(R6)*
- Comprendre la représentation binaire des images
  - — Numériser une image sous forme d'une grille de pixels *(R7)*
  - — Reconstituer une image à partir d'une grille de pixels *(R8)*
  - — Discuter du lien entre la qualité de l'image et sa résolution *(R9)*
  - — Discuter des avantages et inconvénients d'un algorithme de compression d'image *(R10)*
- Pour en savoir plus
  - — Expliquer l'intérêt et utiliser un algorithme de chiffrement basique (chiffre de César) *(R11)*
  - — Expliquer l'intérêt et utiliser un code correcteur et détecteur d'erreur basique (bit de parité) *(R12)*
  - — Utiliser un code pour la représentation de notes de musique, de vidéo, du braille *(R13)*

Algorithmique

Du point de vue de l'algorithmique, l'élève doit être capable de :

- Lire et comprendre un algorithme simple permettant de résoudre un problème donné
  - — Expliquer un algorithme simple *(A1)*
  - — Simuler l'exécution d'un algorithme simple *(A2)*
  - — Adapter un algorithme simple existant *(A3)*
- Concevoir un algorithme simple permettant de résoudre un problème donné
  - — Écrire un algorithme comme une suite d'instructions *(A4)*
  - — Identifier les instructions de base à disposition *(A5)*
  - — Définir des instructions non ambiguës *(A6)*

  - — Définir et utiliser des expressions de manière appropriée dans un algorithme *(A7)*
  - — Définir et utiliser des variables de manière appropriée dans un algorithme *(A8)*
  - — Utiliser des structures de contrôles (instructions conditionnelles ou boucles) de manière appropriée dans un algorithme *(A9)*
  - — Combiner des instructions, des structures de contrôle, des variables pour écrire un algorithme *(A10)*
- Utiliser un algorithme simple pour résoudre un problème donné
  - — Identifier les entrées/sorties d'un algorithme simple *(A11)*
  - — Discuter des conditions d'utilisation d'un algorithme simple *(A12)*
  - — Comparer l'efficacité de deux algorithmes *(A13)*
  - — Identifier une suite d'instructions qui constituent un tout réutilisable *(A14)*
  - — Réutiliser une suite d'instructions en définissant une fonction *(A15)*

Programmation

En programmation, l'élève doit être capable de :

- Comprendre qu'un langage de programmation est une façon de décrire un algorithme de sorte qu'il soit compréhensible par une machine
  - — Identifier la correspondance entre les instructions de base d'un algorithme et celles du langage de pro- grammation à utiliser *(P1)*
  - — Identifier et utiliser les types de données adéquats *(P2)*
  - — Traduire un algorithme simple dans un langage de programmation *(P3)*
  - — Rédiger un programme simple dans un langage de programmation visuel *(P4)*
  - — Exécuter sur machine un programme simple *(P5)*
  - — Établir les cas de base à tester *(P6)*
  - — Vérifier que l'exécution du programme produit les résultats attendus sur base d'un jeu de données de test *(P7)*
  - — Argumenter de l'intérêt d'utiliser plusieurs jeux de données de test *(P8)*

Matériel

Concernant le matériel, l'élève doit être capable de :

- Comprendre le rôle des différents composants d'un ordinateur dans l'exécution d'un programme
  - Lister les composants de base d'un ordinateur (processeur, mémoire de travail, mémoire à long terme, périphériques d'entrée-sortie) *(M1)*
  - Expliquer la fonction de chaque composant de base d'un ordinateur *(M2)*
  - Discuter de la différence entre la mémoire à long terme et la mémoire de travail *(M3)*
  - Simuler l'exécution d'une séquence d'instructions pour illustrer le rôle des composants de base d'un ordinateur (par exemple, somme de 3 nombres) *(M4)*
  - Discuter de la manière dont une opération de haut niveau utilise les composants de base (par exemple, la visualisation d'une vidéo) *(M5)*

Réseau et sécurité

Pour la thématique réseau et sécurité, l'élève doit être capable de :

- Expliquer comment accéder à des données sur une machine distante
  - Décrire le rôle du navigateur comme programme qui affiche une page web distante (aller la chercher et interpréter le code) *(S1)*
  - Expliquer que le contenu et la mise en forme d'une page web reposent sur un langage de balises *(S2)*
  - Identifier les balises qui permettent d'insérer des liens hypertextes et des images dans une page web *(S3)*
  - Identifier les composants d'un lien internet (protocole, nom de domaine, nom du document) *(S4)*
  - Discuter de la relation entre le nom de domaine et l'adresse IP *(S5)*
  - Décrire le découpage des données en paquets et leur acheminement via des relais *(S6)*

  - — Distinguer Web, Internet et réseau afin d'utiliser ces termes à bon escient *(S7)*
- Pour en savoir plus
  - — Décrire l'impact de l'architecture du web sur la sécurité sur le pistage des utilisateurs (tracking) *(S8)*
  - — Décrire l'impact de l'architecture du web sur la sécurité sur la confidentialité des données et leur intégrité *(S9)*
  - — Identifier les balises qui permettent d'insérer du code dans une page web *(S10)*
  - — Expliciter le rôle des cookies *(S11)*

Annexe B : Réponses à la question « J'estime être capable d'enseigner… »

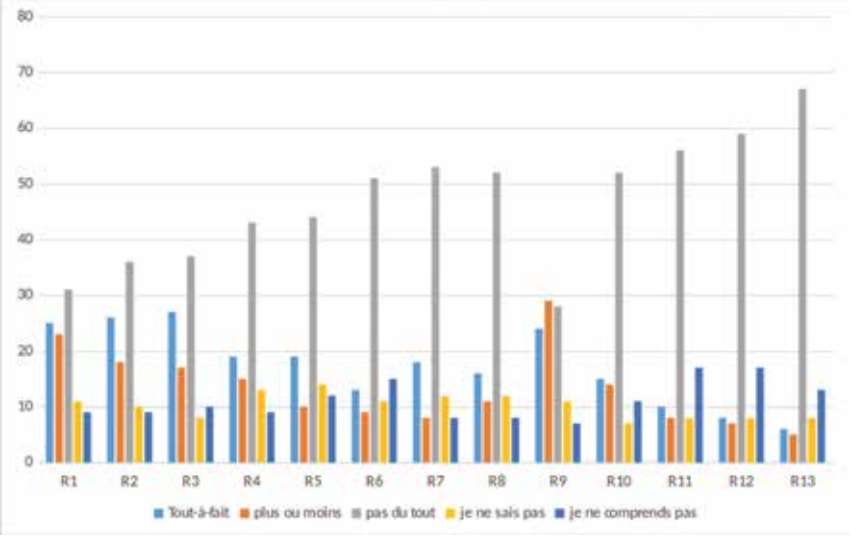


Figure 8 : Résultats pour la thématique Représentation des données.

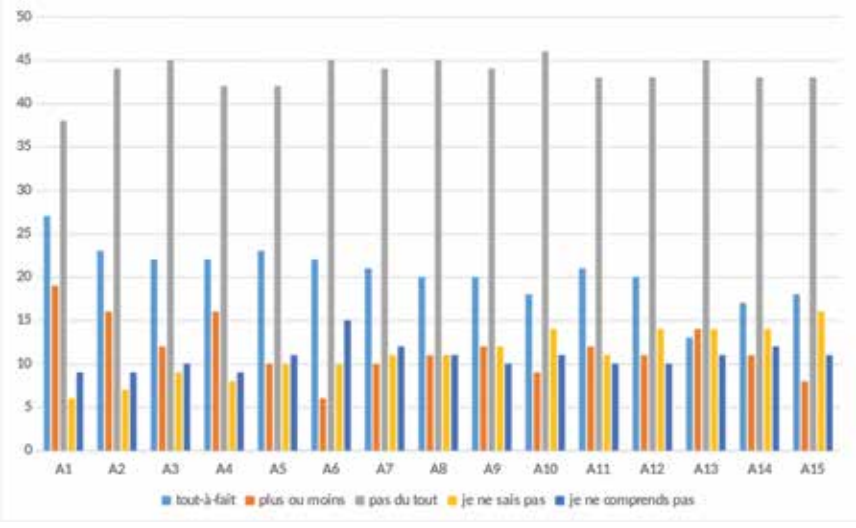


Figure 9 : Résultats pour la thématique Algorithmique.

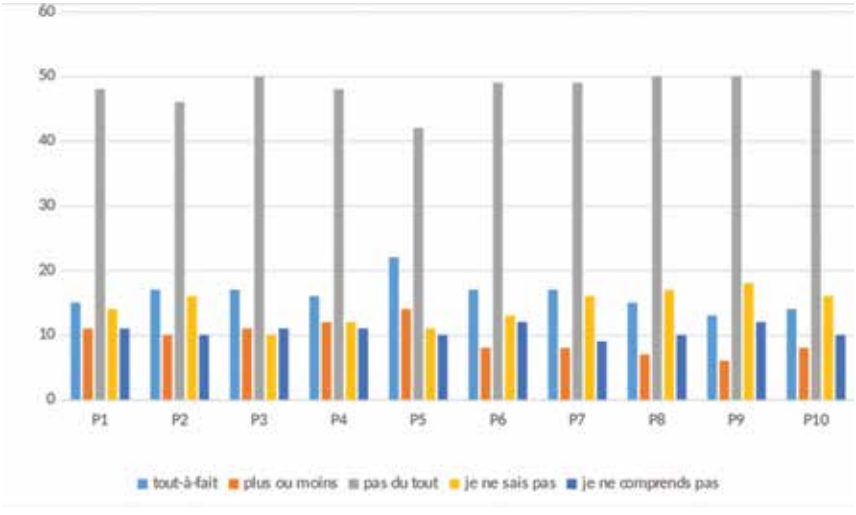


Figure 10 : Résultats pour la thématique Programmation.

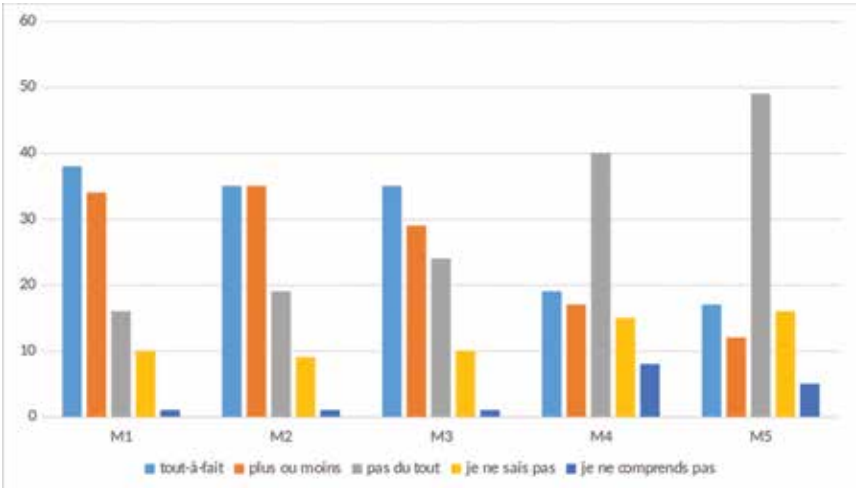


Figure 11 : Résultats pour la thématique Matériel.

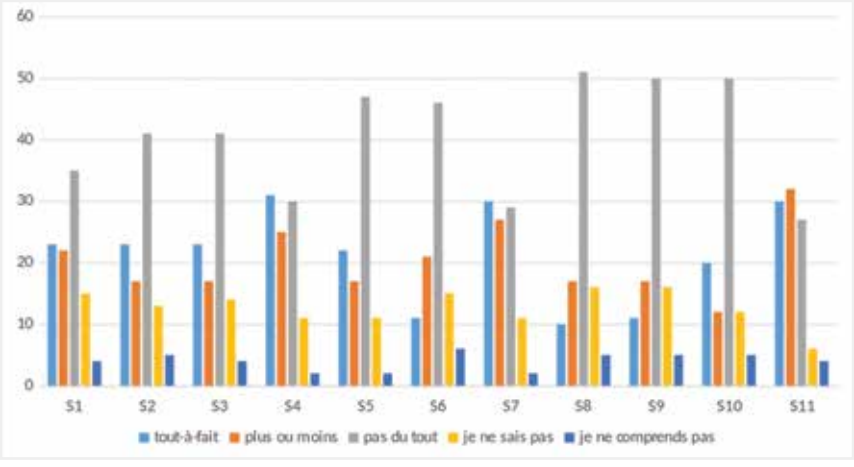


Figure 12 : Résultats pour la thématique Réseau et sécurité.

Florent Tasso & Monique Ouassa Kouaro

Université d'Abomey-Calavi, Bénin
tassoflorent@gmail.com, mkouaro@gmail.com

Intégration de l'enseignement de l'informatique dans les établissements d'enseignement secondaire du Bénin

Résumé
Cette réflexion tente d'appréhender la place de l'enseignement de l'informatique dans les établissements d'enseignement secondaire au Bénin. La démarche méthodologique adoptée est de type descriptif et analytique. Elle exploite pour le recueil des informations l'analyse documentaire, l'observation directe et les entretiens individuels approfondis et de groupe réalisés avec les enseignants, les élèves et les cadres du Ministère de l'Enseignement Secondaire et de la Formation Technique et Professionnelle. La triangulation des informations dans la perspective de la Théorie Unifiée de l'Acceptation et de l'Utilisation de la Technologie (TUAUT) de Venkatesh et al. (2003) révèle que l'informatique peine à être introduit dans les établissements d'enseignement secondaire au Bénin malgré les nombreuses initiatives mises en œuvre dans ce sens. Dans les rares cas d'établissements qui ont réussi à intégrer l'informatique dans les curricula de formation, l'enseignement de cette discipline est noyé dans les autres enseignements scientifiques et techniques, affectant de fait les compétences informatiques des acteurs du système éducatif.

Mots clés : intégration, informatique, déterminants, enseignement secondaire, Bénin

1 Introduction

L'institution scolaire n'est pas en marge de l'explosion des technologies de l'information et de communication (TIC). L'intégration des TIC est d'autant plus inévitable dans le domaine de l'éducation qu'elle permet de favoriser l'accès à l'information ainsi que la réussite des apprenants, de rehausser le professionnalisme du personnel enseignant, d'encourager le leadership des gestionnaires, de favoriser les collaborations Sud-Sud et Nord-Sud, et,

qu'elle offre de multiples solutions pour contrer plusieurs problèmes actuels de l'éducation en Afrique (Karsenti, 2006). Les TIC apparaissent en effet comme des outils d'apprentissage et d'enseignement qui élargissent le champ des options éducatives et des moyens d'optimiser la capacité des différents acteurs du système éducatif. Pour amener les élèves à exploiter efficacement le potentiel des TIC en éducation, la maîtrise de l'informatique s'avère indispensable.

L'enseignement de l'informatique dans les établissements d'enseignement secondaire apparaît dès lors comme un moyen d'initier les élèves aux fonctions principales de l'ordinateur, dans une perspective d'« alphabétisation technologique », tout en développant à la fois leur esprit critique, leurs capacités en matière de déontologie, de comportement en société, et leurs aptitudes à agir et à créer aussi bien au niveau personnel qu'en collaboration avec d'autres ou au sein d'une équipe (Kalogiannakis, 2014).

Cependant, force est de reconnaître que les discours actuels sur l'intégration des TIC de façon globale ou plus spécifiquement sur l'introduction de l'enseignement de l'informatique en contexte éducatif africain semblent beaucoup plus idéologiques qu'empiriques (Karsenti & Collin, 2010). Les usages des outils informatiques en éducation ne sont pas encore ancrés dans la culture et les pratiques professionnelles et il serait illusoire de croire qu'ils le seront uniquement par effet de diffusion de compétences techniques chez les apprenants et chez les enseignants (Béziat, 2012).

Dans le cas spécifique du Bénin, si les travaux de Dakpo *et al.* (2008) soulignent une prise de conscience de l'importance pédagogique des TIC par les différents acteurs du système éducatif, Dahé (2011) fait observer que de nombreux défis entravent l'intégration des outils informatiques dans l'enseignement secondaire public. Selon ce dernier auteur, sur près de 3 725 établissements d'enseignement secondaire et professionnel que comptait le Bénin en 2011, moins de 250 collèges et lycées étaient informatisés et seulement 20 avaient accès à Internet ou plutôt à une connexion bas débit d'environ 56 Ko/s. C'est donc à juste titre que Tasso (2015) fait observer que l'intégration des TIC en éducation reste encore problématique au Bénin, et que les établissements d'enseignement public peinent particulièrement à se mettre au pas de la mondialisation du savoir qu'impose la révolution du numérique. Le diagnostic de Lishou (2017) confirme ce point de vue, car il démontre que des programmes éducatifs conçus pour être dispensés aux moyens d'outils numériques n'existent pas, encore moins des ressources

pédagogiques numériques adaptées et produites par les enseignants de tous les ordres.

Le tableau de l'introduction de l'informatique dans le système éducatif du Bénin est donc globalement peu reluisant. Néanmoins, cette réflexion s'intéresse aux expériences et initiatives mises en œuvre en matière d'intégration d'outils informatiques dans les établissements d'enseignement secondaire, en insistant sur la place de l'enseignement de l'informatique dans les curricula de formation. À partir d'une analyse de la politique éducative du Bénin dans le domaine de l'introduction de l'informatique dans l'enseignement secondaire, elle tente également de proposer de nouvelles mesures en vue d'une véritable éducation à l'informatique dans les lycées et collèges.

2 Matériels et méthodes

Cette recherche, qui tente d'analyser l'intégration de l'informatique dans les établissements d'enseignement secondaire au Bénin, adopte une démarche méthodologique de nature descriptive et analytique. Les enquêtes par questionnaire ont été réalisées avec les enseignants et les élèves identifiés dans quatre des douze départements (Atlantique, Borgou, Littoral et Ouémé) que compte le pays. Elles ont été complétées par 7 entretiens de groupe effectués avec des enseignants et élèves et 27 entretiens individuels approfondis avec les responsables ou promoteurs d'établissements d'enseignement secondaire public et privé ayant implémenté avec succès ou non l'enseignement de l'informatique et des responsables du Ministère de l'Enseignement Secondaire et de la Formation Technique et Professionnel (MESFTP). Au total, un échantillon de 284 personnes a été approché. La phase de collecte des informations de terrain a couvert la période du mois d'août 2017. Le traitement des données terrain a été fait à partir de l'analyse de contenu des discours et du logiciel SPSS.22. L'approche théorique adoptée pour l'analyse des informations a été la Théorie Unifiée de l'Acceptation et de l'Utilisation de la Technologie (TUAUT) de Venkatesh *et al.* (2003).

3 Résultats et discussion

3.1 Politique éducative du Bénin en matière d'introduction de l'informatique dans les établissements d'enseignement secondaire

La triangulation des informations révèle que de nombreuses initiatives ont été entreprises dans l'enseignement secondaire du Bénin dans le but de promouvoir l'introduction de l'informatique. Plusieurs projets de construction/équipement de salles informatiques dans les établissements, de formation des enseignants, d'élaboration d'un programme d'enseignement en informatique, etc. ont été ainsi mis en œuvre au cours de ces trois dernières décennies. L'analyse documentaire des travaux de Dahè (2011), Dakpo *et al.* (2008) et Lishou (2017) permet de citer à titre d'exemple :

– le Projet « Global Learning and Observations to Benefit the Environment » (GLOBE) : mis en œuvre en octobre 1995 dans certains établissements d'enseignements primaire et secondaire du Bénin, ce projet est implémenté depuis 1998 sous le nom de « Projet d'Équipement des Établissements pour la Recherche et l'Étude sur l'Environnement » dans tous les départements administratifs du pays. Il a permis d'équiper 107 établissements et d'initier 231 enseignants à l'utilisation basique de l'ordinateur et à la navigation sur Internet.
– le Projet Education IV : à travers sa composante « introduction des TIC dans le système éducatif béninois », ce projet entré en vigueur en 2005 devrait assurer la construction et l'équipement de salles informatiques dans les lycées et collèges d'enseignement technique et professionnel. À la suite d'une gestion peu efficace, ce projet a pris fin en décembre 2009 après un an de prorogation avec un taux d'exécution de 9 %.
– le Projet d'Introduction de l'Informatique dans les Établissements Secondaires (PIIES) : il a entre autres pour objectifs d'instaurer une culture de la recherche d'information et de l'acquisition du savoir par l'informatique ; de considérer une stratégie Internet comme partie intégrante des programmes éducatifs, des emplois du temps. Le PIIES a réussi à doter quelques lycées et collèges d'ordinateurs sans toutefois les connecter à Internet.
– le Projet de Renforcement de l'Encadrement Pédagogique et de Développement des Technologies de l'Information et de la Communication

en Education (PREPD-TICE) : le Projet de Renforcement de l'Encadrement Pédagogique et de Développement des TICE a été opérationnalisé dans les établissements de Formation Technique et Professionnelle. Il a permis de former des enseignants à l'utilisation pédagogique des TIC et d'initier les élèves des lycées techniques et professionnels à l'intégration des TIC dans leurs pratiques d'apprentissage.

– le Projet de Développement des TIC : il résulte de la fusion du Projet d'Introduction de l'Informatique dans les Établissements Secondaires (PIIES) et du Projet de Renforcement de l'Encadrement Pédagogique et de Développement des TICE (PREPD-TICE).
– le Projet d'Initiation à la Formation Professionnelle à l'Entreprenariat et au Leadership (PIFPEL) : ce projet mis en œuvre par l'arrêté 2009 N°116/MESFTP/DC/SGM/DPP/SA du 6 avril 2009 portant institution, organisation, gestion et financement des activités d'initiation professionnelle dans les établissements secondaires généraux a permis d'initier plusieurs élèves à l'outil informatique. Il a contribué à la construction et à l'équipement d'une vingtaine de salles en ordinateurs et réseaux Internet dans les lycées et collèges ; à la formation de près de 300 enseignants et plus de 4 000 apprenants à l'utilisation de l'outil informatique.

Il ressort toutefois de la triangulation des données des entretiens et des observations de terrain que les différents projets initiés n'ont pas véritablement permis d'introduire de façon durable l'enseignement de l'informatique dans les lycées et collèges. Les difficultés de maintenance de l'équipement informatique acquis se sont souvent accompagnées d'une nette dégradation de l'enseignement de l'informatique, avec la limitation et/ou la disparition des cours. Les propos d'un responsable de collège privé approché dans la ville de Cotonou attestent ce point de vue :

> En 2012, mon établissement a bénéficié grâce à l'appui d'une organisation non gouvernementale d'une salle informatique équipée d'une trentaine d'ordinateurs. Nous avions alors initié l'enseignement de l'informatique dans nos programmes de formation pour une durée de deux heures par semaine notamment pour les classes de sixième, cinquième et quatrième. Il s'agissait d'initier les apprenants aux fonctionnalités de l'ordinateur, à l'utilisation des logiciels Word et Excel et de l'Internet. Malheureusement depuis l'année passée, nous avons été obligés d'interrompre l'enseignement de l'informatique à cause des problèmes de maintenance de l'équipement informatique, d'accessibilité à Internet et de prise en charge du professeur d'informatique.

Ce témoignage met en évidence une expérience d'introduction de l'enseignement de l'informatique dans un établissement secondaire privé qui n'a malheureusement pas fait long feu du fait de l'existence de quelques obstacles. Par ailleurs, le croisement des entretiens réalisés montre que l'informatique peine également à être intégrée dans les curricula de formation des lycées et collèges à cause de plusieurs facteurs d'inertie comme : l'insuffisance d'énergie électrique (30 % de taux d'électrification en milieu rural), le manque d'équipement informatique, l'insuffisance de compétences techniques voire technopédagogiques des enseignants, le scepticisme de certains responsables d'établissements et enseignants quant au bénéfice pédagogique associé à l'utilisation des ordinateurs en classe, les effectifs pléthoriques d'élèves (70 parfois même 100 par classe). Dans certains cas, l'introduction de l'enseignement de l'informatique est parfois reléguée au second rang du fait de la persistance des problèmes traditionnels de l'école béninoise. Les propos d'un directeur d'établissement secondaire public approché à Abomey-Calavi traduisent ce point de vue : « *L'introduction de l'informatique dans les collèges apparaît certes comme une préoccupation importante, mais pour le moment mon établissement est surtout confronté à l'insuffisance de salles de classe, de mobiliers et d'enseignants* ».

En outre, au stade actuel de l'intégration de l'outil informatique dans tous les secteurs de la vie active, l'inexistence d'un document fixant le cadre législatif et réglementaire de l'usage des TIC dans les établissements d'enseignement secondaire apparaît comme un défi majeur. Pour autant, une autorité en charge du Ministère de l'Enseignement Secondaire et de la Formation Technique et Professionnel interviewée à Cotonou fait observer que « *d'ici 2021, des actions concrètes seront mises en œuvre en vue d'introduire de façon durable l'informatique dans les lycées et collèges. C'est ce qui explique la prise en compte du numérique en éducation dans le programme d'action du gouvernement* ». En effet, le 24^{e} projet phare du Programme d'Action du Gouvernement 2016–2021 porte sur la généralisation de l'usage du numérique par l'éducation et la formation. Il vise d'une part à développer l'usage du numérique et les capacités humaines dans le secteur de l'éducation et de la formation professionnelle, technique, initiale et continue et d'autre part à améliorer la qualité de l'éducation par les TIC.

Malgré l'existence de ces facteurs de résistance à l'intégration de l'enseignement de l'informatique dans les établissements d'enseignement secondaire, il est observé chez les élèves (tout comme chez les enseignants) un engouement particulier pour l'informatique. Près de 72 % des élèves

interviewés ont été initiés à l'outil informatique dans les cybercafés. L'informatique apparaît ainsi à la fois comme une science et une technologie que les élèves vivent dans leur quotidien, à travers les réseaux sociaux et les jeux.

Dans le secteur privé, il est nécessaire cependant de préciser que certains établissements ont entrepris des initiatives d'intégration d'outils informatiques dont la dynamique mérite d'être soutenue. C'est notamment le cas du Groupe CERCO (doté d'une plateforme e-learning) ou encore du complexe scolaire Hélios de Bohicon, qui ont réussi à faire de l'enseignement de l'informatique une discipline. La masse horaire de cet enseignement qui n'est pas sans conséquence sur les compétences des élèves est d'au moins deux heures par semaine.

3.2 Compétences informatiques acquises par les acteurs du système éducatif d'enseignement secondaire

Dans un contexte particulier comme celui du Bénin, où l'informatique peine à être intégrée dans les établissements d'enseignement secondaire, les compétences informatiques acquises par les acteurs de ce système restent très peu diversifiées. Dans les rares établissements où l'informatique est introduite dans les programmes de formation, le croisement des informations de terrain fait apparaître que son enseignement ne se limite qu'à une simple utilisation de base de l'ordinateur (écriture de textes, utilisation de quelques logiciels, tableur, etc.) et d'Internet. L'enseignement de l'informatique permet aux élèves de connaître les composantes de l'ordinateur et les objets manipulés comme le clavier, l'écran, la souris, le lecteur de cédérom, ou encore les icônes de l'interface. Il offre aux apprenants la possibilité d'acquérir des compétences élémentaires en matière d'utilisation de l'ordinateur. Selon le témoignage d'un élève en classe de sixième : « *grâce au cours d'informatique que nous recevons dans mon établissement, je sais aujourd'hui allumer et éteindre un ordinateur. Je peux également créer et sauvegarder un document, ce qui me paraissait pourtant mystérieux avant* ». Et un autre en classe de quatrième de renchérir : « *[…] l'initiation à l'informatique reçue en classe me permet sans assistance d'aller au cyber et de faire des recherches documentaires sur Internet. Je suis d'ailleurs plus apte à préparer mes exposés* ». Cette initiation à l'informatique a ainsi l'avantage de permettre aux élèves de faire de l'autoapprentissage,

de l'autoévaluation (production de documents, échanges par courrier électronique, etc.) et de la recherche documentaire (ressources en ligne et hors ligne : dictionnaires, encyclopédies, cédéroms ou services culturels, etc.).

La triangulation des informations de terrain révèle de fait que l'enseignement de l'informatique dans les lycées et collèges du Bénin permet aux élèves d'acquérir des compétences en matière de production de documents ; de gestion des fichiers ; de développement d'idées créatives ; d'utilisation de la messagerie électronique, de tableur et de système de bases de données ; de recherche d'informations sur Internet, comme l'illustre la figure 1.

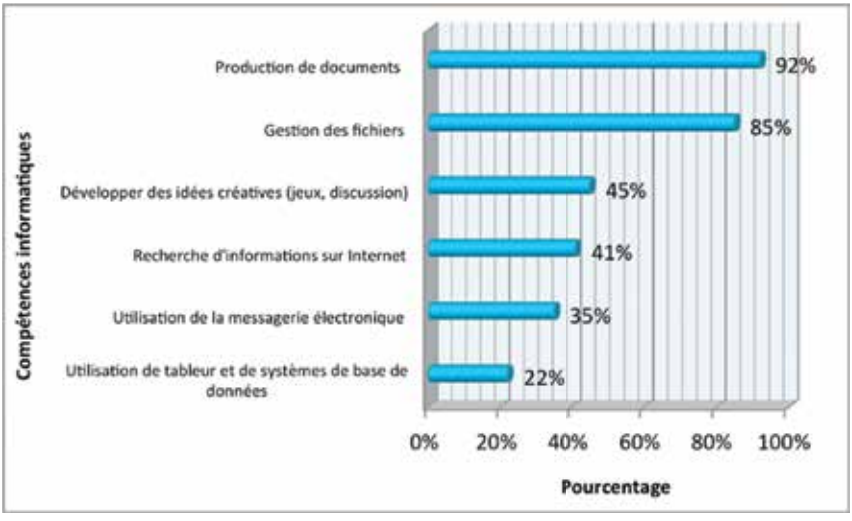


Figure 1 : Compétences acquises par les élèves suite à l'enseignement de l'informatique en classe.
Source : Données de terrain, 2017

La lecture de ce graphique révèle que 92 % des élèves interrogés et dont les établissements exploitent les outils informatiques estiment que l'enseignement de l'informatique leur permet d'acquérir des compétences en matière de production de documents, contre 85 % pour la gestion des fichiers. Les compétences spécifiques liées à la recherche d'informations sur Internet et à l'utilisation de la messagerie électronique représentent respectivement 41 % et 35 %. Ces deux dernières proportions s'expliquent par les difficultés d'accès des établissements équipés d'outils informatiques à Internet. Les compétences les moins acquises par les élèves sont celles relatives à l'utilisation

de tableurs et de systèmes de base de données. L'enseignement de l'informatique a donc l'avantage de permettre aux élèves d'acquérir plusieurs types de compétences. Selon Bullat-Koelliker (2003), ces compétences peuvent être d'ordre intellectuel (utilisation d'informations et jugement critique) ; méthodologique (exploitation efficace des outils informatiques) ; personnel (mise en œuvre de la pensée créative et structuration de l'identité des élèves) ; social (communication et coopération efficace) et réflectif (résolution de problèmes, métaconnaissances et contrôle exécutif).

Par ailleurs, l'enseignement de l'informatique en classe offre la possibilité d'une transformation profonde de la relation pédagogique enseignant-élève, car il contribue à former des élèves d'un type nouveau, encadrés par des enseignants ayant des pratiques pédagogiques nouvelles. Selon les propos d'un enseignant :

> L'enseignement de l'informatique offre l'avantage de rapprocher les apprenants de l'enseignant. Lors des cours d'informatique, l'enseignant est plus proche des élèves : il les suit pratiquement un à un pour mieux les accompagner dans le processus de construction de leur savoir informatique. C'est parfois l'occasion de faire éclore chez certains élèves leur créativité en informatique.

Dans cette perspective, Karsenti (2007) précise que l'intégration pédagogique des TIC dans le domaine de l'éducation conduit à avoir des élèves ayant une meilleure maîtrise des compétences fondamentales et des technologies.

D'ailleurs, 82 % des enseignants interrogés estiment que l'introduction de l'informatique dans les curricula de formation est indispensable pour une formation scolaire adaptée aux réalités du XXI^{e} siècle. Il s'agit d'une technologie utile non seulement aux étudiants mais aussi aux enseignants, car elle permet de développer de « nouvelles » pratiques pédagogiques enrichies (Bonnot *et al.*, 2013). L'enseignement de l'informatique offre ainsi à ces acteurs un potentiel d'usages pédagogiques « innovants » liés à l'information et aux technologies qui lui sont associées. Dans la dynamique de la construction du savoir, l'exploitation des connaissances acquises en informatique offre aux élèves, la possibilité d'accéder au potentiel documentaire d'Internet. Les propos d'une élève en classe de quatrième confirment ce point de vue :

> Depuis la classe de sixième où j'ai commencé par bénéficier des cours d'informatique, j'ai compris qu'Internet pouvait aider les apprenants à réussir sur le plan scolaire. Pour mes cours d'Histoire et Géographie et de Sciences de la Vie et de la Terre, j'ai

> souvent fait des recherches sur la toile qui m'ont beaucoup aidée à mieux comprendre certaines situations d'apprentissages.

Dans ce sens, l'enseignement de l'informatique contribue à améliorer le rendement scolaire des élèves. En outre, au niveau des responsables d'établissements, l'intégration de l'informatique dans les lycées et collèges favorise un meilleur pilotage et une bonne gouvernance. Elle offre la possibilité de mieux gérer les établissements en permettant à leurs responsables de disposer de données fiables et récentes sur les matériels didactiques, les matériels scolaires, le nombre et les qualifications des enseignants. Elle facilite également la production de statistiques scolaires et permet de pallier le manque de données sur la gestion de l'éducation, favorisant ainsi l'élaboration de diagnostics clairs pour la prise de décisions politiques (Lishou, 2017). L'informatique contribue à rehausser la qualité de l'éducation grâce à l'adoption des méthodes pédagogiques de pointe, d'accroître le rendement de l'apprentissage et de réformer les systèmes d'éducation ou d'en améliorer la gestion.

De façon globale, le croisement des données de terrain montre que l'acceptation et l'introduction de l'informatique dans certains lycées et collèges sont sous-tendues par : un souci d'amélioration de la performance des élèves ; l'influence sociale (l'enseignement de l'informatique apparaît comme un prestige social et une source d'attraction selon certains responsables d'établissements) ; une condition de facilitation de l'apprentissage pour les élèves. Ces résultats s'inscrivent bien dans la perspective de la Théorie Unifiée de l'Acceptation et de l'Utilisation de la Technologie (TUAUT) de Venkatesh *et al.* (2003).

Toutefois, il convient de préciser que les facteurs de résistance à l'introduction de l'enseignement de l'informatique dans les lycées et collèges du Bénin sont presque similaires aux obstacles observés dans la plupart des pays d'Afrique. En effet, les travaux de Karsenti *et al.* (2012), qui ont porté sur des initiatives en cours dans plus de cent écoles d'Afrique, montrent que l'introduction de l'informatique dans les systèmes éducatifs reste encore faible. Ces auteurs précisent en effet qu'en tant que *«* région nettement en retard en matière d'adoption, d'utilisation et d'innovation des TIC, l'Afrique ne permet pas encore à ses populations de bénéficier d'une meilleure éducation, voire des possibilités et opportunités d'investissement qu'offrent les TIC en éducation ».

Au regard de cette situation, une véritable intégration de l'informatique dans les établissements d'enseignement secondaire dans le contexte béninois nécessite un changement en profondeur des conceptions de tous les acteurs en présence, voire de l'enseignement au niveau des contenus ainsi que de la forme des activités. Le nouveau rôle de l'enseignant dans ladite situation d'enseignement demande de nouvelles compétences que l'enseignant doit acquérir. L'intégration de l'informatique dans les collèges et lycées oblige donc comme le soulignent Abouhanifa *et al.* (2008) une mobilisation personnelle plus importante qu'en environnement traditionnel d'enseignement.

4 Conclusion

Cet article s'est intéressé à l'analyse de l'intégration de l'enseignement de l'informatique dans les établissements d'enseignement secondaire du Bénin. Les résultats obtenus révèlent que de nombreuses initiatives ont certes été mises en œuvre dans le but de promouvoir l'introduction des Technologies de l'Information et de la Communication (TIC) ou plus spécifiquement de l'informatique dans les lycées et collèges, mais qu'elles n'ont souvent pas permis d'atteindre les résultats escomptés. Dans les rares établissements où l'informatique est enseignée, son introduction dans les curricula de formation est reléguée au second rang. La triangulation des informations de terrain montre d'ailleurs que les compétences informatiques acquises notamment par les élèves ne sont que dérisoires. Elles se limitent généralement à une utilisation basique de l'ordinateur et de l'Internet, entravant de fait l'exploitation judicieuse du potentiel d'usages pédagogiques « innovants » associé aux outils informatiques. Pourtant, l'enseignement de l'informatique offre l'avantage de développer la créativité des élèves, de concourir à la formation d'un type nouveau d'apprenants adapté au développement technologique et de contribuer à l'amélioration du rendement académique.

Dans le but de promouvoir une intégration durable de l'enseignement de l'informatique dans les lycées et collèges du Bénin, quelques mesures méritent d'être prises. Il s'agit notamment de : l'électrification (solaire) des établissements secondaires ; la détermination des autorités politico-

administratives ; l'adoption d'une politique sectorielle en matière d'introduction des TIC dans le système éducatif ; l'équipement informatique et la connexion Internet des établissements d'enseignement secondaire ; l'initiation de classe intelligente pilote ; le recrutement d'enseignants spécialisés en informatique ; la formation et le renforcement des capacités des enseignants à l'utilisation des TIC ; l'intégration de l'informatique dans les curricula de formation ; la création de bibliothèques numériques connectées à Internet ; la création d'une plateforme de gestion et de suivi des statistiques scolaires ; la mise en place d'une plateforme web d'apprentissage ; l'allocation d'un budget alloué à la maintenance de l'équipement informatique ; la résolution des problèmes récurrents à l'école béninoise (insuffisance d'infrastructures pédagogiques, d'enseignants et de mobiliers).

Références

Abouhanifa, S., Benmadani, M., Khalfaoui, M., Hanini, M. et Kabbaj, M. (2008). L'outil informatique : Analyse des obstacles qui entravent son utilisation efficace en classe du secondaire. Association EPI.

Bloch, L. (2013). Une expérience scientifique et didactique : enseigner l'informatique aux biologistes. Dans Vétois, J. (sd). *Enseignement, informatique, TIC et société*, (113–114), ISBN 978–2-343-02473-8.

Bonnot, S., Boulc'h, L. and Delgoulet, C. (2013). Usage des outils informatiques et des plateformes d'enseignement par les étudiants : le rôle du stéréotype de genre. In *L'orientation scolaire et professionnelle*, 42(4).

Bullat-Koelliker, C. (2003). Les apports des TIC à l'apprentissage. Ce qu'en pensent les enseignants qui utilisent les ateliers d'informatique avec leurs élèves. Mémoire de Diplôme d'Étude Supérieure Spécialisée. Faculté de Psychologie et des Sciences de l'Education, Université de Genève.

Dakpo, P. C., Akouété-Hounsinou, F. et Azonhè, T. (2008). L'intégration des TIC dans l'enseignement : quelles perspectives pour l'école béninoise ? Dans Touré, K., Tchombé, T. M. S. et Karsenti, T. (dir.). *ICT and Changing Mindsets in Education*. Bamenda, Cameroun : Langaa ; Bamako, Mali : ERNWACA/ROCARE.

Dayé, K. A. T. (2011). Intégration des TIC dans l'enseignement secondaire général public au Bénin : état des lieux et défis à relever. Communication présentée au Colloque international INRP. *Le travail enseignant au XXIe siècle. Perspectives croisées : didactiques et didactique professionnelle*.

Kalogiannakis, M. (2014). L'enseignement de l'informatique à l'école primaire, au collège, au lycée général et au lycée professionnel dans la Grèce en crise. Adjectif.net [En ligne]. Consulté le 10 juillet 2017. <http://www.adjectif.net/spip/spip.php?article295>.

Karsenti, T. (2006). De l'importance de l'intégration pédagogique par les enseignants du primaire, du secondaire et du tertiaire. Assises de l'éducation et de la formation (14, 16 et 17 septembre). Paris : OIF.

Karsenti, T. et Colin, S. (2010). Quelle place pour les TIC en formation initiale d'enseignants de français ? Le cas de l'Afrique. In *Revue internationale des technologies en pédagogie universitaire*, 7(3), 32–47.

Karsenti, T., Collin, S. et Harper-Merrett, T. (sd.) (2012). Intégration pédagogique des TIC : Succès et défis de 100+ écoles africaines. Ottawa : ON-IDRC.

Lishou, C. (2017). Étude de faisabilité de l'introduction des Technologies de l'Information et de la Communication dans l'éducation au Bénin : sous-secteurs de l'enseignement primaire et secondaire général. Rapport provisoire. Cotonou : UNICEF.

Présidence de la République du Bénin (2016). Programme d'Action du Gouvernement 2016–2021. Projets phares. Bénin révélé. Le nouveau départ. Cotonou, Bénin.

Raby, C. (2005). Le processus d'intégration des technologies de l'information et de la communication. In Karsenti, T. et Larose, F. (dir.), *L'intégration pédagogique des TIC dans le travail enseignant : recherches et pratiques*. Sainte-Foy : Presses de l'Université du Québec. (pp. 79–95).

Tasso, B. F. (2015). Intégration des TIC dans l'enseignement à l'Université d'Abomey-Calavi au Bénin. Thèse de Doctorat en Sociologie du développement, FLASH, UAC.

Venkatesh, V., Morris, M.G., Davis, G.B. et Davis, F.D. (2003). User Acceptance of Information Technology : Toward a Unified View. MIS Quarterly, 27(3), 425–478.

III.
Quels enseignements, avec quels outils ?

Sylvie Tissot & Mireille Bétrancourt

Université de Genève
sylvie.tissot@unige.ch, mireille.betrancourt@unige.ch

La formation des nouveaux étudiants à l'usage des TIC : méthode des invariants ou apprentissage des procédures ?

Résumé
Si elle apparaît encore comme une nécessité, la formation à l'utilisation des logiciels de base (bureautique, OS, outils de l'étudiant) pour les étudiants entrant à l'université peut être proposée sous différentes formes qui dépendent également de la politique institutionnelle à cet égard. La recherche présentée dans cet article a été conduite dans le cadre de l'évaluation d'un dispositif de formation à distance mis en place 6 ans plus tôt et adoptant une approche de type tutoriel (leçon, exercices interactifs) orientée sur l'apprentissage des procédures à reproduire sur le logiciel cible. Sur la base de la littérature, un dispositif alternatif basé sur la méthode des invariants a été mis en place, ciblant la compréhension de concepts du traitement de l'information et la réalisation d'exercices corrigés manuellement par les tuteurs (visaTICE). Une recherche-action comparant deux groupes de participants volontaires suivant l'une ou l'autre des approches a été menée durant une année académique complète. Les résultats montrent que les deux formations aboutissent à des performances équivalentes au certificat ECDL qui mesure les compétences opératoires sur les logiciels cibles. En revanche, les résultats concernant la certification visaTICE, qui cible la compréhension des concepts, montrent une large supériorité des participants au dispositif du même nom par rapport à ceux qui ont suivi la formation orientée procédures. Toutefois, les participants à la formation visaTICE ont regretté un guidage insuffisant sur le niveau procédural, et de manière symétrique, les tuteurs ont dû fournir un tutorat important pour l'accompagnement des étudiants et la correction des exercices. Sur la base de ces résultats, un nouveau dispositif de formation a été conçu pour concilier compréhension des invariants et guidage procédural et dans lequel le rôle des tuteurs a été repensé.

Mots clés : formation à distance, apprentissage de progiciels, évaluation quasi-expérimentale

1 Introduction

Cette recherche orientée vers l'action a pour point de départ le constat, dans les années 2005, du niveau fortement hétérogène des étudiants de 1re année entrant à la Faculté de Psychologie et Sciences de l'éducation (FPSE) de l'université de Genève en termes de maîtrise des outils standards de l'étudiant et notamment bureautique.

1.1 Contexte et questionnement

Le constat ci-dessus n'est pas propre au contexte suisse, puisque la revue de littérature menée par Baron et Bruillard (2008) montre que la plupart des jeunes possèdent une compréhension limitée du fonctionnement des outils qu'ils utilisent quotidiennement, et qu'une minorité développe des usages créatifs (programmation, création de sites web et de blogs). Les enquêtes récentes montrent que ce phénomène persiste en Suisse : le rapport social FORS 2016[1] indique que seuls 20 % des jeunes de 16 à 29 ans déclarent avoir créé un blog ou un site web. En outre, les enquêtes bisannuelles JAMES[2] mettent en lumière l'existence d'une fracture digitale qui reproduit les inégalités sociales, mais également des différences entre cantons.

Dans ce contexte, la FPSE a décidé dès 2006 d'offrir aux étudiants de 1re année issus de ses deux filières (Psychologie et Sciences de l'éducation, soit environ 500 entrants) une formation facultative et non créditée pour acquérir des compétences de base dans les logiciels transversaux (au départ essentiellement bureautiques), sachant qu'aucune formation n'était dispensée au secondaire genevois. Cette formation se déroulait entièrement en ligne, avec un tutorat assuré par des étudiants plus avancés des mêmes filières. Toutefois, les activités de la formation étaient jugées par les étudiants comme très répétitives, peu engageantes, et les tuteurs regrettaient d'être cantonnés à un rôle de support social et de *helpdesk* (assistance technique).

1 <http://forscenter.ch/fr/service-de-donnees-et-d-information-sur-la-recherche/social-indicators-3/social-report/>

2 <https://www.zhaw.ch/de/psychologie/forschung/medienpsychologie/mediennutzung/james/#c77096>

Encore aujourd'hui, la question de la formation des étudiants aux TIC est l'objet de vives discussions (Baron, Bruillard & Drot-Delange, 2015), qui soulignent la tension entre une approche orientée vers la performance opératoire et une approche visant l'acquisition de concepts informatiques. La recherche présentée ici[3] a pour objectif de comparer deux approches de formation alternatives (schématiquement orientée apprentissage procédural ou orientée compréhension) pour développer des compétences dans la maîtrise des logiciels de bureautique (suite MS Office Mac et PC ou OpenOffice et système d'exploitation macOS ou Windows). L'étude visait à explorer deux questions de recherche :

— Quel est l'impact du type de formation sur le résultat de l'apprentissage des étudiants et leur perception de la formation ?
— Comment les tuteurs voient-ils leur rôle dans chacune des formations ?

1.2 Deux dispositifs correspondant à deux approches de formation

Initialement, c'est la solution MEDIAplus des Éditions ENI qui a été évaluée et choisie pour cette formation. Elle correspond, selon la terminologie mentionnée par Depover, Karsenti & Komis (2007) à un tutoriel classique comportant des leçons apportant une description déclarative des procédures à exécuter et des exercices corrigés automatiquement dans le logiciel cible, selon une progression linéaire.

