Pensée informatique : points de vue contrastés

Béatrice Drot-Delange, Jean-Philippe Pellet, Yannis Delmas-Rigoutsos, Éric Bruillard

Pensée informatique : points de vue contrastés

▶ **Béatrice DROT-DELANGE** (INSPÉ Clermont-Auvergne),
Jean-Philippe PELLET (HEP Vaud),
Yannis DELMAS-RIGOUTSOS (Université de Poitiers),
Éric BRUILLARD (EDA, Université de Paris)

■ **RÉSUMÉ** • Issu d'une table ronde organisée lors du colloque Didapro 7 – DidaSTIC à Lausanne en février 2018, ce texte présente différentes analyses contrastées autour de la notion maintenant très populaire de pensée informatique. Il fait le point sur les possibles définitions de la pensée informatique et sur les nombreuses discussions qui lui ont été consacrées. Il adopte trois points de vue : un point de vue historique, un point de vue informatique et un point de vue épistémologique. Le lien avec l'apprentissage de la programmation ou plus largement avec l'apprentissage de l'informatique est également discuté.

■ **MOTS-CLÉS** • pensée informatique, programmation, apprentissage.

■ ***ABSTRACT*** • *Resulting from a round table organized during the Didapro 7 – DidaSTIC conference in Lausanne in February 2018, this text presents various contrasting analyses around the now very popular notion of computational thinking. It reviews the possible definitions of computational thinking and the many discussions that have taken place about it. It adopts three points of view: a historical point of view, a computer-scientist point of view and an epistemological point of view. The link with learning computer programming or, more broadly, with learning informatics is also discussed.*

■ ***KEYWORDS*** • *computational thinking, computer programming, computer science.*

Béatrice DROT-DELANGE, Jean-Philippe PELLET,
Yannis DELMAS-RIGOUTSOS, Éric BRUILLARD

1. Introduction

Lors du colloque Didapro 7 – DidaSTIC, qui s'est tenu à Lausanne en février 2018, une table ronde sur la « pensée informatique » a été organisée. Elle a suscité beaucoup d'intérêt parmi les participants et de nombreux échanges ont été amorcés. Comme les actes de la conférence avaient été conçus avant celle-ci (Parriaux *et al.*, 2018), aucun écrit ne rend compte de ce qui a été présenté et discuté, ce qui nous a semblé dommage, notamment au vu des réactions des participants. Au plan international, la notion de *computational thinking* est maintenant abondamment reprise. À titre d'illustration, une page de blog intitulée *Computational Thinking Papers* (https://csedresearch.wordpress.com/computational-thinking/) liste des articles de recherche et des rapports sur cette notion. En se limitant aux articles dont le titre contient le terme *computational thinking*, elle rapporte près de cent articles rien que sur 2018. Côté français, il n'y a pas encore de texte de synthèse autour de cette notion et les références sont rares et éparses. C'est pourquoi nous avons pensé que la table ronde pouvait servir de base à l'élaboration d'un tel texte pour le monde francophone, sur la notion de pensée informatique.

La table ronde posait deux questions, sur lesquelles chaque intervenant a apporté son éclairage. La structure du texte reprend cette trame. La première concernait la notion de pensée informatique, comment la définir et la caractériser. À cette question correspond une partie, constituée par les points de vue des trois participants, que l'on complète par une mise en perspective avec la littérature de recherche en langue anglaise. La seconde interrogeait les implications pour l'enseignement de cette « pensée ». Nous avons décidé de synthétiser les différentes interventions en un texte commun, qui ne distingue plus les différents points de vue, très largement convergents. Enfin, ce texte se termine par quelques éléments prospectifs, tentant d'aller au-delà de la pensée informatique.

2. Qu'est-ce que la pensée informatique ?

Comme nous l'avons mentionné, beaucoup de textes font maintenant référence au *computational thinking* ou à la pensée informatique. Mais comment peut-on la définir ? Cette première section confronte trois éclairages complémentaires : un point de vue plutôt historique porté par Béatrice Drot-Delange, un point de vue d'informaticien détaillé par Jean-Philippe Pellet et enfin un point de vue épistémologique présenté par Yannis Delmas-Rigoutsos.