L'objectif de MEDIAplus est d'acquérir les fonctionnalités d'un logiciel bien précis en situation (simulation de l'environnement Microsoft ou Open Office). La formation pour chaque logiciel est organisée en modules, puis chaque module contient des « points ». Un « point » est un petit exercice sous forme d'une question dans l'application ouverte. Pour chaque point, une vidéo (la leçon) explique les notions pour répondre à la question ou réaliser la tâche. La figure 1 montre la forme et le type d'activité à réaliser dans le logiciel Excel.

3 Recherche menée dans le cadre du travail de fin d'étude du Master of Advanced Studies de formateur d'enseignant, par le premier auteur et Lydia Curtet.

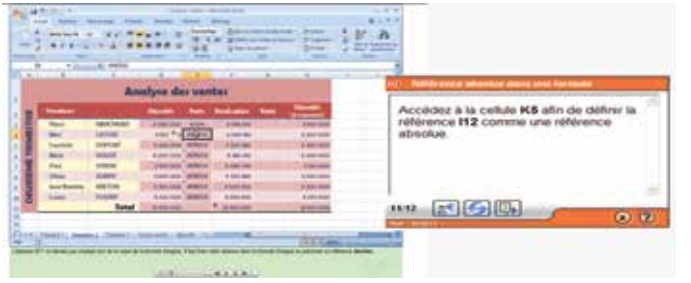


Figure 1 : MEDIAplus : activité dans Excel.

La méthodologie du dispositif repose toujours sur un seul même modèle : un exercice proposé, une leçon s'y rapportant sous la forme d'une vidéo (traduite sous forme de texte), puis une fois l'exercice réalisé, il est testé par le système, un retour est fourni et une solution est disponible sous forme de vidéo.

Ainsi l'apprenant peut observer, lire ou écouter, ensuite il s'exerce dans le logiciel par imitation. Un feedback sous forme de feu vert en cas de réussite, ou de feu rouge en cas d'échec, lui est restitué. L'apprenant ne peut pas continuer s'il n'a pas réussi l'exercice en cours. L'apprenant effectue les points de la leçon selon un guidage strict bien défini.

Compte tenu du taux d'abandon élevé en cours de formation, un sondage a été mené auprès des étudiants en mai 2011 sur cette formation. Les témoignages rapportés ont suscité des interrogations sur la pertinence de la formation concernant l'acquisition pérenne des savoirs et leur transfert vers d'autres logiciels.

> Cette formation est intéressante mais j'ai l'impression que l'on oublie vite les notions apprises !
> Les leçons sont très lentes et répétitives. Il était parfois difficile de suivre.
> […] beaucoup de motivation au début puis la formation devient bien vite secondaire par rapport aux autres exigences universitaires […] la formation PowerPoint m'a irritée fortement, je n'arrivais pas à la finir et ne pouvais pas accéder au module Excel, qui m'encourageait plus.

Comme réponse à ce questionnement, il est décidé de tester le dispositif visaTICE, qui a été développé à l'Université de Liège selon une méthodologie dite des invariants du traitement de l'information numérique (Vandeput, 2011 ; Poisseroux *et al.*, 2009). L'idée principale est de se détacher des

particularités des interfaces des systèmes (logiciels) pour se focaliser sur les principes et concepts fondamentaux (et donc invariants) dans l'objectif d'assurer une pérennité des apprentissages malgré les changements d'interface.

La plateforme visaTICE propose pour chaque tâche présentée, des leçons sous la forme d'un livre, cette métaphore permettant de communiquer l'organisation des contenus de la formation (figures 2 et 3).

Le livre (électronique) est donc structuré en cinq ou six chapitres. Chaque chapitre contient des activités variées telles que mises en situation, réflexions, activités, résumés, exercices, lectures. Les activités sont suivies d'une réponse que l'utilisateur peut dévoiler après avoir cherché à y répondre.

Figure 2 : visaTICE : le livre.

Figure 3 : visaTICE : aperçu d'un chapitre du livre.

La plateforme visaTICE se présente donc comme un système hypermedia selon la terminologie de Depover *et al.* (2010), dans lequel la navigation est libre. L'accès à l'information est facilité mais l'apprenant-utilisateur doit mettre en œuvre des stratégies pour comparer et intégrer les différents éléments.

1.3 Le tutorat pour les deux formations

L'accompagnement humain dans une formation à distance, souvent désigné par tutorat, assume différentes fonctions dans le soutien à l'apprenant, liées aux processus d'apprentissage (soutien cognitif, métacognitif et

socio-affectif), ou aux conditions pratiques telles que les aspects organisationnels et techniques (Depover et Quintin, 2011). La qualité du tutorat est un facteur déterminant de la persévérance et la réussite des étudiants dans les formations à distance (Depover et Quintin, 2011). C'est pourquoi des tuteurs sont engagés chaque année pour suivre, aider, encourager les étudiants. La classe d'un tuteur comprend une quinzaine d'étudiants.

Le rôle des tuteurs dans la formation MEDIAplus consiste à suivre la progression des étudiants, à répondre aux questions, à les encourager et les stimuler. Ils doivent également attribuer les parcours successifs de la formation et rédiger chaque semaine un rapport d'activité. Ils sont également disponibles une heure par semaine en salle informatique pour recevoir les étudiants. La condition pour devenir tuteur est d'être en Bachelor 2e année et d'avoir préalablement suivi la formation MEDIAplus et réussi la certification ECDL Start.

Pour la formation MEDIAplus, l'accompagnement se fait à l'aide d'une plateforme externe (choix de Moodle). Cette plateforme possède les outils de communication synchrones et asynchrones pour que les tuteurs interagissent avec leurs étudiants. Des forums avec des thèmes aussi bien pour les apprenants, leur classe virtuelle et les tuteurs ont été mis en place. Un wiki entre tuteurs, un glossaire et un système de FAQ sont également disponibles. En plus de cet espace d'échanges, les tuteurs se réunissent une fois par semaine pour discuter des problèmes rencontrés par leurs étudiants et de la gestion de leur classe. Au cours de cette rencontre, la coordinatrice insuffle une réelle motivation aux tuteurs pour encourager à leur tour leurs étudiants.

Dans la formation visaTICE, le rôle du tuteur (appelé *coach*) consiste également à stimuler les étudiants mais aussi, si cela s'avère nécessaire, à leur apporter de l'assistance dans la résolution des activités. Le tuteur fournit les solutions des exercices. On peut donc imaginer qu'il est aussi celui qui « corrige » selon des modalités imposées. Dans le cadre de l'étude, ce sont les mêmes tuteurs de la formation MEDIAplus qui ont rempli le rôle de *coach* pour visaTICE. La plateforme visaTICE basée sur Moodle comprend un forum général pour tous les apprenants, pas uniquement les étudiants de l'université de Genève.

2 Méthode

L'étude s'est déroulée sur une année (2012–2013) selon une méthodologie comparative de terrain impliquant 108 étudiants volontaires. Les participants ont été affectés de façon aléatoire (après contrôle de la filière et des scores au pré-test) à l'un des deux dispositifs : 55 pour MEDIAplus, 53 pour visaTICE. Le pré-test et le post-test, identiques, comportaient trois exercices à effectuer dans trois logiciels de la suite Office (traitement de texte, tableau, présentation). En fin d'année, les étudiants restants (N = 53) se sont présentés à la certification proposée par visaTICE et la Certification ECDL Start (*European Computer Driving Licence*), offertes aux participants de l'étude. En outre, des questionnaires ont été administrés aux étudiants pour connaître leur perception du dispositif. Les résultats ne seront pas rapportés dans cet article. Des entretiens ont également été menés avec 6 étudiants (un homme et 5 femmes, tous inscrits en Sciences de l'éducation) et 8 tutrices exerçant leur tutorat dans chacune des formations. La question posée aux tutrices était « exprimez-vous sur votre rôle de tuteur pour chacune des formations ». Les réponses aux entretiens ont été traitées de manière globale et n'ont pas fait l'objet d'une analyse systématique de contenu.

3 Résultats

Au préalable, il faut noter que 42 étudiants (25 dans le groupe MEDIAplus, 17 dans le groupe visaTICE) sur les 108 inscrits ont abandonné en cours de formation, soit 38.9 % de l'effectif. Ce taux, qui peut paraître important, n'est pas si éloigné des pourcentages d'échec/abandon en première année de Bachelor Universitaire. L'échantillon final est donc de 66 participants, 30 dans le groupe MEDIAplus et 36 visaTICE, dont 53 ont passé les deux certifications et 57 les deux pré- et post-tests.

La partie suivante décrit la réussite de tous les participants à la certification proposée par la plateforme visaTICE ainsi qu'à la certification ECDL à laquelle prépare la formation MEDIAplus.

3.1 Comparaison pré/post-test

La figure 4 présente les résultats en pré- et post-test pour chacun des deux groupes de participants (N = 57).

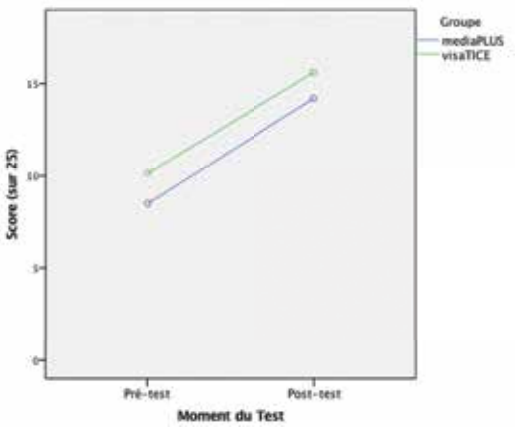


Figure 4 : Résultats des participants aux pré-tests et post-tests en fonction de leur groupe de formation.

Une ANOVA à mesures répétées a été conduite sur la variable score au test en fonction du moment du test (variable intra-sujets à deux modalités) et du groupe de formation (visaTICE ou MEDIAplus). Les résultats de cette analyse montrent que la progression des étudiants entre pré-test et post-test est significative ($F(1, 55) = 79.68$, $p < .0001$, $eta^2 = 1$) mais qu'elle ne dépend pas du type de formation (interaction et effet du groupe, $F < 1$). Les étudiants ont donc amélioré leurs capacités à exécuter des opérations concrètes de façon similaire dans les deux formations.

3.2 Certification visaTICE

La figure 5 montre le taux de réussite de chaque groupe (visaTICE et MEDIAplus) aux tests de la certification visaTICE sur les différents logiciels.

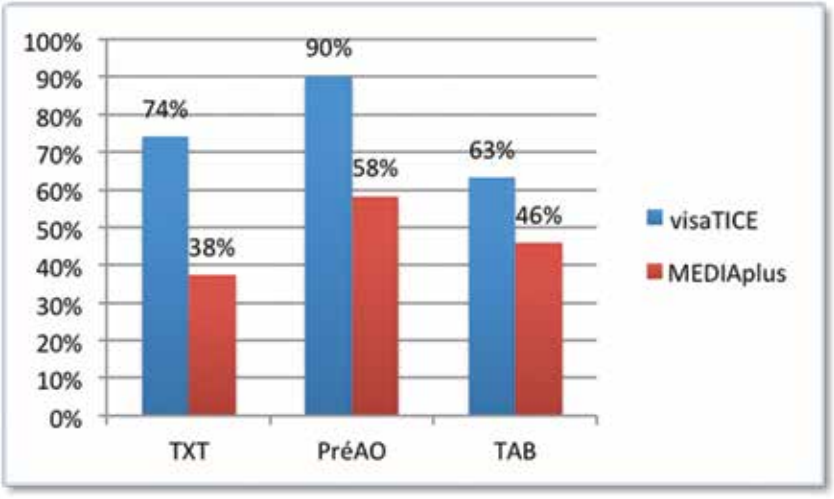


Figure 5 : Résultats de la Certification visaTICE pour les modules TXT (traitement de texte), et PréAO (logiciel de Présentation), TAB (Tableur).

Comme on peut le lire sur la figure 5, le taux de réussite pour la certification visaTICE est nettement meilleur pour les étudiants ayant suivi la formation visaTICE (N = 36) que pour les étudiants ayant suivi la formation MEDIAplus (N = 30).

L'écart entre les moyennes des groupes visaTICE et MEDIAplus est plus faible pour le module Tableur que pour les modules Présentation et Traitement de texte. Ceci peut s'expliquer dans les deux formations parce que les modules tableurs portent sur des éléments procéduraux, alors que les deux autres modules de certification font intervenir plus de notions conceptuelles, qui ne sont pas abordées dans la formation MEDIAplus.

3.3 Certification ECDL

La certification ECDL offre des scores de réussite en termes de pourcentage de réussite de chaque module Windows, texte, tableau et présentation (suite MS Office ou OpenOffice). La figure 6 montre la moyenne des taux de réussite individuels pour chaque module de la certification ECDL pour les participants des deux groupes.

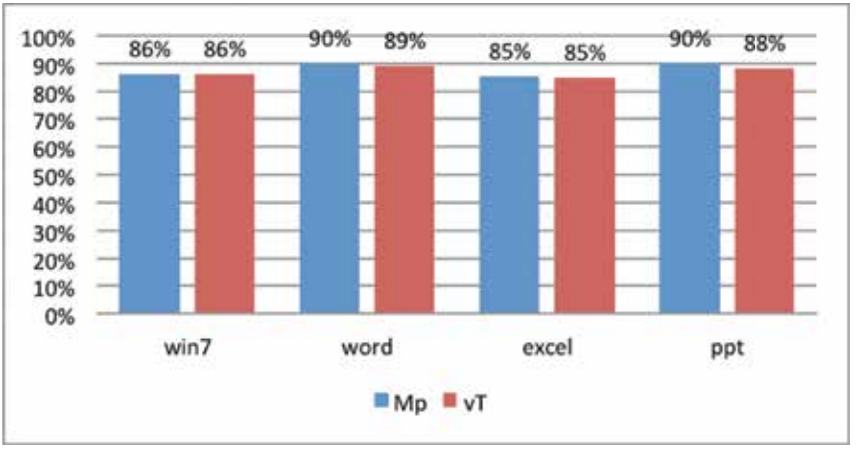


Figure 6 : Résultats de la Certification ECDL pour chacun des modules (Windows 7, Word, Excel, PowerPoint) pour les groupes MEDIAplus (Mp) et visaTICE (vT).

Nous remarquons que les taux de réussite dans chaque module sont élevés et presque identiques pour les étudiants des deux formations. Quelle que soit la formation, les éléments procéduraux nécessaires à la réussite de la certification ECDL ont donc été acquis.

3.4 Résultats des entretiens avec les étudiants

Parallèlement aux tests et certifications, des entretiens auprès de 6 étudiants en Sciences de l'éducation ont été réalisés afin de compléter les résultats quantitatifs sur la question de l'apprentissage. Six entretiens ont donc été réalisés, trois avec des étudiants ayant été formés sur visaTICE et trois autres sur MEDIAplus. Les questions portaient sur les représentations de l'usage des TIC, sur l'effet de la formation sur ces dernières, sur leurs pratiques futures, sur l'apprentissage réalisé.

Perception des étudiants par rapport à MEDIAplus

> Je pense que c'est aussi des exercices qui sont utiles, par rapport à ce qu'on pourrait faire plus tard, non seulement pour maintenant en tant qu'étudiant mais par la suite (E6)

Cette citation révèle une perception positive de l'utilité de la formation, qui est confirmée par les résultats aux épreuves ECDL et au post-test. Sur la forme toutefois, la formation MEDIAplus suscite une certaine monotonie, qui va entraîner chez certains de la lassitude et de l'ennui.

> […] c'est toujours un peu la même chose, c'est peut-être un peu répétitif […] ça devient un peu lassant, c'est toujours la même enfin le même principe […] ça explique lentement (E4)

Les étudiants s'interrogent sur le processus d'apprentissage et sur le fait que de « ne pas chercher » peut entraver l'ancrage du savoir, qu'ils apprennent plutôt le produit de l'apprentissage mais pas la démarche.

> […] le fait de pouvoir voir la réponse et faire la même chose après je sais pas si c'est bien pour apprendre […] c'est long (E5)
> On a toutes les réponses tout de suite ce qu'il faut faire […] c'est une forme de formatage […] question/réponse […] on est plus tenté à vouloir tout de suite la facilité […] on ne peut pas penser autrement (E3)

Ces citations renvoient aux recherches qui montrent, dans une approche constructiviste de l'apprentissage, que les dispositifs de formation doivent susciter une démarche de questionnement de façon à ce que les apprenants puissent « faire sens » des apprentissages, étape nécessaire pour une véritable appropriation des savoirs (Giordan, 1999 ; Tynjälä, 1998).

Perception des étudiants par rapport à visaTICE

Les étudiants déclarent que la formation visaTICE pose les fondamentaux et leur permet de mieux comprendre le fonctionnement des outils bureautiques, comme on peut le voir chez cette étudiante :

> […] par rapport à la théorie au moins on peut mettre un mot exactement […] la théorie ça donne vraiment des mots précis […] c'est quand même réfléchi comme programme […] vraiment conçu […] vraiment pensé […] on privilégie la réflexion (E3)

Toutefois, certains étudiants ont du mal avec l'autonomie et souhaitent un peu plus de guidage bien que l'idée de « chercher par eux-mêmes » est le principe de cette formation.

> On nous laisse très très autonomes […] le corrigé pour moi est encore une consigne […] lorsqu'on a une question on est bloqué pour tout le reste […] on nous laisse un petit peu tâtonner et surtout essayer de chercher d'autres possibilités […] on est

> obligé de tâtonner […] au risque de se décrocher complètement pour tous les corrigés j'avouerais que ça m'a beaucoup frustrée (E3)

Cette étudiante ne semble pas avoir recherché de l'aide auprès de ses pairs ou de son tuteur. Les capacités d'autorégulation des étudiants débutant leurs études universitaires étant parfois embryonnaires, le dispositif doit prévoir de les soutenir dans leur recherche d'aide (Cosnefroy, 2010).

3.5 Perception des tutrices

Huit tutrices (sur les l0) ont été invitées à s'exprimer sur leur rôle dans chacune des formations. Chacune suivait un nombre similaire d'étudiants de chaque formation, ce qui lui donnait la possibilité de comparer.

À propos de MEDIAplus

Le rôle du tuteur dans la formation MEDIAplus porte essentiellement sur la dimension socio-motivationnelle, il doit motiver, relancer et ne pas se décourager suite à des non-réponses, des abandons.

> […] de la déception car beaucoup de mes étudiants ont abandonné, et c'est plus un travail où il faut relancer alors que les étudiants ne répondent pas même pas (A)
> Dans MEDIAplus, j'ai eu moins de questions avec ces étudiants. Cela concernait surtout la relance des étudiants et le suivi des évolutions (C)
> J'ai l'impression d'être moins proche de mes étudiants MEDIAplus. Mais j'ai essayé de les motiver régulièrement, de leur rappeler les échéances de leur montrer l'opportunité et la chance qu'ils avaient de suivre une telle formation (utilité dans les études et dans la vie quotidienne, gratuité, etc.) (M)

Le tuteur est souvent contacté pour des questions d'ordre technique mais son rôle pédagogique est limité, ce qui rend aussi la tâche « plus simple ».

> Le tutorat dans cette formation est plus simple puisqu'il y a plus des questions de cadrage dans l'avancement que de correction. Il s'agit également de répondre à des problèmes techniques plus que de contenu (R)

À propos de visaTICE

Les témoignages montrent que les tutrices se sentent plus investies dans leur travail et ont un vrai rôle pédagogique. Elles renseignent les étudiants sur les contenus et corrigent les travaux. Les tutrices ont apprécié les relations

qu'elles ont pu établir avec leurs étudiants et sont convaincues que cette méthode d'apprentissage renforce l'autonomie de l'étudiant dans ses apprentissages et permet d'acquérir des savoirs en profondeur. D'ailleurs, les corrections leur prennent beaucoup de temps ainsi que les interactions avec les étudiants.

> […] j'ai accompagné l'étudiant durant sa formation, l'aider s'il a besoin d'aide, le laisser faire les exercices de manière autonome tout en étant présente si besoin, l'informer des échéances et des résultats de son parcours. Il m'a fallu plus de temps à consacrer aux étudiants sachant que visaTICE demande plus de feedback que MEDIAplus (N)
> Il faut accompagner l'étudiant. C'est difficile car on doit corriger des exercices mais on n'a pas forcément les connaissances nécessaires, et cela prenait beaucoup de temps. Mais le rapport avec les élèves était super, ils avaient l'air très motivés, je trouve que le tutorat visaTICE était plus enrichissant bien qu'il soit plus compliqué à gérer (A)
> Je me suis sentie vraiment impliquée auprès de mes étudiants visaTICE, je leur ai régulièrement envoyé des feedbacks personnels et je les ai beaucoup encouragés. J'ai l'impression que mon implication a été appréciée, puis plusieurs m'ont remerciée pour mon aide, ma patience, ma rapidité de réponse (M)

Il est à signaler que peu d'activités étaient présentes sur le forum de la plateforme visaTICE et que peu d'étudiants ont posté des messages ; ils s'adressaient directement par mail à leur tutrice. Il était d'ailleurs plus difficile à la coordinatrice de se rendre compte de l'activité des échanges étudiants/tuteurs par mail qu'à travers un forum.

Pour résumer, la formation selon la méthode des invariants semble donc permettre une meilleure compréhension des concepts sans nuire à l'exécution procédurale. Concernant les « défauts » des deux formations, les étudiants dans MEDIAplus se sont plaints d'un dispositif lent, répétitif, avec des blocages décourageants. Dans visaTICE, les plus fortes critiques concernaient le manque de guidage concret pour la réalisation des exercices, les obligeant à chercher par eux-mêmes dans d'autres ressources. Cette difficulté trouvait écho dans la perception des tuteurs de devoir passer un temps très important pour la correction des exercices. Ces derniers appréciaient toutefois leur rôle plus riche, incluant des aspects pédagogiques, plus en accord avec le rôle multidimensionnel des tuteurs en formation à distance (Depover, De Lièvre, Peraya *et al.*, 2011).

4 Proposition d'un dispositif repensé : UniTICE

Les résultats de l'étude confirment l'intérêt d'une approche qui aborde les principes invariants qui gouvernent le fonctionnement opératoire des logiciels (Vandeput, 2011 ; Poisseroux *et al.*, 2009). Ils suggèrent néanmoins qu'une approche intermédiaire, conciliant explications des principes et guidage procédural, serait souhaitable pour satisfaire aux différentes contraintes de moyens d'encadrement et de disponibilité des étudiants, dans la lignée des constats de Baron *et al.* (2015).

Suite aux expériences des dispositifs précédents, le projet de construire une nouvelle plateforme « home made » a mûri et s'est inscrit dans le cadre d'un stage du master MALTT avec la réalisation du premier module sur le traitement de texte. Ce nouveau dispositif suit un certain nombre de principes décrits ci-après.

4.1 Une seule plateforme pour l'enseignement et l'accompagnement

Les étudiants travaillaient sur deux plateformes distinctes, visaTICE (téléchargement des exercices, contenu des savoirs) et Moodle (plateforme d'accompagnement et de suivi, forum, échéances et dépôt des exercices), et devaient naviguer entre les deux. Les étudiants se perdaient entre les deux plateformes et demandaient « où déposer les exercices », par exemple. Ces deux environnements manquaient d'unité et l'apprenant avait une vision « partielle, morcelée et ‹a-structurée› de la somme d'information à laquelle il peut accéder » (Peraya, 2003, p. 12). Il a donc été décidé de fusionner les deux environnements en une seule plateforme de type LMS (en l'occurrence Moodle).

4.2 Un renforcement de l'activité sur le forum

Le forum prévu comme outil de communication entre pairs et le tuteur était très peu utilisé. On constate que les étudiants n'ont pas d'objectif à communiquer avec les autres, mais peut-être faut-il susciter des discussions par les pairs par rapport à une activité précise. Dans la conception d'un dispositif technologique d'apprentissage à distance, le « tuteur aide au

travail collaboratif en intervenant dans le choix des technologies à utiliser (chat, forum, échanges de fichiers, etc.), en proposant une planification du travail, en rappelant les consignes, les délais… » (Docq & Daele, 2002).

Dans UniTICE, le tuteur incite aussi la collaboration en lançant une discussion dans le forum en lien avec l'activité à réaliser pour que cet outil devienne un réel instrument au service de l'activité, adapté au scénario pédagogique mis en place. De plus, afin de motiver l'étudiant à participer au forum de sa classe, il est prévu que sa participation hebdomadaire au forum compte dans la note finale de son travail. La plateforme LMS devenant le lieu principal d'apprentissage, on peut s'attendre à ce que les étudiants utilisent plus spontanément le forum pour communiquer avec leur tuteur.

4.3 Un guidage subtil pour l'apprenant

Dans ce dispositif, des chemins d'apprentissage sont insérés afin de guider au mieux l'apprenant. Le chemin d'apprentissage est créé avec des modules SCORM à partir du logiciel EXe, qui permet de développer des structures d'apprentissage comprenant différents blocs (texte, activité, exercice, etc.) hiérarchisés et reliés entre eux, de manière à mettre sur pied des chemins d'apprentissage « maison » (Figure 6).

Figure 7 : Chemin d'apprentissage (paquet Scorm).

Des exercices adaptés aux étudiants et des corrigés détaillés

La formation visaTICE a été créée à destination des élèves de l'enseignement secondaire qui visent des études supérieures. Les contenus de certains exercices n'étaient donc pas toujours adaptés aux besoins de nos étudiants universitaires. L'exercice enfantin du jeu de la bataille navale pour identifier

les cellules dans une feuille de calcul en est une illustration. Il a semblé plus pertinent de proposer des contenus de la formation en lien avec le contexte de l'étudiant, en proposant des exercices portant sur des éléments qu'ils peuvent rencontrer au cours de leurs études. D'autre part, afin de pallier le manque d'informations procédurales, il a été décidé de fournir à l'étudiant un accès à une bibliothèque numérique sur les logiciels bureautiques (bibliothèque Educatic, actuellement la bibliothèque Eureka de chez ENI).

Feedbacks personnalisés pour les exercices

Selon la même logique que visaTICE, les étudiants reçoivent dans UniTICE une évaluation de leur travail avec un retour formatif directement sur la plateforme Moodle. Pour faciliter le travail du tuteur et assurer l'homogénéité des évaluations, des grilles d'évaluation sont construites et fournies aux tuteurs.

Accompagnement des tuteurs

Comme pour les deux dispositifs précédents, ce dispositif propose un accompagnement par des tuteurs, élément indispensable de toute formation à distance, *a fortiori* pour un public d'étudiants en première année de formation présentielle. Le rôle du tuteur est davantage valorisant et intéressant dans ce dispositif. Le tuteur intervient sur son forum en lançant des sujets de discussion propres à chaque activité, le tuteur stimule l'étudiant pour réaliser ses travaux, mais assume un vrai rôle pédagogique en fournissant de l'assistance, des retours formatifs à ses étudiants. Le tuteur contacte par visioconférence son étudiant ou le reçoit s'il a besoin de monstrations ou de renseignements complémentaires.

Une fois par trimestre, le tuteur rencontre ses étudiants en présentiel pour faire un point de situation et présenter le module suivant. La première présentielle en début d'année est très importante pour définir les règles du jeu, pour montrer la plateforme d'apprentissage, pour mettre en confiance l'étudiant dans cette formation à distance.

Le tuteur, bien qu'avancé en informatique et possédant l'habilité de transférer les connaissances d'un système à l'autre ou d'un logiciel à l'autre, estime être plus efficace sur son forum si les étudiants de sa classe virtuelle possèdent le même système d'exploitation que lui. En effet, le tuteur fournit de l'aide avec des captures d'écran ou vidéos qu'il réalise sur

son ordinateur portable. La répartition des étudiants par classe en fonction de ce critère est à considérer.

Une réunion hebdomadaire de tuteurs déjà existante dans les dispositifs précédents permet de renforcer l'esprit d'équipe, de mettre en commun les questions rencontrées, partager les expériences de tutorat, de stimuler également les tuteurs dans une formation à distance.

5 Discussion et conclusion

L'étude nous a permis de comparer l'apprentissage réalisé avec deux formations fort différentes : le dispositif MEDIAplus est centré sur l'acquisition de procédures, sous la forme de tutoriel, alors que le dispositif visaTICE est basé sur le principe des invariants et délivré sous la forme d'un hypermédia accompagné d'exercices intégratifs, corrigés par les tuteurs humains. L'évaluation quantitative montre que les notions conceptuelles véhiculées par le dispositif visaTICE sont bien acquises, sans que cela ne soit préjudiciable à l'acquisition de savoirs procéduraux. Les taux de succès à la certification ECDL sont très élevés, montrant que les deux formations tiennent leur promesse. Les entretiens avec les étudiants ou les tuteurs montrent une préférence pour le dispositif visaTICE basé sur la compréhension du sens des apprentissages (Tynjälä, 1998) et qui permet aux tuteurs de véritablement accompagner le processus d'apprentissage (Depover & Quintin, 2011). Toutefois ce dispositif a été jugé peu guidant, que ce soit pour les étudiants, qui ont dû chercher les informations dans d'autres sources, ou les tuteurs, qui ont passé énormément de temps en correction. Le nouveau dispositif proposé, UniTICE, a été conçu en prenant ces différents éléments d'évaluation en compte.

Concernant le renouvellement des formations, nous reprendrons les particularités suivantes de chaque dispositif. Dans le dispositif MEDIAplus, la notion de guidage est à conserver. En effet, cette formation, destinée aux premières années de la FPSE qui ont besoin d'une mise à niveau des TIC, est suivie en parallèle des études et doit les mener le plus directement au but. De plus, les résultats des sondages montrent que les étudiants apprécient toujours le cadrage du « collège » même s'ils intègrent l'université. Par ailleurs, le dispositif basé sur le « savoir-faire » permet aux étudiants

qui ont des niveaux plus faibles d'acquérir un bon niveau de connaissances. Dans ce dispositif, nous retiendrons également le tutorat par des pairs qui ont suivi la formation auparavant. Cet accompagnement permet d'assister de façon efficace les étudiants et de les stimuler jusqu'à la fin de la formation à distance.

Dans le dispositif visaTICE, la diversité des activités permet à l'étudiant d'être confronté à différentes situations, un véritable atout pour l'apprentissage. De plus, la méthode des « invariants » utilisée dans ce dispositif est très intéressante notamment pour asseoir les fondamentaux, pour se détacher des aspects de l'interface d'un logiciel et donc transposer des savoirs d'un logiciel à l'autre. Nous retiendrons aussi que visaTICE vise l'autonomie de l'étudiant dans les apprentissages, et demande une recherche des savoir-faire. Cependant, pour certains étudiants en difficultés, cette exigence demande beaucoup d'efforts et peut causer des abandons.

Bien que des évaluations quantitatives et qualitatives aient été menées, l'étude présentée dans cet article avait pour objectif premier l'évaluation d'un dispositif de terrain en vue d'une reconception. Les instruments de mesure utilisés ont été conçus ad hoc et les entretiens ont été analysés de façon globale sans suivre de méthodologie d'analyse systématique de contenus. Alors qu'un nouveau dispositif se met en place, il serait utile de conduire une analyse des apprentissages et des perceptions des étudiants avec UniTICE ainsi que d'explorer l'expérience des tuteurs quant à leur rôle dans ses dimensions pédagogique, socio-motivationnelle, organisationnelle, et technique.

Dans le souci de répondre aux besoins actuels des étudiants par rapport aux TIC, nous étudions un questionnaire plus large sur les compétences numériques qui permettra de déterminer quels nouveaux modules de formation sont à intégrer (programmation web, images/vidéos, compétences informationnelles, etc.). Au début de cette année, les pré-tests effectués ont montré que les besoins en connaissances bureautiques restent toujours présents. Par ailleurs, nous constatons que les étudiants utilisent de plus en plus de tablettes (Android, iPad, Windows) pour suivre les cours et n'ont pas forcément un autre ordinateur. De nouvelles études sont nécessaires pour évaluer si ces nouveaux usages doivent susciter de nouveaux besoins de formation aux TIC (comme le stockage des données, le transfert de données, la sécurité des données, etc.).

Références

Baron, G.-L., & Bruillard, E. (2008). Technologies de l'information et de la communication et indigènes numériques : quelle situation ? Revue STICEF, 15, 1–12. Consulté sur <http://sticef.univ-lemans.fr/num/vol2008/09r-baron/sticef_2008_baron_09.htm>.

Baron, G.-L., Bruillard, E., & Drot-Delange, B. (dir.) (2015). *Informatique en éducation : perspectives curriculaires et didactiques*. Clermont-Ferrand : Presses universitaires Blaise-Pascal.

Cosnefroy, L. (2010). Se mettre au travail et y rester : les tourments de l'autorégulation. *Revue Française de Pédagogie, 170*, 5–15.

Depover, C., & Quintin, J.-J. (2011). Le tutorat et sa mise en œuvre. Dans C. Depover, B. De Lièvre, D. Peraya, J.-J. Quintin et A. Jaillet (dir.), *Le tutorat en formation à distance* (p. 39–42). Bruxelles : De Boeck.

Depover, C., De Lièvre, B., Peraya, D., Quintin, J.-J., & Jaillet, A. (2011). *Le tutorat dans la formation à distance*. Bruxelles : De Boeck.

Depover, C., Karsentis, T., & Komis, V. (2007). *Enseigner avec les technologies. Favoriser les apprentissages, développer des compétences*. Québec, Canada : Presses de l'Université du Québec.

Docq, F., & Daele, A. (2002, mai). *Le tuteur en ligne, quelles conditions d'efficacité dans un dispositif d'apprentissage collaboratif à distance*. Présentation lors du 19e Colloque de l'AIPU – Louvain-la-Neuve. Consulté sur <www.ipm.ucl.ac.be>.

Giordan, A. (1999). *Apprendre*. Paris : Belin.

Peraya, D. (2003). De la correspondance au campus virtuel : formation à distance et dispositifs médiatiques. In B. Charlier et D. Peraya (dir.). *Technologie et innovation en pédagogie : dispositifs innovants de formation pour l'enseignement supérieur* (p. 79–92). Bruxelles : De Boeck.

Poisseroux, J., Lassaux, E., & Vandeput, E. (2009). TacTIC pour une intégration réussie des technologies en Haute École. In Baron, G.L, Bruillard, E. & Pochon, L.O. (Eds.), *Informatique et progiciels en éducation et en formation. Continuités et perspectives*. Paris : INRP, Coll. Technologies nouvelles et éducation.

Tynjälä, P. (1998). Traditional Studying for Examination versus Constructivist Learning Tasks : do learning outcomes differ ? *Studies in Higher Education*, *23*(2), 173–189.

Vandeput, E. (2011). Méthodologie d'identification des invariants du traitement de l'information numérique. *Communication proposée au Colloque International Didapro 4*, Université de Patras, Patras, Grèce.

Christelle Pauty-Combemorel

Laboratoire ÉDucation et Apprentissages (EDA), Université Paris Descartes
christelle.pauty@etu.parisdescartes.fr

Utilisation d'un jeu vidéo dans le cadre de l'enseignement des SVT : le cas de Minetest

Résumé
Cet article s'inscrit dans le cadre d'une thèse de doctorat qui s'intéresse à la construction de la culture numérique de collégiens bénéficiant d'un équipement informatique pouvant être utilisé en classe. Nous nous focalisons ici sur la manière dont Minetest est utilisé en cours de Sciences de la Vie et de la Terre (SVT). Nous avons cherché à comprendre comment les élèves d'une classe de 6e (secondaire Niveau 6) utilisaient ce logiciel durant sept séances de cours et apprenaient autant au plan disciplinaire qu'informatique et ce, bien que ce dernier ne soit pas l'objectif principal de leur enseignant. Notre méthodologie s'appuie sur des observations en classe ainsi que sur des entretiens informels avec les élèves et l'enseignant. Nos résultats montrent que l'utilisation de ce logiciel offre d'une part, la possibilité aux élèves de développer, par la manipulation et l'expérimentation, une compréhension sur certains phénomènes scientifiques qu'il serait difficile d'observer en raison de contraintes temporelles et spatiales ; d'autre part, d'acquérir des compétences techniques en informatique par la répétition d'actions.

Mots clés : appropriation, collégiens, Minetest, SVT

1 Introduction

Nous nous concentrons dans cet article sur la manière dont les élèves d'une classe de sixième (secondaire Niveau 6) bénéficiant chacun d'un ordinateur portable pouvant être utilisé en cours sont amenés à utiliser Minetest en cours de Sciences de la vie et de la Terre (SVT), discipline scolaire regroupant plusieurs sciences telles que la biologie, la botanique et la géologie. Ce jeu vidéo a été employé à des fins pédagogiques par leur professeur pour

leur permettre de développer des compétences scientifiques prescrites par les programmes d'enseignement.

2 Contexte

Minetest est un jeu vidéo de type bac à sable figurant parmi les ressources matricielles libres des ordinateurs portables dont chaque élève de notre corpus bénéficie et peut utiliser en classe. Ce jeu ressemble dans son esthétique et sa jouabilité à Minecraft. Pour filer la métaphore en lien avec les jeux de l'enfance, des matériaux sous forme de blocs cubiques peuvent être utilisés par les joueurs dans un espace délimité pour réaliser toutes sortes de constructions. Ces blocs peuvent être sélectionnés à partir d'un inventaire. Il est aussi possible de les créer, de collecter divers matériaux (bois, eau, métaux, pierre…), voire de les détruire. Il n'y a pas d'objectifs prédéfinis. Les joueurs bénéficient d'une grande liberté puisque « c'est en modifiant des espaces, des lieux, des personnages, qu'ils produisent et façonnent l'architecture de leur partie, agissent et créent leur propre histoire » (Voulgre, 2013).

Ce jeu, qui n'a l'origine pas été conçu pour l'éducation, a été détourné par l'enseignant de SVT pour « servir des finalités sérieuses non anticipées par [ses] concepteurs » (Alvarez, Djaouti, & Rampnoux, 2016, p. 35). Il donne des « opportunités d'apprentissage » (Roberge, 2016, p. 8) puisqu'il permet à ses utilisateurs d'incarner un personnage, d'interagir avec d'autres personnes, de gérer la construction d'un environnement. Il offre aussi l'opportunité de « faire des tests en toute sécurité, d'apprendre les fonctionnalités d'un logiciel par tâtonnements » (Voulgre, 2013). Il peut, par conséquent, être considéré comme un lieu possible d'apprentissage et de collaboration (Quinche, 2015).

Minetest possède, il nous semble, des points communs avec la notion de micromonde. Il se distingue néanmoins de Scratch, un logiciel de programmation de type micromonde, dans lequel deux éléments insécables sont intégrés dans « une interface unifiée qui allie un éditeur de texte et une fenêtre de visualisation » (Libert & Vanhoof, 2016, p. 3). Il offre l'opportunité aux élèves de construire leurs savoirs en faisant leurs propres expériences (Papert, 1981).

Son emploi dans le cadre de la classe repose sur des principes de ludification. Comme beaucoup de jeux vidéo populaires actuellement tels que *Call of duty*, *Starcraft*, ou encore *Minecraft,* Minetest permet aux joueurs de personnaliser leurs pratiques de jeu en leur permettant d'une part, de déterminer s'ils souhaitent jouer en mode « solo » ou en « multijoueurs » ; d'autre part, de choisir parmi trois « degrés de ludicité » (D'Afflon, 2013). Un but du jeu doit être défini, ce qui suppose l'existence de règles (degré 1) et d'une forme de gratifications positives et/ou négatives (degré 2). Enfin, « un jeu peut définir des conditions de défaite, sans toutefois déterminer des conditions de victoire » (d'Afflon, 2013, paragr. 5).

3 Déroulement des activités proposées aux élèves

L'enseignant s'appuie sur les pratiques ludiques des jeunes de jeux vidéo pour qu'ils puissent, déclare-t-il à plusieurs reprises, « apprendre en s'amusant », « apprendre autrement » des notions de biologie et de géologie.

L'enseignant emploie ce logiciel avec l'ambition de susciter l'intérêt de ses élèves, faciliter leur compréhension de certaines notions de biologie (cycle de vie, chaînes et réseaux alimentaires, etc.) et de géologie (érosion, volcanisme, etc.) par la manipulation et la visualisation de phénomènes qui seraient difficilement observables pendant une heure de cours pour des raisons économiques, de temps (la vitesse des phénomènes géologiques, par exemple) ou d'espace (la dimension gigantesque ou microscopique de certains objets scientifiques). Dans cette perspective, plusieurs activités pédagogiques sont proposées aux apprenants. L'accent est davantage mis sur la compréhension de phénomènes scientifiques par la manipulation et la visualisation que sur l'informatique.

Les trois premières séances avaient pour but de faire reconstruire virtuellement l'établissement scolaire et ses alentours aux élèves en restant fidèle à la réalité afin de préparer l'arrivée des nouveaux collégiens. Il s'agissait aussi de leur permettre de se familiariser avec le logiciel en manipulant les touches de contrôle permettant de se déplacer, construire, détruire et discuter par l'intermédiaire du tchat.

L'objectif des deux séances suivantes était de « faire une mini-représentation » pour « répéter comme au théâtre » la présentation faite à

l'occasion d'un salon annuel réunissant les professionnels de l'éducation et de l'innovation autour des Technologies de l'information et de la communication. Des « missions » en lien avec le programme d'enseignement et les activités pédagogiques réalisées en cours ont été distribuées aux cinq groupes formés (construire un enclos et y mettre des poules, créer une énigme et la cacher, etc.).

Les deux dernières séances visaient d'une part, à explorer un nouvel environnement virtuel créé par l'enseignant et à « faire une description aussi fidèle que possible de la planète » en rédigeant une synthèse sur un document de traitement de texte ; d'autre part, à faire le bilan des séances réalisées sur Minetest en rappelant les notions disciplinaires importantes à retenir.

Nous avons cherché à comprendre comment les élèves d'une classe de sixième utilisaient ce logiciel en cours de SVT et apprenaient autant au plan disciplinaire qu'informatique et ce, bien que ce dernier ne soit pas l'objectif principal de leur enseignant.

Afin de répondre à cette interrogation, nous montrerons comment les élèves ont détourné les consignes et les réglages du jeu. Cela nous conduira à réfléchir à la manière dont ils développent des compétences scientifiques et informatiques. Enfin, nous proposerons des éléments de discussion et quelques perspectives.

4 Méthodologie

Le collège, dans lequel nous avons réalisé nos observations, est situé en zone urbaine dans la proche banlieue parisienne. Il accueille près de cinq cents élèves et une trentaine de professeurs.

Nous avons observé de façon hebdomadaire de janvier à juin 2016 une classe de sixième constituée de 25 élèves (9 filles et 16 garçons) en SVT et en histoire-géographie dans la mesure où il s'agissait du seul groupe qu'ils avaient en commun les jours où nous pouvions être présente dans les locaux. Minetest est uniquement employé en SVT.

Parmi les séances observées en SVT, le logiciel a été employé dans sept d'entre elles, dans le cadre de plusieurs activités pédagogiques. Cinq séances se sont déroulées en classe entière et deux en demi groupe.

Nous avons réalisé des observations directes en nous focalisant d'une part, sur le contexte, les objectifs pédagogiques déclarés ou non et le type d'activité ; d'autre part, sur les apprenants en observant leur manière de manipuler les technologies et les interactions qu'ils pouvaient avoir avec les TIC, entre eux et avec l'enseignant.

Par ailleurs, nous avons réalisé des entretiens informels avec l'enseignant de SVT avant et après chaque séance pour comprendre les enjeux informatiques et disciplinaires. Nous avons aussi effectué trois entretiens collectifs de vingt minutes environ avec 11 élèves sur les 20 observés à la fin de l'année scolaire 2016. Nous les avons notamment questionnés sur leurs usages de Minetest au collège et en dehors ainsi que sur les représentations qu'ils ont de certaines notions (craft, pixel, programmation, etc.).

5 Formes de détournement des élèves

Nous avons repéré que la plupart des élèves transgressaient les consignes données par l'enseignant afin de satisfaire des critères ergonomiques d'utilisabilité et d'utilité du logiciel. Ces détournements semblent tolérés, voire parfois incités, par l'enseignant, pour permettre aux élèves de se familiariser avec le logiciel.

5.1 Au plan des consignes

Si chaque élève dispose de son propre ordinateur pour travailler et qu'une grande majorité suit les consignes de l'enseignant en réalisant les tâches prescrites de façon autonome, une phase de tâtonnement a néanmoins été nécessaire pour notamment se familiariser avec les combinaisons de touche.

Lors de la première séance, certaines élèves ont construit leur « maison » pour se protéger des « monstres » du jeu en apportant un soin particulier à l'esthétique des matériaux qu'elles sélectionnaient. Plusieurs garçons se battaient virtuellement contre les « monstres ».

Quelques rares élèves, par ailleurs considérés par leurs enseignants comme étant en échec scolaire, ont aussi des difficultés dans la mise en œuvre du jeu pour utiliser les raccourcis de base. Cela, il nous semble, les

a amenés à mettre en œuvre des formes de stratégies d'évitement (oubli du matériel) pour travailler avec des camarades plus expérimentés et ainsi atténuer leur appréhension.

5.2 Au plan des manipulations techniques

Bien que la majorité des raccourcis par défaut demeurent similaires à ceux de Minecraft, les élèves ayant une expérience disent repérer quelques différences, notamment avec les graphismes. Un élève déclare à ce propos : « c'est vrai que c'est beaucoup pixelisé et tout ça sauf que c'est plus lent. La jouabilité est plus lente… ».

Cela pourrait être expliqué par le fait que l'enseignant déclare avoir « intentionnellement » distingué certains réglages en les modifiant dans le but de « mettre [les élèves] sur un pied d'égalité » dans la découverte du fonctionnement du logiciel et ainsi ne pas privilégier ceux ayant l'habitude de jouer à Minecraft.

Nous avons observé quelques élèves déclarant avoir une expérience de Minecraft modifier les réglages du logiciel dès leurs premières utilisations en classe, sans doute, pour se rapprocher de leurs pratiques ludiques. Dans la mesure où chaque touche du clavier correspond à une action, modifier les paramètres peut conduire à des bogues. Cela est expliqué par l'enseignant aux élèves par le fait qu'en fonction des changements opérés plusieurs raccourcis peuvent entrer en tension et ainsi produire d'autres résultats que ceux attendus.

Ainsi, nous avons repéré des élèves qui tentaient de transposer ce qu'ils connaissaient du fonctionnement de Minecraft à Minetest. Une élève souhaitant accéder à l'inventaire actionne le mode rapide du jeu en appuyant sur la touche « E » au lieu de « I ». Un autre, lors de la sixième séance, explique que « sur Minecraft, il faudrait faire F3 pour obtenir la température ». Il constate que la commande F3 ne donne aucun résultat visible et que seule la commande */status* permet d'accéder aux informations recherchées.

Nous constatons donc l'existence d'un double détournement. En effet, l'enseignant a détourné le jeu pour des objectifs disciplinaires, ce qui l'a conduit à simplifier le jeu lui-même et à modifier les touches de contrôle du logiciel. De leur côté, les élèves ayant une expérience de Minecraft ont cherché à retrouver le jeu qu'ils connaissaient en modifiant les réglages opérés par leur professeur.

6 Les compétences en jeu

6.1 Des compétences scientifiques

L'usage scolaire de Minetest permet à l'enseignant de faire travailler ses élèves de façon transversale sur les cinq thèmes au programme de SVT en 6e : « caractéristiques de l'environnement proche et répartition des êtres vivants », « peuplement d'un milieu », « origine de la matière des êtres vivants », « des pratiques au service de l'alimentation humaine », « diversité, parenté et unité des êtres vivants » (MEN, 2008).

Durant les premières séances, les élèves construisent, à la demande de leur enseignant, un enclos en superposant des blocs. Ils y insèrent des poules et un coq qu'ils font se reproduire avant d'ajouter un loup. Ils disent comprendre ainsi les facteurs à l'origine des variations de l'effectif d'une population (reproduction et prédation).

Par ailleurs, les élèves doivent durant la sixième séance « explorer une nouvelle planète ». L'enseignant leur a donné pour objectif d'observer et de recenser les caractéristiques biotiques (êtres vivants) et abiotiques (température, taux d'humidité, vitesse du vent, en utilisant la commande */ status*) de la planète. Ils développent ainsi des capacités (observer, recenser, organiser l'information) attendues dans les programmes d'enseignement.

6.2 Des compétences techniques en informatique

Au début des activités, l'enseignant a laissé du « temps libre » aux élèves pour découvrir par eux-mêmes l'univers et le fonctionnement des touches de contrôle qui sont nécessaires pour agir dans le monde virtuel. Si un vocabulaire informatique (« serveur », « navigateur », « barre d'adresse ») est employé durant les premières séances par l'enseignant, l'accent est mis sur les manipulations nécessaires à effectuer pour pouvoir accéder au monde virtuel généré par l'enseignant (sélectionner « client », saisir l'adresse du serveur de l'enseignant dans la barre de recherche nommée « Adresse / port », indiquer son pseudonyme et son mot de passe, cliquer sur « rejoindre ») plutôt que sur sa compréhension. Aucun concept informatique n'est défini.

Les apprenants sont donc plutôt amenés à développer, de façon indirecte, des compétences techniques de base en informatique soit par tâtonnement, soit en répétant lors de chaque séance une série d'actions identiques : se repérer dans le poste de travail pour ouvrir le logiciel, se connecter au serveur Minetest de l'enseignant en sélectionnant le nom du réseau indiqué parmi ceux proposés puis en saisissant l'adresse IP du serveur, s'identifier, utiliser les touches du clavier dans le monde virtuel pour se déplacer, construire ou détruire.

En outre, l'enseignant a choisi, pour des raisons de temps (une heure de cours dure cinquante-cinq minutes), de faire travailler les élèves en utilisant le mode créatif du jeu. Cela leur permet d'avoir accès à toutes les ressources du jeu de façon illimitée.

Bien que les tables de craft donnant la possibilité aux joueurs de combiner des ressources pour construire de nouveaux objets soient présentes, seuls deux élèves jouant à Minecraft en dehors de la classe emploient le terme « craft » durant les entretiens. Ils le mobilisent et le définissent par l'action de « faire la forme de l'outil qu'on veut crafter » en combinant plusieurs ressources accessibles dans l'inventaire. Un craft est le résultat de cette action. Cela n'est pas sans nous rappeler le concept d'algorithme que certains chercheurs définissent « presque comme une recette de cuisine » dans laquelle il est nécessaire de combiner plusieurs ingrédients (des blocs) pour obtenir un résultat (un craft). Le mésusage des tables de craft en classe pourrait s'expliquer par le fait qu'il ne paraît pas essentiel de créer des objets dans un monde créatif dans lequel il est possible en ouvrant son inventaire de sélectionner une hache pour l'utiliser.

7 Discussion

Employer les technologies de façon quotidienne apparaît pour les enseignants observés comme un moyen d'une part, de capter l'attention de leurs élèves en suscitant leur intérêt ; d'autre part, « de favoriser le climat de la classe ». Nos observations sur Minetest rejoignent celles faites sur Audacity dans plusieurs disciplines scolaires (Combemorel-Pauty, 2016). Elles indiquent que bien que des technologies sont employées comme moyen pour comprendre des notions scientifiques ou acquérir des compétences

prescrites par les programmes d'enseignement, les apprenants développent des habiletés manipulatoires soit par la répétition d'actions, soit par tâtonnement. Il est en effet nécessaire pour faire les activités prescrites par les enseignants de mobiliser certaines fonctionnalités. C'est pourquoi les professeurs disent « prendre du temps » en début de séquence pour expliciter, pas à pas, oralement ou par écrit, le fonctionnement des logiciels employés.

Minetest génère des rétroactions qui sont basées « sur un ensemble de règles sous-jacentes, modèles et opérations comme réponses aux manipulations de l'utilisateur des objets et des paramètres constituants cet environnement » (Djelil, 2016, p. 79). Les environnements créés par les élèves continuent à vivre, y compris en leur l'absence, ce qui peut conduire à ce que les apprenants nomment « des bogues », c'est-à-dire des dysfonctionnements qu'ils pensent relever d'un problème informatique.

L'enseignant donne un sens en lien avec les notions disciplinaires aux rétroactions du jeu. Pour reproduire le toit du collège, les élèves auraient, selon lui, choisi des matériaux en grès pour être fidèles à la couleur du toit réel. Toutefois, dans la mesure où le jeu prend en compte les phénomènes d'érosion, les blocs de grès utilisés ont subi une altération rapide en raison des conditions climatiques ainsi que de la durée des journées qui alternent toutes les quinze minutes entre le jour et la nuit. Cela a conduit à l'effondrement du toit. Une élève retient à ce propos que : « sur Minetest, quand il pleut et qu'on a mis un toit de sable alors ça tombe ».

Les types de matériaux choisis pourraient donc être considérés comme des variables qui conditionneraient les rétroactions du jeu. Une phase d'échanges a donc été nécessaire pour l'enseignant pour permettre à ses élèves de comprendre que les matériaux choisis pouvaient donner lieu à des rétroactions informatiques reflétant des phénomènes naturels.

8 Conclusion et perspectives

Pour conclure, il nous semble que l'usage scolaire de logiciels de type bac à sable tels que Minetest et Minecraft sont des entrées possibles pour aborder les trois approches, souvent hybridées et complémentaires, du domaine de l'informatique (Bruillard, 2008). Il est possible dans ces jeux dans des temps très courts de (1) lancer des traitements dans le cycle classique don-

nées/traitement/résultat, (2) d'interagir avec la machine, (3) d'être connecté à des réseaux informatiques *via* notamment les serveurs employés ou à des réseaux sociaux par l'intermédiaire de l'univers virtuel lui-même et de son tchat.

Un approfondissement de notre recherche serait, il nous semble, nécessaire pour comprendre comment l'utilisation de ce type de logiciel façonne (ou non) la représentation des élèves des notions disciplinaires en jeux ainsi que de l'informatique.

Références

Alvarez, J., Djaouti, D., & Rampnoux, O. (2016). *Apprendre avec les serious games ?* Canopé éditions. DL 2016, cop. 2016.

Bruillard, É. (2008). Acteurs et territoires de l'éducation à l'information. Un point de vue « informatique ». Consulté à l'adresse <http://ertecolloque.files.wordpress.com/2009/01/eb-tr-education_culture_info_6.doc>.

Combemorel-Pauty, C. (2016). Transmettre une culture numérique à des élèves équipés en ordinateurs portables par un département : le cas de l'utilisation du logiciel Audacity. In *Quelles éducations au numérique, en classe et pour la vie ?* Namur. Consulté à l'adresse <http://didapro6.sciencesconf.org/87558/document>.

D'Afflon, A. (2013). Sur l'échelle de la ludicité. Création et gamification, The gamefulness scale. Creation and gamification. *Hermès, La Revue*, *n° 62*(1), 41–47. Consulté à l'adresse <http://www.cairn.info.frodon.univ-paris5.fr/revue-hermes-la-revue-2012-1-p-41>.

Djelil, F. (2016, décembre 14). *Conception et évaluation d'un micromonde de Programmation Orientée-Objet fondé sur un jeu de construction et d'animation 3D*. Université Blaise Pascal – Clermont-Ferrand II. Consulté à l'adresse <https://tel.archives-ouvertes.fr/tel-01511201/document>.

Libert, C., & Vanhoof, W. (2016). Micro-mondes et concurrence par passage de messages : vers une nouvelle façon d'aborder la programmation ? In *Didapro 6*. Consulté à l'adresse <https://didapro6.sciencesconf.org/83092/document>.

MEN. (2008). Programmes de l'enseignement des SVT au collège. Consulté à l'adresse <http://media.education.gouv.fr/file/special_6/52/9/Programme_SVT_33529.pdf>.

Papert, S. (1981). *Jaillissement de l'esprit : ordinateurs et apprentissage*. [Paris] : Flammarion. impr. 1981, cop. 1981.

Quinche, F. (2015). Les jeux vidéo, lieux d'apprentissage et de collaboration. L'exemple de Minecraft. In S. la direction de L. Corroy, *Jeunes et Médias, Les Cahiers francophones de l'éducation aux médias – n° 7 – été 2015 : Les jeux vidéo : détournements, usages créatifs et enjeux pédagogiques*. Éditions Publibook.

Roberge, M. (2016). De Minecraft à « mindcraft » : comment effectuer le pont entre concepts quotidiens et concepts scientifiques dans des situations d'apprentissage utilisant un jeu vidéo. <http://hdl.handle.net/11143/8557>.

Voulgre, E. (2013). Le bac à sable, un espace pour jouer et apprendre des notions de programmation. Consulté 15 novembre 2017, à l'adresse <http://www.adjectif.net/spip/spip.php?article237>.

Béatrice Drot-Delange[a] & Françoise Tort[b]

a. Université Clermont Auvergne, ACTé
beatrice.drot-delange@uca.fr
b. ENS Paris-Saclay, STEF
francoise.tort@ens-paris-saclay.fr

Concours Castor, ressource pédagogique pour l'enseignement de l'informatique ? Étude exploratoire auprès d'enseignants[1]

Résumé

La question que nous posons dans cette communication est celle de savoir dans quelle mesure les défis du concours Castor peuvent constituer une ressource pour les professeurs dans le cadre de l'enseignement de l'informatique. Pour apporter des éléments de réponse exploratoires à ce questionnement, nous croisons l'analyse faite par des enseignants d'informatique et sciences du numérique (ISN) de défis du concours Castor avec les commentaires des concepteurs concernant les savoirs informatiques mobilisés par ces défis.

Les résultats obtenus montrent que les seuls énoncés des défis ne permettent pas toujours aux enseignants, malgré une expérience importante dans l'enseignement de l'informatique, de cerner quels sont les champs de l'informatique mobilisés ou potentiellement mobilisables dans les défis. Plus précisément, nous identifions des défis « ambitieux », « opaques » ou « ambigus » selon l'écart entre les analyses des enseignants et les intentions des concepteurs. Ceci pose question d'une part sur la mise en valeur du potentiel pédagogique de chaque défi et d'autre part sur la rédaction des solutions par les concepteurs et leur valorisation. L'article se conclut par des pistes de recherche.

Mots clés : ressources pédagogiques, enseignement de l'informatique, discipline scolaire, domaines de l'informatique

1 Cette recherche bénéficie d'un financement dans le cadre du projet ANR ReVEA.

1 Introduction

En France, l'informatique est désormais présente dans les programmes de l'enseignement primaire et du collège depuis la rentrée 2016 et au lycée pour certaines filières depuis 2012. Cette introduction s'est faite dans un contexte où peu de ressources existent, où les enseignants sont peu formés à l'informatique et à son enseignement, où les accompagnateurs des enseignants du premier degré sont sensibles à l'importance de l'enseignement de l'informatique sous réserve qu'ils aient eu dans le passé une expérience de cet enseignement (Villemonteix, 2017).

Parallèlement, depuis 2011, le concours Castor[2] informatique France connaît un succès qui ne se dément pas, d'année en année. Les défis du concours mettent en jeu des concepts, méthodes et principes propres à la science informatique, tout en étant accessibles à des élèves qui n'ont pas de prérequis dans ce domaine (Tort & Dagiene, 2012). Le concours se déroule sur le temps scolaire et à l'initiative des enseignants qui inscrivent leurs classes. Toutefois les défis ne sont pas conçus pour mesurer des savoirs ou des compétences acquis et ne correspondent à aucune forme canonique scolaire. La manière de résoudre le problème n'est pas explicitée et doit être découverte et construite par le candidat (Tort, Drot-Delange & Mano, 2017). Les défis se présentent sous la forme de courts énoncés de problèmes à résoudre, comprenant souvent des images ou des diagrammes, et des artefacts interactifs plus ou moins complexes (Tort & Drot-Delange, 2015).

La question que nous posons dans cette communication est celle de savoir dans quelle mesure les défis du concours Castor peuvent constituer une ressource pédagogique pour les professeurs dans le cadre de leur enseignement de l'informatique.

2 Problématique

Les connaissances professionnelles des enseignants relèvent selon Shulman (2007) de trois types : des connaissances disciplinaires, des connaissances

2 <http://castor-informatique.fr>

pédagogiques et des connaissances curriculaires des contenus à enseigner. Dans cet article, nous nous intéresserons plus particulièrement à la question des connaissances disciplinaires. Shulman (2007) considère que ce type de connaissances renvoie « à la quantité et à l'organisation de la connaissance *per se* dans l'esprit du professeur » (p. 104). Une des difficultés de l'informatique est qu'il n'existe pas d'ontologie de cette science (Bruillard, 2016), malgré les nombreuses propositions de classification qui sont proposées régulièrement (Denning & Martell, 2015). De plus, l'informatique n'est présente dans l'enseignement obligatoire secondaire que depuis 2012, sous forme d'enseignements optionnels au lycée, et comme chapitre aux programmes d'autres disciplines (mathématiques et technologie) au collège. Les enseignants prenant en charge ces enseignements n'ont pas (ou rarement) de formation initiale dans ce domaine (Baron *et al.*, 2015). On comprend dès lors que l'élaboration de connaissances disciplinaires en informatique telles que les définit Shulman, est une tâche complexe pour les enseignants d'informatique.

Les concepteurs des défis du Concours Castor sont des acteurs ayant une expertise forte dans le domaine de la science informatique. En France, il s'agit de membres de l'association France-IOI[3] et de l'INRIA (Institut national de recherche en informatique et en automatique). Les organisateurs cherchent à couvrir l'ensemble des champs de la science informatique et, désormais, de la pensée computationnelle (Dagiene, Sentance & Stupuriene, 2017) et élaborent des classifications permettant aux concepteurs de situer chaque défi dans un domaine de l'informatique.

Dagiene, Sentance et Stupuriene proposent la classification suivante. Elle est construite selon deux dimensions, la première est celle des connaissances, la seconde celle des habiletés (*skill*). Les connaissances relèvent de 5 domaines : « algorithmes et programmation », « données, structures de données et représentations », « processus informatiques et matériels », « communication et réseau », « interactions, systèmes et société ». Ces domaines sont précisés par des mots clés permettant de mieux cerner leur contenu. Les habiletés sont construites sur les compétences considérées comme relevant de la pensée informatique, et identifiées notamment par Selby et Woollard (2013), à savoir l'abstraction, la pensée algorithmique, la décomposition, l'évaluation et la généralisation.

Une organisation possible des connaissances en informatique est donc proposée par les initiateurs du concours eux-mêmes. Nous posons la

3 Fondée en juin 2004 dans le but de développer la sélection et l'entraînement de l'équipe de France aux Olympiades Internationales d'Informatique. <http://www.france-ioi.org>

question de savoir si cette organisation des connaissances est perceptible par des enseignants qui ont d'une part acquis une certaine familiarité avec le concours, puisqu'ils inscrivent régulièrement leurs élèves, et d'autre part qui enseignent l'informatique en lycée.

Nous faisons l'hypothèse que les défis du concours Castor constituent des ressources pour les enseignants, à condition que ces défis mettent en œuvre des savoirs informatiques connus et reconnus par les enseignants. Toutefois, si l'identification des connaissances et compétences couvertes par une ressource, ici un exercice, nous semble une condition nécessaire à ce qu'un enseignant décide de l'exploiter dans son enseignement, elle n'est certainement pas suffisante. L'adéquation de l'exercice pour traiter du concept identifié est aussi un facteur de choix. Sur ce point, les défis du castor introduisent une forme que l'on peut qualifier d'innovante. Les études menées n'ont pas encore véritablement évalué leur pertinence sur la compréhension, et l'apprentissage par les élèves. Il s'agit donc de se demander dans quelle mesure les enseignants repèrent les notions et savoirs du domaine informatique présents dans ces défis et si cette perception correspond aux intentions des concepteurs. Autrement dit, les intentions des concepteurs en termes de savoirs informatiques mis en jeu dans les défis sont-elles suffisamment lisibles dans les énoncés du concours Castor pour être exploitées par les enseignants ?

Pour apporter des éléments de réponse exploratoires à notre questionnement, nous proposons de croiser l'analyse faite par des enseignants des défis du concours Castor avec celle des concepteurs, du point de vue des connaissances disciplinaires. Nous nous intéressons à la perception par les enseignants du contenu informatique des défis, que nous confrontons aux intentions des concepteurs.

3 Méthodologie

Nous avons demandé à des enseignants de faire une analyse d'un corpus de 19 défis retenus par nous. Nous présentons d'abord la manière dont les données ont été collectées puis la méthode d'analyse mise en œuvre.

3.1 Présentation des corpus de données

Plusieurs corpus de données ont été constitués et étudiés. Le premier corpus comporte les énoncés d'une sélection de défis des éditions passées du concours (corpus 1). Le second corpus correspond aux analyses de ces défis effectuées et rédigées par les enseignants (corpus 2). Le troisième corpus est constitué des textes des solutions des défis rédigées par les concepteurs et publiées après le concours (corpus 3).

Tableau 1 : Liste des défis retenus pour l'expérimentation[4].

N°[5]	Année du concours	Pays concepteur	Niveau de classe	Titre du défi
1	2012	Russie	4e-3e	Affichage digital
2	2011	Slovaquie	4e-3e	Les amis de Lucie
3	2012	France	4e-3e	Anonymisation
4	2013	France	6e-5e	Arroser la plante
5	2012	Tchécoslovaquie	6e-5e	Bébrocarina
6	2013	France	6e-5e	Chercher-remplacer
7	2013	France	6e-5e	Choix des invités
8	2012	France	6e-5e	Course de grenouilles
9	2012	France	6e-5e	Chez le dentiste
10	2012	France	6e-5e	Dessine un castor
11	2013	Slovaquie	6e-5e	Dessiner un dé
12	2012	Tchécoslovaquie	6e-5e	Labyrinthe
13	2013	Suède	4e-3e	Réseau pas cher
14	2013	France	6e-5e	Robot peintre
15	2013	Slovénie	2e	Rumeurs
16	2012	France	2e	Territoire de Castor
17	2012	Lettonie	1re-Tale	Transport de troncs
18	2013	France	6e-5e	Trier par poids
19	2012	France	2e	Trombone rouge

Corpus 1 : énoncés de défis sélectionnés

Les défis retenus proviennent des concours des années 2011 à 2013 (*cf.* Tableau 1). Ils ont été choisis parmi ceux qui semblaient poser par leur

4 Les énoncés, solutions et explications des défis sont consultables en ligne : <http://castor-informatique.fr/questions/?menu=1>.

5 Numérotation introduite pour la présente recherche, afin de faire plus facilement référence aux défis dans la suite du texte.

contenu et leur forme des questions de translittératie, notamment du fait des interfaces et images utilisées. En effet, cette recherche a été initiée dans le projet ANR Translit, qui avait pour objet la translittératie, et se poursuit au sein du projet ANR REVEA concernant les ressources vivantes en éducation et en formation. Notons que la sélection effectuée reste représentative des défis proposés ces années-là.

Corpus 2 : analyses des défis par les enseignants

Nous avons demandé à des enseignants impliqués dans ce projet de recherche (*cf.* Tableau 2) de procéder à une analyse des défis retenus. Ces professeurs ont pour point commun d'enseigner l'informatique dans le cadre de l'enseignement de spécialité ISN (Informatique et Sciences du Numérique) en classe de terminale, mais aussi comme enseignement optionnel en classe de seconde ou de première (Informatique et Création Numérique).

Leur analyse a été recueillie au moyen d'un document fournissant une grille qu'ils devaient compléter. La grille comportait le nom du défi, l'année du concours et le niveau de classe minimal auquel le défi a été proposé. Les enseignants ont complété quatre rubriques : (1) lister les notions ou les concepts informatiques mobilisés par le défi (ils pouvaient préciser s'il n'y avait pas de notions, selon eux, ou alors s'il s'agissait plutôt de notions mathématiques) ; (2) décrire la « meilleure » stratégie de résolution possible, et mentionner s'il existait selon eux d'autres stratégies de résolution ; (3) préciser si les élèves étaient susceptibles d'avoir déjà rencontré ces notions dans leur scolarité et, si oui, dans quelle matière ou dans quel cours ; (4) préciser l'intérêt de ces défis du point de vue de l'apprentissage de l'informatique, et la difficulté estimée selon eux.

Tableau 2 : Enseignants participant à la recherche.

Enseignant[6]	**Discipline**	**Ancienneté dans l'enseignement**	**Enseigne ISN depuis**	**Statut**
Christophe	Mathématiques[7]	33 ans	2012	Agrégé
François	Mathématiques	18 ans	2012	Agrégé
Isabelle	Mathématiques	11 ans	2012	Agrégée

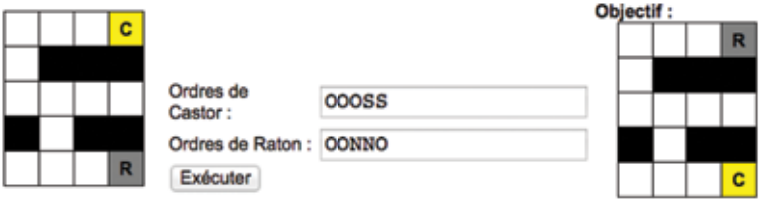
6 Les prénoms ont été modifiés, le genre conservé.

7 En France, les enseignants en charge des enseignements d'informatique appartiennent à une autre discipline, majoritairement les mathématiques.

Corpus 3 : textes des solutions des défis

Les auteurs rédigent des solutions aux défis (*cf.* Figure 1). Ces solutions sont publiées dès que la session du concours est terminée et restent accessibles en permanence. Outre la description de la solution elle-même, elle comporte une section intitulée « C'est de l'informatique ! » qui explique en quoi le problème posé, la situation décrite ou la solution apportée relèvent de la science informatique. La longueur de ces textes est très variable d'un défi à l'autre.

Course de grenouille
Les grenouilles programmables de Castor et Raton sont placées dans un labyrinthe. Le but est de programmer les grenouilles pour qu'elles échangent leurs positions en sautant le même nombre de fois et sans entrer en collision. Les grenouilles peuvent être programmées par une séquence de lettres dirigeant leurs sauts : **N** pour Nord, **S** pour Sud, **O** pour Ouest, et **E** pour Est.



Trouvez deux séries de commandes de même longueur (une série pour chaque grenouille) qui permettent aux grenouilles d'échanger leur position sans se rentrer dedans. N'hésitez pas à faire des essais, vous ne pouvez pas perdre de points.

Solution

Voici un exemple de séquences d'ordres que l'on peut donner à la grenouille de Castor et à celle de Raton, respectivement :

OOOSSESSEEOEOE
OONNEEOOONNEEE

C'est de l'informatique !

Chacune des grenouilles se comporte comme un petit robot que l'on peut **programmer**. Ici, un programme est simplement une séquence de directions, chaque direction étant l'une des lettres N, S, E ou O.

Ce qui est intéressant dans ce sujet, c'est que le temps entre en compte. En effet, il y a deux grenouilles, donc il faut **synchroniser** leurs actions pour éviter une collision. De manière générale, les programmes qui ont besoin de prendre en compte des interactions avec l'environnement, et donc qui dépendent du temps sont souvent bien plus difficiles à écrire que les autres.

Figure 1 : Le défi « Course de grenouille » et sa solution.

3.2 Méthode d'analyse des corpus

Les thèmes abordés dans les corpus 2 et 3 ont fait l'objet d'une analyse thématique de contenu à l'aide du logiciel NVivo 10. Les thèmes intéressant notre étude sont les notions et les savoirs informatiques cités par les enseignants (corpus 2) et les concepteurs des défis (corpus 3).

Nous avons procédé d'abord de manière ascendante en respectant le plus possible les expressions et termes retenus par les enseignants et concepteurs. L'unité de codage est le groupe de mots. Un premier regroupement a ainsi été effectué et a fait émerger les catégories présentées dans le tableau 3.

Tableau 3 : Premières catégories identifiées dans les discours.

Notions et savoirs	**Nombre de références encodées**
Algorithme et programmation	45
Codage de l'information	6
Graphes et algorithmes	25
Intelligence artificielle	1
Système, contrainte, plan	17
Tableau, Base de données	16
Utilisation d'outil	1

Tableau 4 : Correspondances entre nos catégories initiales et la classification de Dagiene et *al.* (2017).

Catégories initiales	**Classification de Dagiene et *al.* (2017)**
Algorithme et programmation	Algorithme et programmation
Codage de l'information	Communication et réseaux Données, structures et représentations
Graphes	Données, structures et représentations
Intelligence artificielle	Interactions, systèmes et société
Système, contrainte, plan	Algorithmes et programmation Processus informatiques et matériels
Tableau, Base de données	Données, structures et représentations
Utilisation d'outil	Processus informatiques et matériels

Cette classification a ensuite été reprise et comparée à celle proposée par Dagiene et *al*. (2017). Il s'est avéré que la catégorisation proposée était

compatible avec celle de ces auteurs (*cf.* Tableau 4). Les différences entre les deux peuvent s'expliquer par le fait que nos propres catégories ont été élaborées à partir des discours des enseignants et des concepteurs analysant et expliquant des cas d'études très précis. Autrement dit, les connaissances repérées et mobilisées par eux sont contextualisées et la formulation est liée à cette contextualisation. Ce n'est pas le cas de la classification proposée par Dagiene et *al*. qui a pour objet de couvrir les champs de l'informatique de manière exhaustive.

4 Résultats

Pour mettre en évidence les domaines informatiques repérés par les enseignants dans les défis sélectionnés, nous comparons le codage du corpus 2 (analyse des défis par les enseignants) à celui du corpus 3 (rubrique « C'est de l'informatique »). Le codage a été effectué selon les catégories de contenus informatiques proposées par Dagiene et *al*. (*cf.* Tableau 4).

Les résultats obtenus sont présentés ci-après. La section 4.1. donne une vue globale de la perception de l'ensemble de la sélection des défis par les enseignants et selon les intentions des concepteurs. La section 4.2 présente le résultat d'une analyse fine par enseignant et par défi et aboutit à une typologie des défis. Enfin, la section 4.3 compare les mots utilisés par les enseignants et ceux utilisés par les concepteurs.

4.1 Vue globale des domaines repérés par les enseignants et visés par les concepteurs

La figure 2 montre une vue globale de la perception de l'ensemble des défis par les trois enseignants et les intentions des concepteurs en matière de notions informatiques présentes dans les défis. Un défi n'est comptabilisé qu'une seule fois dans un même domaine, même si le codage du discours de plusieurs professeurs le rattache à ce domaine. Un défi peut appartenir à plusieurs domaines.

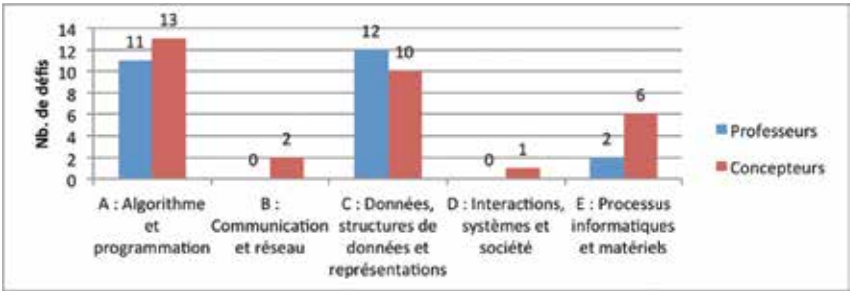


Figure 2 : Domaine des notions citées dans les corpus 2 et 3 pour les 19 défis étudiés.

Globalement, les défis proposés dans la sélection relèvent principalement des domaines « Algorithmes et programmation » (domaine A) et « Données, structures de données et représentations » (domaine C). Ce sont, en effet, les domaines les plus couverts par les défis du concours, et ce même dans les autres pays (Vanicek, 2014). L'étude des résultats du codage tant de l'analyse produite par les enseignants que de la rubrique « C'est de l'informatique » des concepteurs montre une attribution aux domaines de l'informatique relativement similaires.

Cependant, deux domaines sont absents de l'analyse faite par les enseignants : « Communication et réseau » (domaine B) et « Interactions, systèmes et société » (domaine D) alors qu'ils sont présents chez les concepteurs. Enfin, un domaine est sous-représenté chez les enseignants, « Interactions, systèmes et sociétés » (domaine E), par rapport aux intentions des concepteurs.

Finalement, les concepteurs tendent à citer davantage de notions informatiques que les enseignants, rattachant ainsi les défis à davantage de domaines informatiques. Ceci peut s'expliquer par le fait que les enseignants interrogés se sont focalisés sur les notions et savoirs directement mobilisés dans le défi, alors que les concepteurs explorent les applications et références possibles bien au-delà du contexte du défi.

4.2 Écarts entre perceptions des enseignants et intentions des concepteurs

Observons maintenant défi par défi l'analyse faite par les enseignants par rapport aux concepteurs. On constate des différences dans l'appréciation des notions informatiques de chaque défi (*cf.* Tableau 5).

Les résultats montrent plusieurs cas de figure, selon les accords/désaccords entre enseignants et avec les concepteurs. Nous en proposons la typologie suivante. Les libellés proposés pour chaque catégorie qualifient la capacité du défi à « se laisser interpréter » en termes de notions informatiques.

La première catégorie est celle où les domaines mentionnés par les enseignants et les concepteurs sont identiques. C'est le cas de 5 défis (3, 12, 13, 14 et 18). Nous qualifions ces défis de « transparents », car les enseignants accordent les mêmes domaines de l'informatique à ces défis que les concepteurs.

La deuxième est celle où tous les enseignants sont d'accord entre eux, mais où les concepteurs citent davantage de domaines que les enseignants. C'est le cas de 5 défis (4, 7, 9, 11 et 17). Nous qualifions ces défis d'« ambitieux », car les concepteurs y voient davantage de potentialité que les enseignants.

La troisième est celle où les domaines mentionnés par les enseignants peuvent différer entre eux et/ou avec ceux des concepteurs. C'est le cas de 4 défis (5, 8, 15 et 16). Pour le défi 8, tous les enseignants mentionnent le domaine A, ainsi que les concepteurs, mais l'un des enseignants cite 2 autres domaines, dont un seul est présent chez les concepteurs. Nous qualifions ces défis d'« opaques », car ne permettant pas aux enseignants de clairement cerner les domaines informatiques en jeu.

Enfin, 5 défis ne sont pas classés dans des domaines de l'informatique par au moins un des enseignants (1, 2, 6, 10 et 19). Nous considérons alors que les intentions des concepteurs sont « ambigües » pour les enseignants. Nous reprenons plus en détail les propositions faites par les enseignants en nous demandant quelles sont alors les notions mobilisées par le défi, puisqu'au moins un des enseignants considère qu'il ne s'agit pas d'informatique.

Isabelle, pour le défi 1 (Affichage digital), précise qu'elle « ne voit pas » quelles sont les notions mobilisées et qu'elle considère davantage ce défi comme un jeu. Pourtant, Christophe indique comme notion la « négation, le OU inclusif ». François mentionne « l'enchainement d'instructions ».

Tableau 5 : Domaine(s) relevé(s) dans les commentaires des enseignants (corpus 2) et des concepteurs (corpus 3) pour chacun des défis (corpus 1).

	Christophe	**François**	**Isabelle**	**Concepteurs**	**Catégorie**
1 : Affichage digital	A	A	**	ACE	Ambigu
2 : Les amis de Lucie	AC	**	C	C	Ambigu
3 : Anonymisation	C	C	C	C	Transparent
4 : Arroser laPlante	A	A	A	ADE	Ambitieux
5 : Bébrocarina	C	C	A	AB	Opaque
6 : Chercher_Remplacer	C	**	AC	A	Ambigu
7 : Choix des Invités	C	C	C	AC	Ambitieux
8 : Course de Grenouille	ACE	A	A	AE	Opaque
9 : Chez le dentiste	C	C	C	AC	Ambitieux
10 : Dessine Castor	**	A	E	ABCE	Ambigu
11 : Dessiner un dé	A	*	A	AE	Ambitieux
12 : Labyrinthe	A	A	A	A	Transparent
13 : Réseau pas cher	*	*	C	C	Transparent
14 : Robot Peintre	A	A	A	A	Transparent
15 : Rumeurs	C	*	*	E	Opaque
16 : Territoire de Castor	C	A	A	C	Opaque
17 : Transport_Troncs	C	C	C	AC	Ambitieux
18 : Trier par Poids	A	A	A	A	Transparent
19 : Trombone rouge	C	A	**	C	Ambigu

Légende
* les données ne sont pas disponibles, l'enseignant n'ayant pas rempli la grille pour ce défi.
** l'enseignant explique qu'il ne s'agit pas d'informatique.
A : Algorithmique et programmation, B : Communication et réseau,
C : Données, structures de données et représentation, D : Interactions, systèmes et société,
E : Processus informatiques et matériels.

François précise pour le défi 2 (Les amis de Lucie) qu'il s'agit selon lui « essentiellement d'un travail sur la consigne », que c'est un exercice « qui ne teste pas grand-chose ». Dans le même temps, Isabelle mentionne la notion de voisin et les graphes. Christophe évoque également les graphes mais aussi l'organisation des données, le test et le parcours d'un tableau.

Pour le défi 6 (Chercher Remplacer), il s'agit selon François d'un raisonnement logique et il n'indique pas de connaissances informatiques à mobiliser. Christian évoque le tri de l'information et l'utilisation d'un outil, tandis qu'Isabelle mentionne les chaines de caractères et les correspondances de chaines.

Christophe indique pour le défi 10 (Dessine un castor) qu'il ne sait pas quelles sont les notions mobilisées et qu'il ne sait pas définir les stratégies à mettre en œuvre pour le résoudre. Isabelle pour ce même défi évoque un « phénomène de rétroaction, la réponse d'un système ». Elle précise aussi qu'« en informatique, je ne vois pas trop ». François considère qu'il s'agit de mettre en place une démarche algorithmique.

Le défi 19 (Trombone rouge) relève, selon Isabelle, de la logique et elle ne voit pas quelles sont les notions informatiques mobilisées. Il est à noter que Christophe et François ne lui attribuent pas le même domaine de connaissances. Selon Christophe, le défi s'inscrit plutôt dans le domaine « Données, structures de données et représentation ». Il précise que les notions sont celles de l'organisation des données, de la lecture de graphe et du tri de l'information. Selon François, le défi s'inscrit dans le domaine « Algorithmique et programmation ». Il s'agit en effet de « mettre en place une démarche algorithmique » pour résoudre ce défi.

Nous avons pu constater les différences de représentation entre enseignants, mais aussi entre enseignants et concepteurs sur les contenus informatiques mobilisés ou mobilisables dans ces défis. Nous nous intéressons maintenant aux termes employés par les uns et les autres dans leurs discours.

4.3 Comparaison des termes employés par les enseignants et les concepteurs

Comparons les mots employés par les concepteurs dans la rubrique « C'est de l'informatique » (corpus 3) à ceux utilisés par les enseignants dans leurs commentaires (corpus 2).

Tout d'abord, nous nous intéressons aux mots du domaine informatique les plus fréquemment cités par les concepteurs (le premier quartile). La figure 3 les mentionne. Nous recherchons ensuite si ces mots sont présents dans le discours des enseignants.

On constate que certains mots employés par les concepteurs ne sont pas présents dans les écrits des enseignants. C'est le cas par exemple du mot « langage ». D'autres mots sont proportionnellement bien plus présents chez les concepteurs. C'est le cas du mot « programme » qui est cité trois fois plus souvent par les concepteurs que par les enseignants. On peut avancer l'hypothèse que la notion de programme et celle de langage sont associées dans l'esprit des enseignants à des langages de programmation usités et connus (Scratch, Python, Java, C, etc.). Or ces langages sont absents des défis du Castor. Lorsque les concepteurs utilisent le mot langage, c'est à propos d'instructions proposées par eux (par exemple les lettres N S E O pour désigner des ordres de déplacements directionnels).

Procédons de manière identique pour les enseignants (*cf.* Figure 4). Les mots « jeu » et « logique » apparaissent dans les discours des enseignants, alors qu'ils ne sont pas présents chez les concepteurs. Sans que le terme de « jeux » soit une appréciation péjorative du défi en tant que tel, cela pourrait être la reconnaissance d'une forme de ressource, certes non canonique, qui peut susciter l'intérêt des élèves. Le mot « logique » est employé par les enseignants pour souligner que ce ne sont pas d'abord des notions informatiques qui sont mises en œuvre mais une capacité à faire preuve de logique. On constate d'ailleurs que ce terme n'est pas employé par les concepteurs.

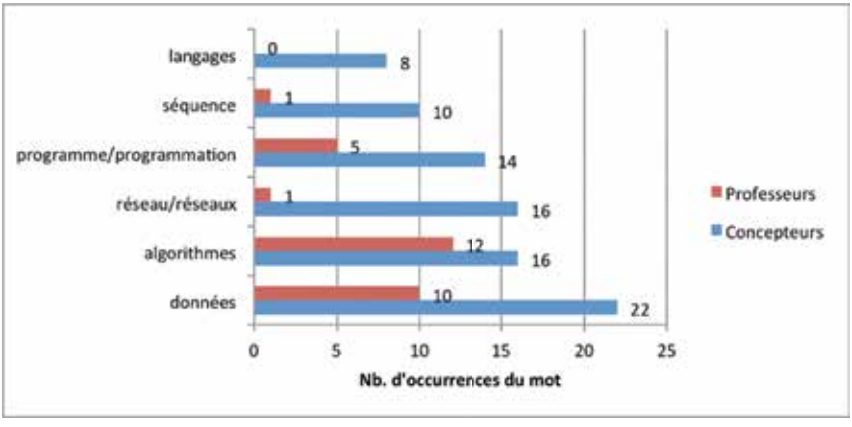


Figure 3 : Les mots du domaine informatique les plus fréquemment mentionnés (1er quartile) par les concepteurs et leur présence dans les commentaires des enseignants.

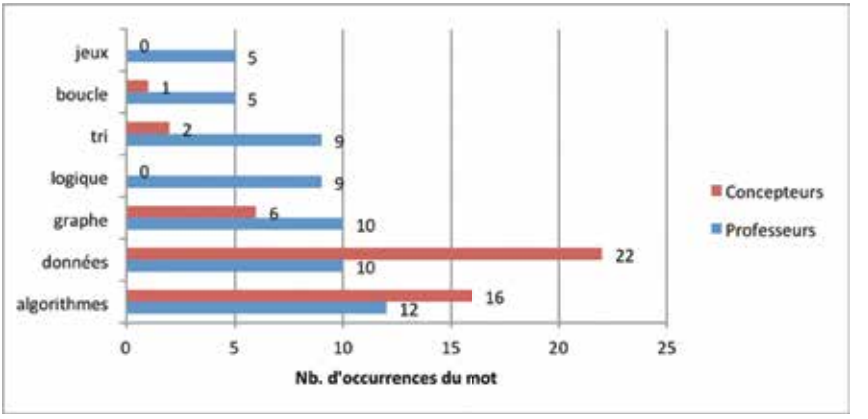


Figure 4 : Les mots du domaine informatique les plus fréquemment mentionnés (1er quartile) par les enseignants et leur présence dans la rubrique « C'est de l'informatique ».

5 Discussion

Les professeurs impliqués dans cette étude enseignent l'informatique en lycée depuis plusieurs années, ils ne sont donc pas représentatifs de l'ensemble des enseignants qui sont amenés, ou qui vont l'être, à enseigner l'informatique. Nous considérons leur expertise comme un atout dans cette recherche, notamment dans leur compréhension fine des défis. L'écart entre leurs connaissances disciplinaires et celles des concepteurs est sans aucun doute moins grand que celui qu'auraient des enseignants dépourvus de cette expérience. Comparer leurs discours à celui des concepteurs a donc un sens.

Considérer les défis du concours en tant que ressources mobilisables dans un enseignement de l'informatique nous a fait nous interroger sur la structure des connaissances sous-jacentes selon les enseignants et selon les concepteurs. Dans le cas de l'informatique, cette structure ne va pas nécessairement de soi et est encore en débat dans de nombreux pays lors de l'élaboration des curricula.

Les résultats obtenus montrent que les enseignants ne perçoivent pas toujours, à partir de l'énoncé du défi, l'intérêt de celui-ci pour l'apprentissage des notions informatiques.

Certains domaines semblent toutefois plus facilement repérables que d'autres, c'est notamment le cas du domaine A (« Algorithme et programmation »), voire du domaine C (« Données, structures de données et représentations »). Outre le fait que, de manière générale, les défis mettent plus souvent ces domaines en jeu, on peut aussi supposer que les enseignants interrogés en ont une meilleure connaissance du fait de leur formation et des programmes d'ISN qu'ils sont amenés à enseigner.

On constate aussi des écarts dans les termes employés tant par les concepteurs que par les enseignants. Les premiers emploient plus souvent les termes de réseaux, de programme, de séquences, de langage alors que les seconds emploient davantage les termes de logique, de tri, de boucle, de jeu.

Nous avons pu constater que certains défis étaient « transparents » pour les enseignants, au sens où l'analyse de leurs écrits aboutissait à la même attribution de domaines informatiques que celle des concepteurs. A l'inverse, certains défis restent « ambigus », au sens où au moins un des enseignants ne perçoit pas l'intention des concepteurs en termes de notions informatiques. Les intentions des concepteurs en matière d'intérêt du défi vis-à-vis de l'informatique sont parfois ambitieuses, et ne sont pas toutes perceptibles par les enseignants.

On peut alors craindre que les élèves non plus ne perçoivent pas en quoi ces défis relèveraient de la science informatique, avec un risque de résultats contreproductifs par rapport aux objectifs du concours Castor, à savoir la promotion de l'informatique auprès des élèves. Ce risque a déjà été constaté par Taub, Armoni et Ben-Ari (2012) dans le cas de l'informatique sans ordinateur. Les auteurs étudient si les objectifs assignés à cet enseignement sont remplis auprès d'élèves de collèges (grade 7), à savoir s'ils ont une représentation plus claire de l'informatique et s'ils ont envie d'en continuer l'étude plus tard. Les résultats obtenus montrent que les élèves sont moins attirés par l'informatique et la considèrent comme moins intéressante que ce qu'ils déclaraient avant de faire les activités sans ordinateur. L'hypothèse faite par les auteurs est que les élèves ne comprennent pas de quelle manière les activités proposées sont représentatives de l'informatique étudiée dans l'enseignement secondaire ou d'une éventuelle future carrière. Les élèves ne feraient pas le lien entre les activités proposées et les concepts informatiques, faute d'explicitation. Pourtant, là aussi, les documents à destination des enseignants les précisent.

Comment rendre plus « transparentes » les intentions des concepteurs du concours Castor et faciliter l'exploitation pédagogique et disciplinaire des défis ? Une solution mise en œuvre par les organisateurs du concours en Grande-Bretagne consiste à utiliser explicitement la classification proposée dans Dagiene *et al.* (2017). Ainsi, dans le livret présentant l'ensemble des défis du concours et leurs solutions[8], il est fait mention pour chaque défi de deux catégories : la compétence relevant de la pensée informatique sollicitée et le domaine informatique couvert. Il s'agit en quelque sorte de fournir des métadonnées pour chaque défi, pour faciliter le travail de l'enseignant lorsque celui-ci souhaite s'approprier ces défis comme ressources pour son enseignement.

Cette question de la documentation de ressources dites « clés en main » en informatique n'est pas nouvelle (Drot-Delange, 2013). Elle se pose avec d'autant plus d'acuité que les enseignants n'ont pas toujours une connaissance des concepts. Cette question a déjà été soulevée par les initiateurs du mouvement *Computer Unplugged*. La première édition de leur manuel (Fellows, Bell & Witten, 1996) livrait des activités, sans instruction particulière pour les enseignants. Mais il s'est vite avéré qu'un minimum d'explications était nécessaire pour des enseignants qui n'avaient pas toujours la culture informatique et mathématique nécessaire à la prise en main et à l'exploitation de ces activités (Bell, Rosamond & Casey, 2012). Une version écrite par des enseignants pour les enseignants a donc été éditée (Fellows, Bell & Witten, 2002) comprenant des activités testées en classe. C'est la même logique qui prévaut dans Calmet, Hirtzig et Wilgenbus (2016) où les défis du concours Castor sont recensés par cycle et intégrés dans les progressions proposées par l'ouvrage.

6 Conclusion et perspectives

Nous souhaitions savoir dans quelle mesure les défis du concours Castor pouvaient constituer des ressources à destination des enseignants dans le cadre d'un enseignement de l'informatique. Pour cela, nous avons postulé

8 2016 *Competition complete answer book* : <http://www.bebras.uk/uploads/2/1/8/6/21861082/uk-bebras-2016-answers.pdf>. Consulté le 4 septembre 2017.

que la structure de connaissances informatiques sous-jacentes aux défis devait correspondre à celles des enseignants.

Nous avons montré que les énoncés des exercices seuls ne permettaient pas toujours aux enseignants, possédant une expérience importante dans l'enseignement de l'informatique, de cerner quels étaient les champs de l'informatique mobilisés ou potentiellement mobilisables dans les défis. Ceci en rend donc complexe l'utilisation dans une classe à des fins pédagogiques pour l'enseignant.

L'existence de défis « ambitieux », « opaques » ou « ambigus » pose question d'une part sur la mise en valeur du potentiel pédagogique de chaque défi et d'autre part sur la rédaction des solutions et des rubriques « C'est de l'informatique ». La documentation associée aux défis et la communication des métadonnées décrivant un défi permettraient probablement aux enseignants de mieux cerner les intentions des concepteurs. Le chantier de la construction d'une classification est en cours dans les équipes de recherche. Il conviendrait que celles-ci n'associent pas seulement des concepteurs mais aussi des enseignants pour s'assurer de l'utilité de cette classification.