2.1. Un point de vue historique (Béatrice Drot-Delange)

Les débats, récurrents, sur la nécessité ou non d'introduire l'informatique dans les programmes scolaires mettent en avant des arguments bien connus : considération des besoins de la société pour son industrie liée à l'informatique, renouvellement de l'attractivité des filières scientifiques de l'enseignement supérieur, etc. Ces arguments partagent les caractéristiques de se situer à un niveau macro et à moyen et long terme. D'autres arguments concernent les bienfaits supposés que pourrait apporter cet enseignement aux élèves eux-mêmes, de manière plus ou moins directe. Il s'agit de former les élèves au monde dans lequel ils évoluent et évolueront, en leur donnant les prémices d'une culture informatique et numérique. Les transferts de compétences supposés sont aussi souvent mentionnés, qui s'opéreraient de l'apprentissage des notions informatiques vers d'autres domaines, comme les mathématiques. Les débats sur la pensée informatique relèvent de cette logique de transférabilité.

La question d'une forme de pensée singulière est présente dès les années 1980, lors des débats sur l'introduction d'un enseignement d'informatique en France. Arsac souligne que :

> *« l'informatique s'appuie sur des méthodes de pensée originales dont l'apport est enrichissant et la valeur culturelle certaine. »* (Arsac, 1981).

Faisant l'historique de la méthode informatique, il précise que :

> *« C'est vers 1965 qu'a commencé à émerger l'idée que l'informatique était plus qu'un phénomène technique, et qu'elle reposait sur une forme de pensée originale qui lui appartenait en propre. Mais c'était alors une intuition plus qu'une évidence tirée des faits. »* (*ibidem*).

En 1983, les orientations décidées par la direction des écoles concernant l'introduction de l'informatique à l'école primaire mentionnent les trois finalités générales de cet enseignement : éveil humain et social, éveil technologique et éveil logistique (Boulle, 1984). Cette dernière finalité met l'accent sur le logiciel et la pensée logistique ou algorithmique.

Un consensus semble s'être élaboré parmi les partisans d'un enseignement de l'informatique au primaire et au secondaire sur l'idée

qu'apprendre l'informatique forme les élèves à une pensée singulière liée à la résolution de problèmes. Rogalski insiste ainsi sur l'apport de l'informatique à la résolution de problèmes dans d'autres disciplines grâce à ses outils théoriques (les concepts), les outils techniques renvoyant à l'environnement informatique.

> *« C'est cet aspect de l'informatique comme outil de pensée qui a été mis en avant dans l'introduction institutionnelle d'un enseignement de l'informatique dans le second cycle des lycées. L'initiation de culture générale à l'informatique implique ainsi une part importante de modélisation de situations-problèmes. »* (Rogalski, 1987, p. 7).

Les élèves seraient alors à même de transférer cet apprentissage de la modélisation pour la résolution de problèmes à d'autres domaines. Toutefois, la revue de littérature menée par Baron (1990) souligne que ces transferts de compétences ne sont pas prouvés lors d'expérimentations cherchant à les mesurer.

Dans un article de 2006, Jeannette Wing relance l'intérêt pour la pensée informatique, qu'elle désigne sous l'expression *computational thinking*. Elle précise dans cet article que « *adopter un mode de pensée informatique conduit à résoudre des problèmes, à concevoir des systèmes et à comprendre le comportement humain différemment, en s'appuyant sur les concepts fondamentaux de la discipline informatique et en y incluant une panoplie d'outils intellectuels qui reflètent l'étendue de la science qu'est l'informatique.* » (Wing, 2006).

Elle propose, en guise d'approche de la pensée informatique, une liste d'attitudes et de compétences permises par l'adoption d'une pensée informatique (l'adaptation en français de son texte est disponible en ligne à https://interstices.info/la-pensee-informatique). Elle prend soin de distinguer pensée informatique et apprentissage de la programmation, ce qui n'est pas toujours le cas des tenants de la pensée informatique.