Les perspectives de cette recherche sont de trois ordres. Nous avons proposé dans cette communication une typologie des défis basée sur la complétude des domaines de l'informatique repérés par les enseignants. On peut se demander si ces défis présentent structurellement des caractéristiques qui les distinguent les uns des autres, ou si ce sont les connaissances des enseignants qui font la différence. Ensuite, nous souhaitons focaliser le questionnement sur l'activité de l'élève lors de la résolution de défis. Il s'agirait alors d'identifier les stratégies de résolution mises en œuvre et l'analyse qu'en font les enseignants. Enfin, dans une démarche complémentaire à celle présentée ici, on évaluera la pertinence des défis proposés par rapport aux domaines qu'ils sont censés couvrir.

Références

Baron, G.-L., Drot-Delange, B., Grandbastien, M., & Tort, F. (2015). L'enseignement de l'informatique dans l'enseignement secondaire en France : un retour de balancier ? Dans G-L. Baron, E. Bruillard,

B. Drot-Delange (eds.). *Informatique en éducation : perspectives curriculaires et didactiques* (p. 83–101). Clermont-Ferrand : Presses Universitaires Blaise Pascal.

Bell, T., Rosamond, F. et Casey, N. (2012). Computer Science Unplugged and Related Projects in Math and Computer Science Popularization. Dans H. L. Bodlaender, R. Downey, F. V. Fomin et D. Marx (dir.), *The Multivariate Algorithmic Revolution and Beyond* (vol. 7370, pp. 398–456). Berlin, Heidelberg : Springer Berlin Heidelberg.

Bruillard, É. (2016). Quelle informatique à repenser et à construire pour les élèves de l'école primaire ? Dans F. Villemonteix, G.-L. Baron et J. Béziat (dir.), *L'école primaire et les technologies informatisées. Des enseignants face aux TICE.* (pp. 29–37). Villeneuve d'Asq : Presses Universitaires du Septentrion.

Calmet, C., Hirtzig, M. et Wilgenbus, D. (2016). *1, 2, 3… Codez !* Le Pommier.

Dagiene, V., Sentance, S. et Stupuriene, G. (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. *Informatica*, 28(1), 23–44.

Denning, P. J. et Martell, C. H. (2015). *Great Principles of Computing*. The MIT Press.

Drot-Delange, B. (2013). Enseigner l'informatique débranchée : analyse didactique d'activités (pp. 1–13). Présenté à Actualité de la Recherche en Education et en Formation, Montpellier : France.

Fellows, M. R., Bell, T. et Witten, I. (1996). *Computer Science Unplugged… offline activities and games for all ages : Original Activities Book*. Computer Science Unplugged.

Fellows, M. R., Bell, T. et Witten, I. (2002). *Computer science unplugged*.

Selby, C. & Woollard, J. (2013). Computational thinking : the developing definition, available via internet : <http://eprints.soton.ac.uk/356481>. Consulté le 30 avril 2016.

Shulman, L.-S. (2007). Ceux qui comprennent. *Éducation et didactique*, 1(1), 97–114.

Taub, R., Armoni, M. et Ben-Ari, M. (2012). CS unplugged and middle-school students' views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education* (TOCE), 12(2), 8.

Tort, F. et Dagiene, V. (2012). Concours Castor : découvrir l'informatique autrement/*E-dossier de l'audiovisuel : L'éducation aux cultures de l'information*/.

Tort, F. et Drot-Delange, B. (2015). Visual Literacy in Introductory Informatics Problems. Dans A. Brodnik et J. Vahrenhold (dir.), *Informatics in Schools. Curricula, Competences, and Competitions* (vol. 9378, pp. 175–182). Springer International Publishing.

Tort, F., Drot-Delange, B. et Mano, M. (2017). Filles et informatique : qu'en est-il du concours Castor ? Dans J. Henry, A. Nguyen et É. Vandeput (dir.), *L'informatique et le numérique dans la classe. Qui, quoi, comment ?* (pp. 69–84). Namur : Presses Universitaires de Namur.

Vanicek J., (2014) Bebras Informatics Contests ; Criteria for Good Tasks Revised. Dans Y. Gülbahar and E. Karatas (Eds) : *ISSEP 2014, LNCS* 8730 (pp. 17–28). Springer.

Villemonteix, F. (2017). L'enseignement de l'informatique à l'école primaire vu par les acteurs de l'accompagnement des pratiques pédagogiques. Dans J. Henry, A. Nguyen et É. Vandeput (dir.), *L'informatique et le numérique dans la classe. Qui, quoi, comment ?* (pp. 139–152). Namur : Presses Universitaires de Namur.

Dimitri Racordon & Didier Buchs

CUI, Université de Genève
dimitri.racordon@unige.ch, didier.buchs@unige.ch

Démystifier les concepts informatiques par l'expérimentation

Résumé
L'informatique s'étant aujourd'hui disséminée dans tous les domaines de la vie quotidienne, une compréhension générale de ses principes est devenue un outil indispensable. Malheureusement, force est constater que cet objectif est encore loin d'être atteint pour une majeure partie de la population, même dans les jeunes démographies. Ce document résume brièvement les expériences que nous avons conduites dans le cadre de campagnes de promotion des sciences informatiques. Nous en extrayons une approche plus générale et détaillons les avantages d'une telle démarche.

Mots clés : algorithmique, spécifications, apprentissage

1 Introduction

Malgré son omniprésence, l'informatique est une science peu connue du grand public. De plus, elle est employée comme synonyme de beaucoup d'autres termes qui ne la représentent pas en tant que science, mais plutôt son application à un domaine spécifique. Par exemple, les termes « numérique » ou encore « digital » sont souvent considérés comme équivalents, alors qu'ils correspondent plus à des applications ou technologies issues de la science elle-même. Outre la confusion qui plane au-dessus des termes, les concepts informatiques eux-mêmes sont également relativement méconnus. À tort, il est souvent admis qu'une maîtrise de l'utilisation d'une technologie implique une certaine connaissance de son fonctionnement. Par exemple, les principes sur lesquels repose le fonctionnement d'Internet sont généralement peu connus du grand public, alors que son utilisation est désormais monnaie

courante. Le mysticisme qui entoure l'informatique constitue un autre facteur intéressant. Si aujourd'hui les bases des mathématiques, de la physique ou encore des sciences humaines sont enseignées à tous, il en est tout autre pour l'informatique. Ce malheureux constat révèle que des termes tels qu'« algorithme », « complexité » ou encore « décidabilité » n'ont que peu si ce n'est aucune signification pour le grand public. Dans la communication de Baron et Bruillard (2001), un historique des difficultés de l'enseignement de l'informatique est relaté. Le problème apparaît comme étant lié à la fois à la démocratisation de l'usage des outils informatiques, de leur passage à une approche plus intuitive que raisonnée et la tardive émergence de la science elle-même. Une problématique similaire semble avoir existé dans l'enseignement d'autres sciences formelles comme les mathématiques.

Jacques Baudé, président d'honneur de l'EPI (Association Enseignement Public et Informatique – France) l'exprime de cette manière[1] :

> On nous dit encore, qu'il n'est pas besoin de savoirs savants, qu'il suffit de cliquer. Bel argument pour des marchands mais le rôle des enseignants n'est-il pas d'ouvrir ou d'entrouvrir, tant que faire se peut, les boîtes noires, de donner aux élèves les moyens de comprendre ce qu'ils font, de prendre du recul, et non pas d'en faire de simples consommateurs toujours dépassés ?

Dans ce document, nous proposons de définir un cadre permettant l'appréhension des fondements de l'informatique. Notre méthode s'appuie sur l'expérimentation comme un moyen de contrecarrer le mysticisme qui entoure cette science, en associant une signification concrète et familière à ses concepts, puis en offrant la possibilité de les manipuler de façon ludique et intuitive.

Nous ne sommes pas spécialisés en didactique et nos réflexions sont essentiellement basées sur l'expérience d'enseignement universitaire. Néanmoins, notre domaine d'activité en recherche s'intéressant aux aspects formels de l'informatique, ce qui par ailleurs explique notre goût pour l'abstraction et la modélisation formelle des systèmes, notre force est d'être à la croisée des approches théoriques et pratiques. Nous avons également une bonne connaissance des langages de programmation et des méthodes de développement du logiciel, nous donnant les outils nécessaires pour lier ce que nous appelons les concepts fondamentaux de l'informatique et leur application dans des domaines concrets. En outre, nous avons eu l'occasion de mettre notre démarche

1 <http://www.epi.asso.fr/revue/editic/jb-asti.htm>

d'apprentissage en pratique dans le cadre de diverses campagnes de promotion de l'université, et en particulier des études en informatique.

2 Résolution de problème : quoi et comment

Comme nous l'avons précisé en introduction, notre domaine d'activité principal s'intéresse aux aspects formels de l'informatique. Pour cette raison, nous choisissons de privilégier une vision abstraite et de masquer les aspects plus technologiques. Par là, nous entendons que nous préférons développer une compréhension des propriétés des algorithmes, ainsi que des données qu'ils manipulent, plutôt que de nous focaliser sur leurs applications. Lorsque nous aborderons nos expériences sur le terrain dans le chapitre 3, nous expliquerons par exemple comment nous avons pu tirer partie de robots, une application très technologique de l'informatique, pour discuter d'algorithmes répartis.

Ces aspects abstraits sont souvent nécessaires car ils précisent les informations que le contexte d'exécution des programmes demande. En tant que développeurs de logiciels, il nous paraît clair que ces notions sont primordiales. Nous pensons malheureusement qu'elles constituent le plus souvent un obstacle à la compréhension des fondements des problèmes à résoudre si le public ne parvient pas à percevoir l'implication concrète d'une opération ou propriété décrite de manière abstraite. L'acquisition de cette capacité n'est pas triviale. Le défi est cependant loin d'être insurmontable, et nous pensons qu'il peut être grandement simplifié en structurant de manière détaillée et systématique le processus de création d'un algorithme. Ce processus peut être décomposé en trois étapes, que nous détaillerons plus loin :

1. Classification des données
2. Description des propriétés de l'algorithme
3. Description des opérations de l'algorithme

L'une des difficultés de l'informatique est qu'elle requiert une extrême précision dans la description de ces trois étapes, entre autres parce que l'ordinateur doit être programmé avec soin pour pouvoir fournir des résultats

satisfaisants. Néanmoins cette nécessité peut être vue comme une opportunité, car hormis des travaux très manuels, beaucoup d'autres activités n'encouragent pas cette dimension. Précisons également qu'en utilisant une formulation abstraite, nous nous rendons indépendants d'une technologie ou d'un langage de programmation particulier. Nous utilisons donc une notation dite formelle (ou mathématique), dont le principal avantage se situe au niveau de son universalité. Alors qu'il existe d'innombrables langages de programmation et plateformes, tous accompagnés de leurs subtilités respectives, une notation formelle permet de fédérer les syntaxes et sémantiques de ces différents systèmes.

Pour exprimer ces problèmes, nous allons donc nous concentrer sur la description des données à traiter ainsi que sur les opérations à effectuer. Les opérations seront également abstraites, en utilisant un système permettant l'expression de leur état avant et après exécution de l'opération.

2.1 Classification des données

La représentation des données est une composante souvent négligée lorsque l'on parle d'algorithme. Elle est néanmoins essentielle, non seulement parce qu'elle permet une description précise des domaines d'entrée et de sortie d'un algorithme, mais aussi car elle peut contribuer à l'efficacité de l'algorithme lui-même. Beaucoup d'algorithmes simples vont travailler sur des données élémentaires. Historiquement, comme l'informatique était principalement utilisée pour résoudre des problèmes numériques, les algorithmes se contentaient de manipuler des nombres. Mais avec l'arrivée d'applications dans des domaines bien plus variés, une telle simplicité n'est plus suffisante. Il devient donc nécessaire de considérer d'autres types de données, plus complexes, qu'il convient de décrire minutieusement. Il existe une myriade de structures de données différentes. En fait, un pan important de la recherche en informatique consiste même à les étudier. Nous appelons cette étape la **classification**.

2.2 Descriptions des propriétés : le quoi

Décrire les propriétés d'un algorithme permet de réfléchir au problème à résoudre. Il convient ici d'identifier précisément l'objectif de l'algorithme,

ainsi que la nature des données qu'il traitera. Nous appelons cette étape le **quoi**. Même si les propriétés d'un algorithme pourront avoir des implications importantes sur la description de ses opérations, comme nous en discuterons plus loin, il est important de ne pas se soucier durant cette étape de la méthode de résolution qui sera développée à l'étape suivante. Certes, en fonction de la difficulté que représente cette dernière, il conviendra peut-être d'ajouter des propriétés (ou contraintes) sur les données à traiter pour simplifier le problème. Mais en distinguant bien ces deux étapes (description des propriétés *puis* des opérations), il nous sera donné l'opportunité de comprendre pourquoi un problème est trop difficile (voire impossible) à résoudre étant donné les propriétés des données qu'il manipule.

2.3 Descriptions des opérations : le comment

Forts de la description abstraite d'un problème, il nous est désormais possible de fournir une procédure permettant de le résoudre. Cette procédure est généralement fournie sous la forme d'une liste d'opérations (ou instructions) à exécuter. Ces dernières correspondent en réalité à des transformations qui sont appliquées de manière successive aux données d'entrées. Nous appelons cette étape le **comment**.

C'est durant cette étape, et en fonction des propriétés décrites à l'étape précédente, que nous serons en mesure d'identifier la complexité de l'algorithme. Certains problèmes peuvent s'avérer trop difficiles ou même impossibles à résoudre. Le *problème de l'arrêt* (Turing, 1937) est exemple connu de problème indécidable. Ce dernier consiste à déterminer si un programme termine ou non, à partir de la description de ses opérations. D'autres problèmes peuvent être montrés indécidables ou trop coûteux en général, mais toutefois disposer d'une solution si l'on contraint davantage leurs données d'entrées (c'est-à-dire en modifiant le quoi).

2.4 Un exemple : le tri de valeurs

Afin d'illustrer les étapes décrites ci-dessus, nous proposons de nous intéresser au problème du tri de valeurs. Il s'agit ici de construire un algorithme permettant de trier une liste de valeurs (numériques ou non). Nous allons procéder avec la même approche méthodique que nous avons présentée,

c'est-à-dire en identifiant les données à manipuler, en décrivant le quoi puis finalement le comment.

En parallèle aux explications *en prose* de la classification, du quoi et du comment, nous nous efforcerons également d'adopter une notation formelle, dans le but de bien illustrer la correspondance entre notations et signification.

La classification

Pour pouvoir représenter des listes, il nous faut un moyen d'associer un rang à chaque valeur. Pour se faire, nous admettrons que les données d'entrée de l'algorithme seront fournies sous la forme d'un ensemble de paires rang, valeur.

Formellement, nous utiliserons E pour représenter le type (ou domaine) de données à trier. Nous utiliserons les nombres naturels $\mathbb{N}$ pour représenter le rang d'une valeur dans une liste, et définirons le domaine d'entrée et de sortie de notre algorithme par $\mathcal{P}(\mathbb{N} \times E)$. Par exemple, la liste de lettres [d, a, p] serait représentée par l'ensemble $\{(d, 0), (a, 1), (p, 2)\}$.

Le quoi

Forts d'une description précise des domaines d'entrée et de sortie de notre algorithme de tri, nous pouvons désormais spécifier les contraintes de l'opération. Il convient tout d'abord de réfléchir aux contraintes qu'il pourrait exister sur le type de données que nous souhaitons manipuler. Étant donné que nous souhaitons exprimer un algorithme de tri de valeurs, il paraît donc naturel de ne considérer que des valeurs qui peuvent être comparées. Parce qu'historiquement il est coutume de définir le tri de valeurs *numériques*, il est souvent admis qu'une telle relation existe entre tout couple de valeurs. Or, il n'est pas nécessairement évident ou même possible de comparer les éléments d'autres types d'ensembles. Par exemple, il n'y aurait aucun sens à vouloir trier une liste des fruits et des animaux. Et quand bien même notre liste ne serait composée que de fruits, il conviendrait encore de définir un critère de comparaison, comme la taille ou le poids. Nous contraindrons donc l'ensemble des valeurs à trier à disposer d'un tel critère.

Formellement, soit E le domaine des données à trier, nous admettrons qu'il existe une relation $< \subseteq E \times E$ décrivant si un élément $e_1 \in E$ est plus petit qu'un élément $e_2 \in E$ (noté $e_1 < e_2$).

Enfin, nous considérerons que notre algorithme aura bel et bien correctement trié la liste qui lui aura été présentée si et seulement si :

1. Toutes les valeurs en entrée auront été préservées en sortie (i.e. pas de perte d'information)
2. Aucune valeur absente en entrée n'aura été ajoutée en sortie
3. Pour tout couple de paires rang, valeur, la paire avec le rang le plus petit sera associée à la valeur la plus petite.

Par exemple, si la liste $\{(d, 0), (a, 1), (p, 2)\}$ était présentée à notre algorithme, nous nous attendrions à obtenir la liste $\{(a, 0), (d, 1), (p, 2)\}$ en sortie.

Formellement, soit $l, l' \in \mathcal{P}(\mathbb{N} \times E)$ les listes d'entrée et de sortie respectivement, $\forall (i, e) \in l, \exists (i', e) \in l'$ (1re contrainte), $||l|| = ||l'||$ (2e contrainte) et $\forall (i_1, e_1), (i_2, e_2) \in l', i_1 < i_2 \Rightarrow e_1 < e_2$ (3e contrainte).

Le comment

La transformation de la description du quoi à une description du comment sous-entend généralement l'introduction d'algorithmes. Il est intéressant de constater que pour un énoncé de problème il correspond généralement plusieurs algorithmes. Dans certains cas il n'en existe pas ou que sur une version limitée du problème ; on dit alors que le problème est indécidable (domaine théorique de la calculabilité). Dans d'autres cas, les algorithmes possibles peuvent s'avérer très ou même trop coûteux en temps ou en espace (domaine théorique de la complexité). On peut dire que ce sont des limites naturelles liées aux possibilités d'automatisation.

Il existe une multitude d'algorithmes de tri qui respectent les contraintes que nous avons spécifiées ci-dessus. Dans le cadre de notre exemple, nous allons nous contenter d'en décrire un seul, appelé **bubblesort**. Le principe de cet algorithme est de déplacer successivement les valeurs inférieures de la liste vers le début de celle-ci, comme s'il s'agissait de bulles dans un liquide :

```
1. for let i in 0 .. |l| do
2.   for let j in i .. |l| do
3.     let (i, el) in l
4.     let (j, er) in l
5.     if el > er do
6.       l := l - {(i, el), (j, er)} | {(i, er), (j, el)}
```

Les premières lignes de l'algorithme décrivent deux boucles, c'est-à-dire une répétition d'instructions. L'objectif est de répéter les lignes 3 à 6 pour chaque paire de valeurs dans la liste à trier. Les lignes 3 et 4 extraient la paire de couples à comparer. La ligne 5 vérifie que la valeur à gauche ne soit pas plus grande que celle à droite, le cas échéant la ligne 6 inverse les valeurs associées au rang des couples extraits.

L'algorithme proposé ci-dessus est indépendant du langage de programmation que l'on pourrait utiliser pour l'implémenter. C'est là que nous souhaitons arrêter notre démarche intellectuelle, car nous sortirions du domaine abstrait et aurions à nous heurter aux problématiques liées à la technologie (sémantique du langage, mémoire à disposition, etc.). Plus tard, nous verrons même qu'en restant à ce niveau d'abstraction, il est possible *d'exécuter* cet algorithme de manière bien plus concrète, avec des objets physiques. Ceci souligne donc l'intérêt d'appliquer des abstractions.

3 Notre action sur le terrain

Dans le cadre de campagnes de promotion de l'université, et en particulier des études d'informatique, nous avons tenté d'appliquer la démarche que nous avons présentée ci-dessus pour mener diverses présentations et activités interactives. Entre autres, nous avons participé en 2016 à l'événement *Science Me!* (s. d.), une compétition annuelle dont le but est de présenter les sciences au grand public, par le biais d'une présentation scénarisée. En mettant en scène une sorte de simili de programme culinaire, nous avons expliqué la signification du terme *algorithme*, en tirant un parallèle avec une recette de cuisine. Nous avons également apporté notre contribution dans diverses journées d'initiation, notamment plusieurs éditions du TecDay (*Un collège se transforme en gigantesque laboratoire*, 2017), où nous animons chaque année une activité durant laquelle des élèves de 15 à 20 ans sont invités à mettre au point puis expérimenter un algorithme réparti.

Ces deux expériences ont été accueillies avec succès et nous ont confortés dans la pertinence de notre approche. Elles nous ont également permis de raffiner notre approche au cours du temps. Par exemple, nous

nous sommes rendu compte que l'expérimentation était une composante essentielle à la compréhension.

3.1 Nuit de la science

Une première expérience consistait en la mise à disposition des collégiens d'un environnement de programmation de robot afin de réaliser une tâche simple. Cette approche était très satisfaisante en termes d'enthousiasme et d'investissement des collégiens aux tâches proposées. Néanmoins nous déplorions à chaque événement que le public perçoive notre activité comme *un jeu avec des robots*, plutôt que d'un atelier de découverte de l'algorithmique.

C'est ce constat qui nous a poussés à nous éloigner de la technologie. Notre activité se focalisait trop sur le simple fait de déplacer des robots, et pas sur une réflexion sous-jacente quant à l'algorithme décrivant la tâche à réaliser, car les étudiants pouvaient se contenter *d'essayer* des séries d'instructions jusqu'à parvenir à la solution escomptée.

3.2 Science Me

Science Me! est une compétition amicale regroupant plusieurs équipes de scientifiques souhaitant partager leur vision de la science par le biais de présentations mises en scène. Introduire l'informatique dans ce type de cadre représente souvent un défi, car à l'inverse de sciences telles que la physique ou la chimie, il est difficile en informatique de présenter des résultats spectaculaires à un public profane. C'est pourquoi au lieu de partir des applications plus communes de l'informatique, telles que la programmation ou encore la robotique, nous avons fait le choix de miser sur une activité *a priori* complètement annexe : la cuisine. Nous avons représenté une recette de gaspacho sous forme de réseau de Petri (Reisig, 1985), un formalisme informatique permettant d'exprimer formellement tout type de processus. La figure 1 présente le réseau complet. Lors de la présentation, nous avons ensuite illustré pas à pas le déroulement de l'algorithme, en réalisant tout simplement notre recette. Non seulement ce format nous a permis d'expliquer la notion d'algorithme, mais également d'introduire une notation formelle. Les retours du public ont été très positifs et ont clairement confirmé

l'intérêt de notre approche. En utilisant un domaine *a priori* connu de tous pour illustrer un concept informatique, les notions que nous souhaitions transmettre se sont avérées bien plus faciles à communiquer que si nous étions partis d'une application plus directement liée.

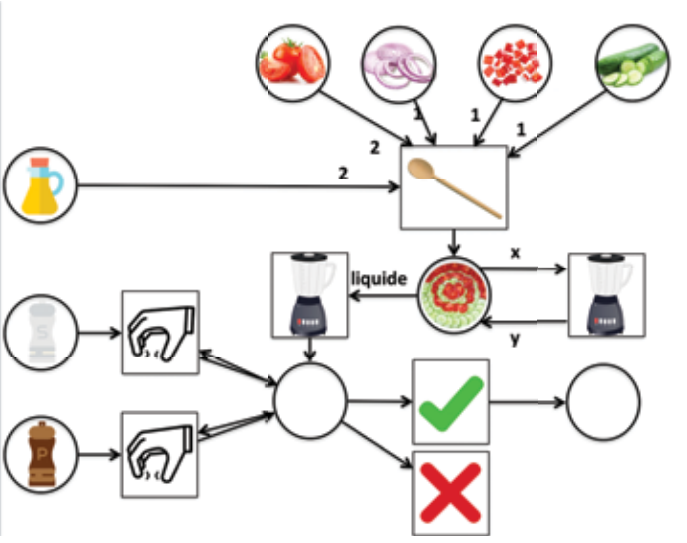


Figure 1 : Réseau de Petri représentant la préparation d'un gaspacho.

3.3 TecDay

Le TecDay est initiative de l'Académie Suisse des Sciences Techniques (SATW). Il s'agit d'une journée d'activités organisée dans les collèges de toute la Suisse, dont le but est d'offrir aux élèves la possibilité de rencontrer des chercheurs et ingénieurs afin que ceux-ci leur présentent leurs travaux. Sur deux années consécutives, nous avons proposé aux étudiants du collège Sismondi puis à ceux du collège Rousseau de se familiariser avec la notion d'algorithme. L'expérience consiste à mettre au point un algorithme distribué d'élection de leader. Dans un premier temps, le problème et ses contraintes sont présentés aux élèves, qui sont alors invités à trouver par eux-mêmes une solution. Les forces et faiblesses de ces solutions sont discutées en groupes, puis nous proposons notre propre alternative. Avec le support d'une plaquette (illustrée par la

figure 2) représentant la mémoire interne dont disposerait un ordinateur, nous exécutons notre algorithme en groupe, dans le but de bien comprendre chaque étape de son déroulement.

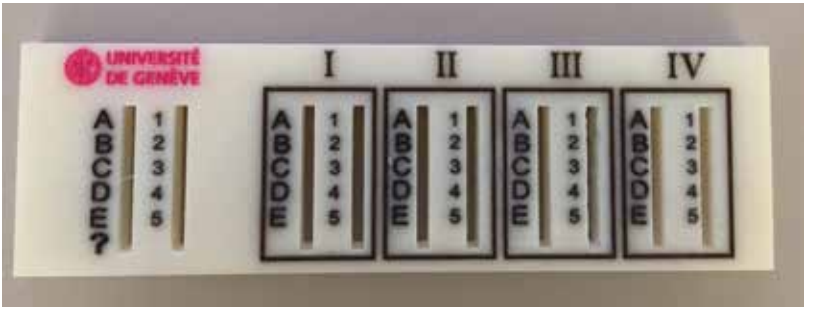


Figure 2 : Plaquette représentant la mémoire d'un ordinateur.

Les retours de cette expérience ont également été des plus encourageants, et ce sur les deux années. Un aspect intéressant du format du TecDay est qu'il nous permet de faire participer le public, en premier lieu de par une recherche de solution par rapport à un problème donné, puis par l'expérimentation.

4 Démarche générale

Dans nos trois expériences, l'élément clé est que nous avons approché des concepts informatiques abstraits (algorithmes et structures de données) par le biais d'une application concrète, parfaitement à la portée du public. Ceci permet de démystifier les termes, en proposant de les illustrer avec des notions plus familières, et de les expérimenter plus directement. Parce que nous faisons abstraction des considérations technologiques, notre approche est en fait très théorique. Néanmoins, parce que nous la lions très fortement à un domaine familier, elle peut être approchée avec bien plus de simplicité. En outre, si nos interventions au TecDay et à la nuit de la science peuvent être perçues comme plus proches d'une application classique de l'informatique, notre participation à *ScienceMe!* démontre que

des domaines *a priori* très éloignés peuvent toutefois servir de supports appropriés pour l'illustration de concepts abstraits.

Nous en déduisons une démarche plus générale :

1. Identifier un domaine d'application familier du public ciblé
2. Identifier les concepts informatiques qui peuvent être appliqués au domaine
3. Mettre au point une expérience pratique permettant d'illustrer les concepts identifiés

En partant du domaine d'application, nous nous assurons que le public puisse comprendre l'intérêt de l'approche informatique. Par exemple, dans le cas de la recette de cuisine, il apparaît évident le besoin de disposer d'une marche à suivre détaillant les étapes de réalisation. L'introduction de l'algorithmique devient alors naturelle. De la même manière, de la réflexion autour des contraintes liées à la création d'une procédure d'élection, il apparaît évident le besoin de décrire précisément les objectifs et contraintes associés.

Si nos expériences se sont principalement concentrées sur la notion d'algorithme, notre approche peut être appliquée à d'autres concepts informatiques. Par exemple, nous évoquions dans l'introduction qu'Internet reposait sur des concepts relativement méconnus. En partant de l'Internet comme domaine d'application familier, une possible activité consisterait à étudier les différents composants nécessaires à son fonctionnement. Une première étape inviterait le public à identifier ces composants, dans le cadre d'une discussion de groupe. Puis, une seconde étape consisterait à distribuer le rôle de chaque composant, avant de jouer le transfert d'une donnée d'un serveur à un client.

5 Mise en œuvre ludique d'algorithmes

Dans le chapitre précédent, nous avons donné une démarche générale pour l'approche de concepts informatiques. Dans ce chapitre, nous proposons d'illustrer cette démarche avec un exemple d'activité pédagogique. Nous reprendrons l'algorithme de tri présenté dans le chapitre 2, et adapterons son processus de création à notre activité.

Notre objectif sera de trier un jeu de 52 cartes dans l'ordre classique, c'est-à-dire de l'as au roi, de la couleur pique suivie de cœur, carreau et trèfle.

La classification

Dans le chapitre 2, nous avons représenté les listes par l'intermédiaire de couples rang, valeur. Nous ferons de même ici, en utilisant un paquet de cartes simplement numérotées de 1 à 52. Ce second jeu de cartes représentera les rangs, alors que le premier représentera des valeurs. Dès lors, nous pourrons introduire deux notions fondamentales : les ensembles (ou types) ainsi qu'une poignée d'opérations sur ces ensembles.

Dans le chapitre 2, nous avons exprimé formellement les listes sous la forme d'un produit cartésien $\mathbb{N} \times E$. En fait, ceci correspondra dans notre activité à former l'ensemble de tous les couples de cartes qui contiennent une carte de chaque paquet. Nous pourrons même représenter cela physiquement, sous la forme d'une sorte de support que le public pourra utiliser pour associer les cartes (voir figure 3).

Nous pouvons déjà souligner l'illustration par des moyens concrets d'une notion abstraite : le produit cartésien. En fait, il est intéressant de constater qu'en fournissant des supports pour plus un nombre arbitraire n de cartes, il est possible de représenter des n-uplets, c'est-à-dire des représentants de produits cartésiens de n ensembles (i.e. $S_1 \times S_2 \times \cdots \times S_n$).

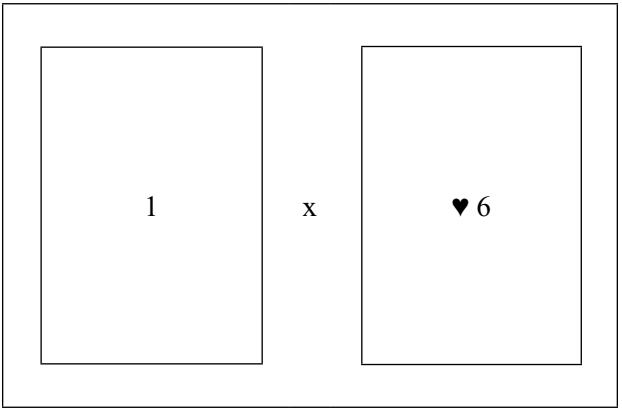


Figure 3 : Support pour couple rang, valeur.

Le quoi

Dans le cadre de cet exemple, décrire le quoi représente une sous-activité très intéressante. En effet, s'il paraît très simple d'énoncer l'objectif d'un tri, déterminer toutes les contraintes qui décrivent son succès n'est pas si évident. Il sera ici l'occasion de mener une discussion de groupe et inviter le public à se rendre compte de l'imprécision probable d'une description profane. Une erreur fréquente sera sans doute de ne considérer que l'ordre de la liste produite par l'algorithme, mais pas la préservation des valeurs qui auront été fournies en entrée. Sans cette condition, la liste vide serait une réponse parfaitement acceptable à n'importe quelles données d'entrée.

Un autre axe sera de discuter de l'ordre lui-même. Traditionnellement, un jeu de cartes est trié de l'as au roi, en partant de la couleur pique, suivie de cœur, carreau et trèfle. L'énonciation précise d'un tel ordre représente également un exercice intéressant.

Le comment

Finalement, la dernière étape de l'activité consistera à décrire un moyen de parvenir à réaliser l'objectif principal (i.e. le tri du jeu de cartes), tout en respectant les contraintes qui auront été identifiées lors de l'étape précédente. Il sera là encore l'occasion d'entreprendre une discussion de groupe pour tenter de coucher sur papier une méthode de résolution. Tout comme nous le faisons au TecDay, nous pourrons ensuite proposer une ou plusieurs solutions, et discuter de leurs tenants et aboutissants. Il est à noter que le tri constitue un problème qui se prête extrêmement bien à ce genre d'exercice, car il n'est pas très compliqué ni d'identifier différents algorithmes, ni d'en comprendre les implications, notamment en termes de complexité.

Muni des jeux de cartes et des supports permettant de représenter des éléments de produit cartésien, le public finira par être invité à expérimenter un ou plusieurs des algorithmes qui auront été élaborés au préalable.

6 Conclusion

Tout comme les mathématiques et la physique, l'informatique souffre souvent d'une image trop abstraite, ayant tendance à dissuader le grand public

de s'y intéresser. Néanmoins, il apparaît évident que l'enseignement de ses concepts fondamentaux est devenu indispensable dans le contexte actuel. Dans ce document, nous avons montré qu'en adoptant une approche moins théorique et plus basée sur la réflexion et l'expérimentation, l'apprentissage de notions jugées *a priori* complexes et très abstraites peut être grandement facilité. En particulier, nous avons montré qu'il était relativement aisé d'introduire la notion d'algorithme dans des domaines familiers du grand public, et ainsi démystifier le terme ainsi que le concept qu'il représente. Nous avons également montré qu'une approche similaire pouvait être utilisée pour l'introduction d'autres notions et concepts. Et finalement, nous avons illustré l'application de notre démarche dans la réalisation d'une activité pédagogique.

Notre démarche est issue de réflexions sur les expériences que nous avons menées lors de campagnes de promotion de l'université. Ce cadre représente pour nous une opportunité intéressante d'expérimenter diverses approches pédagogiques, afin de raffiner notre discours et notre approche.

Références

Baron, G.-L., & Bruillard, E. (2001). Une didactique de l'informatique ? *Revue française de pédagogie*, 163–172. Reisig, W. (1985). *Petri nets : An introduction*. New York, NY, USA : Springer-Verlag New York, Inc.

Schiper, A. (2016). *Découvrir le numérique : une introduction à l'informatique et aux systèmes de communication*.

Presses polytechniques et universitaires romandes.

Science me! (s. d.). <http://scienceme.unige.ch>. (Accessed : 2017–07-12).

Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, *2*(1), 230–265.

Un collège se transforme en gigantesque laboratoire. (2017). <goo.gl/LA8xh9>. (Accessed : 2017–07-12).

Sandra Nogry

Laboratoire Paragraphe, Université Cergy-Pontoise
sandra.nogry@u-cergy.fr

Comment apprennent les élèves au cours d'une séquence de robotique éducative en classe de CP ?

Résumé
À l'école primaire, la robotique éducative est maintenant proposée pour introduire l'apprentissage de l'informatique. La présente étude vise à documenter le processus d'apprentissage au cours d'une séquence de robotique proposée et mise en œuvre par une enseignante de CP. À partir de vidéos des séances enregistrées dans le cadre du projet DALIE, l'activité des élèves a été analysée en vue de caractériser les difficultés rencontrées, les genèses instrumentales et conceptuelles qui apparaissent progressivement au cours de la séquence, ainsi que les médiations qui concourent à ces genèses. Cette étude permet ainsi de mettre en évidence la dynamique des apprentissages dans une telle séquence.

Mots clés : robotique éducative, école élémentaire, apprentissage, approche génétique

1 Introduction

La transmission d'une culture informatique pour tous est devenue ces dernières années un enjeu sociétal. L'objectif du projet ANR DALIE (didactique et apprentissages de l'informatique à l'école primaire) est d'étudier comment mettre en place un curriculum d'informatique à l'école primaire. Plusieurs questions y sont abordées : quelles connaissances enseigner dans les différents cycles ? Comment les enseignants mettent-ils en œuvre des séquences d'enseignement de l'informatique ? Quelles connaissances les élèves construisent-ils ?

Au sein de ce projet, la présente étude s'est focalisée sur l'analyse de l'activité d'apprentissage dans une séquence de robotique éducative au CP. Dans ce domaine, différentes études ont montré l'efficacité de séquences de

robotique proposées ou co-élaborées avec les chercheurs pour apprendre certains concepts informatiques ainsi que la démarche de résolution de problème. Ainsi, dès la maternelle, les élèves peuvent apprendre le concept de séquence et développer des capacités à planifier, à gérer plusieurs commandes de programmation, à réaliser un algorithme et à déboguer (Komis & Misirli, 2015 ; Bers *et al.*, 2014). Néanmoins, peu d'études portent sur des séances conçues par les enseignants eux-mêmes. Comment les enfants apprennent-ils durant cette séquence de robotique éducative conçue par l'enseignant ? Quelles sont les genèses instrumentales et conceptuelles durant cette séquence ? Quels sont les facteurs qui concourent à ces genèses ?

2 Cadre théorique

Dans la perspective des théories de l'activité, notre unité d'analyse est l'activité, définie par P. Falzon comme « ce qui est fait, ce qui est mis en jeu par le sujet pour réaliser l'objectif qu'il se fixe dans une situation donnée. L'activité ne se réduit pas au comportement, elle inclut l'activité intellectuelle, ainsi que les discours sur l'action, les interactions avec autrui ». Il s'agit de documenter les finalités de l'activité des élèves, le déroulement de cette activité et les médiations entre le sujet et l'objet de l'activité. Ces médiations peuvent être de nature sociale (interactions entre élèves ou avec l'enseignant) ou instrumentale, les instruments mobilisés étant les artéfacts utilisés associés à un schème défini comme organisation invariante de l'action (Rabardel, 1995).

Cette analyse a été articulée à une analyse micro-génétique du développement des connaissances (Siegler, 2006). Cette approche se fonde sur l'analyse d'épisodes d'activité durant lesquels des transformations de connaissances ou des changements conceptuels peuvent être mis en évidence. Pour ce faire, le chercheur doit mettre en place des observations denses de chaque épisode et faire un usage intensif et opportuniste de toutes les données disponibles afin d'inférer les changements réalisés ainsi que les mécanismes en jeu. En articulant ces deux approches, il s'agit de mettre en évidence les genèses instrumentales et conceptuelles qui ont lieu durant la séquence d'apprentissage proposée.

3 Méthodologie

L'étude s'est déroulée dans une classe de CP composée de 18 élèves âgés de 6 ans. L'enseignante, une maître-formatrice expérimentée, était volontaire pour participer au projet de recherche.

La séquence observée a été conçue et mise en œuvre par l'enseignante. Elle visait à faire programmer des déplacements dans l'espace. Les robots *Bee-Bot* étaient utilisés par groupe de 3 élèves. La séquence proposait une démarche d'investigation fondée sur une approche inductive ; elle se déroulait en deux phases.

(1) Faire parcourir au robot un chemin tracé au sol avec des Kapplas (séances 1 à 4)
Dans cette première phase, les consignes étaient minimales, les groupes travaillaient en autonomie durant environ 30 minutes par séance ; ils devaient découvrir par eux-mêmes le robot et en comprendre le fonctionnement, avant de rendre compte de leurs découvertes et de leurs questions lors d'un regroupement en classe entière. Après un premier temps de recherche en autonomie (séance 1), l'enseignante a précisé la consigne en demandant de tracer une ligne droite, avant de laisser les élèves plus libres de leur tracé. L'objectif central de cette première phase était la compréhension des commandes de contrôle.

(2) Dessiner par groupe un parcours à programmer ensuite (séances 5 et 6)
Durant la séance 5, chaque groupe dessinait un parcours sur papier puis programmait le robot pour réaliser son propre parcours tandis que durant la séance 6, le parcours était d'abord dessiné par un groupe avant d'être donné à un autre groupe qui programmait le robot pour qu'il exécute ce parcours. L'objectif était de les amener à programmer une séquence d'instructions.

Les robots utilisés, des *Bee-Bot*, sont des jouets programmables contrôlés à partir d'une interface tangible sur la partie supérieure de l'objet : quatre boutons permettent d'orienter l'objet, trois boutons permettent de contrôler l'exécution du programme (GO : exécution des instructions programmées, PAUSE : interruption momentanée de l'exécution du programme, CLEAR : effacement des commandes enregistrées en mémoire).

Le protocole mis en place dans le projet DALIE consistait à suivre le travail réalisé dans la classe en observant et filmant une séance toutes les deux semaines et en menant des entretiens et *focus groups* avec les enseignants. Dans la classe observée, quatre séances sur six ont été observées et filmées. Le dispositif de captation était composé d'une caméra fixe (plan large sur la salle) et d'une caméra mobile orientée successivement sur le travail des différents groupes d'élèves.

Au total, quatre heures de vidéos ont été enregistrées et analysées. À partir ces enregistrements, une analyse de l'activité des élèves a été réalisée suivant trois étapes :

– identification des épisodes durant lesquels des changements opératoires ou conceptuels sont visibles
– transcription et analyse des séquences dans lesquels apparaissent ces changements,
– mise en évidence de la nature des changements (genèses instrumentales/ conceptuelles)

4 Résultats

Nous avons cherché à identifier les genèses conceptuelles et instrumentales (constitution de nouveaux schèmes associés au robot) ainsi que les difficultés rencontrées dans chacune des phases de la séquence proposée.

4.1 Phase 1 : faire parcourir au robot un trajet dessiné sur le sol

Activité des élèves

Dans cette première phase, les élèves avaient pour consigne de tracer un chemin sur le sol et faire parcourir ce chemin au robot. Durant les trois premières séances, les élèves construisent rapidement un chemin, se saisissent du robot et appuient sur différentes touches pour le faire avancer. Leur activité n'est pas coordonnée : chacun essaie de se saisir du robot sans répartition des tâches ni discussion sur la stratégie à adopter. Les

interactions entre élèves se limitent à exprimer leur déception ou leur joie suivant si l'objectif est atteint ou pas.

Une stratégie de type « pas à pas » semble d'abord privilégiée, l'enfant qui a le robot en main introduit une instruction à la fois et observe le comportement du robot. Le plus souvent, il commence par appuyer sur la commande GO puis sur la flèche avant. Cette séquence de touches est présentée par un élève en regroupement comme étant la procédure à suivre ; aucun élève ne venant le contredire, elle devient ensuite un schème (organisation invariante de l'action) largement partagé au sein de la classe. Dans chaque groupe les élèves constatent régulièrement un écart entre leur intention (voir le robot se déplacer d'un pas) et le déplacement effectif du robot, variable suivant les instructions précédemment entrées par d'autres élèves. Ces déplacements suscitent de l'incompréhension : une élève dit ainsi à l'enseignante « *au début il m'écoutait et là il m'écoute plus du tout* ». Ceci les conduit à adopter une stratégie *essai-erreur* ou à modifier le parcours sur le sol pour l'adapter au déplacement du robot afin de réussir la tâche ; certains en viennent à se désengager de la tâche.

Médiations proposées par l'enseignante

Lorsqu'elle circule dans les groupes, l'enseignante intervient ponctuellement pour amener les élèves à s'interroger sur les relations de cause à effet entre les touches sélectionnées et le comportement du robot puis à anticiper le résultat de leurs actions (« *pourquoi il se déplace comme ça ?* » ; « *et comment vous allez faire pour qu'il se déplace autrement ?* »). Au cours de ces échanges, dans plusieurs groupes les élèves découvrent la commande de contrôle CLEAR et l'utilisent pour arrêter l'exécution du programme avant un nouvel essai. Par son questionnement, l'enseignante introduit également l'idée qu'un programme peut être exécuté plusieurs fois de façon identique. Ces questions suscitent des interrogations et des doutes chez les élèves, amorcent une réflexion poursuivie collectivement en fin de séance.

Des genèses instrumentales et conceptuelles

L'analyse de l'activité des élèves durant les trois premières séances met en évidence l'apparition d'un schème d'action associé au robot (GO-flèche) partagé collectivement au sein de la classe mais celui-ci ne permet pas de contrôler efficacement le déplacement du robot. Les élèves semblent adopter une conception anthropomorphique du robot selon laquelle « la machine

prend en compte les intentions, et traite sémantiquement les opérations en jeu » (Rogalski, 2015) ; ainsi, on peut avancer une interprétation sémantique de la séquence « GO-flèche » : va – en avant, en arrière, etc.). Un manque de compréhension du caractère différé de l'exécution du programme et un manque de compréhension des commandes de contrôle (touches démarrage, PAUSE, CLEAR) sont également notables. L'utilité de la touche de contrôle CLEAR est progressivement découverte dans les différents groupes, mais sa fonction (effacer la mémoire) n'est pas encore comprise. À l'issue de la quatrième séance, l'enseignante s'appuie sur une situation de réussite pour amener les élèves à préciser le rôle des commandes de contrôle et l'ordre dans lequel les utiliser ; une séquence d'action est instituée : remettre à zéro, appuyer sur les flèches, appuyer sur GO.

4.2 Phase 2 : programmer le robot pour qu'il exécute un parcours dessiné

Dans la phase suivante, l'activité est fortement prescrite par l'enseignante : les élèves doivent dessiner sur une feuille le parcours du robot, puis représenter ses différents déplacements par des flèches puis programmer le robot et de vérifier si le trajet parcouru correspond au parcours dessiné. L'enseignante insiste sur l'importance de se mettre d'accord avant d'agir et indique vouloir vérifier chaque étape.

Médiations proposées par l'enseignante

Différentes médiations (dessin, verbalisations) sont introduites afin de conduire les élèves à planifier le déplacement du robot et à différer son exécution. Dans un premier temps, l'enseignante propose un fort étayage : elle guide l'utilisation des commandes de contrôle (« *tu as annulé d'abord ?* »), la planification du trajet (« *d'accord, il va aller tout droit, et combien de fois ?* ») et l'analyse de l'écart entre le parcours dessiné et le déplacement effectué. Elle montre notamment la correspondance entre la représentation des déplacements sur le dessin (les flèches) et le déplacement du robot en comptant chaque pas ; elle analyse avec eux les causes de l'écart et les conduit à modifier le programme ou la représentation du déplacement lorsqu'elle est imprécise. À travers cet étayage, elle structure l'activité des élèves et introduit ainsi une procédure à suivre systématiquement.

Activité des élèves

L'analyse de l'activité des élèves montre que ces prescriptions sont suivies. À partir du dessin qu'ils ont réalisé ensemble, les élèves sont en capacité d'identifier la séquence de déplacements à programmer. Avant de programmer le robot, certains verbalisent les actions à exécuter ensuite. Dans un premier temps, la plupart adoptent une stratégie de sous-programme : ils réalisent un sous-programme pour chaque segment du parcours. La programmation des rotations, représentées par des virages, pose problème ; en effet, le robot ne tourne qu'à angle droit. Ceci les conduit à modifier la représentation du parcours pour l'ajuster aux possibilités du robot (souvent sur les conseils de l'enseignante). Pour identifier les directions à choisir, certains éprouvent le besoin de représenter les déplacements du robot en acte, par des mouvements du corps ou du robot. À l'issue de ces séances, l'ensemble des élèves est capable de produire une séquence d'instruction complète.

Genèses instrumentales et conceptuelles

Dans ces dernières séances, suite à un enseignement explicite par l'enseignante et à un étayage soutenu, les élèves apprennent à utiliser adéquatement les commandes de contrôle. La réalisation graphique du parcours joue un rôle central, elle les oblige à anticiper les déplacements du robot et à envisager une séquence d'actions à programmer ; le concept de séquence ou de programme n'est jamais verbalisé par l'enseignante, mais il apparaît au cours de l'activité comme un concept en acte. Ainsi, la structuration de la tâche, les médiations proposées et l'étayage de l'enseignante conduit les élèves à adopter systématiquement une démarche qui consiste à planifier le déplacement du robot, à réfléchir à la séquence d'instructions à entrer et à observer l'écart entre le parcours prévu et réalisé. Ce faisant, ils perçoivent le caractère différé de l'exécution d'un programme.

5 Discussion

Dans cette étude, nous avons analysé l'activité des élèves dans une séquence conçue et mise en œuvre par une enseignante expérimentée mais peu formée à l'enseignement de l'informatique. La séquence proposée était fondée

sur la démarche d'investigation ; les élèves étaient ainsi invités à découvrir par eux-mêmes le fonctionnement du robot. Face à leurs difficultés, une seconde phase reposant sur la production d'une représentation graphique du parcours à effectuer leur a été proposée. Dans la première phase de découverte, les élèves rencontrent des difficultés telles qu'une conception anthropomorphique du robot ou des difficultés à intégrer le caractère différé de l'exécution – par ailleurs bien documentées dans les travaux sur l'apprentissage de l'informatique dans le secondaire (Rogalski, 2015). Ils rencontrent également une difficulté de compréhension des commandes du *Bee-Bot* ; celles-ci paraissent simples mais nécessitent des connaissances sur les choix de conception du robot (empan de la mémoire, etc.). Cette phase de découverte les conduit à s'interroger sur le fonctionnement du robot, sur l'effet de leurs actions ainsi qu'à découvrir par eux-mêmes une commande de contrôle. Dans la seconde phase, la consigne les amène à changer d'objectif en se centrant moins sur la production d'un résultat (faire parcourir au robot le chemin) que sur la production d'une procédure, le programme à exécuter, à représenter d'abord graphiquement. Ils découvrent ce qu'est une séquence d'instruction et deviennent progressivement capables de mettre en œuvre une démarche de résolution de problème (planification, mise en œuvre, débogage). Les médiations proposées par l'enseignante ainsi que les représentations graphiques à élaborer jouent un rôle central dans cet apprentissage. Les connaissances, habiletés et stratégies sont de même nature que celles mises en évidence par Komis et Misirli (2015), bien que des médiations différentes soient proposées. Cette étude montre comment ces connaissances, habiletés et stratégies apparaissent dans la séquence proposée et souligne les facteurs qui concourent à leur développement.

Références

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering : Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157.

Komis, V. & Misirli, A. (2015). Étude des processus de construction d'algorithmes et de programmes par les petits enfants à l'aide de jouets programmables. In Drot-Delange, B., Baron, G.-L. & Bruillard, E.,

Informatique en éducation : perspectives curriculaires et didactiques. Clermont-Ferrand : Presses Universitaires Blaise-Pascal.

Rabardel, P. (1995). *Les hommes et les technologies.* Paris : Armand Colin.

Rogalski, J. 2015. Psychologie de la programmation, didactique de l'informatique déjà une histoire… In Drot-Delange, B., Baron, G.-L. & Bruillard, E., *Informatique en éducation : perspectives curriculaires et didactiques*. Clermont-Ferrand : Presses Universitaires Blaise-Pascal.

Siegler, R. S. (2006). Microgenetic analyses of learning. In Khun, D. & Siegler, R. (Eds.) *Handbook of child psychology, vol. 2* (464–504). New York : John Wiley & Sons.

Thibault Desprez, Stéphanie Noirpoudre, Théo Segonds,
Damien Caselli, Didier Roy* & Pierre-Yves Oudeyer*

Équipe FLOWERS, Inria BSO
prenom.nom@inria.fr

Poppy Ergo Jr : un kit robotique au cœur du dispositif Poppy Éducation

Résumé
Pour favoriser une meilleure appréhension du monde numérique, le dispositif Poppy Éducation propose un kit robotique pédagogique open source baptisé « kit Ergo Jr ». Il se destine principalement aux élèves et enseignants de la fin du collège et du lycée, mais sa modularité a fait naître des projets à de nombreux niveaux, abordant de nombreuses disciplines. Nous présentons ici la démarche de conception centrée utilisateur qui a conduit son développement ainsi que les objectifs de dissémination qui y ont été associés. Nous présentons le kit en lui-même, composé d'un robot à construire soi-même et d'un livret d'accompagnement pédagogique contenant des activités clé en main. Nous développons certains des usages qui en ont été faits dans le milieu scolaire, et plus particulièrement dans les sections ISN et ICN. Enfin, nous exposons les méthodes et résultats des questionnaires d'utilisabilité que nous avons fait remplir à des élèves de la région Nouvelle Aquitaine ayant pratiqué des activités avec le robot Ergo Jr. Ces questionnaires ont également été complétés par leurs enseignants.

Mots clés : kit robotique, robot, Poppy, Ergo Jr, programmation, *Snap!*, Python, sciences du numérique

1 Introduction

Pour aider chacun à mieux comprendre le monde numérique dans lequel nous vivons et ainsi en faire le meilleur usage possible, l'État français a choisi depuis 2017 d'intégrer, du cycle 1 au cycle 3, dans tous les

* Contributions égales.

programmes scolaires, les sciences du numérique. Depuis cette intégration, il devient indispensable d'identifier et de disséminer des dispositifs pédagogiques pertinents permettant d'acquérir les bases essentielles de l'informatique. Depuis 2015, le projet Poppy Éducation (équipe Flowers, Inria Bordeaux, France), reposant sur le kit robotique pédagogique Poppy Ergo Jr, s'inscrit dans cette volonté, et cherche à aider les élèves à comprendre, à apprendre et à manipuler les éléments fondamentaux des sciences du numérique et de la pensée informatique. Pour cela a été mise en place une démarche de développement centrée utilisateur, impliquant une quarantaine d'enseignants de lycées de Nouvelle Aquitaine intervenant en section Informatique et Sciences du Numérique (ISN) et Informatique et Création Numérique (ICN). Les échanges entre les chercheurs de l'équipe, les enseignants et leurs élèves ont permis de faire évoluer le matériel et de concevoir un livret pédagogique d'activités avec le robot Ergo Jr. Un des objectifs visés par le projet Poppy Éducation était de créer des kits clé en main, facilitant l'auto-formation de l'enseignant et son appropriation du dispositif, avec deux ambitions, que l'enseignant puisse modifier le kit et l'adapter à sa pratique pédagogique et qu'il partage ses réalisations avec la communauté enseignante. Pour évaluer l'impact du kit Poppy Ergo Jr dans le dispositif, nous avons choisi dans un premier temps d'adopter une posture d'observation, afin de recueillir un certain nombre de données qualitatives ; nous avons également utilisé des questionnaires standardisés afin d'estimer l'utilisabilité, et l'expérience utilisateur offerte par le kit. Aujourd'hui, plus de 30 lycées sont équipés du kit Ergo Jr ; et des expérimentations plus avancées avec suivi de cohortes sont en cours de réalisation. Nous présentons ici le kit Poppy Ergo Jr, les résultats des questionnaires standardisés, ainsi que quelques données qualitatives sur la pertinence du kit.

2 Phase de conception

2.1 Objectif & méthode

Inspirés par le kit IniRobot, qui totalisait après un an d'existence environ 700 utilisateurs adultes et 6400 enfants dans 35 villes de France (Roy &

Oudeyer, 2016), nous avons opté pour une stratégie qui favorise une dissémination *bottom-up* et l'auto-formation. Ainsi nous avons mis l'accent sur l'accessibilité du kit, que ce soit par son coût ou par la richesse et la disponibilité des ressources qui lui sont dédiées. Ces paramètres favorisent une plus grande appropriation des outils par les utilisateurs, garante d'une pérennité du dispositif.

Pour aller plus loin, nous avons souhaité favoriser le détournement de la plateforme, ainsi que le partage de ce détournement. Pour cela un certain nombre de choix sur le développement ont été faits (Noirpoudre *et al.*, 2017) : le caractère open source du dispositif, la modularité des pièces mécaniques, ou la programmation multilingue. Cependant, l'impact de ces choix sur la dissémination effective ne pourra être observé que sur le long terme. En revanche, des stratégies de conception ont déjà fait leurs preuves, comme la méthode du développement centré utilisateur (Abras, Maloney-Krichmar & Preece, 2004) que nous avons appliquée pour partie ici.

Tout d'abord, nous avons bâti un groupe de travail avec plusieurs enseignants et ingénieurs constituant nos premières réunions. Nous avons ensuite effectué les premières formations à l'utilisation de la plateforme, et certaines adaptations techniques du kit en ont découlé. Puis s'est organisé un suivi avec les enseignants via téléphone, mail, réunion, forum, observation sur place, etc. De ce suivi ont émergé plusieurs contenus et pratiques pédagogiques. Enfin, nous avons pu optimiser au cours du temps de nombreuses spécificités de la plateforme et centraliser de nombreuses activités.

Figure 1 : Robot à construire + un livret pédagogique.

Figure 2 : Robot Poppy Ergo Jr en cours d'utilisation.

2.2 Le kit Poppy Ergo Jr

Le robot Poppy Ergo Jr (Figure 2) utilisé dans le dispositif Poppy Éducation est issu de la plateforme robotique Poppy (Lapeyre, 2015) et en reprend donc les caractéristiques : cette plateforme est un ensemble de briques matérielles et logicielles open source basé sur l'impression 3D, permettant de construire différents robots, programmables avec de multiples langages (notamment via une API REST), dont l'Ergo Jr. La modularité de la plateforme permet la conception et le partage de projets éducatifs et collaboratifs mettant en jeu des compétences variées, comme la manipulation de multiples technologies (*e.g.* impression et conception 3D, objet connecté, etc.) permettant des connexions entre diverses disciplines, outils et matériaux d'une variété et d'une accessibilité toujours croissante (*cf.* fab lab, MOOC).

Les activités ont été conçues initialement pour deux langages de programmation, *Snap!* (variante de Scratch) et Python, mais il est possible d'utiliser d'autres langages. Les activités peuvent être menées avec les robots physiques ou bien avec leur version simulée. L'utilisation de ce dispositif concerne la fin du collège, le lycée et l'enseignement supérieur. Mais d'autres usages, en primaire, en maternelle, dans des fab labs, ont été observés.

De nombreuses activités sont aujourd'hui disponibles sur le site web <www.poppy-education.org> grâce au partage des enseignants et de leurs élèves. Ces activités abordent des thématiques d'une grande diversité, comme les mathématiques, la physique, les sciences de la vie et de la terre, les sciences humaines, le design, l'art, etc.

Un livret d'activités pédagogiques (Noirpoudre *et al.*, 2016) permet la prise en main du kit robotique. Il se compose d'activités de découverte de la plateforme en elle-même et des concepts de l'informatique et de la robotique (*e.g.* servomoteur, capteur, boucle) ; d'idées d'activités et de « défis » permettant d'exploiter le potentiel du kit. Ces activités visent à favoriser chez les élèves la démarche scientifique, la coopération et la création d'un micromonde d'apprentissage via le robot (Noirpoudre *et al.*, 2017).

Les ressources représentent un vecteur permettant d'orienter certains usages en les facilitant. Ainsi, la plateforme a été modifiée afin d'en accroître les possibilités de personnalisation et d'adaptabilité, notamment en accentuant la portabilité vers d'autres solutions open source. Car, partant du

principe que chaque utilisateur est unique et qu'il possède un bagage théorique et pratique spécifique, il est indispensable que la plateforme puisse s'adapter à ses singularités. Plusieurs observations réalisées par l'équipe Poppy Éducation en témoignent : des projets aboutissant, au pire, à un abandon faute de ressources, ou, au mieux, à des réalisations allant au-delà des possibilités techniques de la plateforme grâce à des ressources externes, comme l'utilisation de différents matériaux (plastique, bois, carton) ou de différentes formes de pièces pour le robot ; des pièces additionnelles (tête, pince, labyrinthe) ; des capteurs (webcam, Makey Makey, Leap Motion) et contrôleurs (Arduino) supplémentaires, etc. Ces détournements réalisés par les enseignants et leurs élèves sont avant tout possibles car l'ensemble des ressources (tutoriel, matériel et logiciel : *Snap!*, Onshape, Meshmixer, etc.) sont accessibles et réutilisables grâce aux licences open source.

3 Phase d'évaluation, les questionnaires

Dans notre démarche de conception centrée utilisateur, la première étape fut d'observer et de recueillir des données qualitatives, permettant de faire évoluer nos premiers prototypes tant sur l'aspect technique que sur l'aspect pédagogique. Nous avons également recueilli un grand nombre de témoignages écrits et/ou oraux sur les usages qu'ont eus les enseignants avec leurs élèves via l'utilisation du robot Ergo Jr.

Par rapport à notre objectif de dissémination, une première mesure de l'utilisabilité globale du dispositif était indispensable. Pour réaliser cette étude, nous avons sélectionné deux questionnaires standardisés traitant de cette question : le SUS (*System Usability Scale*, Brooke, 1996) et l'AttrakDiff (Hassenzahl, Burmester & Koller, 2003). Ces deux questionnaires sont complémentaires et permettent, d'un côté d'identifier d'éventuels problèmes de conception et de l'autre de rendre compte de la perception de l'utilisateur lors des activités. L'intérêt qu'ils soient standardisés est de pouvoir comparer les résultats.

Le lien de ces questionnaires (à compléter en ligne) a été diffusé aux enseignants participant au dispositif Poppy- Éducation, qui l'ont ensuite transmis à leurs élèves. Nous avons collecté 88 réponses : 20 enseignants de 47 ans en moyenne (écart type $S = 14.26$) et 68 élèves (*âge* = 16 ;

S = 2.44), 37 sont issus de section ISN, 12 de ICN, et 18 du collège. Ils ont tous pratiqué des activités durant l'année scolaire 2016–2017 et ont répondu aux questionnaires à la fin de cette même année. Ces activités ont pu être plus ou moins longues, plus ou moins répétées ; pour déterminer ces paramètres, un questionnaire de renseignements additionnels était intégré après les questionnaires d'utilisabilité, 29 critères de discrimination ont pu en être dégagés. Ces critères reflètent différents usages du kit et ont permis d'observer plusieurs différences significatives dans les résultats aux deux questionnaires. Ceux-ci sont visibles via les différents tracés sur les figures 3 et 4. Ici, leur interprétation ne sera vue que d'une manière générale, les données recueillies ne permettant pas une analyse à ce niveau de granularité.

Nous avons également synthétisé certains de ces critères pour former des groupes : *néophyte*, n'a pas utilisé d'autres kits et a pratiqué moins de 6 heures d'activités avec le kit Ergo Jr ; *novice*, n'a pas utilisé d'autres kits et a pratiqué entre 6 et 25 heures d'activités avec le kit Ergo Jr ; *expert*, a utilisé d'autres kits et a pratiqué plus de 25 heures d'activités avec le kit Ergo Jr ; cependant, ces groupes n'ont pas permis d'établir de distinction significative.

3.1 Utilisabilité : le SUS

Le SUS (« the Systeme Usability Scales ») (Brooke, 1996) se compose d'une série de 10 affirmations à évaluer selon une échelle de Likert à 5 points. Le score obtenu varie entre –2 et 2 et permet de graduer l'utilisabilité définie sous la norme ISO 9241–11 par *« le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié »*.

Nous pouvons observer sur la figure 3 que la moyenne générale (axe noir en gras) est positive sur l'ensemble des affirmations hormis la première, cela est peut-être induit par l'environnement scolaire proposant des plannings stricts et de nombreux modules à explorer. La rétrospective de 2013 du SUS (Brooke, 2013) nous apprend que le score moyen obtenu par des dispositifs au SUS est de 68/100 soit 0.72 sur l'intervalle [–2 ; 2]. Dans notre étude, la moyenne générale a atteint 0.72 : 0.57 pour les enseignants (N = 20) et 0.77 pour les élèves (N = 68).

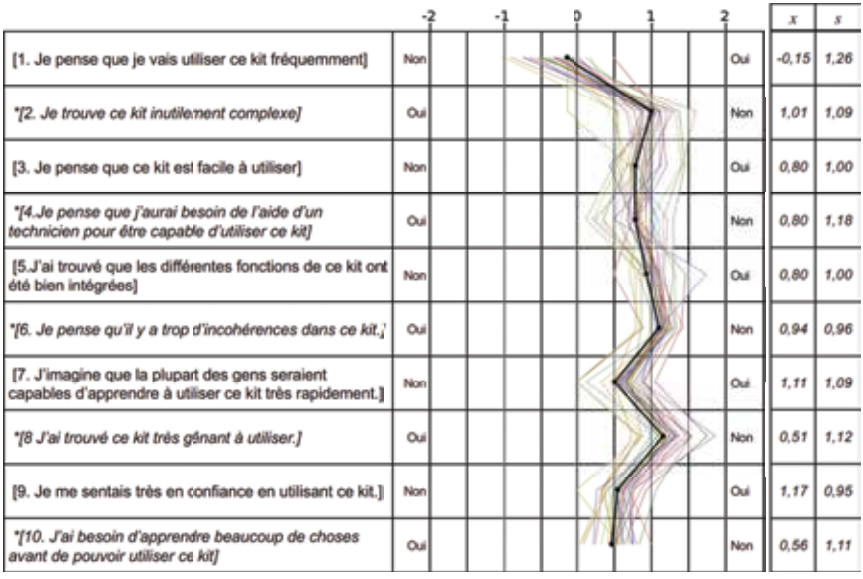


Figure 3 : Diagramme des réponses. L'axe noir, en gras, correspond à la moyenne générale.

Les affirmations 7, 3 et 10 évoquent les questions de prise en main et de capacité d'auto-formation, point que nous souhaitions optimiser. Ici la variabilité observée montre que notre action a eu un impact mais pas sur l'ensemble de la population. Les résultats de l'affirmation 8 montrent une bonne acceptance de la part des utilisateurs.

3.2 Expérience utilisateur : l'AttrakDiff

L'AttrakDiff (Hassenzahl *et al.*, 2003) – possédant une validation en français (Lallemand, Koenig, Gronier & Martin, 2015) – est un questionnaire permettant d'évaluer l'expérience utilisateur suivant 4 échelles : Échelle de Qualité Pragmatique ; Échelle de Qualité Hédonique – Stimulation ; Échelle de Qualité Hédonique – Identité ; Échelle d'Attractivité globale. La première estime le niveau de difficulté perçu par l'utilisateur. La deuxième estime si le dispositif offre une expérience stimulante et novatrice pour l'utilisateur. La troisième estime l'intégration sociale perçue avec l'utilisation du dispositif. La quatrième estime les mêmes

aspects mais d'un point de vue plus global. Pour obtenir ces scores, l'utilisateur est invité à noter 28 paires d'antonymes suivant une échelle à 7 points.

Nous pouvons observer sur la figure 4 le résultat pour nos 29 modalités pour les 28 paires de mots, ici ordonnées par catégorie.

Sur la figure 4, nous pouvons voir qu'une majorité de réponses sont positives, et que plusieurs paires de mots semblent se distinguer, notamment : l'évaluation de *Technique / Humain* en moyenne à $\overline{x} = -0.70$ ($S = 1.38$) et l'évaluation de *Amateur / Professionnel* à $\overline{x} = -0.15$ ($S = 1.45$) quand l'évaluation générale se situe à $\overline{x} = 1.20$ ($S = 0.61$). Dans une moindre mesure, nous observons la même distinction au niveau de *Prévisible / Imprévisible* $\overline{x} = 0.36$ ($S = 1.31$) qui peut s'expliquer par la mise en avant dans les activités de la démarche d'apprentissage par essai-erreur. De la même façon on observe que les termes *Original – Créatif – Plaisant* obtiennent une meilleure évaluation que la moyenne, respectivement : $\overline{x} = 1.73$; 1.75 ; 1.88 ($S = 1.24$; 1.14 ; 1.15).

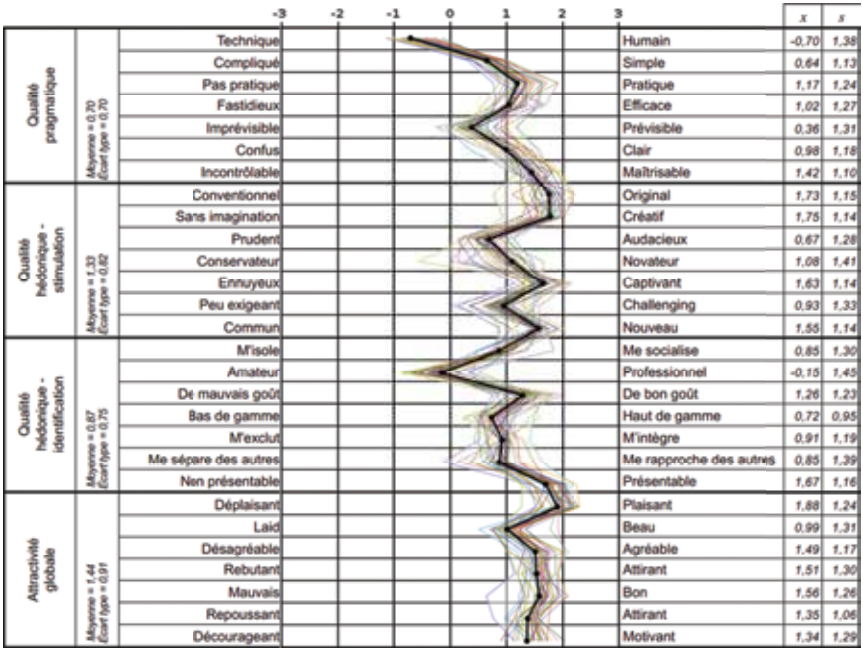


Figure 4 : Diagramme des réponses. L'axe noir, en gras, correspond à la moyenne générale.

Au niveau des 4 échelles, nous observons que l'aspect stimulation $x = 1.33$ ($S = 0.70$) et l'attractivité globale $x = 1.44$ ($S = 0.91$) obtiennent de meilleurs résultats que l'aspect identification $x = 0.87$ ($S = 0.75$) et pragmatique $x = 0.70$ ($S = 0.70$). La moyenne de ces différentes échelles donne un score global de 1.087 ($S = 0.58$) pour l'ensemble de l'échantillon ($N = 88$) ; de 1.057 ($S = 0.58$) pour les enseignants ($N = 20$) ; et de 1.182 ($S = 0.61$) pour les élèves ($N = 68$).

3.3 Interprétation globale

Ces deux questionnaires – standardisés – nous ont permis, d'une part, de rassembler un certain nombre d'éléments quantifiables afin de mieux évaluer les choix et stratégies de conception pris pour le dispositif Poppy Éducation et ce principalement sur le kit Ergo Jr. Cela nous permet de mieux appréhender la pertinence des solutions proposées en fonction des différents usages et perceptions qu'ont les utilisateurs. D'autre part, utiliser ces questionnaires nous permet de proposer une « photographie » des caractéristiques d'utilisabilité perçues par l'utilisateur selon différents critères. Ceci offre la possibilité de comparaison future avec d'autres dispositifs de robotique pédagogique, permettant de mieux classifier ces outils.

4 Limites & perspectives

De nouveaux enseignants souhaitant utiliser le kit Ergo Jr ont été identifiés, cela nous permettra d'effectuer de nouvelles passations des questionnaires présentés ici et ainsi de comparer les résultats à plus large échelle. De plus, les enseignants ayant participé à la phase d'évaluation ont également participé à la phase de conception, et sont tous de la région Nouvelle Aquitaine ; il est donc difficile de généraliser. À noter également qu'il faut distinguer les effets induits par « les nouvelles technologies » et « les sciences du numérique » : Les ordinateurs installés dans les établissements scolaires dans les années 1990 ne produisent plus le même attrait chez les étudiants. Multiplier les essais pour limiter l'effet de nouveauté et élargir la zone géographie représente la prochaine étape

vers la généralisation de nos résultats. Dans ce même objectif a été mis en place un protocole de suivi de cohorte qui sera activé à la rentrée de septembre 2017. Ce protocole nous permettra une analyse longitudinale afin d'observer l'impact de kits robotiques dans les apprentissages et les représentations qu'ont les élèves de la pensée informatique. Enfin, d'autres résultats restent à analyser ou compléter, il est difficile d'aller plus loin aujourd'hui dans leur interprétation sans de nouvelles données comme par exemple avec le score obtenu au SUS par les enseignants (0.57) plus faible que celui traditionnellement relevé (0.72).

5 Conclusion

Le dispositif Poppy Éducation, à travers le kit robotique Ergo Jr, a connu un retour plutôt positif de la part de la communauté scolaire. L'équipe Poppy Éducation a pu observer ces retours et les pratiques qui y ont mené, et cela tant qualitativement que quantitativement. En effet, nous venons de voir que la plateforme Poppy, notamment le robot Ergo Jr, pouvait avoir une grande variété d'usages et que ceux-ci modifiaient significativement la perception qu'en ont les utilisateurs. Cependant, certains éléments du kit, atouts comme faiblesses, font l'unanimité chez ces utilisateurs, dressant ainsi une carte d'identité du kit. Cette photographie vient en complément de la liste des caractéristiques techniques, pour contribuer à une meilleure visibilité des différents dispositifs proposés aux enseignants dans le cadre de l'apprentissage des sciences du numérique. Cette évaluation permet également de poser les fondations d'études plus complexes, notamment sur les apprentissages réalisés par les élèves, leur motivation et leur engagement dans les activités, l'évolution de leur perception des robots, de l'intelligence artificielle, et plus généralement leur appropriation de la pensée informatique.

Références

Abras, C., Maloney-Krichmar, D. & Preece, J. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks : Sage Publications*, *37*(4), 445–456.

Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7. Brooke, J. (2013). Sus : a retrospective. *Journal of usability studies*, *8*(2), 29–40.

Hassenzahl, M., Burmester, M. & Koller, F. (2003). AttrakDiff : Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In *Mensch & computer 2003* (pp. 187–196). Springer.

Lallemand, C., Koenig, V., Gronier, G. & Martin, R. (2015). Création et validation d'une version française du questionnaire AttrakDiff pour l'évaluation de l'expérience utilisateur des systèmes interactifs. *Revue Européenne de Psychologie Appliquée/European Review of Applied Psychology*, *65*(5), 239–252.

Lapeyre, M. (2015). *Poppy : plate-forme robotique open source, imprimée en 3d et totalement modulaire pour l'experimentation scientifique, artistique et pédagogique* (Thèse de doctorat non publiée). Bordeaux.

Noirpoudre, S., Roy, D., Demangeat, M., Desprez, T., Segonds, T., Rouanet, P., Caselli, D., Rabault, N., Lapeyre, M. & Oudeyer, P.-Y. (2016). Livret pédagogique : Apprendre à programmer Poppy Ergo Jr en Snap.

Noirpoudre, S., Roy, D., Desprez, T., Segonds, T., Caselli, D. & Oudeyer, P.-Y. (2017). Poppy education : un dispositif robotique open source pour l'enseignement de l'informatique et de la robotique. In *EIAH 2017, Environnements Informatiques pour l'Apprentissage Humain.* Strasbourg, France.

Roy, D., & Oudeyer, P.-Y. (2016). IniRobot et Poppy Éducation : deux kits robotiques libres pour l'enseignement de l'informatique et de la robotique. In *Colloque Didapro-Didastic 6e édition.* Namur, Belgique.

Thierry Karsenti & Julien Bugmann

Université de Montréal
thierry.karsenti@umontreal.ca, julien.bugmann@umontreal.ca

Un robot humanoïde pour enseigner la programmation : une recherche exploratoire auprès d'élèves ayant des difficultés d'apprentissage

Résumé
Les robots sont de plus en plus présents dans notre société. Il se dit même qu'ils vont devenir « indispensables » à notre quotidien. Au cours des dernières années, on les voit même de plus en plus dans des salles de classe, notamment à des fins d'apprentissage de la programmation. En effet, et même si la présence de robots à l'école remonte à plusieurs dizaines d'années, cette tendance tendrait à s'accroître avec l'émergence de robots dits « éducatifs ». Notre recherche vise à mieux comprendre le rôle d'un robot humanoïde dans l'apprentissage de la programmation pour des élèves ayant des difficultés d'apprentissage, de même que cet usage pour la motivation des élèves dans l'apprentissage de la programmation. Pour atteindre ces objectifs, nous avons mis en place un protocole particulier où un robot humanoïde a été intégré aux cours de programmation d'élèves ayant des difficultés d'apprentissage. Les données recueillies révèlent qu'un robot humanoïde peut participer à l'enseignement-apprentissage de la programmation, même pour des élèves ayant des difficultés d'apprentissage importantes à l'école. Nos données illustrent également les différents impacts sur les élèves du protocole de recherche mis en place.

Mots clés : robotique, robots humanoïdes, programmation, codage, école, apprentissages

1 Introduction

Même si l'apprentissage du code ou de la programmation existe depuis de nombreuses années dans les écoles (voir Papert, 1981), cette pratique a récemment connu un essor important avec des initiatives comme « Hour

of Code » (*une heure de code*) à laquelle plus de 450 millions d'apprenants ont déjà participé[1]. Par ailleurs, et pour être en accord avec les compétences du XXI[e] siècle que tous les élèves devraient se préparer à maîtriser (van Laar, van Deursen, van Dijk et de Haan, 2017), plusieurs pays offrent un enseignement obligatoire du code. C'est le cas notamment pour l'Angleterre, certaines provinces du Canada, certains États des États-Unis, la France, la Grèce, la Norvège et la Suède. Pourquoi de telles initiatives dans ces pays ? Parce qu'il est important que les élèves apprennent à coder afin qu'ils comprennent davantage le monde qui les entoure et afin qu'ils soient mieux préparés au monde technologique dans lequel ils évolueront demain (Karsenti et Bugmann, 2017a). Autrement dit, pour éviter de « subir le code », il semble important que les élèves puissent le maîtriser (voir OCDE, 2015). Pour ce faire, de nombreux logiciels existent désormais afin de participer à cet apprentissage, tant dans la salle de classe qu'en dehors de l'école[2]. En plus des logiciels, plusieurs robots participent également à l'apprentissage de la programmation en permettant notamment une concrétisation physique des programmes générés par les élèves, à l'inverse des logiciels « classiques » tels que *Scratch* ou *Swift Playgrounds* qui génèrent un scénario visuel sur écran (voir Karsenti et Bugmann, 2017b). Alors que le phénomène des « robots » en éducation n'est pas nouveau[3], on constate, depuis quelques années, une forte multiplication de ces technologies en classe, au point que l'on retrouve aujourd'hui, dans le commerce, plus d'une vingtaine de robots destinés à la salle de classe (voir Karsenti et Bugmann, 2017a). Et lorsque l'on sait l'importance de l'apprentissage du code et les nouvelles compétences à maîtriser dans les années à venir par les élèves (Fonction publique de l'Ontario, 2016 ; van Laar *et al.*, 2017), il n'est pas étonnant de voir se multiplier ce type d'outils dans les établissements scolaires. L'usage des robots revêt un intérêt particulier pour l'apprentissage du code, notamment parce qu'il participe de façon très particulière à la motivation des élèves (voir section 3).

1 <https://hourofcode.com/fr>

2 Pour une liste des principaux logiciels utilisés, voir : <http://karsenti.ca/11coder.pdf>

3 Par exemple, Seymour Papert, dans les années 1980 avait lui aussi lancé la mode de la programmation à l'école en développant sa tortue LOGO (voir Papert, 1981).

2 Contexte théorique

Plusieurs études ont montré qu'apprendre la programmation comporte de multiples avantages, par exemple l'apprentissage des mathématiques (Temperman *et al.*, 2014), les habiletés en résolution de problèmes (voir Keane, Chalmers, Williams et Boden, 2016) et la créativité (Falloon, 2016 ; Moreno-León, Robles et Román-González, 2016 ; OECD, 2015 ; Romero, Lille et Patiño, 2017 ; Smith, Sutcliffe et Sandvik, 2014). Pour accroître la motivation des élèves dans l'apprentissage de la programmation, plusieurs chercheurs se sont penchés sur l'usage des robots (voir Alimisis, 2012 ; Gaudiello et Zibetti, 2013 ; Keane *et al.*, 2016 ; Komis et Misirli, 2016) et ont démontré leur potentiel motivationnel auprès des apprenants (Greff et Melgarejo, 2017 ; Janiszek, Boulc'H, Pellier, Mauclair et Baron, 2011 ; Kaloti-Hallak, Armoni et Ben-Ari, 2015). Dans un contexte où l'usage de robots pour l'apprentissage du code est susceptible de participer à la motivation des élèves, certains chercheurs se sont plus particulièrement intéressés aux robots humanoïdes dont le potentiel pour l'apprentissage du code et la motivation à apprendre à coder semble encore plus important (voir Keane *et al.*, 2016). Pourquoi un robot humanoïde ? Tout simplement parce que ce robot peut accroître, encore plus, la motivation des élèves, par rapport à l'usage d'un autre type de robot. Ce type de robot est, par ailleurs, rarement utilisé pour l'apprentissage du code. En effet, il est plus fréquemment utilisé auprès d'enfants atteints de troubles du spectre de l'autisme (Caudrelier et Foerster, 2015 ; Centelles, Assaiante, Etchegoyhen, Bouvard et Schmitz, 2012).

3 Objectif de recherche

Cette étude vise à mieux comprendre le rôle d'un robot humanoïde dans (a) l'apprentissage de la programmation et (b) la motivation pour l'apprentissage de la programmation pour des élèves ayant des difficultés d'apprentissage. Pour atteindre cet objectif, nous avons mis en place un protocole de recherche particulier (voir section 4.2) où un robot

humanoïde a été intégré aux cours de programmation d'élèves ayant des difficultés d'apprentissage. L'originalité de notre recherche est multiple. Elle se situe (1) d'abord dans l'usage des robots pour l'apprentissage du code, voire (2) dans l'usage d'un type particulier de robot pour l'apprentissage du code (un robot humanoïde), mais aussi (3) dans le contexte particulier de cet usage (les élèves identifiés comme ayant des difficultés particulières d'apprentissage).

4 Méthodologie

4.1 Participants

Ce sont en tout 83 élèves (48 filles et 35 garçons), âgés de 11 à 17 ans, qui ont participé à cette recherche. L'âge moyen des participants était de 12,5 ans ; 64 élèves étaient du niveau primaire et 19 du secondaire. Ces élèves avaient tous été identifiés comme ayant des difficultés particulières d'apprentissage. Dans le contexte scolaire canadien, ils sont qualifiés d'élèves en « adaptation scolaire »[4].

4.2 Contexte et dispositif de la recherche

Dans le cadre de cette recherche, nous avons choisi d'utiliser le robot humanoïde NAO[5] dans deux contextes d'enseignement au Québec (Canada) où l'on retrouvait des élèves identifiés comme ayant des difficultés particulières d'apprentissage. Nous avons donc proposé aux élèves un programme d'apprentissage de la programmation à l'aide du robot humanoïde NAO. Ce programme, que nous avons nommé « Maître NAO » (voir Figure 1), comportait dix niveaux progressifs à réaliser, composés chacun de trois étapes intermédiaires (donc, 30 étapes en tout). Pour programmer

4 Ministère de l'Éducation, du Loisir et du Sport (2006). *Programme de formation de l'école québécoise*. Gouvernement du Québec.

5 <https://www.ald.softbankrobotics.com/en/robots/nao>

le robot NAO, les élèves avaient à utiliser le logiciel *Choregraphe*, un logiciel de programmation qui, à l'aide de boîtes de commandes à *glisser-déplacer,* leur permettait de contrôler le robot. Ce logiciel, qui peut toutefois s'avérer particulièrement complexe, offre la possibilité aux utilisateurs de voir le programme réalisé sur un robot virtuel et, donc, de travailler sur les tâches de programmation même lorsque le robot NAO n'était pas avec eux. Les élèves utilisant le logiciel *Choregraphe* peuvent donc s'initier à la programmation par la manipulation et l'organisation de boîtes de commandes, comme celles présentes, notamment, lors de l'utilisation de logiciels tels que *Scratch*. À titre d'exemple, avec *Choregraphe*, l'utilisateur crée un programme en liant des boîtes de commandes entre elles et en leur attribuant un certain nombre de caractéristiques. Par exemple, si un élève souhaite programmer NAO pour qu'il s'assoie, qu'il parle, puis qu'il se relève, la séquence illustrée à la Figure 2 serait à réaliser. Le logiciel *Choregraphe* permet aussi de visualiser rapidement les différentes commandes qui peuvent être appliquées au robot. Tout d'abord, le fait qu'il doive s'assoir par l'application de la boîte « *Sit down* », puis le fait qu'il parle avec la boîte « *Say* » et enfin qu'il se lève avec la boîte « *Stand Up* ». Aussi, en rentrant dans les paramètres dans la boîte de programmation « *Say* », l'utilisateur va pouvoir préciser ce qu'il souhaite que le robot dise (Figure 3). Par l'articulation cohérente de plusieurs boîtes de commandes à ordonner et à paramétrer, les élèves peuvent donc programmer le robot NAO et rendre plus concrète leur création. Enfin, il semble important de noter que peu d'études ont porté sur la programmation du robot NAO par des élèves du secondaire, voire encore moins pour des élèves du primaire. En effet, la plupart des études sur la programmation du robot NAO ont été réalisées par des étudiants universitaires (voir Nijimbere, 2014).

Figure 1 : Illustration du programme « Maître NAO »
mis en place auprès des participants à l'étude.

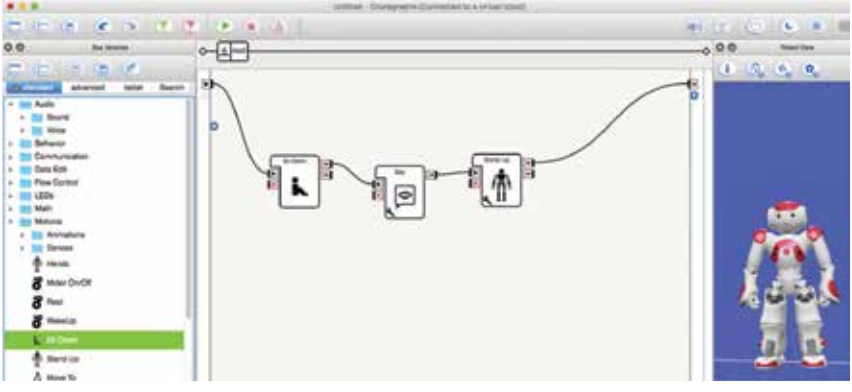


Figure 2 : Utilisation du logiciel de programmation *Choregraphe*
pour programmer le robot NAO.

Figure 3 : Contenu de la boîte « *Say* » et texte que devra dire le robot.

4.3 Instruments de collecte de données

Dans le cadre de cette recherche exploratoire et afin d'atteindre notre objectif de recherche, nous avons eu recours à plusieurs outils de collecte des données. Il semble toutefois important de rappeler la particularité de nos participants qui avaient tous des difficultés particulières d'apprentissage. Cela limitait, en soi, l'usage de certains outils comme des questionnaires. Par exemple, plusieurs études ont montré qu'il est particulièrement difficile d'exploiter des échelles psychométriques auprès de tels élèves (voir Avramidis, Strogilos, Aroni et Kantaraki, 2017). Nous avons donc principalement utilisé quatre instruments de collecte de données : (a) des observations vidéographiées des séances d'apprentissage de la programmation avec le robot humanoïde NAO (11 séances de 75 minutes) ; (b) des entretiens de groupe avec les élèves (n = 14) ; (c) de brefs entretiens individuels avec les élèves (n = 43) ; (d) des captures vidéos du travail de programmation réalisé par les élèves sur le logiciel *Choregraphe* avec lequel le robot humanoïde NAO fonctionne ; (e) le suivi des niveaux réalisés par les élèves (Maître NAO).

4.4 Stratégies d'analyse des données recueillies

Les observations des séances de programmation, les entretiens de groupe et les entretiens individuels ont permis de générer des données qualitatives (verbatim des élèves). L'analyse de ces verbatim a été effectuée à l'aide du

logiciel QDA Miner 5[6]. Elle a consisté en une analyse de contenu (voir L'Écuyer, 1990 ; Miles et Huberman, 2003) dont le codage semi-ouvert a été construit à partir des réponses des participants en lien avec les principaux objets de recherche (apprentissage de la programmation ; motivation pour l'apprentissage de la programmation).

5 Résultats

Les données recueillies dans le cadre de cette recherche exploratoire ont d'abord permis de constater que les 83 élèves ayant des difficultés d'apprentissage qui ont participé à cette recherche ont tous été capables de programmer le robot humanoïde NAO. En effet, l'analyse des niveaux atteints par les élèves révèle qu'après seulement 75 minutes (une séance), les élèves du secondaire avaient, en moyenne, atteint le niveau 4 (Figure 4), tandis que les élèves du primaire avaient atteint en moyenne le niveau 5 (Figure 5). Ils étaient ainsi capables d'interagir verbalement et physiquement avec le robot (Niveau 1), de trouver et d'activer une commande dans le logiciel de programmation *Choregraphe* (Niveau 2), de comprendre le fonctionnement du logiciel et de manipuler le robot virtuel (Niveau 3), d'effectuer de la programmation procédurale en créant une séquence de commandes (Niveau 4), mais aussi d'apprendre à effectuer une programmation séquentielle en créant une séquence de commandes intégrant des évènements tels que la réponse d'un utilisateur à une question (Niveau 5). Par ailleurs, les élèves ont effectué, tout au long du dispositif, du « debugging » (Komis et Misirli, 2013) pour comprendre ce en quoi le programme effectué ne fonctionnait pas adéquatement.

Au primaire, on constate que certains élèves ont même été capables d'atteindre le niveau 8 dès la séance 1 (voir Figure 5). Cette différence entre les deux groupes s'explique notamment par l'expérience en programmation des élèves du primaire. En effet, tous avaient déjà réalisé plus de 5 séances d'une heure de programmation à l'aide du logiciel *Scratch*[7], l'un des plus populaires dans les écoles primaires en Amérique du Nord.

6 <http://provalisresearch.com/fr/produits/logiciel-d-analyse-qualitative/>
7 <https://scratch.mit.edu/>

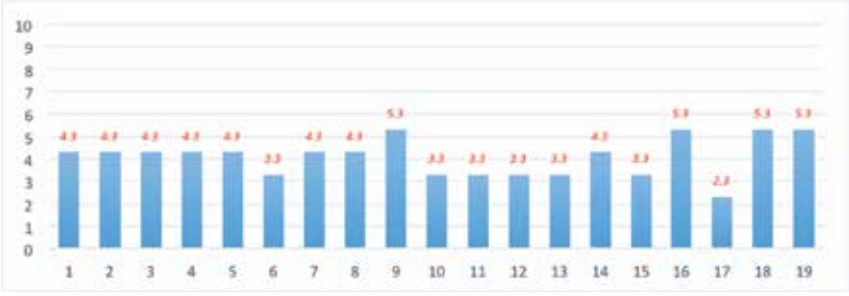


Figure 4 : Niveaux atteints (sur 10.3, soit 10 niveaux et 3 étapes) par les élèves du secondaire à l'issue de la première séance avec le robot NAO.

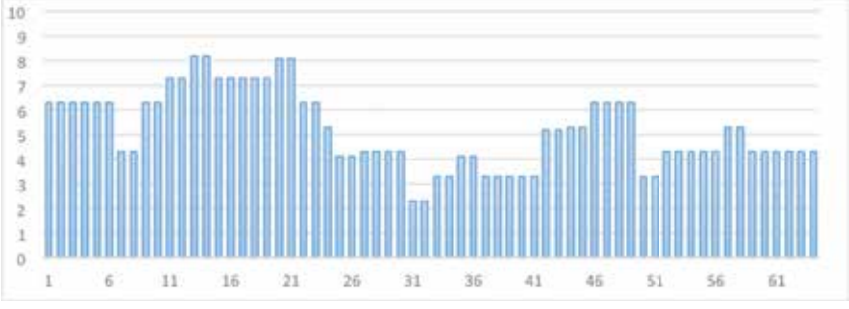


Figure 5 : Niveaux atteints (sur 10.3) par les élèves du primaire à l'issue de la première séance avec le robot NAO.

La Figure 6 révèle, quant à elle, le score atteint par les élèves du secondaire à la fin des séances de programmation du robot humanoïde NAO. Rappelons que nous avions proposé 10 niveaux, composés chacun de 3 tâches, à effectuer. Le score maximal pouvant être atteint était donc de 10.3 (les 3 tâches du niveau 10). Les données obtenues montrent que plus de 50 % des élèves ont complété le niveau 10, et que seulement 5 élèves n'ont pas réussi à atteindre le niveau 9 (élèves 3, 7, 12, 15 et 18).

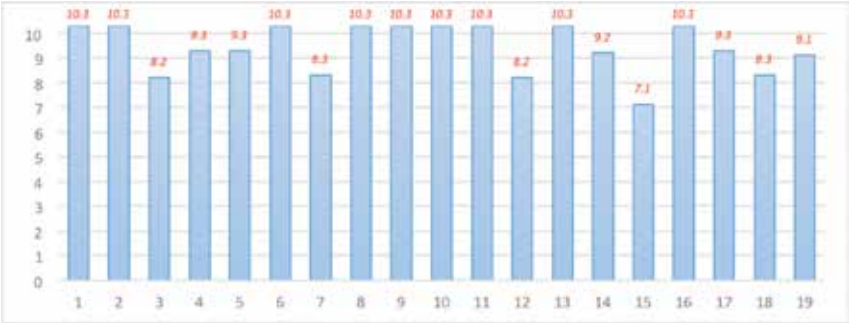


Figure 6 : Niveaux atteints (sur 10.3) par les élèves du secondaire à l'issue de la dernière séance avec le robot NAO.