> *« La programmation permet d'implémenter la pensée informatique, mais celle-ci ne s'y réduit pas. (...) Il s'agit bien d'une pensée, pas simplement de calcul mécanique au sens de routinier. (...) La pensée informatique est un moyen mécanique pour les humains de résoudre des problèmes. »* (Wing, 2006, p. 35)

Elle reviendra à plusieurs reprises sur ce texte afin de le préciser, l'engouement qu'il a suscité n'ayant d'égal que les interprétations

différentes auxquelles il a donné lieu. Ainsi en 2010, elle précise qu'il s'agit de savoir formuler un problème, pas seulement dans le domaine des mathématiques, mais dans tous les domaines de la vie, qui admette une solution informatique. Il s'agit selon elle de la nouvelle littératie du XXI^{e} siècle, qui consiste à discerner ce qui relève d'une solution informatique ou non, à choisir les moyens de la mettre en œuvre, à appliquer ou à adapter les outils ou techniques à de nouveaux usages, etc. En 2014, elle insiste en disant que la pensée informatique ne concerne pas seulement la résolution d'un problème mais aussi sa formulation. On peut faire le lien avec l'idée de modélisation présente tant chez Arsac que chez Rogalski. Elle indique également qu'il s'agit de penser comme un informaticien. La pensée informatique comme étant celle des informaticiens est également reprise par le groupe de travail européen CompuThink (https://ec.europa.eu/jrc/en/computational-thinking). La pensée informatique est, selon ce groupe, l'utilisation des concepts de l'informatique pour formuler et résoudre un problème.

La pensée informatique ainsi définie, celle des informaticiens, parfois limités aux programmeurs, est-elle aussi l'approche proposée par Papert ? Papert prône l'importance de l'apprentissage d'un langage de programmation, comme moyen d'enseigner aux enfants à penser :

> *« [...] certaines utilisations d'une technique informatique puissante, voire d'un état d'esprit informatique, peuvent fournir aux enfants de nouvelles façons d'apprendre, de penser [...] »* (Papert, 1981, p. 30).

Mais son propos concerne bien davantage les idées fondamentales que les enfants peuvent découvrir ou s'approprier dans des situations informatisées que l'apprentissage de l'informatique pour elle-même (Papert, 2000). Bers (2017) revient sur les trois rôles que peuvent jouer les ordinateurs en lien avec des idées fondamentales selon Papert.

– Un rôle neutre : certaines idées fondamentales sont indépendantes de l'existence de l'ordinateur, elles existaient bien avant l'informatique et celle-ci n'entraîne pas de modification majeure sur ces idées.

– Un rôle libérateur : certaines idées existaient avant l'informatique, mais l'ordinateur les libère en les rendant plus puissantes et plus accessibles à un grand nombre. La modélisation en serait un exemple.

– Un rôle incubateur : un petit sous-ensemble d'idées sont nées avec l'ordinateur, elles ne pourraient pas être connues sans l'ordinateur.

Lors de sa conférence au colloque Didapro 7 – DidaSTIC, Bers (2018) indique la dimension protéiforme de la pensée informatique, dans la

lignée des travaux de Papert : pensée abstraite, systémique, logique et séquentielle, algorithmique, résolution de problèmes, apprentissage par l'erreur... Il ne s'agit pas seulement d'apprendre à faire quelque chose, mais aussi d'aborder différemment des concepts ou notions, d'exprimer des idées, et finalement de s'exprimer. Elle est à l'origine d'environnements (ScratchJr et Kibo) qui visent à permettre aux enfants dès le plus jeune âge d'être créatifs, de s'exprimer et de découvrir des idées fondamentales (Bers, 2017).

Le rapide historique, non exhaustif, montre les enjeux autour de la notion de pensée informatique, comme légitimation d'un enseignement de l'informatique dans le système scolaire. Les travaux rapidement mentionnés renvoient à la question de la spécificité et de la caractérisation de la pensée informatique par rapport à d'autres formes de pensée.

2.2. Un point de vue informatique (Jean-Philippe Pellet)

Pour toute communauté professionnelle ou scientifique, c'est à la fois une aubaine et un risque de voir se développer un intérêt massif pour ce qui constitue ses thématiques et ses problématiques propres. C'est une aubaine, parce que c'est l'occasion de faire mieux connaître la discipline. Il y a tout à coup des leviers à actionner : politiques, financiers, collaboratifs. On a l'occasion de parler de notre milieu, de notre passion, ceci en ayant tout à coup une considération du grand public différente et une pertinence nouvelle. Et d'un autre côté, un tel intérêt est aussi un risque — de l'intérêt massif soudain s'ensuit un discours public parfois dépourvu de la connaissance des notions et concepts des professionnels.