Les données recueillies lors des observations vidéographiées et des entrevues individuelles et de groupe témoignent également de l'impact de ce dispositif sur l'apprentissage de la programmation par les élèves. Par exemple, plusieurs élèves (74/83) ont affirmé, lors des observations vidéographiées et des entrevues individuelles et de groupe réalisées après deux séances de programmation seulement, être en mesure de programmer le robot humanoïde NAO, voire posséder « beaucoup plus » de « connaissances » permettant de faire se déplacer le robot, de le faire interagir, d'utiliser des conditions, etc. Enfin, les observations vidéographiées et les entrevues individuelles et de groupe réalisées ont également permis de mieux comprendre dans quelle mesure ce dispositif a aussi permis de stimuler le sentiment de compétence chez de nombreux élèves (72/83) qui rencontraient bon nombre de difficultés dans l'apprentissage de matières plus classiques à l'école (le français, les mathématiques), mais qui excellaient dans la programmation du robot NAO. Il est cependant à noter que les élèves ont parfois eu de la difficulté avec la manipulation physique du robot (niveau 10 en mode *Animation*) qui requiert, pour être accomplie, une perception aiguë de l'équilibre du robot, tout en assurant une bonne coordination à l'ensemble des mouvements programmés dans le logiciel *Choregraphe*. Par ailleurs, les données recueillies illustrent également que les élèves ont particulièrement apprécié les séances avec le robot humanoïde NAO et que ce dispositif a permis d'accroître leur motivation pour l'apprentissage du code. C'est ce que soulignent 79 des 83 élèves lors des observations vidéographiées et des entrevues individuelles et de groupe réalisées : « *j'ai aimé ça programmer*

NAO », « *j'ai aimé ça parce que […]* », « *j'ai aimé le programmer pour qu'il danse* », « *[…] est le fun, c'est d'apprendre à le programmer, […]* »…

6 Discussion

L'une des principales forces de la présente étude réside assurément dans la méthodologie de recherche particulière utilisée. En effet, l'usage de différentes méthodes de collecte de données semble être, en soi, un avantage majeur pour enrichir et trianguler les résultats obtenus. Les choix méthodologiques effectués ne sont pourtant pas sans limites. Tout d'abord, le fait de travailler à partir des perceptions peut constituer une limite que nous avons tenté de pallier par plusieurs méthodes de collecte de données, y compris les niveaux atteints par les apprenants. De plus, pour réduire ce biais méthodologique, les analyses effectuées ont systématiquement comparé les réponses des différents types de répondants, mettant en exergue leurs points de divergence lorsque nécessaire. Une autre limite de l'étude est liée à l'échantillon des participants qui n'était pas aléatoire, c'est-à-dire que notre choix des participants n'avait pas pour objectif de représenter un sous-ensemble de la population interrogée. Nous avons plutôt misé sur un échantillon de convenance, soit un échantillon non probabiliste qui n'aspire pas à être représentatif, mais simplement à utiliser les répondants disponibles ou volontaires et aisément interrogeables.

Cette recherche, menée dans deux écoles du Québec, a montré qu'il était possible d'amener des élèves vers de multiples apprentissages et bénéfices à l'aide d'un robot humanoïde de type NAO. En effet, et alors que les initiatives se multiplient pour enseigner la robotique et la programmation à l'aide de robots tels que *Dash*, *Bee-Bot* ou *Ozobot*, il apparaît que l'utilisation d'un robot d'un nouveau genre, plus proche de l'être humain, mais peut-être aussi plus compliqué à manipuler ou à comprendre, pourrait avoir un impact encore plus fort sur des élèves de primaire et de secondaire ayant des difficultés d'apprentissage. On constate même que les élèves de primaire se sont avérés particulièrement performants dans ce projet, et ce, au bout d'une seule séance seulement de programmation du robot humanoïde NAO. Certes, certains défis subsistent, tels que les difficultés techniques qui peuvent être rencontrées ou encore le nécessaire appui de la

direction scolaire à obtenir, mais comme nous l'avons montré, l'utilisation de ce robot, via notre programme d'initiation appelé « *Devenez un Maître NAO* » a mis en évidence de nombreux bénéfices pour les apprenants. Ils ont ainsi connu, par exemple, une forte augmentation de la motivation à se rendre en classe, des bénéfices en termes de collaboration, d'entraide, et cela a même eu un impact positif sur leur réussite scolaire en général par une forte valorisation sociale de leur quotidien scolaire. En effet, certains élèves n'ont pas hésité à publier fièrement leurs réalisations et l'avancée de leur programmation sur les réseaux sociaux. Enfin, il semble important de faire remarquer que les données recueillies révèlent également que ce dispositif a eu plusieurs autres impacts chez les élèves, qui ne sont toutefois pas présentés dans ce texte, comme le développement de leur capacité à collaborer entre eux dans la réalisation d'une même tâche ou encore leur capacité à s'entraider dans la réalisation de tâches différentes, voire dans le développement de la créativité des élèves, grâce notamment au degré de liberté qui leur était permis dans la réalisation de certaines tâches de programmation (par exemple, au Niveau 6). Cela sera discuté dans de futures publications. Finalement, face au succès rencontré par ce dispositif dans les écoles visitées, nous poursuivons aujourd'hui le projet avec de nouveaux niveaux, plus complexes, et destinés à amener les élèves à devenir des « *NAO PRO* ». Reste aujourd'hui à interroger et définir la place, et la forme, que pourrait prendre un tel dispositif dans les curriculums officiels…

Références

Alimisis, D. (2012). Integrating robotics in science and technology teacher training curriculum. Dans *Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics Integrating Robotics in School Curriculum*, Riva del Garda (Trento, Italy) (pp. 170–179). Repéré à <http://www.terecop.eu/TRTWR2012/trtwr2012_submission_28.pdf>.

Avramidis, E., Strogilos, V., Aroni, K. et Kantaraki, C. T. (2017). Using sociometric techniques to assess the social impacts of inclusion : Some methodological considerations. *Educational Research Review*, 20, 68–80. <http://dx.doi.org/10.1016/j.edurev.2016.11.004>.

Caudrelier, T. et Foerster, F. (2015). Contribution des robots sociaux aux thérapies des troubles du spectre autistique : une revue critique. État de l'art : avantages et challenges en robotique sociale. Dans *Actes du TER « Cognition, affects et interaction »* (p. 25–32) Repéré à <https://hal.archives-ouvertes.fr/cel-01110281/file/TER2015.pdf>.

Centelles, L., Assaiante, C., Etchegoyhen, K., Bouvard, M. et Schmitz, C. (2012). Understanding social interaction in children with autism spectrum disorders : does whole-body motion mean anything to them ?. *L'Encéphale*, 38(3), 232–240. <http://dx.doi.org/10.1016/j.encep.2011.08.005>.

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. on the iPad. *Journal of Computer Assisted Learning*, 32(6), 576–593. <http://dx.doi.org/10.1111/jcal.12155>.

Fonction publique de l'Ontario. (2016). *Définir les compétences du 21e siècle pour l'Ontario. Compétences du 21e siècle. Document de réflexion*. Repéré à <https://pedagogienumeriqueenaction.cforp.ca/wp-content/uploads/2016/02/Ontario-21st-century-competencies-foundation-FINAL-FR_AODA_EDUGAINS_Feb-19_16.pdf>.

Gaudiello, I. et Zibetti, E. (2013). La robotique éducationnelle : état des lieux et perspectives. *Psychologie française*, 58(1), 17–40. <http://dx.doi.org/10.1016/j.psfr.2012.09.006>.

Greff, É. et Melgarejo, B. (2017). Les mini-drones : de nouveaux outils pour s'initier à la programmation informatique. *La nouvelle revue de l'adaptation et de la scolarisation*, (78), 239–254.

Janiszek, D., Boulc'H, L., Pellier, D., Mauclair, J. et Baron, G.-L. (2011). De l'usage de Nao (robot humanoïde) dans l'apprentissage de l'informatique. Dans G.-L. Baron, É. Bruillard et V. Komis (dir.), *Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques* (p. 231–239). Athènes : New Technologies Editions.

Kaloti-Hallak, F., Armoni, M. et Ben-Ari, M. M. (2015). Students' attitudes and motivation during robotics activities. Dans *Proceedings of the Workshop in Primary and Secondary Computing Education* (p. 102–110). ACM. <http://dx.doi.org/10.1145/2818314.2818317>.

Keane, T., Chalmers, C., Williams, M. et Boden, M. (2016). The impact of humanoid robots on students' computational thinking. Dans

Australian Conference on Computers in Education (ACCE 2016), Brisbane, Australia (Vol. 29).

Karsenti, T. et Bugmann, J. (2017a). Faut-il apprendre à coder à l'école ?. *École branchée*, 19(3), 32–35.

Karsenti, T. et Bugmann, J. (2017b). Les écoles canadiennes à l'heure du code ? *Éducation Canada*, 57(1), 14–19.

Komis, V., et Misirli, A. (2013). Étude des processus de construction d'algorithmes et de programmes par les petits enfants à l'aide de jouets programmables. In *Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif*.

Komis, V., et Misirli, A. (2016). The environments of educational robotics in Early Childhood Education : towards a didactical analysis. *Educational Journal of the University of Patras* UNESCO Chair.

L'Écuyer, R. (1990). *Méthodologie de l'analyse développementale de contenu. Méthode GPS et concept de soi*. Sainte-Foy, QC : Presses de l'Université du Québec.

Miles, M. B. et Huberman, A. M. (2003). *Analyse des données qualitatives*. Bruxelles : De Boeck Supérieur.

Moreno-León, J., Robles, G. et Román-González, M. (2016). Code to learn : Where does it belong in the K-12 curriculum ?. *Journal of Information Technology Education : Research*, 15, 283–303. <http://dx.doi.org/10.28945/3521>.

Nijimbere, C. (2014). Apprendre l'informatique par la programmation des robots : cas de Nao. *Sticef*, 21, 63–109. Repéré à <http://sticef.univ-lemans.fr/num/vol2014/09-nijimbere/sticef_2014_nijimbere_09p.pdf>.

OECD. (2015a). Schooling redesigned : Towards innovative learning systems. <http://dx.doi.org/10.1787/9789264245914-en>

OECD. (2015b). Students, computers and learning : Making the connection. PISA : OECD Publishing. <http://dx.doi.org/10.1787/9789264239555-en>.

Papert, S. (1981). *Jaillissement de l'esprit scientifique : ordinateurs et apprentissage*. Paris : Flammarion.

Romero, M., Lille, B. et Patiño, A. (2017). *Usages créatifs du numérique pour l'apprentissage au XXI^e siècle*. Presses de l'Université du Québec.

Smith, N., Sutcliffe, C. et Sandvik, L. (2014). Code club : Bringing programming to UK primary schools through Scratch. Dans *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (p. 517–522). New York, NY : ACM. <http://dx.doi.org/10.1145/2538862.2538919>.

van Laar, E., van Deursen, A. J. A. M., van Dijk, J. A. G. M. et de Haan, J. (2017). The relation between 21st-century skills and digital skills : A systematic literature review. *Computers in Human Behavior*, 72, 577–588. <http://dx.doi.org/10.1016/j.chb.2017.03.010>.

Patrice Frison[a,b], Moncef Daoud[b] & Michel Adam[b]

a. Laboratoire IRISA
b. Université de Bretagne Sud
patrice.frison@univ-ubs.fr, moncef.daoud@univ-ubs.fr,
michel.adam@univ-ubs.fr

Transition didactique de l'activité débranchée à la programmation avec AlgoTouch

Résumé

L'apprentissage de la programmation peut se faire de différentes manières. On distingue habituellement les activités débranchées de l'activité branchée de programmation. Cependant le passage à l'activité de programmation est difficile. De nombreux obstacles d'ordre technique sont à surmonter : l'apprentissage d'un langage de programmation, la pratique d'un environnement de développement, la compilation, l'exécution et le débogage. Ce papier propose une démarche pour passer progressivement de l'activité débranchée à la programmation en utilisant AlgoTouch, un outil de programmation par démonstration. Cet outil permet de manipuler directement les objets de la programmation (variables, tableaux, indices) pour réaliser un algorithme donné. De plus, il permet d'enregistrer un suite d'opérations et le programme est produit automatiquement. À partir d'une activité débranchée, nous montrons comment transformer progressivement les objets et actions de cette activité en objets numériques et opérations de la machine.

Mots clés : algorithmes, programmation pour débutants, programmation par démonstration, activités débranchées, visualisation d'algorithmes

1 Introduction

L'enseignement de la programmation doit désormais être dispensé depuis l'école primaire. En France, depuis la réforme, la communauté scientifique s'intéresse à l'éducation de l'informatique pour tout cycle d'apprentissage (Baron & Drot-Delange, 2016). Il ne s'agit pas de proposer la programmation

et l'apprentissage d'un langage pour les débutants. L'objectif est de découvrir des concepts et des méthodes spécifiques à la science informatique. L'approche la plus communément admise pour cet apprentissage consiste à développer d'abord des activités débranchées[1]. Elles consistent à réaliser des algorithmes sans utiliser de machines. L'idée étant de distinguer le concept d'algorithme, permettant de résoudre un problème, de celui de programme, exécuté par une machine, associé à un langage de programmation (voir Drot-Delange, 2013, pour une analyse de différentes expériences). Ces concepts (algorithme, machine, langage, information) sont les bases pour l'enseignement de la discipline informatique d'après la SIF (2016).

Cependant le passage à l'activité de programmation est difficile (Guzdial, 2004). De nombreux obstacles sont à surmonter : l'apprentissage d'un langage de programmation, dont la syntaxe est parfois contraignante, la pratique d'un environnement de développement pour l'écriture du code, la compilation, l'exécution et le débogage. Pour surmonter ces difficultés, des environnements particuliers ont été développés pour les débutants dont le plus connu est Scratch[2]. De plus, de nombreux systèmes de visualisation ont été proposés pour animer des algorithmes (voir Sorva, Karavirta & Malmi, 2013, pour une étude bibliographique récente).

Mais, comme le souligne Boisvert (2009), l'écriture d'un programme suit un processus qui est rarement décrit dans les manuels ou dans les cours. En effet, en général, le principe d'un algorithme est donné, puis son fonctionnement est illustré et enfin le code est montré directement. Par contre, l'expérience montre que des apprenants ont souvent le syndrome de la page blanche lorsqu'il s'agit d'écrire le programme dont on vient de décrire le principe de l'algorithme. Comme le notent Hundhausen, Farley et Brown (2009), même avec des environnements spécifiques, « les novices ont des difficultés à construire des boucles et à référencer correctement des éléments de tableau dans ces boucles avec l'usage d'index ».

Il y a donc un vide à combler entre la réalisation d'une activité débranchée et l'écriture du programme associé. Dans cet article, nous proposons de montrer comment passer progressivement d'une activité débranchée à la réalisation d'un programme en utilisant un outil de programmation par démonstration appelé AlgoTouch développé par l'un des auteurs (Frison, 2015). À partir d'une activité débranchée, nous montrons comment transformer progressivement les objets et actions de cette activité en objets

1 <http://csunplugged.org>
2 <http://scratch.mit.edu>

numériques et opérations de la machine. Dans un premier temps, l'apprenant va réaliser l'algorithme sur lequel il travaillait avec les éléments de la programmation (variables, opérateurs, etc.) disponibles avec AlgoTouch. Dans un second temps, ayant constaté que certaines opérations se répètent, il va commencer à enregistrer ces opérations pour pouvoir les répéter automatiquement. AlgoTouch se chargera de créer les parties du programme associé. Ainsi, un algorithme complet pourra être transformé progressivement en programme.

Cet article présente d'abord brièvement le logiciel AlgoTouch. Puis, nous exposerons comment passer de l'activité débranchée à un programme en illustrant par un exemple complet. Enfin, en conclusion, nous présenterons quelques perspectives.

2 Le logiciel AlgoTouch

AlgoTouch utilise la métaphore du tableau noir, c'est-à-dire la manipulation directe sur l'écran des objets de la programmation (variables, tableaux, indices). Par des gestes simples, l'utilisateur peut déplacer ces éléments, glisser-déposer des valeurs dans des cases de la mémoire (variables ou tableaux), augmenter les valeurs d'index, lancer des comparaisons, effectuer les opérations de base (addition, soustraction, multiplication, division). De plus, il permet d'enregistrer une séquence d'actions, de rejouer la séquence enregistrée, et enfin d'achever la construction de l'algorithme (cas de sortie). En fait, lorsque le mode d'enregistrement est activé, un programme est créé et produit au fur et à mesure que l'on « joue » avec les éléments de l'algorithme. Le système gère également les instructions conditionnelles (Frison, 2015).

AlgoTouch est dédié à la conception d'algorithmes manipulant uniquement des entiers ou des caractères. La structure de base d'un algorithme est une simple boucle. Lors de la construction d'un algorithme, le concepteur doit définir quatre parties : l'initialisation des variables, la répétition de la boucle, son corps et l'après-boucle. Il est possible de créer des sous-algorithmes définis sous forme de macro. Ils facilitent ainsi la création d'algorithmes non limités à une seule itération. La structure de boucle a été

définie dans un pseudo-langage inspiré par le langage Eiffel (Meyer, 2009). Cette structure est composée de 4 parties distinctes (figure 1).

```
From
   <Instructions>
Until
   <Conditions de sortie>
Loop
   <Instructions>
Terminate
   <Instructions>
End
```

Figure 1 : Structure d'une boucle.

La partie *From* permet de définir clairement les instructions nécessaires avant l'itération. La partie *Until* définit la ou les conditions de sortie de la boucle. Ces conditions sont placées en séquence sans opérateur. La première condition est évaluée, si elle est vraie, la boucle s'arrête. Sinon, la seconde condition est évaluée et ainsi de suite. La partie *Loop* constitue le cœur de l'itération. Elle est exécutée quand toutes les conditions de sortie sont fausses. Enfin la partie *Terminate* permet de définir les instructions éventuelles à effectuer après l'itération.

Dans la méthode proposée, l'utilisateur crée ces parties dans l'ordre inverse : le corps (partie *Loop*), la poursuite de la boucle (partie *Until*) et enfin l'initialisation (partie *From*) et la terminaison (partie *Terminate*). La raison est de définir progressivement l'algorithme du cas général jusqu'aux détails : à chaque étape, l'outil sera utilisé. L'idée principale derrière la méthode est que l'utilisateur exécute les opérations de la séquence du corps avec des éléments réels de l'algorithme (variables, tableaux) sur un cas typique, puis il rejoue la séquence avec des données différentes (notamment pour traiter des instructions conditionnelles). Il utilise l'outil pour indiquer qu'il faut placer une condition de sortie, et les instructions correspondantes seront produites automatiquement dans la partie *Until*. Pour terminer la construction de l'algorithme, il va définir comment initialiser les variables dans la partie *From*. Ensuite, il peut vérifier que l'algorithme fonctionne correctement en exécutant le programme complet. Dans certains algorithmes, des opérations doivent être exécutées après la sortie de la boucle principale : il enregistre ces instructions qui sont codées dans la partie *Terminate*.

3 Transition du débranché vers la programmation

Dans de nombreuses activités débranchées, l'apprenant doit résoudre un problème par une série de manipulations et en utilisant des règles simples (peser des objets, déplacer des cartes, déplacer des jetons). Dans le cas de la résolution d'un problème par une machine, les « objets » disponibles sont des variables ou des tableaux et les opérations possibles sont limitées : affectation d'une variable, opérations simples (addition, soustraction, multiplication, division, reste de la division), comparaisons de valeurs pour prendre des décisions et répétition d'un ensemble d'opérations. La méthode que nous proposons consiste à transformer progressivement l'activité débranchée en ajoutant des règles permettant de remplacer les manipulations sur les objets de l'activité par des opérations sur les variables d'un algorithme. Pour illustrer notre propos, nous allons traiter un exemple de tri. Un apprenant doit trier, par ordre croissant, des cartes contenant des nombres entiers, en présence d'un tuteur. Dans un premier temps, nous allons décrire plusieurs activités débranchées successives, puis les premières activités branchées avec AlgoTouch, enfin l'automatisation du tri et la création du programme.

3.1 Activités débranchées

Tri de cartes : faces visibles

Le tuteur dispose cinq cartes, faces visibles, devant un apprenant (*cf.* figure 2). Celui-ci doit les ranger par ordre croissant. En fait, en général, l'apprenant effectue ce travail assez rapidement sans vraiment besoin d'aide. Si le tuteur lui demande comment il a procédé, les explications sont assez floues, il peut difficilement décrire l'algorithme qu'il a utilisé.

Tri de cartes : faces cachées

Le tuteur définit maintenant de nouvelles règles. Il dispose les cartes, faces cachées, et il indique à l'apprenant qu'il ne peut consulter que deux cartes à la fois, puis les cacher à nouveau (*cf.* figure 3).

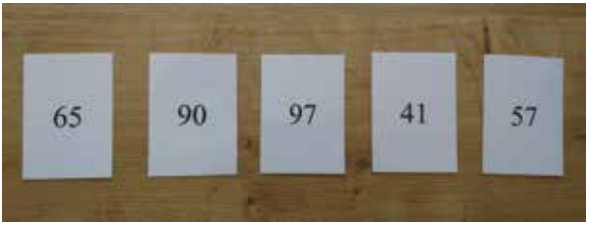


Figure 2 : Cartes à trier : faces visibles.

Figure 3 : Cartes à trier : faces cachées ; seules deux cartes peuvent être consultées à la fois.

Cette fois, l'apprenant est un peu perturbé, il retourne les cartes, deux par deux, au hasard, en tentant de mémoriser leur contenu.

Le tuteur apporte une aide pour organiser les choses en indiquant un principe simple : comparer la première carte avec la deuxième, si la deuxième est plus petite, échanger. Puis comparer, la première carte avec la troisième, échanger si besoin. Recommencer jusqu'à comparer la première carte avec la dernière en échangeant si besoin. Les figures 4 et 5 illustrent la manipulation avec les cartes 1 et 4.

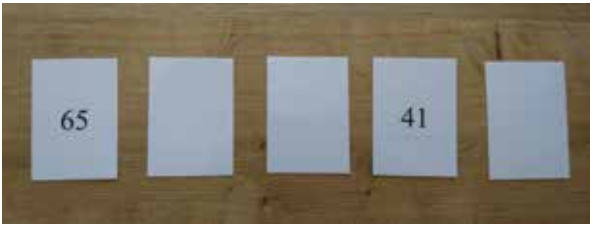


Figure 4 : Comparaison des cartes 1 et 4.

Figure 5 : Échange des cartes 1 et 4.

À la fin de ce passage, la première carte est la plus petite. Il recommence maintenant en comparant la seconde carte avec les suivantes pour placer correctement le second minimum. D'un point de vue pratique, le tuteur propose à l'apprenant de repérer avec l'index de sa main gauche, la position de la carte qui va contenir le minimum et avec l'index de la main droite, la position de la prochaine carte à comparer (*cf.* figure 6). En fait, cette manière de faire s'avère indispensable si le nombre de cartes est plus élevé ! Cette pratique sera mise à profit dans l'activité branchée.

Figure 6 : Itération du processus à partir de la carte 2. Ici, comparaison de la carte 2 avec la carte 4. L'apprenant repère la carte 2 avec l'index de la main gauche et la carte 4 avec l'index de la main droite.

Il recommence ce processus pour placer chaque carte des minimums successifs. L'apprenant comprend vite le principe et effectue les opérations pour trier les cinq cartes (*cf.* figure 7). Il se rend compte que cela prend du temps (notion de *complexité*). À ce stade, le tuteur peut en profiter pour lui demander combien de comparaisons ont été effectuées. La réponse est 4+3+2+1 soit 10 pour cet exemple avec 5 cartes. Le calcul du cas général avec `n` cartes peut être proposé.

Figure 7 : Résultat du tri des cartes.

3.2 Activités branchées

Dans cette étape, il s'agit de passer dans le monde du numérique en utilisant AlgoTouch. Les cartes seront remplacées par des nombres stockés dans des cases de la mémoire. La notion de *variable* est introduite. Le tuteur crée donc maintenant 5 variables (`a,b,c,d,e`) contenant chacune un entier, les mêmes valeurs que dans le cas débranché. Le résultat est affiché sur la figure 8.

Figure 8 : Données à trier dans des variables simples.

Tri de variables par déplacement

Le but du jeu consiste à trier les variables de la plus petite à la plus grande, c'est-à-dire, placer les variables en les ordonnant par ordre croissant de la gauche vers la droite. Pour compliquer le jeu, le tuteur cache le contenu des variables (mode aveugle disponible dans AlgoTouch), comme dans l'activité débranchée.

Comment savoir si le contenu d'une variable est plus petit que celui d'une autre ? Le tuteur montre alors qu'un ordinateur possède un *comparateur* (*cf.* figure 9).

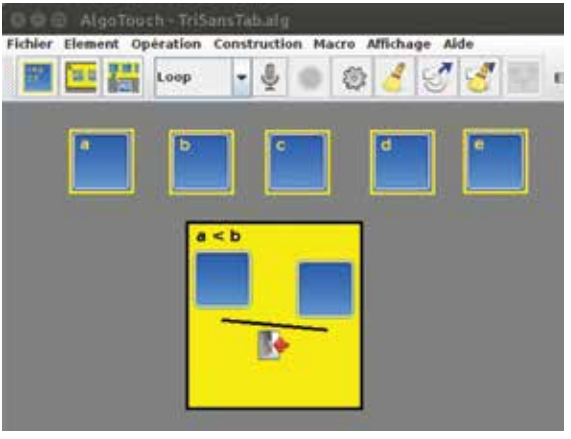


Figure 9 : Comparaison de variables en mode aveugle. La valeur de `a`, placée sur le plateau de gauche, est plus petite que la valeur de `b`, placée sur le plateau de droite.

Avec AlgoTouch, il s'utilise comme une balance : l'utilisateur place sur chaque plateau le contenu à comparer.
AlgoTouch indique si le contenu de la variable `a` est plus petit que celui de la variable `b`, ou plus grand, ou égal.

Disposant de ces nouvelles règles du jeu, l'apprenant comprend qu'il peut utiliser le même algorithme que celui défini pour le tri de cartes faces cachées. Il procède au tri des variables et pour finir, le tuteur rend visible à nouveau le contenu des variables pour vérifier que le résultat est correct (voir figure 10).

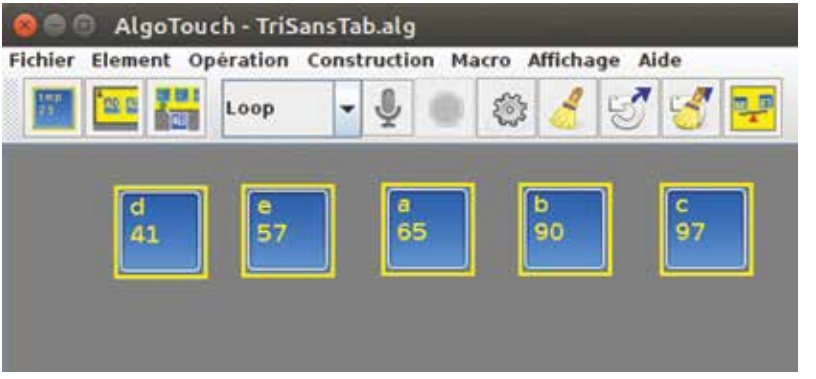


Figure 10 : Les variables sont rangées par ordre de valeur croissante. Remarquer que les variables ont été déplacées visuellement en fonction de leur contenu.

Tri du contenu de variables

À l'issue de l'étape précédente, le tuteur fait remarquer que les variables (`a,b,c,d,e`) ont bien été triées en les replaçant de gauche à droite, par exemple (`d,e,a,b,c`) comme illustré sur la figure 10. Cependant, il serait souhaitable que ce soit le contenu des variables (`a,b,c,d,e`) qui soit réorganisé par ordre croissant. La raison en est assez simple, du point de vue de l'ordinateur, les variables ne peuvent pas être déplacées puisque ce sont des cases à des emplacements (adresses) fixes en mémoire. Par contre, il est possible de modifier le contenu d'une variable.

Le tuteur propose maintenant de revoir l'algorithme de tri des variables en échangeant le contenu des variables sans les déplacer en utilisant une variable intermédiaire, notée `tmp` par exemple. En fait, ils viennent de découvrir le motif classique, en programmation, pour échanger deux variables. Après cela, l'apprenant réalise aisément le tri des valeurs des variables (`a,b,c,d,e`) en reprenant l'algorithme précédent mais en échangeant les contenus de deux variables si besoin au lieu de les déplacer. La figure 11 montre le résultat de la manipulation.

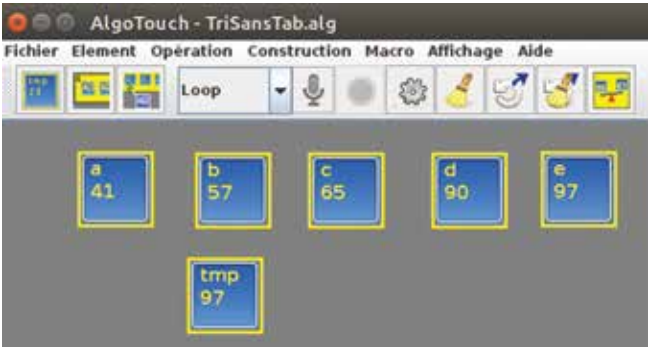


Figure 11 : Les variables sont triées par ordre de valeur croissante.

Tri manuel d'un tableau

Lors de l'étape précédente, le tuteur a montré qu'il était possible de réorganiser les données de 5 variables pour les classer par ordre croissant en utilisant les opérations de l'ordinateur : *l'affectation* et la *comparaison*. La question qui se pose alors est la suivante : comment faire pour que l'algorithme fonctionne indépendamment du nombre de variables ? En d'autres termes, comment appliquer le même principe de tri avec 20, 100 ou 1000 variables ?

Le tuteur introduit alors la notion de *tableau*. L'idée est la suivante : pourquoi ne pas associer un indice à un nom de variable. Les variables porteront ainsi le même nom et leur indice permettra de les différencier. Ces variables forment ce qu'on appelle habituellement un tableau. Avec AlgoTouch, le tuteur va alors créer un tableau, nommé `t`, par exemple, de 10 cases. AlgoTouch ajoute une constante appelée `t.length` contenant la taille du tableau, soit la valeur 10.

Le tuteur demande à l'apprenant ce qui change par rapport au tri des variables. L'apprenant indique, qu'*a priori*, c'est pareil sauf que `a` est identifié par `t[0]`, `b` par `t[1]`, etc. Il précise que cette fois, il doit comparer `t[0]` à `t[1]`, puis à `t[2]`, etc. Le tuteur indique alors qu'il serait judicieux de remplacer l'indice par une variable (`j`) qui serait incrémentée pour repérer la variable suivante du tableau `t`.

AlgoTouch définit la notion de *variable d'index* associée à un tableau. Le tuteur crée la variable d'index `j` associée au tableau `t`. AlgoTouch identifie par une flèche l'indice du tableau repéré par `j`. De plus, AlgoTouch crée une icône pour représenter `t[j]`, c'est une case comme une variable

mais dont le contenu est calculé en fonction de la valeur de `j`. Lorsque l'utilisateur manipule cette icône, il manipule en fait `t[j]` (*cf.* figure 12).

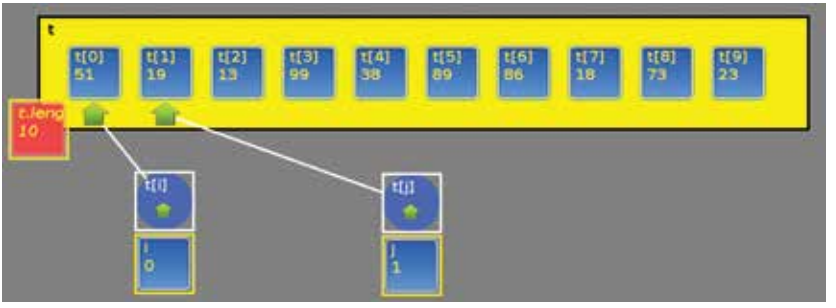


Figure 12 : Tableau de 10 valeurs à trier. À la création du tableau `t`, la constante `t.length` est définie automatiquement. Deux variables d'index `i` et `j` permettent d'accéder respectivement à `t[i]` et `t[j]` grâce aux icônes placées au-dessus. De plus, des flèches indiquent les éléments correspondants.

3.3 Automatisation

Dans cette partie, l'apprenant dispose maintenant de toutes les notions pour pouvoir automatiser l'algorithme de tri des données d'un tableau. Pour avancer, le tuteur remarque qu'il est souhaitable de disposer d'un deuxième indice, par exemple `i`, pour mémoriser l'endroit où va être placé le prochain minimum. Le tuteur indique que les variables d'index jouent le même rôle que les index (les doigts) utilisés dans la version débranchée pour repérer les éléments à comparer.

Dans cet exemple, le contenu de `t[0]` est comparé avec `t[1]` puis `t[2]` etc. Pour généraliser, il faut maintenant comparer `t[i]` successivement avec `t[j]` pour `j` allant de `i+1` au dernier indice du tableau.

Échange automatisé

Dans un premier temps, le tuteur montre qu'il compare toujours `t[i]` avec `t[j]`. Lorsque `t[j]` est inférieur, il doit échanger `t[i]` et `t[j]`. Cet échange peut donc être automatisé.

Avec AlgoTouch, il suffit d'actionner le bouton « enregistrer », puis d'effectuer les actions nécessaires et enfin d'arrêter l'enregistrement. Le tuteur crée alors une *macro opération* qu'il renomme *Swap*. Il peut alors ré-exécuter cette macro lorsqu'il souhaite échanger `t[i]` et `t[j]`.

Le tuteur vient donc d'introduire une nouvelle notion, fondamentale en programmation, la notion de *sous- programme* : suite d'instructions identifiée par un nom que l'utilisateur peut ré-exécuter. Cette notion permet aussi d'introduire la notion de décomposition d'un programme complexe en un ensemble de programmes plus simples, eux-mêmes éventuellement décomposés en sous-programmes.

Placement du minimum automatisé

Pour illustrer concrètement la notion de décomposition d'un algorithme en sous-algorithmes plus simples, le tuteur reprend l'exemple du tri du tableau de valeurs. Il rappelle qu'à une certaine étape, il faut placer dans `t[i]` la plus petite valeur des données allant de l'indice `i` au dernier indice du tableau. Il va créer une macro appelée *PlaceMin* pour représenter cet algorithme.

Lorsqu'il exécute manuellement les différentes opérations de l'algorithme *PlaceMin*, il s'aperçoit qu'il répète toujours la même séquence d'instructions : (1) comparer `t[i]` avec `t[j]`, (2) si `t[j]` est inférieur à `t[i]`, échanger `t[i]` et `t[j]`, (3) incrémenter `j`.

Le tuteur vient d'identifier la notion *d'itération ou de boucle*. Dans cet exemple, la séquence répétitive sera enregistrée dans le bloc *Loop*. Une fois cette séquence enregistrée, AlgoTouch offre la possibilité de l'exécuter à nouveau. Le tuteur peut donc s'apercevoir que le corps de l'algorithme fonctionne correctement : à chaque exécution, la valeur de `t[i]` est échangée avec `t[j]` si besoin et l'indice `j` est incrémenté. AlgoTouch identifie visuellement par une flèche l'indice du tableau repéré par `j` (*cf.* figure 13). Il voit donc la flèche se déplacer vers la droite à chaque exécution du corps de boucle jusqu'à ce que la flèche devienne rouge et déborde du tableau.

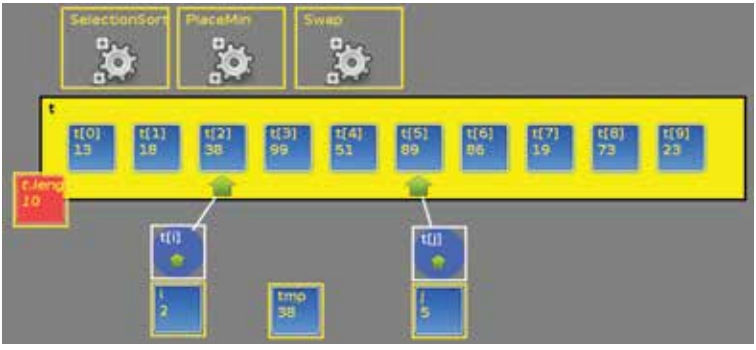


Figure 13 : Tableau en cours de tri. Les données à gauche de l'indice `i` sont triées. Le tableau est parcouru (indice `j`) pour placer la plus petite valeur en `t[i]`. La figure montre également les icônes des trois macros utilisées pour ce tri. Ces macros sont exécutables indépendamment par un double clic.

```
Define Swap
From
Until
Loop
  tmp = t[j];
  t[j] = t[i];
  t[i] = tmp;
Terminate
End
```

```
Define PlaceMin
From
  j = i + 1;
Until
  (j >= t.length)
Loop
  if (t[j] < t[i]){
    Swap;
  }
  j = j + 1;
Terminate
End
```

```
Define Selection-
Sort
From
  i = 0;
Until
  (i>=t.length)
Loop
  PlaceMin
  i = i + 1;
Terminate
End
```

Figure 14 : Programmes produits par AlgoTouch. Le premier, *Swap*, échange les contenus de `t[i]` et `t[j]`. À noter que puisque la clause *Until* est vide, le corps de la boucle ne s'exécute qu'une seule fois. Dans une version ultérieure d'AlgoTouch, la structure de la boucle sera remplacée par le corps du programme. Le second, *PlaceMin*, place en `t[i]` la valeur minimum de la suite. Le dernier, *SelectionSort*, trie tout le tableau en utilisant l'algorithme de sélection du minimum à chaque étape.

Le tuteur montre alors qu'il faut quitter la boucle et enregistrer la condition de sortie associée (*Until*). Un comparateur apparaît, il faut donc comparer l'indice `j` avec la valeur de la taille du tableau (`t.length`). La condition de sortie est que `j` est supérieur ou égal à `t.length`. Il reste

maintenant à enregistrer la séquence d'instructions à exécuter avant la première itération (*From*), soit `j = i+1`. Enfin, il doit éventuellement enregistrer la partie terminale (*Terminate*). Dans le cas présent, elle est vide. À l'issue de cette phase, le tuteur a construit les différentes séquences de la macro *PlaceMin*. Le code qui a été fabriqué automatiquement est illustré sur la figure 14.

Tri automatisé d'un tableau

À l'issue de cette étape, l'apprenant sait placer la bonne valeur à l'indice `i` en appelant la macro *PlaceMin*. Dans un premier temps, le tuteur propose de jouer avec cette macro. D'abord, il mélange les valeurs du tableau `t` et il initialise `i` à `0`. Il lance l'exécution de la macro *PlaceMin*, la plus petite valeur se retrouve en première position. Il incrémente `i` et appelle à nouveau *PlaceMin*. La seconde valeur est correctement placée. Le tuteur identifie la boucle comme étant la séquence suivante : (1) *PlaceMin*, (2) incrémenter `i`.

Le tuteur va alors créer le programme de tri (en fait la macro *SelectionSort*). Il enregistre d'abord le corps de la boucle vu précédemment (*Loop*). Puis exécute cette séquence plusieurs fois pour vérifier que l'algorithme se déroule correctement. Il arrête la boucle lorsque l'indice `i` déborde du tableau (*Until*). Pour la partie *From*, il initialise `i` à `0`. Pour la partie *Terminate*, elle est vide dans ce cas.

Le programme correspondant *SelectionSort* est donné dans la figure 14. Il utilise les différentes macros : *Swap* et *PlaceMin*. Le tuteur fait remarquer que chacune d'elles est très simple et facile à comprendre. Ceci illustre ainsi la notion vue précédemment qui consiste à décomposer un problème en sous-problèmes plus simples.

4 Conclusion et perspectives

Le logiciel AlgoTouch a été développé comme preuve de concepts. Il fonctionne sur plusieurs plateformes en utilisant la souris, mais l'usage d'un tableau interactif permet de manipuler directement les éléments d'un programme. Il facilite la création d'algorithmes simples sans avoir besoin de passer par un langage de programmation. Ainsi, il permet la transition d'une

activité débranchée à la programmation (activité branchée). Par étapes successives, les objets et actions des activités débranchées, sont remplacées par des variables et des opérations de la programmation. Dans ce papier, nous avons montré comment traiter un algorithme de tri.

Par ailleurs, nous avons aussi développé plusieurs scénarios de transitions d'activités débranchées vers des activités branchées : codage binaire, addition binaire, calcul d'une moyenne, etc. Ils ont été mis au point et testés avec quelques élèves de collège en classe de troisième pendant deux demi-journées. Pour ces exemples, AlgoTouch se révèle particulièrement bien adapté. Les élèves ont maîtrisé très rapidement l'outil. Ils ont vite compris comment réaliser des affectations, des opérations, des comparaisons. Ils ont aussi pris l'habitude d'utiliser des index sur des algorithmes opérant sur des tableaux. Enfin, ils ont pu apprendre progressivement les instructions du langage d'AlgoTouch puisque le système les produit automatiquement au fur à mesure des actions effectuées. À titre d'exemple, à la fin de l'expérience, ils ont réussi avec succès à réaliser un algorithme simple directement dans le langage de programmation.

De nouvelles expérimentations doivent être réalisées afin d'avoir des retours d'enseignants, d'élèves et d'étudiants pour améliorer le système. Nous pensons en particulier que le logiciel permet de faire comprendre les mécanismes de la programmation à tout public en un séminaire d'une journée par exemple.

Références

Baron, G.-L., & Drot-Delange, B. (2016, novembre). L'éducation à l'informatique à l'école primaire. *1014, Bulletin de la Société informatique de France*, *8*, 73–79. Consulté sur <https://hal.archives-ouvertes.fr/hal-01403598>.

Boisvert, C. R. (2009). A visualisation tool for the programming process. In *Proceedings of the annual acm sigcse conference on innovation and technology in computer science education* (pp. 328–332). New York, NY, USA : ACM. Consulté sur <http://doi.acm.org/10.1145/1562877.1562976> doi : 10.1145/1562877.1562976.

Drot-Delange, B. (2013, août). Enseigner l'informatique débranchée : analyse didactique d'activités. In *AREF* (p. 1–13). France. Consulté sur <https://archivesic.ccsd.cnrs.fr/sic_00955208> (ACTI Axe1).

Frison, P. (2015). A teaching assistant for algorithm construction. In *Proceedings of the 2015 ACM conference on innovation and technology in computer science education* (pp. 9–14). New York, NY, USA : ACM. Consulté sur <http://doi.acm.org/10.1145/2729094.2742588> doi : 10.1145/2729094.2742588.

Guzdial, M. (2004). Programming environments for novices. *Computer science education research*, *2004*, 127–154.

Hundhausen, C. D., Farley, S. F. & Brown, J. L. (2009). Can direct manipulation lower the barriers to computer programming and promote transfer of training ? : An experimental study. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *16*(3), 13.

Meyer, B. (2009). *Touch of class : learning to program well with objects and contracts*. Springer.

SIF. (2016, novembre). Enseigner l'informatique de la maternelle à la terminale. *1024, Bulletin de la Société informatique de France*, *9*, 25–33.

Sorva, J., Karavirta, V. & Malmi, L. (2013, novembre). A review of generic program visualization systems for introductory programming education. *Trans. Comput. Educ.*, *13*(4), 15 :1–15 :64. Consulté sur <http://doi.acm.org/10.1145/2490822> doi : 10.1145/2490822.

Chryssa Tsourapi[a], Vassilis Komis[a] & Georges-Louis Baron[b]

a. Université de Patras
chrisatsou@gmail.com, komis@upatras.gr
b. Université René Descartes – Paris 5
georges-louis.baron@parisdescartes.fr

L'étayage des enseignants de l'école maternelle au cours des activités de programmation avec le logiciel ScratchJr

Résumé
Cette étude qualitative, qui fait partie du projet de recherche DALIE (Didactique et Apprentissage de l'Informatique à l'Ecole), s'intéresse à explorer l'étayage de l'enseignant, afin de soutenir ses élèves de 4–6 ans, au cours d'activités de programmation informatique avec le logiciel ScratchJr. De quelle manière le rôle de l'enseignant est-il déterminant pour les acquis d'apprentissage ? Nous identifions trois types principaux de manifestation d'étayage : l'étayage cognitif, l'étayage technique et l'étayage émotionnel. Dans ce travail, nous mettons l'accent sur les concepts de programmation qui demandent un soutien spécial de la part de l'enseignant. Il résulte que les notions par rapport au script, au programme et à la séquence sont celles qui nécessitent le plus l'intervention de l'enseignant et exigent donc une attention didactique particulière.

Mots clés : ScratchJr, programmation, codage, maternelle, étayage, scénario éducatif

1 Introduction

La prise en compte de la dimension sociale d'apprentissage est clairement exprimée dans la théorie de Vygotsky par le concept de la Zone Proximale du Développement (ZPD), selon laquelle l'élève peut arriver à un niveau de compétences supérieur quand il est soutenu par un adulte, notamment par la parole (Vygotsky, 1978). C'est dans ce cadre que se fonde l'étayage,

une notion complexe qui reflète les profits cognitifs acquis par l'apprenant dans la ZPD avec l'aide d'un interlocuteur d'un niveau de développement supérieur.

L'étayage, premièrement défini par Wood, Bruner et Ross (1976), décrit la procédure dans laquelle l'élève est motivé à résoudre un problème dont la résolution serait impossible sans le soutien de l'enseignant. Il s'agit d'une notion difficile à délimiter mais fortement présente au terrain de la didactique contemporaine. Certains parlent d'un outil et d'autres d'une stratégie ou d'une technique (Van Der Stuyf, 2002 ; Roehler & Cantlon, 1997). D'après Mercer (1995), l'intervention de l'enseignant est déterminante pour le renforcement d'une connaissance réciproque avec l'élève. Il identifie une série de techniques d'étayage : les questions directes et indirectes, la confirmation, le refus, la répétition, la formulation longue, la récapitulation, etc. L'étayage se manifeste plutôt dans les situations ouvertes. Yelland et Masters (2007) distinguent trois différents types d'étayage pendant la résolution de problèmes avec des outils informatiques : le soutien technique, le soutien cognitif et le soutien émotionnel. En plus, Mercer et Fischer (1992) soulignent l'importance de l'étayage, étant donné qu'il permet aux enseignants d'atteindre les objectifs du programme en adoptant des techniques linguistiques particulières.

Le développement de la pensée informatique (Wing, 2006) par la programmation et la robotique pédagogique constitue ces dernières années un objectif d'enseignement dans plusieurs systèmes éducatifs (Komis & Misirli, 2016 ; Baron & Drot-Delange, 2016). L'apprentissage à travers la résolution des problèmes à l'aide de logiciels de programmation est un nouveau domaine d'enseignement et de recherche (Bers *et al.*, 2013). Falloon (2016), qui a notamment exploré l'usage de ScratchJr chez les petits enfants, soutient que le codage de base permet aux enseignants de CP (cours préparatoire) de développer chez leurs élèves des compétences d'un niveau supérieur. La bonne planification des activités qui se basent sur la résolution des problèmes et proposent le codage dans un environnement collaboratif est la clé pour les meilleurs résultats cognitifs (Komis, Touloupaki & Baron, 2017 ; Flannery *et al.*, 2013). Le rôle de l'enseignant est souligné pour les meilleurs acquis cognitifs (Kim & Hannafin, 2011 ; Portelance *et al.*, 2015).

2 Questionnement de recherche

La présente recherche s'inscrit dans le contexte des activités de programmation et vise à répondre au questionnement suivant : de quelle manière l'enseignant intervient-il dans les acquis d'apprentissage par rapport à la pensée informatique quand le travail concerne des environnements numériques de programmation et des jeunes âges ?

Nous avons adopté une telle problématique car elle nous paraît être une question d'actualité, et parce que la majorité de travaux relatifs publiés explorent plutôt l'activité du côté de l'élève que de celui de l'enseignant. L'objectif de notre recherche a été d'étudier l'étayage de l'enseignant pendant l'apprentissage de la programmation à l'école maternelle. Il se focalise sur les deux questions suivantes :

- De quelle manière se manifeste l'étayage de l'enseignant chez les enfants de grande section de maternelle pendant des activités de programmation avec ScratchJr ?
- Quels sont les concepts de programmation qui requièrent un étayage particulier ?

3 Méthodologie

En ce qui concerne la méthode de travail, huit enseignantes des écoles maternelles publiques de la région de Patras en Grèce faisant partie de l'équipe grecque du Projet DALIE[1] ont été formées à la programmation avec le logiciel ScratchJr par l'équipe de recherche du laboratoire de Didactique des Sciences et des TICE du Département des sciences de l'éducation-section préscolaire de l'université de Patras. Les outils utilisés sont le logiciel ScratchJr, des tablettes et le scénario éducatif. Suite à leur formation, les enseignantes ont appliqué des scénarios éducatifs, créés par l'équipe de recherche dans le but d'enseigner des concepts de base de programmation et de développer la pensée informatique chez les petits enfants. La recherche a

1 <http://www.unilim.fr/dalie/>

été menée pendant les mois d'avril et de mai 2016 dans les écoles publiques concernées de la région de Patras en Grèce. Les élèves impliqués n'avaient aucune familiarisation précédente avec le logiciel ScratchJr. Le scénario éducatif était planifié dès le début d'après le programme officiel grec pour être adapté au développement des enfants, il constitue donc par définition une sorte d'étayage cognitif.

Ce scénario, élaboré dans le cadre du projet DALIE, vise à familiariser les enfants à la programmation avec le logiciel ScratchJr. Il comporte trois activités d'initiation aux concepts de base en programmation, une activité d'application (consolidation) des connaissances acquises et une activité d'évaluation. La dernière, de forme ouverte, se déroule à la fin de la procédure. Il s'agit d'un problème de programmation sur lequel l'élève travaille individuellement et doit programmer le comportement de deux personnages sur la même scène. Tout le processus d'enseignement a été filmé. C'est cette activité d'évaluation qui constitue le corpus de données de cette recherche, car nous avons constaté que les élèves ne pouvaient pas gérer les concepts seuls malgré leur familiarisation précédente. Nous avons choisi d'étudier les vidéos des activités provenant de trois enseignantes dont l'approche didactique semblait différente. Plus précisément, nous avons analysé 30 séquences vidéo, 10 de chaque classe/enseignante. Nous avons utilisé le logiciel d'analyse qualitative Nvivo 8 (Gibbs, 2002) pour effectuer le codage et approfondir sur l'exploration du corpus de données.

Il s'agit d'une étude de cas puisque ce travail vise à examiner les spécificités d'un groupe particulier étudié (Cohen *et al.*, 2013). Nous avons effectué une analyse de contenu du corpus (le discours des enseignantes) afin d'éclairer comment le soutien pédagogique est effectué par l'enseignante et d'identifier les notions de programmation sur lesquelles elle met l'accent. Cette forme d'analyse nous a paru adaptée car elle permet la catégorisation du corpus en codant notamment le discours (Neuendorf, 2016). Plus précisément, nous avons choisi l'analyse de contenu thématique selon laquelle nous avons codé le discours de l'enseignante enregistré sous l'angle du sens (Creswell, 2013). Les parties du discours présentant des points communs au niveau conceptuel ont été codées dans la même catégorie. De cette façon nous pouvons répondre aux questions de manière descriptive et détaillée.

4 Analyse et résultats

Une action d'étayage est l'intervention de l'institutrice nécessaire afin que l'élève puisse avancer dans la résolution du problème. Nous rappelons les trois types d'étayage selon Yelland & Masters (2007) qui ont été reconnus dans notre analyse : le soutien technique, le soutien cognitif et le soutien émotionnel.

Premièrement, en ce qui concerne l'étayage technique, les enseignantes consacrent un temps important aux élèves ayant des difficultés de manipulation des tablettes, notamment pour attacher les commandes en puzzle parce que l'écran tactile ne réagit pas toujours de manière appropriée. Ce type d'intervention est plutôt tactile. Si par exemple l'élève a du mal à supprimer une commande incorrecte ou un personnage parce qu'il n'arrive pas à appuyer sur le petit « x » à cause de sa faible motricité fine, la maîtresse intervient. Toutefois, si elle intervient pour conseiller à l'élève de supprimer une commande, il s'agit d'un étayage à caractère cognitif.

Nous observons aussi un étayage émotionnel tout au long de la procédure. L'encouragement, la motivation, le maintien de l'attention, la confirmation, constituent des éléments essentiels au cours de l'enseignement qui incitent au progrès. Des exemples indicatifs de ce type d'étayage sont les expressions : « *bravo* », « *continue comme ça* », « *c'est ça* », « *très bien* », « *il faut réfléchir un peu* », etc.

Finalement, le type d'étayage figurant le plus souvent dans le corpus et sur lequel nous mettons l'accent dans ce travail, c'est le soutien cognitif. Dans cette catégorie s'inscrivent tous les efforts des enseignantes pour consolider chez les élèves des notions relevant de l'informatique et de la programmation, mais nous abordons également des notions mathématiques, comme la direction et du langage, comme la description orale.

Ce soutien cognitif est effectué en grande partie par la correction des erreurs, qui est aussi liée au contenu en jeu. Enfin, nous envisageons les stratégies didactiques adoptées comme des liens indispensables entre les enseignantes, les élèves et la nouvelle connaissance pour la réalisation des objectifs d'enseignement.

Puisque l'étayage cognitif occupe une grande partie des interventions étudiées, nous avons procédé à une analyse plus détaillée. Le modèle suivant (Figure 1) reflète les axes de programmation auxquels les élèves

semblent avoir besoin du support de l'adulte ainsi que cela résulte du codage détaillé du corpus.

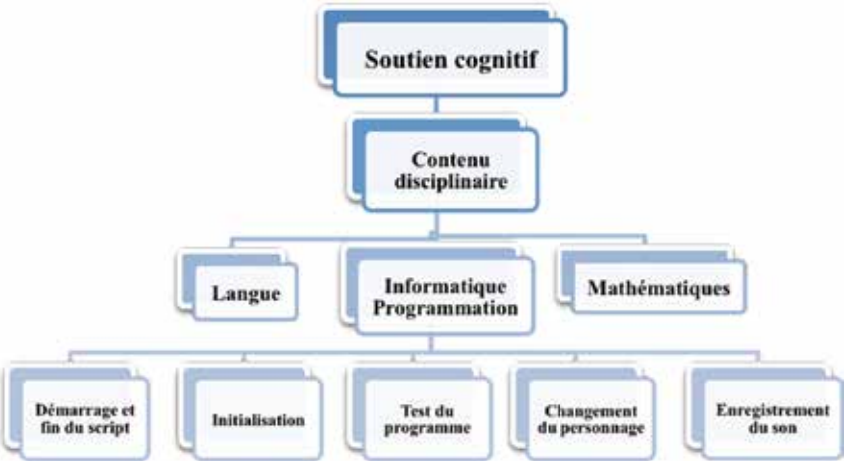


Figure 1 : Les notions de programmation soutenues par l'enseignante.

Le soutien cognitif concerne surtout l'accompagnement pédagogique des notions provenant du contenu disciplinaire. D'abord, il y a l'étayage par rapport à la programmation. Dans ce cas-là, la maîtresse aide l'enfant à dépasser des obstacles éventuels qui empêchent la résolution du problème. Les notions de démarrage et de fin du script, d'initialisation, de test du programme, de changement du personnage, d'enregistrement du son sont celles qui posent les problèmes les plus fréquents.

Nous estimons, toutefois, que l'intervention concerne la totalité des concepts proposés par le scénario éducatif parce que les enseignantes tendent plusieurs fois à intervenir même lorsque l'élève est en fait capable d'avancer seul à la résolution du problème, ce qui peut être expliqué par le stress de la réussite du projet et le temps limité.

Le tableau suivant, qui découle d'une analyse « Matrix Coding » sur Nvivo, illustre le nombre d'apparitions de l'intervention des enseignantes par rapport aux notions de programmation au cours de l'activité. Les trois lignes du tableau représentent les trois enseignantes et les colonnes, les notions de programmation.

Tableau 1 : Apparition des interventions quant aux notions de programmation.

	Enregistrement du projet	Démarrage et fin du script	Initialisation	Suppression du personnage	Introduction du fond	Introduction du personnage	Test du programme	Changement du personnage	Commandes de mouvement	Enregistrement du son	Nouveau projet	Paramètre	Plein écran	Drapeau vert
Ens. 1	1	**15**	9	5	2	2	**15**	6	0	5	1	2	0	0
Ens. 2	0	13	**30**	3	3	2	**24**	6	7	2	0	0	0	1
Ens. 3	0	1	10	0	0	0	**30**	**19**	0	9	0	0	1	1
Total	1	**29**	**49**	8	5	4	**69**	**31**	7	16	1	2	1	2

Nous observons que le support est assez faible quant aux commandes de gestion de l'environnement, comme les boutons plein écran, nouveau projet, enregistrement du projet, etc. Le concept de paramètre est aussi très peu mentionné même s'il s'agit d'un concept assez compliqué, parce qu'il ne fait pas partie de l'enseignement proposé dans cette activité. Ensuite, le drapeau vert, qui en général pose des problèmes aux utilisateurs à cause de sa double présence sur l'interface – ce qui peut être constaté par les pré- et post-tests des enfants – à ce moment ne constitue pas d'obstacle, car c'est le seul bouton en mode de plein écran, donc les enfants peuvent aussi y appuyer intuitivement.

En ce qui concerne les notions de programmation, celles qui nécessitent plus de support spécial (les plus fréquentes dans le tableau 1) sont les notions de script (démarrage et fin), de séquence et de programme. Tout d'abord, les enseignantes demandent directement ou indirectement aux enfants de « *compléter leur script* », « *d'ajouter le démarrage et la fin du script* », « *si leur script est prêt* », « *s'ils ont oublié quelque chose* ». Il est clair, à travers cette analyse, que même si le langage de ScratchJr est simple et intuitif (les briques sont organisées en forme de puzzle de manière séquentielle afin que les fautes syntaxiques soient évitées), les enfants oublient très souvent les commandes de contrôle, drapeau vert et fin. Nous attribuons la fréquence de ces interventions au fait que le logiciel permet l'exécution du script tout en appuyant sur n'importe quelle commande du script, ce qui occulte l'importance des blocs du drapeau vert et de la fin.

L'enseignement du démarrage et de la fin d'un script contient la notion de séquence, notion de base de la programmation. La séquence ne pose pas de problème quand les élèves arrivent à construire leurs scripts correctement. Ils comprennent que les commandes choisies seront exécutées dans la scène dans l'ordre présenté. Il y a cependant des exemples dans le corpus où la maîtresse essaie de faire comprendre à l'élève la séquence, c'est-à-dire associer les commandes choisies aux actions exécutées.

Les enseignantes insistent beaucoup sur le fait qu'il faut tester souvent le programme construit, afin de résoudre le problème. Le test continu du programme permet d'ailleurs l'ajout ou la suppression des commandes pour améliorer la résolution, ou même le débogage. Elles rappellent assez souvent aux élèves ayant des difficultés qu'ils sont capables d'associer conceptuellement les actions dans la scène aux commandes mises dans l'espace de programmation, c'est-à-dire de comprendre la syntaxe du programme.

Presque chaque fois que l'enseignante guide l'enfant, elle lui rappelle aussi de faire retourner le personnage à sa position initiale. Il y a des enfants qui, en testant leur programme, voient le personnage s'éloigner de l'objectif et se souviennent seuls de faire l'initialisation. Cette dernière pose aussi des problèmes, car les enfants tendent à glisser manuellement le personnage à sa position initiale. L'intervention de l'enseignante dépend bien sûr de la performance de l'élève impliqué, mais aussi de son style pédagogique, c'est-à-dire selon qu'elle est plutôt directive ou constructiviste. L'une des participantes adopte plutôt la méthode de programmation pas à pas, elle est donc plus directive et les deux autres promeuvent l'expérimentation de l'élève.

Enfin, la catégorie qui inclut l'encouragement, le rappel et l'étayage pour le changement de personnage apparaît également assez fréquemment. Les enseignantes interviennent plusieurs fois pour préciser quel personnage se trouve chaque fois dans l'espace de programmation, afin de faire comprendre à l'élève celui sur lequel il doit programmer. D'une part, elles guident directement l'enfant pour appuyer sur l'icône du personnage souhaité et, d'autre part, elles laissent entendre qu'il y a besoin d'amener l'autre personnage sur l'espace de programmation.

D'une façon ou d'une autre, elles considèrent leur aide à ce moment-là comme déterminante pour la résolution du problème. Le changement de caractère est difficile à assimiler parce qu'il impose la compréhension de la notion du programme, qui est assez compliquée. Ceci est dû au fait que l'espace de programmation change chaque fois que l'on change de

personnage. De plus, la terminologie des enseignantes peut causer de la confusion, parce qu'elles utilisent le même mot, soit « le programme » soit « le script », pour le même objectif.

5 Discussion

Le présent travail a cherché à éclairer les manifestations d'étayage à l'école maternelle à travers des activités relatives à l'apprentissage de la programmation. Nous avons organisé un enseignement des notions de base aux enfants de grande section de l'école maternelle et étudié l'étayage fourni par les enseignantes et avons analysé l'activité finale du scénario, une évaluation, au cours de laquelle l'intervention de l'enseignante devrait être limitée.

Les résultats montrent que celle-ci est forte mais pas forcément d'une manière directive. La complexité des notions traitées, le programme chargé des écoles maternelles et la durée limitée de la recherche peuvent expliquer cette contradiction, car il n'y avait pas assez de temps pour la consolidation des connaissances des élèves. Les institutrices devenaient aussi plus directives pour que leurs élèves puissent résoudre le problème final dans le cadre du projet de recherche. À ce moment-ci, il convient de souligner le fait que les enseignantes n'ont pas la même approche didactique, comme il apparaît dans le tableau 1.

L'étayage constaté correspond à celui de Yelland et Masters (2007) et comporte trois dimensions : premièrement, le soutien technique, qui contient des aides didactiques orales et tactiles de l'enseignant, au cas où l'élève a du mal à gérer l'écran de la tablette, ou l'interface du logiciel. Nous constatons aussi le soutien émotionnel, où l'enseignante essaie d'encourager l'élève et maintenir son intérêt. Enfin, il y a le soutien cognitif, c'est-à-dire les aides au plan des connaissances et des stratégies didactiques adoptées. Nous concluons que l'étayage joue un rôle déterminant dans le résultat final de l'activité. En fait, la majorité des élèves arrivent à résoudre le problème avec l'accompagnement de l'enseignante toujours présente.

Une attention particulière dans notre analyse a été portée sur les spécificités du contenu disciplinaire, c'est-à-dire les concepts de programmation et le langage utilisé. La sémantique du langage nécessite d'un soutien

particulier (Komis, Touloupaki et Baron, 2017), c'est pourquoi ses trois composants, c'est-à-dire les opérations (par exemple les scripts), les objets (les personnages) et les emplacements (les fonds), constituent les objectifs cognitifs de base du scénario éducatif. D'un point de vue sémantique, il s'agit d'exécution des actions par l'objet dans un certain endroit (Fay et Mayer, 1988). En même temps, les concepts de la syntaxe causent quelques fois l'intervention de l'enseignante afin de soutenir ses élèves. Nous constatons que ces notions de programmation sont assez compliquées pour les petits enfants, nous considérons donc le rôle de l'enseignant déterminant à l'apprentissage.

Pour finir, soulignons les limites de cette recherche : il s'agit d'une étude de cas sur un échantillon faible. L'étude ne prétend donc pas à la généralisation de ses résultats. Elle tente néanmoins d'explorer de façon approfondie les interactions entre les enseignants, les apprentis et les connaissances.

Nous souhaitons à l'avenir approfondir ce travail en examinant aussi les moyens que les enseignants adoptent pour construire l'étayage. De plus, il serait intéressant d'analyser les apports cognitifs de l'étayage sur la programmation en examinant une équipe d'élèves soutenus par l'enseignante et une autre équipe dont les membres travailleraient seuls pour les comparer. Finalement, un sujet de recherche pourrait être la différenciation de l'étayage entre les élèves de grande et de moyenne section de l'école maternelle.

Références

Baron, G.-L., & Drot-Delange, B. (2016). L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. *Revue française de pédagogie*, 195, pp. 51–62.

Bers, M., Flannery, L., Kazakoff, E. & Sullivan, A. (2013). Computational thinking and tinkering : Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, pp. 145–157.

Cohen, L., Manion, L., & Morrison, K. (2013). *Research methods in education*. Routledge.

Creswell, J. W. (2013). *Research Design : Qualitative, Quantitative, and Mixed Methods Approaches*. Sage.

Fay, A. L., & Mayer, R. E. (1988). Learning Logo : A cognitive analysis. In R. E. Mayer (Ed.), *Teaching and learning computer programming : Multiple researcher perspectives.* (pp. 55–74).

Flannery, L.P., Kazakoff, E.R., Bontá, P., Silverman, B., Bers, M.U., & Resnick, M. (2013). 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).

Gibbs, Graham R. (2002). *Qualitative Data Analysis : Explorations with Nvivo*. Open University, Buckingham.

Kim, M., & Hannafin, M. (2011). Scaffolding problem solving in technology enhanced learning environments (TELEs) : Bridging research and theory with practice*, Computers & Education, 56* (2), pp. 403–417.

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).

Komis, V., & Misirli, A., (2016). The environments of educational robotics in Early Childhood Education : towards a didactical analysis, *Educational Journal of the University of Patras UNESCO Chair,* 3(2), pp. 238–246.

Mercer, N. (1995). *The guided construction of knowledge : Talk amongst teachers and learners*. Multilingual matters.

Mercer, N., & Fisher, E. (1992). How do teachers help children to learn ? An analysis of teachers' interventions in computer-based activities. *Learning and instruction*, *2*(4), pp. 339–355.

Neuendorf, K. A. (2016). *The content analysis guidebook*. Sage.

Nisbet, J. & Watt, J. (1984). Case study. In J.Bell, T. Bush, A. Fox, J. Goodey and S. Goulding (eds), *Conducting Small-Scale Investigations in Educational Management*. (pp. 79–92).

Portelance, D. J., Strawhacker, A., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*. pp. 1–16.

Roehler, L. R., & Cantlon, D. J. (1997). Scaffolding : A powerful tool in social constructivist classrooms. *Scaffolding student learning : Instructional approaches and issues, 1*.

Van Der Stuyf, R. R. (2002). Scaffolding as a teaching strategy. *Adolescent learning and development*, *52*(3), pp. 5–18.

Vygotsky, L. (1978). *Mind in society*. Cambridge, MA : Harvard University Press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), pp. 33–35.

Wood, D., Bruner, J., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Child Psychiatry, 17,* pp. 89–100.

Yelland, N., & Masters, J. (2007). Rethinking scaffolding in the information age. *Computers & Education, 48*(3), pp. 362–38.

Sévastiani Touloupaki[a], Georges-Louis Baron[a] & Vassilis Komis[b]

a. Laboratoire EDA, Université Paris Descartes
sevina.touloupaki@gmail.com, georges-louis.baron@parisdescartes.fr
b. Équipe ICTE, Université de Patras
komis@upatras.gr

Un apprentissage de la programmation dès l'école primaire : le concept de message sur ScratchJr

Résumé

Ce travail est inscrit dans le cadre du projet DALIE de l'Agence Nationale de la Recherche française et son objet d'étude a été d'analyser la manière dont l'usage du logiciel ScratchJr peut permettre aux élèves de primaire de s'approprier des notions de programmation. Dans ce contexte, nous avons choisi de mener une étude exploratoire dans une école primaire française auprès de douze élèves de cours préparatoire. Nous avons élaboré un scénario pédagogique de dix séances (une activité initiale, une activité de préparation psychologique, sept activités d'apprentissage, commençant par la familiarisation avec les concepts de l'interface et se terminant à l'apprentissage des concepts clés de programmation tels que la répétition et les messages, et une activité d'évaluation) qui visait au développement des compétences de programmation chez les jeunes élèves. Même si nous avons étudié plusieurs concepts différents de programmation comme la séquence, la synchronisation, l'itération, etc., nous avons choisi de nous concentrer dans cet article seulement sur l'analyse du concept des messages. Pour cela, nous allons analyser d'une part les réponses des élèves aux entretiens semi-directifs avant et après la mise en place du scénario pédagogique et d'autre part les programmes développés par les derniers au cours de la séance d'évaluation. Nous allons du coup plus loin que ce qu'on avait fait à Ioannina Touloupaki et al. (2016), où avaient été présentés les résultats qui ressortent seulement par les programmes élaborés au cours de la séance d'évaluation par les élèves de l'école maternelle en Grèce et du primaire en France.

Mots clés : didactique de programmation, ScratchJr, message, école primaire, programmation visuelle, programmation orientée objet

1 Introduction

L'apprentissage de la programmation pendant la petite enfance revient à l'ordre du jour depuis quelques années. Plus précisément, nous traversons une période florissante en ce qui concerne les environnements de programmation destinés aux plus jeunes élèves et cela facilite l'étude de l'apprentissage de programmation dès l'école élémentaire.

Pourtant l'apprentissage de la programmation a une longue histoire à l'école élémentaire, qui remonte aux années 60 et à la création du langage Logo par Seymour Papert, Wallace Feurzeig et leur équipe. Ces derniers ont défendu à l'époque l'émergence et la fondation d'une pensée logique, qui renforce la construction de procédures mathématiques (Feurzeig, 2010).

Cet auteur nous explique que l'expérimentation avec Logo permet à l'élève d'améliorer sa pensée critique et de participer au processus éducatif d'une manière active. Il nous explique que l'objectif principal de l'élaboration du Logo était le développement d'une série des compétences qui pourraient avoir des effets positifs dans d'autres domaines.

Selon Clements et Gullo (1984), l'apprentissage de la programmation pourrait jouer un rôle important pour l'amélioration de certains aspects du processus de résolution des problèmes. Ils montrent que l'apprentissage de la programmation peut renforcer le développement d'une pensée divergente et améliorer les résultats des participants aux épreuves métacognitives en limitant le nombre des erreurs réalisées. Selon Clements et Meredith (1992), l'apprentissage de la programmation dans un environnement Logo renforce également le développement des compétences en mathématiques, en résolution de problèmes et en langage. Plus près de nous, Duncan *et al.* (2014) soulignent également le fait que la programmation pourrait faciliter l'apprentissage de concepts utiles pour d'autres matières, comme la littérature ou les mathématiques.

Dans un ouvrage récent, Misirli et Komis (2016) décrivent les résultats de l'expérimentation d'un scénario pédagogique fondé sur l'usage des jouets programmables Bee-Bot. Cette recherche a eu deux objectifs principaux : d'une part, de construire des concepts de base de programmation tels que la séquence, l'algorithme, le programme et la commande et d'autre part, de développer des compétences liées à la pensée informatique chez les jeunes enfants. Il s'agit d'une étude qui a été menée dans 34 classes de maternelle en Grèce, auprès de très jeunes élèves âgés de 4 à 6 ans.

Misirli et Komis (2016) montrent que les jeunes enfants arrivent à résoudre un problème d'orientation dans l'espace en deux dimensions à l'aide du jouet programmable, après la mise en place d'un scénario pédagogique adapté à leurs besoins. Pour aller plus en détail, ils sont arrivés d'abord à verbaliser un algorithme, puis à passer à la programmation (par cartes et sur le jouet programmable) et enfin à le déboguer lorsqu'il y avait une erreur de programmation. Un pseudo-langage qui comprend des cartes qui affichent les commandes qui apparaissent sur le robot a été utilisé en tant que stratégie didactique adéquate, afin de favoriser la description de l'algorithme ainsi que la détection et la correction des erreurs.

Le langage de programmation Logo a beaucoup évolué au cours des années qui ont suivi et plusieurs versions sont disponibles en ce moment dans le marché. Un de ses descendants actuels est le langage de programmation Scratch, qui a constitué la base pour la création de ScratchJr, son petit frère, destiné aux élèves de 5 à 7 ans.

1.1 Le logiciel ScratchJr

ScratchJr a été créé par le laboratoire Média du MIT, l'université Tufts et PICO (Playful Invention Company) afin d'initier les élèves de la grande section de l'école maternelle jusqu'au CE1 aux notions de programmation. L'application est disponible en anglais sous forme d'application gratuite, pour iPad et pour tablette Android. Le code et l'interface ont été simplifiés pour qu'ils puissent correspondre au développement cognitif, personnel, social et affectif des jeunes élèves (Resnick *et al.*, 2013).

ScratchJr, comme son ancêtre Scratch, utilise aussi un environnement de type « Drag and Drop », c'est-à-dire que les enfants doivent cliquer et glisser les icônes sur l'espace de programmation afin de créer des scripts. Il y a une palette de briques colorées qui comprend six catégories différentes. Par exemple, les blocs jaunes permettent de décider du mode de départ du programme, les flèches bleues des déplacements sur l'écran, etc. Les icônes sont assez larges pour que les enfants puissent les utiliser facilement. L'interface est conviviale, joyeuse, amicale et ludique de manière à motiver les jeunes enfants à programmer avec le logiciel. En utilisant ScratchJr, les enfants peuvent exprimer leur créativité, pendant qu'ils programment leurs histoires et leurs jeux interactifs. Ils peuvent modifier facilement leurs

caractères, ajouter leurs propres voix ou des sons, et même insérer des photos d'eux-mêmes (Resnick *et al.*, 2013).

ScratchJr a été créé afin de pouvoir s'intégrer à la classe et promouvoir trois catégories d'apprentissage à travers la programmation : la résolution de problèmes, le développement des connaissances fondatrices (la séquence, l'estimation, la composition, la décomposition, etc.) et l'apprentissage de la lecture, de l'écriture et des mathématiques. Plus précisément, les scripts de programmation sont écrits en poursuivant la direction de l'écriture, de gauche à droite. En outre, le logiciel met à la disposition des utilisateurs une grille, qui facilite une exploration quantitative (mesurer, compter, etc.). La communauté de ScratchJr fournit également du matériel curriculaire et des sources en ligne pour les enseignants, pour soutenir leur travail (Resnick *et al.*, 2013).

Nous pouvons citer les travaux de Dylan Portelance, Marina Umaschi Bers et Amanda Strawhacker (2015) sur l'apprentissage de la programmation à travers le logiciel ScratchJr par les très jeunes élèves. Soixante-deux élèves dès la grande section de l'école maternelle jusqu'à CE1 ont été exposés à un curriculum de six semaines en utilisant le logiciel ScratchJr. Ils ont appris des concepts fondamentaux de programmation et ils ont appliqué ces concepts à leurs propres projets. Les chercheurs s'intéressent à distinguer les blocs de programmation que les jeunes élèves préfèrent utiliser pour leurs projets après avoir appris tous les blocs disponibles sur l'application. L'analyse des données montre que les blocs qui sont utilisés le plus souvent par les élèves sont ceux qui sont responsables du mouvement du personnage dans la scène.

Dans un contexte un peu plus large, Dylan Portelance (2015) a créé sous la direction de Marina Umaschi Bers une activité novatrice qui s'appelle « Code and Tell », dont la conception vise à initier les jeunes élèves aux concepts de base de programmation à travers l'application ScratchJr. C'est une étude exploratoire mise en place afin de comprendre comment cette activité facilite l'apprentissage des concepts de pensée informatique par les jeunes élèves lorsque celui-ci est accompagné d'entretiens directifs entre les élèves. Soixante-six élèves de CE1 ont été exposés à un curriculum de treize séances pour réaliser trois genres de projets différents : collages, histoires et jeux interactifs avec ScratchJr. Au cours de chaque unité du curriculum, les élèves participaient à deux grandes sous-unités : une période consacrée à créer leurs propres projets (soit collages, soit histoires, soit jeux) et une deuxième période consacrée à participer à l'activité « Code

and Tell ». Cette dernière est constituée d'entretiens directifs menés auprès des élèves par groupes de deux, c'est-à-dire que chaque élève posait quatre questions à son binôme, dont les trois premières étaient prédéfinies par le chercheur et la dernière était de leur propre choix. L'analyse des données a conduit l'auteur à créer trois grandes catégories de projets en fonction du comportement des élèves, pendant la présentation de leurs projets : les projets « descriptifs », les projets « démonstratifs » et les projets « imaginatifs ». Il souligne le fait que chaque catégorie a un lien fort avec la pensée informatique. En fonction de la complexité avec laquelle les élèves ont présenté leurs projets, Portelance a créé trois sous-catégories à l'intérieur de chaque grande catégorie, les « simples », les « intermédiaires » et les « complexes » (Touloupaki, 2016).

Figure 1 : L'interface de ScratchJr.

1.2 La singularité de ScratchJr et le concept de message

ScratchJr, comme Scratch, suit un paradigme de programmation orientée objet fondé sur les principes de Smalltalk. Smalltalk a été le premier langage de programmation orientée objet, développé par Alan Kay, en 1971 (Kay, 1993). La programmation orientée objet ou programmation par objets analyse les problèmes de manière à identifier les « acteurs » qui le composent et puis à déterminer ce qu'est et ce que doit faire chaque acteur. Ce paradigme utilise l'objet comme principe unificateur (Cloutier, 1988).

Un objet a un état qui est l'ensemble de ses propriétés et un comportement spécifique, c'est-à-dire un ensemble de réactions qui peuvent être déclenchées par d'autres objets à travers l'envoi de messages. Il est également doté d'un ensemble de scripts qui définissent son comportement. Par

exemple, un objet peut modifier certains de ses attributs à la demande d'un autre. Le programmeur doit décrire quels échanges entre les acteurs (objets) produiront les comportements adéquats pour la résolution d'un problème (Cloutier, 1988). Les objets qui ont les mêmes attributs constituent une classe d'objets (Ben-Ari, 1996).

Dans ce contexte, ScratchJr permet de définir plusieurs personnages, de créer pour chacun d'entre eux des scripts spécifiques et de les faire communiquer en utilisant les messages (Komis, 2016). C'est pourquoi ce paradigme permet de mettre en place en même temps des formes de la programmation concurrente.

En effet, les scripts des personnages sur ScratchJr s'exécutent de manière concurrente et pas l'un après l'autre. L'utilisateur devra avoir des mécanismes permettant de définir l'ordre de l'exécution des scripts des objets, pour résoudre ses problèmes. Un tel mécanisme est rendu assez facile par le concept de messages.

Ce dernier est implémenté par deux commandes : la commande « envoyer 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 la couleur est déclenché.

La commande « envoyer message » se fait dans un script de programmation, comme toutes les autres commandes sur ScratchJr. Son seul paramètre est la couleur. La commande « quand message reçu » sert à déclencher un événement et elle se place au début d'un script. En ce qui concerne la sémantique, lorsque l'exécution du script arrive à l'envoi d'un message de couleur spécifique, ce message est envoyé et les scripts des personnages, qui prévoient l'arrivée d'un message de la même couleur sont déclenchés de manière concurrente.

Figure 2 : Les commandes de messages.

2 Problématique et questionnements de recherche

Nous avons choisi pour objet d'étude, l'analyse de la manière dont l'usage du logiciel ScratchJr peut permettre aux élèves de CP de s'approprier des notions de programmation. Plus précisément, notre problématique est formulée de la manière suivante :

- *Comment initier les enfants de 6 à 7 ans aux notions de base de programmation à travers le logiciel ScratchJr et plus précisément au concept des messages ?*

Les messages sont un concept de programmation de haut niveau, peu étudié dans la littérature scientifique au niveau de la petite enfance. Notre objectif principal a été la compréhension de la fonctionnalité des messages par les élèves et leur utilisation opérationnelle, afin de résoudre un problème donné. Dans ce contexte, une autre raison pour laquelle nous avons choisi d'étudier ce concept était sa capacité potentielle à contribuer au développement d'une pensée informatique chez les jeunes enfants. Plus précisément, nos questionnements sont les suivants :

- Comment les élèves perçoivent-ils les instructions des messages ?
- Comment les élèves arrivent-ils à utiliser les commandes des messages de manière opérationnelle afin de pouvoir résoudre un problème d'évaluation ?
- Comment les élèves arrivent-ils à construire des représentations complètes autour du concept des messages ?

3 Méthodologie

3.1 Recherche de conception

Nous avons choisi de mettre en place une recherche de conception ou *design-based research* en anglais. Il s'agit d'une approche qui met au centre la conception d'un scénario pédagogique (le *design*) et l'évaluation de sa

capacité à transmettre les savoirs choisis par son créateur à travers la mise en place itérative d'une série d'interventions dans un terrain choisi. La recherche de conception permet une compréhension profonde du processus de l'apprentissage (Cobb *et al.*, 2003).

Cobb *et al.* (2003) font référence à cinq de ses caractéristiques principales. D'après eux, une telle recherche a comme objectif le développement d'une série des théories à propos du processus de l'apprentissage et des stratégies didactiques mises en place, afin de pouvoir faciliter ce processus d'apprentissage. Ensuite, la recherche de conception a un caractère interventionniste, car un de ces objectifs est d'investiguer les possibilités d'une amélioration de l'éducation à travers l'étude des stratégies différentes d'enseignement. Puis, une recherche de conception a deux facettes : le potentiel et le réfléchissant. Ces deux facettes constituent une quatrième caractéristique qui est l'itération. La cinquième caractéristique de la recherche de conception est ses racines pragmatiques, c'est-à-dire que les théories qui émergent d'une recherche de conception doivent avoir un sens spécifique pour cette conception.

Au cours de la mise en place d'une recherche de conception, un des objectifs principaux est d'améliorer la conception 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 (Cobb *et al.*, 2003).

Dans ce cadre, le chercheur fait une analyse holistique des interventions éducatives, c'est-à-dire qu'il perçoit les interventions comme des interactions entre les enseignants, les apprenants et les matériels utilisés (The Design-Based Research Collective, 2003).

3.2 Étude de cas

Une étude de cas a été effectuée. Il s'agit d'une méthode mixte de la recherche, qui permet une approche exploratrice, descriptive et explicative du phénomène étudié (Yin, 2003). L'étude de cas peut aussi avoir une direction plutôt qualitative ou quantitative en fonction de la posture du chercheur, de la question de départ, du terrain choisi, etc. Dans le cadre de notre projet, la méthode reste mixte, mais elle est caractérisée par une visée plutôt qualitative.

Nous avons choisi ce type de recherche parce que nous voulions nous concentrer non seulement sur le terrain mais aussi sur les comportements et les interactions de l'échantillon, par rapport au logiciel ScratchJr et aux notions de programmation en général. Nous nous fonderons donc sur des données empiriques de terrain (approche inductive) et les interpréterons pour arriver aux résultats et, à terme, à la construction de modèles et de théories, toujours en rapport avec la population étudiée. Il s'agit d'une stratégie de recherche préférable lorsque les circonstances et le problème de la recherche sont convenables et lorsque le chercheur ne contrôle pas les comportements d'échantillon. L'étude de cas est une méthode empirique qui examine un phénomène contemporain où les frontières entre ce phénomène et le contexte ne sont pas visibles (Yin, 2003).

La formulation de notre questionnement est compatible avec cette méthode, car elle est adaptée pour réaliser une recherche dont les questions de départ sont posées sous la forme « Comment ? » et « Pourquoi ? » (Yin, 2003).

3.3 Le scénario pédagogique

Ce scénario pédagogique forme le noyau de notre projet de recherche et il comprend une série d'activités qui ont comme but l'introduction de concepts préliminaires de programmation comme la séquence et l'itération, mais surtout le concept des messages, à travers le logiciel ScratchJr.

Tout d'abord, nous avons distingué deux grandes catégories de concepts : les concepts d'interface et les concepts de programmation. Parmi les concepts d'interface disponibles sur ScratchJr, nous avons choisi de travailler sur l'accueil, la scène, l'initialisation de la scène, le mode de présentation, le paysage, le drapeau vert, l'enregistrement d'un projet, l'éditeur graphique, les catégories des commandes, l'espace de programmation, le script de programmation et les personnages. Ce sont les concepts de base du logiciel qu'on devait enseigner aux élèves, afin de pouvoir les initier à la programmation.

Ensuite, nous avons distingué deux catégories de concepts de programmation, basées sur la facilité d'acquisition des élèves : les concepts de niveau haut (comme la séquence, les messages, etc.) et les concepts de niveau bas (comme le déplacement du personnage vers la gauche ou la droite, etc.). En rendant compte du développement cognitif, affectif et social des enfants de notre échantillon, nous avons décidé des concepts de programmation que

nous avons explorés dans notre scénario. Plus précisément, notre ambition était d'examiner d'une part les représentations des enfants à propos des concepts choisis, avant l'application du scénario et d'autre part s'il y a du progrès cognitif après l'application du scénario.

Parmi les concepts de programmation disponibles sur ScratchJr, nous avons choisi d'étudier les suivants : les instructions, les valeurs des instructions, la séquence, l'itération (boucle), la programmation concurrente/ la synchronisation, le programme et les messages.

Tableau 1 : Les concepts de programmation en jeu.

Concept de programmation	Description du concept de programmation
Instruction	Chaque instruction oblige un personnage à effectuer une action.
Valeur d'instruction	Plusieurs instructions nécessitent une valeur pour être exécutées.
Programme	Un programme est un ensemble d'instructions destinées à être exécutées par le logiciel ScratchJr.
Script de programmation	Un script de programmation est une connexion de commandes structurées qui spécifie le comportement d'un personnage. Les commandes en ScratchJr sont organisées en forme de puzzle de manière séquentielle. Un script s'exécute de gauche à droite.
Séquence	Une série d'instructions placées l'une après l'autre.
Itération	La répétition est une boucle itérative qui permet de répéter un nombre de fois prédéfini les commandes qui se trouvent à l'intérieur.
Programmation concurrente	En cliquant sur le drapeau vert, tous les scripts s'exécutent simultanément. Cela signifie que les scripts sont exécutés en même temps sans qu'un seul script ait besoin d'attendre la fin de l'autre ou sans que les scripts soient dépendants les uns des autres.
Messages	Les personnages communiquent à l'aide de deux commandes de messages, la commande « envoyer message » et la commande « quand message reçu ». Les deux personnages sont coordonnés lorsqu'ils échangent des messages.

En ce qui concerne le scénario pédagogique, nous avons conçu et réalisé, pendant un mois et demi, sept activités d'enseignement, une activité initiale et une activité finale d'évaluation. Chaque jour, nous avons animé une séance de trente à quarante minutes.

Au cours de l'activité initiale, nous avons d'abord conduit un entretien personnel (pré-test) pour détecter les représentations initiales des élèves à propos des concepts étudiés et d'autre part, une activité de préparation psychologique et cognitive. Pendant la séance de préparation, nous avons présenté aux élèves certains concepts de la programmation. Nous leur avons présenté l'utilisation de la tablette, en leur apprenant les processus du verrouillage et du déverrouillage. Ensuite, nous avons fait une première familiarisation avec la manipulation de la tablette et une initiation au logiciel ScratchJr.

Ensuite, nous sommes passés aux séances d'enseignement. Au cours de la première, nous avons exploré comment ouvrir l'application, démarrer un nouveau projet, insérer et effacer des briques de programmation, utiliser les commandes de mouvement afin de déplacer le personnage dans la scène, définir la position du personnage mais aussi le processus de l'enregistrement d'un projet. Nous avons aussi essayé de comprendre la signification des instructions pour le personnage.

Au cours de la deuxième séance, nous avons présenté le principe « début-fin » de la programmation (séquence) et comment : sélectionner un arrière-plan, présenter un projet en plein écran et introduire des nouveaux personnages dans la scène. Nous avons aussi appris comment faire tourner le personnage à gauche et à droite et comment le faire sauter. Nous avons appris également l'icône pour initialiser la position du personnage.

Pendant la troisième activité, nous avons montré comment enregistrer un son, écrire un message et effacer un personnage sur la scène. Nous avons aussi réalisé une révision de toutes les briques de mouvement apprises jusqu'alors.

Au cours de la quatrième activité, nous avons abordé un concept principal de programmation, celui de la boucle, c'est-à-dire apprendre à répéter une commande et un motif de commandes. Pour enseigner ce concept, nous avons choisi de présenter aux élèves des projets déjà prêts qui contiennent la nouvelle commande et les laisser expérimenter et découvrir la fonctionnalité de la commande à l'aide de nos questions. Au cours de la cinquième activité, nous avons résolu un problème en utilisant la commande de la répétition.

Les sixième et septième activités ont été dédiées au concept de message. Plus précisément, au cours de la sixième nous avons enseigné aux élèves la fonctionnalité des messages. Pour cela, nous avons présenté des projets déjà prêts qui contiennent les commandes des messages et nous les avons laissés expérimenter et découvrir la fonctionnalité de chacune des commandes, à l'aide de nos questions. Pour faciliter la compréhension du concept par les élèves, nous avons choisi d'utiliser une partie du matériel proposé par le site ScratchJr, les images des commandes du logiciel à imprimer. Nous avons donc construit les commandes du ScratchJr en version papier, que les élèves ont utilisées pour comprendre la fonctionnalité des messages et pour résoudre le problème de la septième activité. Dans la figure qui suit, nous présentons le matériel utilisé au cours de ces deux séances.

Figure 3 : Les commandes et les personnages du ScratchJr en version papier (matériel utilisé pour la mise en place du scénario pédagogique).

Au cours de l'activité d'évaluation, nous avons mené un entretien personnel (post-test) avec les questions de l'activité initiale et un même problème à résoudre pour tous les élèves.

Nous avons mis en place l'activité initiale et l'activité finale, afin de pouvoir détecter le développement cognitif des enfants à propos des concepts étudiés, avant et après l'application de notre scénario pédagogique.

Enfin, nous avons mené une séance dédiée à la découverte de l'éditeur graphique, car les élèves voulaient savoir sa fonctionnalité.

3.4 L'échantillon

Notre recherche a été réalisée dans une école primaire en banlieue parisienne auprès de 12 élèves, 8 filles et 4 garçons. Ces élèves étaient les seuls dons les parents avaient signé l'autorisation de participer à notre recherche. Les élèves ont travaillé par groupes de quatre, avec quatre tablettes pour réaliser les activités proposées. Nous avons travaillé avec trois groupes de quatre élèves. Les groupes ont été constitués par l'enseignante de la classe, de manière à pouvoir représenter tous les niveaux scolaires différents présents dans la classe. Les tablettes que nous avons utilisées étaient des iPad. L'enseignante de la classe n'a pas participé à notre étude.

3.5 Techniques et outils de recueil de données

Pour recueillir nos données, nous avons choisi d'utiliser comme techniques l'observation participante et les entretiens semi-directifs. Pour faciliter le recueil de données, nous avons utilisé un enregistrement audiovisuel du projet, ainsi que nos notes personnelles avec les incidents critiques et les grilles d'analyse pour l'activité initiale et l'activité finale. Nous avons également utilisé des captures d'écran des programmes élaborés par les élèves tout au long du processus.

4 Résultats

Une première étude empirique menée en 2016–2017 nous avait permis d'avoir des résultats préliminaires à propos de l'initiation des élèves de CP aux concepts de programmation et plus précisément du concept du message. Nous avons dès lors analysé d'une part les données recueillies au cours de la séance d'évaluation et, d'autre part, ceux recueillis par le pré-test et post-test.

Dans un premier temps, nous avons analysé les programmes élaborés[1] par les élèves face à un même problème d'évaluation et les enregistrements vidéo des séances de l'évaluation. Nous sommes arrivés donc à distinguer trois types de construction du concept de messages en fonction de la performance des élèves face à ce problème : une « construction complète », une « construction partielle » et une « construction incomplète ».

Pour arriver à cette typologie nous avons défini une série des critères qu'un programme devait satisfaire afin de pouvoir être caractérisé « complet ». Dans le Tableau 2 nous présentons les critères pour une construction complète de la notion des messages par les élèves. Les critères sont au nombre de deux (A, B), parce que nous avons choisi de ne pas enseigner le paramètre des commandes des messages, c'est-à-dire la couleur.

La classification des élèves selon la typologie décidée s'est fondée sur l'énumération du nombre des critères que leur programme satisfaisait. Pour que les critères A et B soient satisfaits, il faut que leurs deux parties soient satisfaites aussi. C'est-à-dire le personnage (enfant ou papillon) dont le script doit contenir la commande concernée, mais aussi la position à laquelle se trouve la commande concernée sur le script du personnage.

Tableau 2 : Critères de la construction complète de la notion des messages.

A. La commande « envoyer message » se trouve après la commande de mouvement sur le script du personnage « enfant ».
B. La commande « quand message reçu » se trouve au début du script du personnage « papillon ».

Dans le Tableau 3 nous présentons les types de construction de la notion des messages que nous avons distingués au cours de notre analyse, ainsi que le nombre d'élèves qui appartiennent à chaque catégorie. Les critères qui apparaissent sur le tableau sont les critères satisfaits pour chaque type de construction. Par exemple, bien que pour une « construction complète » nous devions avoir satisfait les deux critères, A et B, pour une « construction partielle » nous devons avoir satisfait seulement le critère B. De la même manière fonctionne le tableau pour le troisième type de construction de la notion des messages.

1 Une partie de cette analyse a été présenté au colloque < http://didinfo2016.etpe.gr/> en Grèce.

Tableau 3 : Types de construction de la notion de messages.

Types de construction	Critères satisfaits	Nombre d'élèves
Construction complète	A & B	8
Construction partielle	B	3
Construction incomplète	—	1

Comme nous pouvons observer sur le Tableau 3, la plupart des élèves de notre échantillon arrivent à construire de manière complète le concept de message. Plus précisément, 8 sur 12 résolvent correctement le problème donné en utilisant les messages de manière opérationnelle. Trois arrivent à une construction partielle et 1 arrive à construire la notion de manière incomplète.

Figure 4 : Exemple d'une construction complète.

En analysant les erreurs des élèves sous le prisme de la théorie de Mayer (1988) sur les savoirs syntaxiques et sémantiques de la programmation, nous pouvons remarquer que ceux qui arrivent à une construction partielle ou incomplète du concept des messages font des erreurs qui concernent surtout la sémantique du langage ScratchJr. Pourtant, nous avons aussi distingué un autre type d'erreur qui émerge dans les programmes des élèves en raison de la non-compréhension de l'énoncé du problème. Nous avons choisi d'appeler ce type d'erreur « erreur compréhension énoncé ».

Figure 5 : Exemple d'une construction partielle.

Dans la figure 5 par exemple, nous pouvons distinguer une « erreur syntaxique », car l'élève a ajouté « envoyer message » (opération nº 1) au script du personnage « papillon » (objet nº 2) et pas au script du bon personnage,

c'est-à-dire du personnage « enfant » (objet nº 1). En plus, il a fait une « erreur sémantique », car il a modifié la couleur du « envoyer message » et d'une telle manière que même s'il avait utilisé cette commande dans le script du personnage « enfant », les deux personnages n'auraient pas pu échanger des messages.

Figure 6 : Exemple d'une construction partielle.