Par le côté « transfert » relatif aux compétences que la pensée informatique semble désigner, par la pluridisciplinarité qu'elle reflète, ou encore par la position assumée de, par exemple, Jeannette Wing (2006) de la présenter comme universellement applicable et utile, il suit que chacun se sent concerné par une discussion autour de la pensée informatique. Il est d'autant plus important d'essayer de cerner les interprétations qui, au-delà du *buzzword*, sont les plus à même d'apporter aux étudiants quelque chose d'aussi fondé, stable et enrichissant que possible.

Le premier constat à faire, même s'il est peut-être évident, est que la « pensée informatique », telle qu'on la décrit aujourd'hui, n'est pas un concept informatique. C'est un concept lié à l'enseignement de l'informatique (et de thématiques connexes), à sa potentielle transversalité, aux

compétences générales développées, et aux potentiels transferts que sa maîtrise ou sa connaissance impliquerait. On pourrait s'aventurer à dire que c'est d'autant plus un thème lorsque l'on parle d'un enseignement destiné à un public général et non à de futurs informaticiens, où la nécessité d'« enrober » des connaissances ou compétences d'un nouveau vocabulaire semble moins présente.

En tant qu'informaticien, la lecture la plus structurante sur la pensée informatique/*computational thinking* est probablement l'article de Peter Denning, « *Remaining Trouble Spots with Computational Thinking* » (Denning, 2017). Denning, qui a été président de la grande *Association for Computing Machinery*, y pose notamment trois questions, relayées brièvement ici.

1. Qu'est-ce que le *computational thinking ?* Denning montre que le concept ne date pas de Jeannette Wing en 2006, mais, comme évoqué ici plus haut, des années 1960 et plus concrètement encore du virage computationnel des années 1970 et 1980. C'est à ce moment que la simulation informatique est devenue un troisième grand moyen de faire des découvertes scientifiques, aux côtés de l'approche théorique et de l'approche expérimentale. Il explique aussi pourquoi il pense que c'est une erreur de déconnecter des savoir-faire computationnels du modèle d'exécution qu'il y a derrière, souvent d'une machine, comme le fait Wing, : parce que c'est dans le but de faire accomplir un certain travail au modèle que l'on met en œuvre ces savoir-faire.

2. Comment mesurer des apprentissages en *computational thinking ?* Il dit que les tests portent traditionnellement sur des savoirs et pas des savoir-faire et que cela pose problème : en effet, la définition de *computational thinking* de Jeannette Wing mentionne presque exclusivement des savoir-faire plutôt que des savoirs.

3. Le *computational thinking* serait-il absolument utile pour tout le monde ? Denning le conteste et se déclare sceptique en ce qui concerne le transfert des compétences. Il argumente que la plus-value de la maîtrise des compétences derrière le *computational thinking* n'est vraiment réelle que pour les gens qui doivent effectivement concevoir des calculs automatiques, de la *computation*. Ceci ne veut en aucun cas dire que le *computational thinking* ne serait pas pour tout le monde — juste qu'il ne faut pas s'aventurer à des promesses irréalistes, générer des attentes démesurées pour ensuite ne pas être en mesure d'y répondre en conséquence.

Ceci dit, il y a bien un avant et un après Wing. Il est parfois difficile de bien cerner le changement de focale que cela provoque sur l'enseignement de l'informatique (ou, d'ailleurs, d'autres disciplines) et la cause de son succès. Dans ce sens, Denning compare la pensée informatique traditionnelle avec la « nouvelle » pensée informatique de Wing et pose la différence d'approche fondamentale qu'il y voit : dans le contexte de la pensée informatique traditionnelle, c'est la pratique de la conception d'algorithmes et de la programmation qui engendre du *computational thinking*, alors que, dans le contexte de la nouvelle pensée informatique, c'est l'apprentissage de certains concepts transversaux qui aident à la résolution de beaucoup de problèmes — dont la programmation, qui n'en est qu'un parmi d'autres. La direction de la causalité est inversée.

Pourquoi l'approche de Wing a-t-elle trouvé autant d'écho cette dernière décennie ? Peut-être qu'une partie de réponse réside dans cette idée en arrière-plan, séduisante mais, à mon avis, trompeuse, qu'on pourrait y trouver, à savoir : il est inutile d'acquérir des connaissances et une pratique disciplinaire importantes pour maîtriser les compétences de plus haut niveau inhérentes à la discipline et pour discuter de leur pertinence dans la formation générale.

2.3. Un point de vue épistémologique (Yannis Delmas-Rigoutsos)

Qu'est-ce que la « pensée informatique » ? Est-ce la pensée des informaticiens ? Lesquels, dans ce cas ? Celle des créateurs de logiciels, qu'on a pu appeler « pensée logistique » ? Celle des scientifiques du domaine informatique ? Les présentations du colloque Didapro 7 – DidaSTIC (Parriaux *et al.*, 2018) montrent combien les points de vue sont, en réalité, divers. Certains envisagent de mettre au centre la résolution de problèmes — c'est très souvent évoqué. Pourquoi pas, mais cette démarche est très loin d'être spécifique à l'informatique. C'est un élément important de la culture des ingénieurs, dont participent nombre d'ingénieurs informaticiens. On pourrait aussi s'attacher à ce qui en découle souvent : l'utilisation de « boîtes à outils » (outils technologiques ou intellectuels).

À notre sens, il ne faut pas espérer pouvoir disposer d'une définition autre qu'en extension de la pensée informatique. L'histoire de l'informatique, y compris pour ses principaux concepts, est nourrie de notions qui ne lui sont pas spécifiques et qui, pour certaines, préexistent de beaucoup : algorithme, programme enregistré, arbitraire du signe linguistique... C'est d'ailleurs ce qu'on retrouve dans l'histoire de nombreuses autres sciences

et techniques : à certains moments convergent de nombreux apports, divers et parfois anciens. Dans le cas de l'informatique, soulignons l'apport d'inspiration de la littérature, en particulier de science-fiction.

C'est une illusion totale de croire qu'on pourrait avoir une définition en intension de l'informatique — là encore, l'informatique ne se distingue pas de nombreuses autres sciences. Une discipline scientifique est un patchwork parce que son histoire, l'histoire des apports qui l'ont construite, est elle-même un patchwork.

Désigner tel outil (intellectuel ou technologique) comme « important » relève d'un choix (Delmas-Rigoutsos, 2018). En particulier ce choix ne peut que dépendre de la destination de cet outil : « important » n'a guère de sens en soi ; il convient de toujours préciser important *pour tel public* (ou pour telle finalité). Pour cette raison, le cœur de ce qu'est la pensée informatique varie en fonction du public visé. La notion de « pensée informatique » admet inévitablement différentes versions, ou plus exactement instanciations.

Pour Denning (2003), il faudrait nécessairement retenir comme centrale l'idée du modèle computationnel, au sens de modèle du calcul. Or l'informatique comporte également de nombreuses autres contributions à la formation générale de l'esprit, et notamment le rapport à l'abstraction, à la structuration des faits, à la représentation des objets de la pensée. Peter Denley, cofondateur et premier Secrétaire général de l'Association for history and computing, explique 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* » (Denley, 1994, p. 34). Plusieurs projets de recherche en histoire ont été l'occasion de l'observer, de manière très concrète. La représentation des données n'était pas pensée computationnellement : de fait, les projets utilisaient leurs bases de données comme on aurait pu le faire quelques années plus tôt avec des fiches cartonnées (avec, certes, le confort d'une forme de traitement de texte). Plusieurs de ces projets de recherche ont périclité, ou échoué à se développer, du fait d'un manque de conceptualisation de leurs jeux de données.

Revenons à la question de la définition de la pensée informatique. Bien sûr, les représentations évoquées sont souvent dirigées (ou devraient l'être) par la possibilité d'opérer des traitements sur les données, et, en ce sens, elles sont donc bien computationnelles, mais il ne s'agit pas là du tout, pour autant, de modéliser le calcul lui-même. D'ailleurs, si l'on

observe la manifestation de cette compétence dans d'autres sciences, il s'agit bien souvent plus de comprendre des phénomènes que de traiter des données, *stricto sensu*. La dimension de l'analyse computationnelle (c'est-à-dire les techniques de recherche d'une représentation adaptée, de construction d'un référentiel) peut être un apport important à de nombreuses autres disciplines scientifiques.