Dans la figure 6 nous pouvons remarquer deux « erreurs sémantiques », une « erreur syntaxique » et une « erreur compréhension énoncé ». L'élève a fait une « erreur syntaxique », car elle a utilisé « envoyer message » dans le script du personnage « papillon », au lieu de l'utiliser seulement sur le script du personnage « enfant ». C'est une « erreur sémantique » en même temps, car il n'est pas possible au personnage qui reçoit le message d'être le personnage qui l'envoie de nouveau après. Selon la sémantique du langage ScratchJr « envoyer message » est une opération qui devrait être effectuée sur un objet. Par ailleurs, le fait qu'elle ait utilisé « envoyer message » dans le script du personnage « enfant », mais pas à la bonne position, nous montre soit qu'elle n'a pas bien compris l'annonce du problème et dans ce cas elle fait une « erreur compréhension énoncé », soit qu'elle n'a pas bien compris la sémantique de la commande « envoyer message » et c'est pourquoi elle ne comprend pas le fait que la position de la commande sur le script du personnage joue un rôle important, car elle définit le moment ou l'autre personnage va déclencher son script.

Figure 7 : Exemple d'une construction partielle.

La figure 7 nous présente une erreur qui est en même temps « syntaxique » et « sémantique », car l'élève a oublié d'utiliser la commande « envoyer message » (opération nº 1) sur le script du personnage « enfant » (objet

nº 1). Cela est une erreur syntaxique, car le concept des messages comprend deux commandes qui sont effectuées sur deux personnages différents et l'élève a utilisé seulement la commande du « quand message reçu » sur le personnage « papillon ». En ce qui concerne la sémantique, il s'agit d'une erreur, car le personnage papillon ne peut pas recevoir un message qui ne lui est pas envoyé par le personnage « enfant ».

Figure 8 : Exemple d'une construction incomplète.

Ensuite, dans la figure 8 nous pouvons distinguer quatre erreurs. D'abord une « erreur syntaxique », car l'élève a oublié d'utiliser la commande du « quand message reçu ». Après, il a fait une « erreur sémantique », car il a utilisé la commande « envoyer message » sur le script du personnage « papillon », qui devrait avoir la commande du « quand message reçu ». Les deux personnages ne vont pas communiquer. En plus, il a fait une autre erreur lorsqu'il a utilisé la commande « envoyer message » sur le script du personnage « enfant ». Il se trouve que c'est le même cas que pour la figure 6 : soit « erreur compréhension énoncé », soit « erreur sémantique ».

En analysant les erreurs des élèves nous pouvons remarquer que la plupart d'entre eux se trompent avec la commande « envoyer message » et sa position dans le script, ou avec le caractère, ici « papillon » ou « enfant », dont le script devrait la contenir. Nous pouvons supposer que la forme de la commande du « quand message reçu » aide les enfants à la poser au bon endroit – c'est-à-dire au début du script.

En faisant une comparaison entre la performance face au problème de l'évaluation et les réponses aux entretiens de post-test, nous pouvons souligner le fait qu'il existe une corrélation assez importante entre les deux, pour une partie de ceux qui arrivent à résoudre correctement le problème de l'évaluation et pour tous les élèves qui n'arrivent pas à le résoudre correctement.

Les 4 élèves dont nous avons analysé les erreurs auparavant éprouvent une série des difficultés en ce qui concerne la verbalisation de leurs connaissances autour du concept des messages. Plus précisément, ils construisent des « représentations incomplètes » des commandes « envoyer message » et

du « quand message reçu » au post-test. Parmi ceux qui arrivent à résoudre correctement le problème, 3 élèves construisent une représentation complète du « envoyer message » et 4 élèves construisent une représentation complète du « quand message reçu ». Pour les autres qui arrivent à résoudre correctement le problème d'évaluation, il n'y a pas de corrélation avec leurs réponses au post-test.

L'analyse nous a également montré que la commande « envoyer message » pose la plupart des problèmes aux élèves tant au cours de la séance d'évaluation, qu'au cours de la séance de post-test. Nous pouvons donc faire l'hypothèse que le savoir autour du « envoyer message » n'est pas complètement construit dans le cerveau des élèves et c'est pour cette raison qu'ils n'arrivent pas à bien verbaliser leurs connaissances autour de cette commande, alors qu'ils arrivent mieux pour la commande du « quand message reçu ».

En faisant une comparaison entre les réponses des élèves à l'entretien personnel avant et après l'application du scénario pédagogique, on peut remarquer le fait que la plupart de nos sujets (8 sur 12) réalisent un progrès cognitif, car leurs réponses à propos des commandes « envoyer message » et « quand message reçu » au post-test sont meilleurs que celles au pré-test. Seulement 4 élèves ne font pas de progrès.

Les difficultés des jeunes à s'exprimer autour des commandes des messages sont tout à fait normales, si on prend en compte le fait que nous avons mené seulement deux séances d'enseignement du concept, à cause de la durée courte de notre étude. Au cours de ces deux séances, nous avons fait attention à la compréhension de la fonctionnalité des messages et leur utilisation opérationnelle face aux problèmes donnés. Si on analyse du coup nos résultats par ce point de vue, nous pouvons dire que notre objectif a été atteint.

5 Discussion

Notre étude a eu un statut exploratoire et souffre de sérieuses limites. L'échantillon est d'abord de faible taille et nos observations ont été de courte durée. Pourtant, nous pensons que leur validité, concernant les difficultés rencontrées, va au-delà de notre étude.

D'autres recherches utilisant des échantillons plus importants seraient nécessaires, afin de pouvoir mieux comprendre le processus de l'appropriation du concept des messages par les jeunes élèves. Nous nous intéressons en particulier à comprendre la manière dont les élèves construisent leurs représentations autour du concept des messages. Il nous paraît intéressant aussi de comprendre pourquoi la commande « envoyer message » pose plusieurs difficultés aux élèves que la commande du « quand message reçu ».

Nous souhaitons donc examiner davantage le concept des messages, mais aussi d'autres concepts qui sont supportés pas le logiciel ScratchJr et appartiennent à la programmation événementielle, comme le concept « Toucher » (*Start on tap* en anglais) ou « Contact » (*Start on bump* en anglais). C'est pourquoi nous avons mené, au cours du dernier semestre, une étude auprès de 22 élèves de deux classes de grande section de l'école maternelle en Grèce, afin de pouvoir examiner également comment réagissent les plus jeunes face au même concept. Les résultats préliminaires de cette étude nous montrent des similarités en ce qui concerne les difficultés rencontrées et les types d'erreurs chez les plus jeunes élèves face au concept de message.

Références

Ben-Ari, M. (1996). Understanding Programming Languages (1st éd.). New York, NY, USA : John Wiley & Sons, Inc.

Clements, D. H., & Gullo, D. F. (1984). Effects of Computer Programming on Young Children's Cognition. American Psychological Association Inc., 76(6), 1051–1058.

Clements, D. H., & Meredith, J. S. (1992). Research on Logo : Effects and Efficacy. Logo Foundation, p. 15.

Cloutier, J.-F. (1988). Apport de différents paradigmes de programmation comme autant d'outils de pensée (p. 195–203). Présenté à Colloque francophone sur la didactique de l'informatique, Paris. Consulté à l'adresse <https://www.epi.asso.fr/fic_pdf/dossiers/d07p195.pdf>.

Cobb, P., Confrey, J., DiSessa, A., Lehrer, R., & Shauble, L. (2003). Design Experiments in Educational Research. Educational Researcher, 32(1), 9–13.

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should Your 8-year-old Learn Coding ? In Proceedings of the 9th Workshop in Primary and Secondary Computing Education (p. 60–69). New York, NY, USA : ACM. <https://doi.org/10.1145/2670757.2670774>.

Feurzeig, W. (2010). Toward a Culture of Creativity : A Personal Perspective on Logo's Early Years and Ongoing Potential. International Journal of Computers for Mathematical Learning, 15(3), 257–265. <https://doi.org/10.1007/s10758-010-9168-4>.

Kay, A. C. (1993). The Early History of Smalltalk. In The Second ACM SIGPLAN Conference on History of Programming Languages (p. 69–95). New York, NY, USA : ACM. <https://doi.org/10.1145/154766.155364>.

Komis, V. (2016). Une analyse cognitive et didactique du langage de programmation ScratchJr (p. 11). Présenté à Didapro6-DidaSTIC, Namur, Belgique. Consulté à l'adresse <https://didapro6.sciencesconf.org/76475/document>.

Mayer, R. E. (Éd.). (1988). Teaching and Learning Computer Programming : Multiple Research Perspectives. Hillsdale, N.J : Routledge.

Misirli, A., & Komis, V. (2016). Construire les notions de l'orientation et de la direction à l'aide des jouets programmables : une étude de cas dans les écoles maternelles en Grèce. In, F. Villemonteix, J. Béziat, G-L.Baron (Ed.), L'école primaire et les technologies informatisées.

Portelance, D. J. (2015). Code and Tell : An exploration of peer interviews and computational thinking with ScratchJr in the early childhood classroom. Tufts University. Consulté à l'adresse <https://ase.tufts.edu/DevTech/resources/Theses/DPortelance_2015.pdf>.

Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. International Journal of Technology and Design Education, 1–16. <https://doi.org/10.1007/s10798-015-9325-0>.

Resnick, M., Kazakoff, E. R., Bonta, P., Silverman, B., Bers, M. U., & Flannery, L. P. (2013). Designing ScratchJr : Support for Early Childhood Learning Through Computer Programming (p. 10). New York, NY, USA. Consulté à l'adresse <http://ase.tufts.edu/DevTech/publications/scratchjr_idc_2013.pdf>.

The Design-Based Research Collective. (2003). Design-Based Research : An Emerging Paradigm for Educational Inquiry, 32(1), 5–8.

Touloupaki, S. (2016). Analyse d'une recherche sur la pensée informatique. Consulté 24 juin 2016, à l'adresse <http://www.adjectif.net/spip/spip.php?article396>.

Touloupaki, S., Konstantinopoulou, M., Nicolos, D., Baron, G.-L., & Komis, V. (2016). Η οικοδόμηση της έννοιας του « μηνύματος » στη γλώσσα προγραμματισμού ScratchJr από μαθητές 5–7 ετών. (p. 8). Présenté à didinfo2016, Ioannina. Consulté à l'adresse <http://www.etpe.gr/custom/pdf/etpe2382.pdf>.

Yin, R. K. (2003). Case Study Research : Design and Methods (3rd Revised edition). Thousand Oaks, Calif : SAGE Publications Ltd.

Étienne Vandeput[a] & Julie Henry[b]

a. Institut Universitaire de Formation des Enseignants (IUFE), Genève, Suisse
etienne.vandeput@hotmail.com
b. Centre de recherche PReCISE – NAmur Digital Institute (NADI) – Université de Namur, Belgique
julie.henry@unamur.be

Apprendre à programmer
Comment les enseignants justifient-ils le choix d'un outil didactique ?

Résumé
De nombreux outils sont utilisés dans un contexte d'enseignement de la programmation. Mais comment ces derniers sont-ils choisis ? Quels sont les critères sur lesquels les enseignants appuient leur choix ? À travers une série d'entretiens semi-dirigés menés auprès d'enseignants d'horizons différents (secondaire, supérieur universitaire et non universitaire), des éléments de réponse sont apportés. Ainsi, un outil n'est pas choisi parce qu'il facilite l'apprentissage, mais plutôt parce qu'il est actuel, parce qu'un supérieur le souhaite ou parce qu'il est utilisé dans le monde du travail. De même, aucune réflexion d'ordre didactique n'est envisagée pour aider l'étudiant à surmonter ses difficultés. Le constat pessimiste dressé par cet article invite à envisager des recommandations à prendre en compte dans la perspective d'une introduction à la pensée informatique dans l'enseignement obligatoire.

Mots clés : enseignement de la programmation, apprentissage de la programmation, choix d'un outil, didactique

1 Introduction

À l'école obligatoire, on a très tôt voulu enseigner la programmation informatique. Mais le manque d'expérience, de repères, les carences en matière de formation des enseignants et la complexité de l'exercice ont eu raison de cette première orientation. Le vent de la pédagogie s'est assez

rapidement mis à souffler dans la direction des usages, voire de l'aide à l'apprentissage. Ce vent semble avoir tourné, du moins momentanément. Il revient dans la direction prise au tout début de l'apparition de l'informatique à l'école, mais pour d'autres raisons. L'une des principales est sans doute que, comme le prévoyaient les optimistes, tout le monde finirait un jour par utiliser les technologies numériques, mais que comme pouvaient l'imaginer les pessimistes, les utilisateurs ne pourraient guère progresser en matière d'exploitation efficace et de comportements prévoyants (sécurité en matière de vie privée, par exemple). Pour endiguer ce courant, la solution que d'aucuns proposent aujourd'hui est donc d'apprendre aux jeunes à coder arguant que cet apprentissage les rendra moins vulnérables aux effets pervers de la vague numérique. Cet argument, rejeté au début des années 80, est aujourd'hui mis en avant avec la nécessité de développer chez les jeunes une pensée qualifiée d'informatique.

La programmation n'a évidemment pas cessé d'être enseignée dans les facultés d'informatique et à la Haute École[1]. C'est actuellement à ce niveau d'études qu'une majorité d'étudiants la découvrent. C'est un défi que leurs enseignants relèvent avec des fortunes diverses. À l'heure de cet hypothétique retour au code, il nous a paru intéressant d'observer les préoccupations didactiques des enseignants dans la formation à la programmation à la fin du secondaire et au début de l'enseignement supérieur. Nous nous sommes focalisés sur un élément particulier : le choix des outils didactiques. Nous précisons, bien entendu ce que nous entendons par outils didactiques. L'idée est de tirer des enseignements utiles dans le contexte où cet enseignement de la programmation est de plus en plus souhaité à l'école obligatoire dans de nombreux pays.

2 Objet de l'étude

Sans prendre position, dans un premier temps, sur le bien-fondé ou non d'un retour au code et sans contester que la programmation soit et reste sans doute à jamais au cœur de l'informatique indépendamment de l'évolution

1 Dénomination utilisée en Belgique pour désigner des études supérieures non universitaires.

galopante de celle-ci, notre étude s'intéresse aux choix didactiques opérés par les enseignants qui sont censés enseigner les bases de la programmation. Nous espérons ainsi pouvoir tirer des leçons utiles aux enseignant(e)s qui peuvent être amené(e)s à reproduire l'expérience à un niveau inférieur. Nous limitons volontairement notre étude à la programmation impérative à cause du lien étroit qu'elle entretient avec l'algorithmique et sachant que c'est essentiellement à ce paradigme de programmation que s'intéressent les projets actuels à l'école obligatoire.

2.1 Public-cible

L'étude a été réalisée en Communauté française de Belgique sur un public-cible constitué d'une dizaine d'enseignants d'université ou de Haute École qui enseignent un cours de base en programmation. Dans le souci de compléter notre information, nous avons également interrogé un enseignant du secondaire supérieur[2], titulaire d'un cours censé donner le goût, l'envie de programmer à des élèves qui se destinent à ce type d'études. Précisons que nous ne parlons pas ici de « sensibilisation » à l'acte de programmation comme on pourrait le faire à travers des expériences liées à l'informatique débranchée ou à travers les expériences constructivistes que permettent certains univers graphiques, mais bien d'un enseignement très objectivé de la programmation.

2.2 Outils didactiques

Nous nous intéressons directement aux outils didactiques choisis pour atteindre les objectifs d'enseignement établis, non sans questionner d'abord les enseignants sur ces objectifs et sur la difficulté subjective d'enseigner/apprendre la programmation. Nous avons donné au mot « outil » l'acception la plus générale qui soit, permettant ainsi aux enseignants de s'exprimer le plus largement possible. On trouvera donc derrière ce mot, le langage, mais aussi ce qui permet de produire du code : un environnement de développement ou un simple éditeur de texte (dans les situations où on se refuse à toute assistance de génération du code, par exemple),

2 Adolescents de 16 à 18 ans.

des diagrammes, des schémas, voire des objets réels qui pourraient participer à l'élaboration d'une meilleure compréhension. Nous nous intéresserons surtout à ce qui motive ce choix. Nous pensons qu'apprendre à programmer se détache difficilement d'un vrai langage, même si nous n'excluons pas de remettre cela en question s'il s'avère que certains se contentent d'un langage de description ou d'un pseudo-langage. Et donc, en particulier, la motivation du choix du langage est un élément important de notre analyse.

3 Questions de recherche

Au départ, notre étude souhaite essentiellement fournir des informations concernant l'importance relative des outils dans le choix de l'enseignant. Nous avions aussi l'ambition de trouver des réponses à des questions telles :

- pourquoi privilégier un pseudo-langage, un langage, un environnement de développement ?
- quelles sont les priorités accordées (programmation structurée, modélisation, structures de données. . .) dans le cadre d'un tel enseignement et pourquoi ?
- y a-t-il des outils qui facilitent la compréhension de certains concepts, qui favorisent certains comportements attendus du « bon programmeur » ?
- y a-t-il des outils qui perturbent la compréhension de certains concepts, qui favorisent certains comportements peu souhaités de la part du « bon programmeur » ?

Mais ce que nous avons découvert nous oblige à faire preuve d'une certaine humilité et montre, en tous cas, que ces questions passent souvent au second plan devant les urgences auxquelles les enseignants doivent faire face, notamment en ce qui concerne la gestion du contexte dans lequel ils évoluent. Nous donnons des détails dans les conclusions et perspectives.

4 Méthodologie

La préoccupation est double : (1) comprendre le choix des enseignants ; (2) les questionner sur les concepts considérés comme délicats à maîtriser par leurs élèves/étudiants et ceux qui leur paraissent compliqués à enseigner. L'idée est également de leur demander d'expliciter les forces et les faiblesses des outils de leur choix dans ces apprentissages/enseignements.

Pour cela, nous avons mené une série d'entretiens semi-dirigés auprès de dix enseignants : un enseignant du secondaire, cinq enseignants travaillant au sein d'une haute école et quatre enseignants, au sein d'une université[3]. Un guide d'entretien détaillé nous a permis d'entamer la conversation à partir de questions plutôt générales et de puiser dans des questions de détails si ceux-ci n'apparaissaient pas dans le discours de l'enseignant(e). L'établissement de ce guide d'entretien n'a pu se faire sans une certaine garantie de bonne couverture des réponses possibles. Pour le concevoir, nous nous sommes inspirés de certaines études (Lahtinen, Ala-Mutka & Järvinen, 2005 ; Milne & Rowe, 2002) qui s'intéressent à la perception subjective des difficultés liées à la compréhension des concepts, mais aussi d'autres études (Good, 2011 ; Guzdial, 2004 ; Pears *et al.*, 2007 ; Sorva, Karavirta & Malmi, 2013) qui proposent une analyse personnelle des outils ou encore de celles qui établissent des taxonomies de ces outils (Kelleher & Pausch, 2005). Les informations tirées de ces études nous ont permis d'orienter le questionnaire pour qu'en quelques questions et sous-questions bien calibrées, nous puissions établir rapidement la cohérence du choix de l'outil dans le contexte décrit. L'entretien était axé sur quatre questions principales :

- Quelles difficultés essentielles identifiez-vous dans l'apprentissage/l'enseignement de la programmation ?
- Quels outils didactiques utilisez-vous pour enseigner la programmation ?
- Quels objectifs poursuivez-vous dans votre cours de programmation ?

3 Pour garantir l'anonymat des enseignants interviewés, chacun d'eux s'est vu attribuer un nom fictif. Ce dernier, associé à un extrait présent dans cet article, permet notamment de mentionner le genre de l'auteur. Son niveau d'enseignement et son ancienneté sont également précisés.

- En quoi les outils choisis vous permettent-ils de rencontrer les objectifs fixés et les difficultés supposées ?

Pour chacune de ces questions, nous disposions d'une série de sous-questions nous permettant de compléter l'information lorsque celle-ci manquait de détails. À titre d'exemple, les sous-questions de la première question étaient :

- Quels sont les concepts (notions) complexes à enseigner ?
- Quels sont les concepts (notions) difficiles à maîtriser (comprendre) ?
- Pourriez-vous classifier ces difficultés, en traduire l'importance ?
- Au-delà des concepts, quelles sont les savoir-faire à acquérir ?
- Quels sont ceux dont l'acquisition n'est pas triviale ?

5 Analyse de données

Nous traiterons ces données en plusieurs points. À chaque fois, nous tenterons de suggérer des solutions didactiques.

5.1 Outils et motivation

Le premier concerne le rapport que peut entretenir l'étudiant, non pas à la programmation elle-même, mais aux méthodes et aux règles de bonne pratique que requiert cette activité. En effet, les étudiants ne perçoivent pas facilement l'intérêt de s'imposer une discipline de travail. La nécessité d'obtenir rapidement un résultat, et singulièrement un programme, prime sur le souci de qualité, voire de bon fonctionnement de celui-ci.

> […] leur demander d'utiliser un outil formel pour prouver qu'un programme fonctionne… je ne pense pas qu'ils comprennent l'intérêt. (Antoine, université, moins de 5 ans)

Ce n'est que dans les situations de constat manifeste des problèmes (le programme ne fonctionne pas ou ne donne pas les résultats attendus) que celles-ci peuvent trouver une part de justification. Et ce constat ne se fait

pas, par exemple, lorsque le programme semble fonctionner avec les données choisies, ce qui renforce la difficulté.

> […] ceux qui sont convaincus de savoir déjà programmer nous disent « ça sert à rien » jusqu'au jour où ça ne marche plus. (Antoine, université, moins de 5 ans)

Face à cette difficulté, on peut imaginer deux remèdes. Le premier est évidemment d'habituer l'étudiant, beaucoup plus tôt dans son cursus, à ce genre d'attitude, à ce savoir-être. Cela plaide évidemment en faveur d'une introduction de la programmation à l'école, sous une forme ou sous une autre, pourvu qu'elle inclue cette dimension de rigueur. Les enseignants eux-mêmes le suggèrent prétextant, sans doute à juste titre, devoir agir trop tard et avec des moyens inadaptés.

> C'est un peu gênant si nous, dans le supérieur, on reprend des choses qui sont du niveau primaire […] (Claire, haute école, plus de 15 ans)

Ainsi, des outils qui apparaissent comme pertinents pour apprendre à programmer se révèlent inadaptés à l'âge mental des étudiants.

> Je crois que nos étudiants seraient vexés si on choisissait des outils qui sont clairement à destination des petits et qu'on les utilisaient avec eux […] ils se diraient « mais ils nous prennent pour qui ceux-là ». (Claire, haute école, plus de 15 ans)

Le second remède, qui est d'ailleurs tout aussi valable dans la première situation, consisterait à choisir des problèmes (défis) types qui conduisent très rapidement à l'échec si cette rigueur n'est pas de mise. Mais les entretiens laissent penser que les enseignants ont peu l'occasion d'y réfléchir.

D'autres commentaires nous font penser que, dans des contextes où les aspects techniques sont très présents, la programmation est mieux acceptée peut-être parce qu'elle constitue une activité plus intellectuelle.

> Moi, je donne des cours de théorie aussi : réseau, etc. et je trouve que la programmation passe beaucoup mieux que ces cours-là parce que les élèves sont plus intéressés. (Nathalie, haute école, plus de 20 ans)

Lorsqu'ils pensent activité de programmation, beaucoup d'étudiants la considèrent comme une occasion de mettre en œuvre des projets qui leur plaisent. Certains outils sont alors perçus comme des entraves ou des éléments perturbateurs.

> Avant qu'ils ne conçoivent le programme en C, un des professeurs de labo (donc sur machine), demande aux étudiants de réaliser sur papier, le programme en pseudo-code. Moi je ne suis pas aussi rigoureux parce que je vois que ça les embête. (Tony, haute école, plus de 20 ans)

5.2 La question du choix

Rappelons que nous souhaitions donner à la notion d'outil l'acception la plus large possible et y inclure tous les éléments qui peuvent constituer une aide ou un support à l'apprentissage de la programmation. On s'intéresse donc à des outils matériels autant qu'à des outils logiciels et en particulier, au(x) langage(s) qui permettent de transformer le fruit d'une réflexion algorithmique en un programme exécutable. Il est à remarquer que certains concepts, voire certaines étapes de la programmation sont vus par certains comme des outils.

> Les spécifications sont à la fois compétences et outils. Même chose pour invariants… c'est plutôt un outil qui est censé aider les étudiants à bien faire leurs boucles […] (Antoine, université, moins de 5 ans)

Mais nous les avons exclus de la réflexion en tant que tels. Le moins que l'on puisse dire, c'est qu'ils sont d'une très grande variété. On ne retrouve pas vraiment de consensus à ce propos. Les outils choisis sont en lien avec la stratégie d'enseignement, voire les impératifs de formation. Parmi les outils évoqués, le plus dépouillé en quelque sorte est la feuille de papier. Elle témoigne d'une volonté délibérée de l'enseignant de faire réfléchir les étudiants. Ce témoignage montre que l'attitude de celui-ci n'est pas toujours le repli face à un outil austère.

> Le meilleur outil de l'informaticien c'est la feuille de papier, le crayon et la gomme. Ça reste dans l'optique « je réfléchis avant de coder ». La plupart des exercices sont faits sur papier et ils ont des séances de laboratoire qui leur permettent de se confronter à la machine. (Nicolas, université, plus de 5 ans)

À l'autre bout du spectre, on trouve le langage. Certains considèrent qu'apprendre à programmer, c'est maîtriser un langage de programmation.

> Moi dans les cours, je n'utilise pas d'outils intermédiaires, je n'utilise pas de représentations graphiques, de structures algorithmiques ou quoi que ce soit d'autre, je reste dans le concret, c'est le code. (Cédric, université, plus de 10 ans)

Et puis, il y a ceux qui ne mettent pas vraiment de frontières entre algorithmique et langage.

> En première année, ils ont un cours d'algorithmique fort orienté vers le C. Les exemples, c'est déjà presque de l'écriture C. (Nicolas, université, plus de 5 ans)

Entre les deux, on observe presque autant de conceptions de l'enseignement de la programmation que d'enseignants. Certains font travailler les étudiants dans des environnements de développement sophistiqués. Le choix est guidé de diverses manières.

> Pour Eclipse, j'ai eu le choix. Le directeur m'a dit : « tu peux utiliser soit Eclipse, soit Netbeans ». Il y a une liberté au niveau du choix des outils. (Tony, haute école, plus de 20 ans)

Ce type de choix effectué plus volontiers dans les Hautes-Écoles que dans les Universités trouve souvent sa justification dans la proximité de la formation avec le monde du travail qui est d'ailleurs parfois partie prenante de cette formation.

> Je suis allé voir ceux qui étaient les plus demandés sur le marché. Je les ai proposés au directeur qui a demandé aux consultants externes ce qu'ils en pensaient et ils ont validé le choix. […] ce ne sont pas les profs en interne qui ont choisi les outils, c'est une proposition faite qui a été validée par les extérieurs, par les gens du métier. (Tony, haute école, plus de 20 ans)

Pour beaucoup d'enseignants de Haute École, le choix des outils ne peut se faire sans un regard vers les entreprises. Il est plus important que l'étudiant connaisse l'outil, même s'il ne s'en sert pas de manière optimale.

> De toute façon, on est quand même bien obligé de passer par les outils les plus courants actuellement. On ne peut pas faire autrement. (Nathalie, haute école, plus de 20 ans)

Quand un choix est fait par la communauté des enseignants, il est parfois revu à la lumière des commentaires des sociétés susceptibles d'engager ces étudiants.

> C'est vrai qu'on se disait qu'avec le C++, une fois qu'ils ont fait ça (*sic*), ils peuvent passer partout. On avait aussi des retours des sociétés qui disaient « écoutez, le C# est quand même fort utilisé et quand ils viennent chez nous, ils ne le connaissent pas ». Donc, nous avons décidé il y a trois ans de changer et de passer au C#. (Nathalie, haute école, plus de 20 ans)

La préoccupation est moins présente à l'Université, où l'objectif est davantage tourné vers la rigueur que vers un choix de langage. Néanmoins, lorsque le cours s'adresse à des étudiants dont la finalité des études n'est pas l'informatique, on peut la retrouver sous des justifications assez proches.

> […] je pense qu'au final le C garde son intérêt, notamment pour les ingénieurs civils puisque c'est de toute façon une exigence des électriciens. Ceux qui iront en électricité ou même en méca ont besoin du C à un moment donné pour les systèmes embarqués etc. Donc autant voir du C. (Nicolas, université, plus de 5 ans)

Sans se préoccuper de l'acte de programmation lui-même, le regard posé est déjà celui d'une informatique utile.

> Le langage C est un choix imposé par la Faculté de Sciences. La formation en math est très appliquée et donc, à partir de la troisième année, ils sont censés utiliser pas mal d'outils informatiques, pas mal de librairies, etc. Une bonne partie des librairies qu'ils utilisent est écrite en C. Et donc, il y avait le souhait de voir du C dès le début pour justement enlever une difficulté chez les étudiants. (Cédric, université, plus de 10 ans)

Quand des difficultés surgissent, il est fréquent que les enseignants changent d'outils. Mais ils sont désemparés quand les changements qu'ils opèrent ne donnent pas les résultats escomptés. Ils constatent alors que la motivation n'est pas nécessairement liée à la rigidité potentielle de ceux-ci.

> Pendant des années, on a fait du Pascal et on a cru que les étudiants n'accrochaient pas, parce qu'ils avaient l'habitude d'un monde informatique très convivial. Et nous, on leur proposait une petite fenêtre sous DOS, en noir, où les accents ne passaient pas correctement ou bien c'était beaucoup de chipotage. On s'était dit que c'était ça qui freinait les étudiants. Donc on a voulu faire quelque chose de plus joli en faisant du HTML. On voyait la même matière qu'avant au travers du JavaScript : les boucles, les conditionnelles. . . et ça n'a rien changé du tout. C'est toujours aussi difficile alors qu'on a essayé de faire des choses beaucoup plus jolies. Ça n'a pas été la solution. (Claire, haute école, plus de 15 ans)

Au bout du compte, il semble que les outils les plus élémentaires (papier, crayon, gomme) soient choisis par ceux qui veulent encourager la réflexion. Les langages et environnements de développement sont surtout choisis pour que l'étudiant y soit accoutumé dans le monde du travail ou de formation qui l'attend sans qu'ils lui permettent nécessairement d'acquérir de bons réflexes de programmation. Une chose semble certaine, le choix du langage n'est pas vraiment guidé par le souci d'apprendre à programmer.

5.3 Le rapport des outils à la didactique

Au vu de ce qui précède, on peut se demander quelle place occupe l'intérêt didactique des outils dans le choix de ces derniers. D'autre part, les contraintes qui pèsent sur l'enseignement de la programmation dans l'enseignement supérieur ne semble plus pouvoir laisser beaucoup de place à cette préoccupation, ce qui serait un argument de poids dans la nécessité d'enseigner à tout le moins l'algorithmique dans l'enseignement secondaire. Nous traitons de cette question dans les conclusions et les perspectives. Nous analysons ici, dans le discours des enseignants, s'ils établissent un lien entre les outils et leur rôle facilitateur ou non de l'enseignement de la programmation. Cette question est à mettre en rapport avec la représentation que se font les enseignants des problèmes liés à cet enseignement. Nous la décrivons en deux ou trois points.

Il y a d'abord la manière dont ils jugent leur public. Elle est parfois dure et sans nuance.

> […] il y a des étudiants très bons, les 30 % qui réussissent… eux, je pense qu'on peut faire n'importe quoi avec eux, ça marchera toujours. (Claire, haute école, plus de 15 ans)

Elle peut traduire un certain fatalisme.

> […] il y a des élèves qui sont doués pour ça et il y en a qui le sont moins. (Étienne, secondaire supérieur, plus de 20 ans)

Elle est parfois mieux identifiée sans qu'on laisse entrevoir de solution au problème sinon celle du temps qui passe.

> Enseigner la programmation reste quelque chose de difficile puisque, soit les étudiants qui perçoivent directement comment faire, ils ont la logique et puis on a en a qui restent bloqués. […] j'attends que mes étudiants mûrissent. Il faut leur laisser le temps. (Julie, haute école, plus de 15 ans)

Il y a la manière de traiter les problèmes lorsqu'ils se présentent. Devant un constat totalement négatif, on s'oriente fréquemment vers des modifications d'outils, de stratégies qui ne s'avèrent pas nécessairement payantes, les raisons du découragement ou du manque d'intérêt étant parfois mal identifiées.

> Aucun outil ne nous satisfait actuellement. […] nous cherchons toujours pour les étudiants plus faibles et qui sont très bons dans les autres matières… Nous avons déjà changé souvent nos méthodologies parce que nous étions persuadés que le Pascal ne fonctionnait pas parce que ce n'était pas joli (*sic*). […] le parcours est long et ça passe par des apprentissages qui sont nettement moins sexy. (Claire, haute école, plus de 15 ans)

Dans les meilleurs cas, on se rend compte que les outils sont limités, mais c'est plutôt rare et la manière d'exprimer la difficulté ne laisse pas forcément entrevoir de manière de contourner ces limites.

> Quand elle *(NDLA : ma collègue)* travaille en pseudo-code, il y a des choses qu'elle sait faire, que je ne sais pas faire en C. (Julie, haute école, plus de 15 ans)

Parfois on devine tout de même une possibilité de contourner la difficulté.

> On peut déjà faire pas mal de choses, pas tout malheureusement, […] avec Algobox et donc quand on programme, enfin quand on traite des problèmes mathématiques, en général on arrive à les coder après par Algobox. Il y a parfois certaines résolutions où je leur dis « là écoutez c'est la limite de l'outil, vous ne pourrez pas le faire tel quel avec Algobox », mais voilà. (Anne, haute école, moins de 5 ans)

Lorsque les avantages d'un outil sont mis en évidence, c'est plutôt pour dégager des facilités d'usage ou un intérêt pédagogique, même si on devine qu'il y a aussi des bénéfices du côté de la facilité de compréhension et de l'aide à la programmation.

> […] on passe très vite à un pseudo-code vraiment en français car mes étudiants ont du mal parfois avec l'anglais, […] un pseudo-code classique « tant que. . . faire » avec des choses très compréhensibles et alors qu'au début, je les orientais vers le Pascal, finalement je trouve que ça amène des difficultés et pas beaucoup d'avantages alors je travaille maintenant avec Algobox qui est très très facile à utiliser et qui est très adapté. (Anne, haute école, moins de 5 ans)

En gros, les enseignants ne choisissent pas un outil parce qu'il va les aider à lever une difficulté, mais plutôt parce qu'un autre outil ne semble pas donner de bons résultats ou parce qu'ils pensent que le fait de ne pas l'avoir rencontré risque de constituer un obstacle dans la vie professionnelle future de leurs étudiants.

La question du rapport des outils à la didactique semble perdre son sens dès lors que la question de la didactique n'est pas posée. La programmation s'enseigne sur base d'une expérience personnelle, mais on sent l'absence

cruelle de réflexion sur la manière de solutionner les problèmes de compréhension bien qu'on le souhaite ardemment. Ce manque est compréhensible dès lors que la formation des informaticiens n'inclut pas la possibilité de suivre une filière didactique et donc d'identifier un lieu où cette réflexion, cette recherche peut avoir lieu.

Bref, il est difficile de qualifier les outils dont on nous a parlé de didactiques, dès lors qu'il semblent davantage générer des problèmes que d'apporter un support aux difficultés des étudiants à gérer l'activité de programmation.

6 Conclusion

Les outils sont choisis en fonction de toute une série de critères de laquelle la facilité d'apprentissage semble absente. On choisit un outil parce que son dépouillement est considéré comme formateur, parce qu'une Faculté le souhaite au nom d'intérêts propres, parce que le monde du travail le demande, etc. Nous n'avons pas entendu parler d'un outil choisi parce que, du point de vue de sa conception, il facilitait l'apprentissage de la programmation. Que du contraire, certains langages comme Pascal qui ont longtemps été considérés comme pédagogiques sont rejetés parce que trop vieux et se voient préférer des langages plus actuels, même si ceux-ci présentent des caractéristiques qui pourraient être considérées comme invalidantes. Par exemple, l'absence de typage avec Python a été soulignée sans que cela paraisse être une difficulté.

L'étude que nous avons menée a été quelque peu polluée par un problème qui est bien identifié. L'étudiant qui entame des études supérieures et qui est confronté à un cours de programmation n'y est pas correctement préparé. Plusieurs phénomènes sont mis en avant : (1) il n'a pas une représentation correcte de la discipline, voire du métier qui peut en découler (Greening, 1998 ; Grover, Pea & Cooper, 2014 ; Grover, Rutstein & Snow, 2016 ; Hewner, 2013 ; Hewner & Guzdial, 2008) ; (2) il a un vécu numérique qui n'est pas nécessairement un capital car les représentations qu'il se fait de certains concepts informatiques sont mauvaises ou biaisées ; (3) et c'est sans doute un élément-clé, il n'a pas vécu (ou si peu), lors de ses études secondaires, de situations d'apprentissage qui lui fassent

correctement percevoir le contexte dans lequel il va être baigné et notamment, celui de systèmes qu'il devra lui-même concevoir et qui fonctionnent de manière formelle, autrement dit très différente de ce qu'il connaît en tant qu'être humain. Ces systèmes exigent rigueur et esprit logique et il semblerait que ces deux dimensions soient beaucoup trop peu développées à l'école secondaire/obligatoire.

Les enseignants du supérieur se retrouvent donc devant des difficultés qu'ils n'ont pas prévues. Face à ces difficultés, et souvent dans l'urgence de devoir obtenir des résultats, ils parent au plus pressé. Nous avons trouvé fort peu d'enseignants qui soient vraiment satisfaits du travail qu'ils font. Mais pour expliquer les difficultés rencontrées, ils cherchent assez souvent des explications dans le passé des étudiants, dans leur vécu comme dans la formation qu'ils ont ou pas reçue. Il est donc remarquable de constater qu'une réflexion d'ordre didactique n'est jamais ou alors très rarement envisagée. L'enseignant n'évoque pas de stratégie pour faire mieux comprendre un concept qui pose problème, pour autant que celui-ci ait été identifié.

> On voit de la théorie avec eux, mais on voit la théorie sur des exemples et on leur donne après plutôt des exercices à faire. (Nathalie, haute école, plus de 20 ans)

On compte davantage sur la répétition des situations, une sorte de drill, pour atteindre la compréhension.

> Il faut donner beaucoup d'exercices pour qu'ils comprennent. (Nathalie, haute école, plus de 20 ans)

Lorsque surgit une difficulté, elle est parfois supprimée et donc évitée ou bien elle donne lieu à un changement d'ordre pédagogique. Faire autre chose, autrement…

> Honnêtement, je ne suis pas certain de maintenir les invariants dans le cours dans les années qui viennent parce qu'au fond, quel est l'intérêt si de toute façon ils ne comprennent pas du tout… est-ce que je ne perds pas mon temps sur ce concept-là ? (Antoine, université, moins de 5 ans)

Même en fin de secondaire, lorsque des cours existent, la difficulté d'apprentissage conduit souvent à un aveu d'impuissance.

> Certains élèves […] inversent la partie gauche et la partie droite de l'affectation. Ça, c'est vrai que j'ai déjà vu plusieurs fois. […] je leur ai déjà dit et ils refont l'erreur

> après, comme si dans leur tête il y avait quelque chose qui ne se passait pas bien… Je ne sais pas trop expliquer. Ça ne se corrige pas toujours aussi facilement que ça. (Étienne, secondaire supérieur, plus de 20 ans)

Le vrai problème ne serait-il pas que l'initiation à la programmation arrive bien trop tard et que les enseignants du supérieur ne bénéficient pas de conditions idéales pour exercer leur métier comme, par exemple, apprendre aux étudiants à concevoir des systèmes efficaces, sachant qu'ils ont déjà une bonne idée de ce qu'est programmer.

> On est fort démuni si on veut faire quelque chose pour adultes qui ressemble quand même à quelque chose pour adultes. Le problème aussi c'est que toutes ces matières-là sont enseignées par des professeurs pour qui ce n'est pas difficile. […] on tombe des nues à voir la difficulté qu'ont les étudiants de comprendre un « si alors sinon ». Pour tous les profs […] c'est quelque chose qui n'a jamais posé problème. Donc il y a une fracture entre le niveau des étudiants et le niveau des professeurs qui leur donnent cours. (Claire, haute école, plus de 15 ans)

Ce constat pessimiste nous invite à envisager, sinon des solutions, une série de recommandations qui pourraient être prises en compte dans la perspective d'une introduction à la pensée informatique dans l'enseignement obligatoire, pour autant qu'elle soit à l'ordre du jour.

7 Perspectives

Il s'avère que les questions posées sur le choix des outils ont souvent été occultées par une énorme difficulté. Beaucoup d'enseignants sont amenés à donner un cours requérant des compétences fondamentales, et d'ailleurs pas forcément liées à l'informatique de manière directe, mais qui sont des compétences requises pour pouvoir développer un cours à ce niveau dans de bonnes conditions. D'une certaine manière, cette situation rappelle celle qu'on a connue voici une trentaine d'années, dans des cours et des formations censés apprendre un usage pertinent des outils progiciels. Alors qu'on visait à développer une méthodologie bien établie de conception des produits, on se heurtait à des problèmes et des besoins beaucoup plus basiques tels un manque de familiarisation avec le clavier, des difficultés à gérer les productions et à les organiser en fichiers et en dossiers, etc. Il en résultait

souvent, pour l'enseignant comme pour l'apprenant, de la frustration, de l'incompréhension et du découragement.

> Tout ce qui est fichier curieusement aussi ils ont du mal. On s'attendrait *a priori* à ce que les jeunes qui sont réputés capables numériquement s'en sortent… et non, il y en a quand même pas mal qui ne savent pas ce que c'est un dossier, qui savent à peine ce que c'est un fichier et qui confondent les deux. (Antoine, université, moins de 5 ans)

Ce sont ces sentiments qui prévalent aussi dans le discours des enseignants. L'absence de compétences fondamentales (au sens où il est difficile de construire quelque chose sans elles) engendre le plus souvent un désintérêt de l'étudiant pour les activités qu'on lui propose. Parmi ces compétences, on relève essentiellement une capacité d'abstraction qui n'est pas acquise par tous au sortir des études secondaires, mais aussi et surtout, un manque d'habitude à traiter des problèmes ou des tâches de manière systématique, le mot « système » prenant ici tout son sens. Si les jeunes sont habitués aux ellipses, aux raccourcis de langages, aux sous-entendus, ils ont rarement sinon jamais rencontré des situations de travail ou d'apprentissage qui leur imposent une discipline semblable à celle que nécessite la programmation (au sens large) des systèmes informatiques et qui ne peut se justifier que dans la qualité finale d'un produit. Ce manque de discipline ou plutôt de perception correcte de la manière dont fonctionnent ces systèmes est probablement responsable d'un énorme bouchon d'incompréhension qui pourrait sauter en incluant, dans une éducation numérique au secondaire, une série d'activités bien pensées dans ce sens. Et les idées ne manquent pas en la matière. Ce qui manque, c'est probablement l'intégration d'un programme bien élaboré dans un cursus accueillant (entendez : qui offre un nombre d'heures suffisant). Espérons que les choses finiront par bouger dans ce sens.

Cela dit, dans tous les contextes où l'informatique et la programmation sont rendus obligatoires à l'école du même nom, il ne faudrait pas rater cette occasion de faciliter une transition vers des études dans le domaine. Faire des élèves, à ce stade, des programmeurs hors pair est probablement moins urgent que de les « éduquer » à la maîtrise de ces systèmes. Et puis, ce qui n'apparaît pas clairement dans le discours, c'est ce réflexe qui conduit l'enseignant à analyser une situation-problème et à tâcher de la solutionner par une approche didactique différente. Trop souvent, c'est l'incompréhension qui règne et une certaine maladresse dans la manière d'envisager des

solutions. Mais comme nous l'avons dit, c'est probablement à un niveau inférieur que ces difficultés d'apprentissage doivent être abordées.

N'empêche, pour revenir à la question principale de cette étude, que peu d'éléments viennent corroborer l'idée d'un choix d'outils pour leurs qualités didactiques.

Références

Good, J. (2011). Learners at the wheel : novice programming environments come of age. *International Journal of People-Oriented Programming (IJPOP)*, *1*(1), 1–24.

Greening, T. (1998). Computer science : through the eyes of potential students. In *Proceedings of the 3rd australasian conference on computer science education* (pp. 145–154).

Grover, S., Pea, R. & Cooper, S. (2014). Remedying misperceptions of computer science among middle school students. In *Proceedings of the 45th acm technical symposium on computer science education* (pp. 343–348).

Grover, S., Rutstein, D. & Snow, E. (2016). What is a computer : What do secondary school students think ? In *Proceedings of the 47th acm technical symposium on computing science education* (pp. 564–569).

Guzdial, M. (2004). Programming environments for novices. *Computer science education research*, *2004*, 127–154.

Hewner, M. (2013). Undergraduate conceptions of the field of computer science. In *Proceedings of the ninth annual international acm conference on international computing education research* (pp. 107–114).

Hewner, M., & Guzdial, M. (2008). Attitudes about computing in postsecondary graduates. In *Proceedings of the fourth international workshop on computing education research* (pp. 71–78).

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming : A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, *37*(2), 83–137.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. In *Acm sigcse bulletin* (Vol. 37, pp. 14–18).

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, *7*(1), 55–66.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., . . . Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, *39*(4), 204–223.

Sorva, J., Karavirta, V. & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, *13*(4), 15.

Index des auteur·e·s

A
Adam, Michel 273

B
Baron, Georges-Louis 291, 303
Bers, Marina Umaschi 21
Bétrancourt, Mireille 167
Buchs, Didier 219
Bugmann, Julien 257

C
Caselli, Damien 245
Chessel Lazzarotto, Frédérique 117
Collard, Anne-Sophie 61

D
Daoud, Moncef 273
Debled-Rennesson, Isabelle 99
Delmas-Rigoutsos, Yannis 31
Desprez, Thibault 245
Dillenbourg, Pierre 17
Drot-Delange, Béatrice 199
Drouillon, Frédéric 83
Dumas, Bruno 61

F
Févotte, Philippe 99
Françoise, Tort 199
Frison, Patrice 273

G
Grandbastien, Monique 99

H
Henry, Julie 61, 129, 325
Hernalesteen, Alyson 61

K
Kalaš, Ivan 23
Karsenti, Thierry 257
Komis, Vassilis 291, 303

L
Langlois, David 99

N
Nogry, Sandra 235
Noirpoudre, Stéphanie 245

O
Ouassa Kouaro, Monique 151
Oudeyer, Pierre-Yves 245

P
Pauty-Combemorel, Christelle 187

R
Racordon, Dimitr 219
Roy, Didier 245

S
Segonds, Théo 245
Smal, Anne 129

T
Tanoh, Hélène 99
Tasso, Florent 151
Tissot, Sylvie 167
Touloupaki, Sévastiani 303
Tsourapi, Chryssa 291

V
Vandeput, Étienne 325