Nous pourrions citer d'autres exemples de cette dimension computationnelle, notamment la simulation. Il semble que ce serait trop réducteur de ne voir là que des outils techniques de calcul. La simulation est aussi un apport intellectuel de l'informatique, ne serait-ce que parce qu'elle nous oblige à préciser les variables de modélisation, les éléments à abstraire, voire à négliger...

L'expression « pensée informatique » a peut-être comme intérêt essentiel, au-delà de son contenu, d'insister sur l'idée qu'il s'agit en propre d'une pensée et non seulement de savoir-faire technologiques et, ce faisant, de promouvoir l'articulation entre l'informatique et les autres disciplines, notamment scientifiques. Dans une des présentations au colloque (Chessel Lazzarotto, 2018), un réemploi en grammaire d'une notion vue en robotique est un exemple, parmi de très nombreux autres, qui montre l'intérêt d'une telle démarche.

De nombreux liens intellectuels, épistémologiques, seraient intéressants à développer. Il y aurait à faire, par exemple, du côté de la cybernétique, qui rentre dans le mouvement intellectuel plus large du structuralisme, qui va des mathématiques avec Bourbaki à l'anthropologie avec Claude Lévi-Strauss, et qui est également au cœur de l'informatique.

Autre exemple : la correspondance preuve-programme (type-calcul), qui a un lien très fort avec l'analyse dimensionnelle en physique, ou avec la combinatoire énumérative en mathématiques. Celle-ci pourrait alimenter une réflexion sur l'apprentissage élémentaire du calcul — peut-être même fonder ces apprentissages. De tels échanges, en approfondissant la correspondance de Curry-Howard, permettent d'ailleurs de mieux comprendre certaines notions ou méthodes de l'informatique, en les généralisant et en les explicitant, par exemple la nécessité pour un code d'être clairement documenté : un code, c'est au moins autant des instructions que du typage, de la spécification, de la documentation. Ce serait, enfin, une occasion de valoriser la dimension expressive, poétique, artistique, du code.

Si l'on élargit un peu le propos, il peut être intéressant d'observer l'informatique comme une matrice disciplinaire, telle que la formalise l'épistémologue Imre Lakatos. Pour cet auteur, les programmes de recherche scientifiques se structurent dans deux directions : d'une part une heuristique[1] positive indique dans quel sens chercher, tandis qu'une heuristique négative correspond à la base théorique (méthodes, vocabulaire, concepts, théories...) sur laquelle s'entendent les participants de la discipline, par large consensus, et qu'ils souhaitent, pour l'heure, ne pas remettre en question. De ce point de vue, si l'on s'intéresse à une notion de pensée informatique à très gros grain ou pour des non-spécialistes, on peut rester, pour l'essentiel, dans le champ de l'heuristique négative : la discipline est stable, bien délimitée, et, pour l'essentiel, assez explicitable. Si, en revanche, on s'intéresse à la science en marche, à la recherche vivante, le champ s'étend à l'heuristique positive, par définition protéiforme et mouvante. À ce niveau, il ne faut pas, d'une décennie à la suivante, espérer trouver les mêmes composantes d'une hypothétique pensée informatique : les contours de la discipline changent perpétuellement. Ce qui, à une époque, semble important peut devenir plus tard secondaire. Si l'on veut poser la question dans le champ de l'éducation, il devient donc essentiel de poser des objectifs pédagogiques : dans quelles directions, avec quelles intentions, souhaite-t-on travailler la pensée informatique ?

2.4. Pensée informatique et *computational thinking* : analyse de revues de questions en langue anglaise

Les questionnements qui viennent d'être évoqués trouvent des échos dans les nombreuses revues de littérature en langue anglaise consacrées au *computational thinking*. Nous en donnons un aperçu ci-après.

Plusieurs revues ont été publiées depuis 2013. On peut citer :

– Grover et Pea, pour l'enseignement primaire et secondaire (Grover et Pea, 2013) ;

– Kalelioglu, Gülbahar et Kukul, s'intéressant à établir un cadre pour la pensée informatique (Kalelioglu *et al.*, 2016) ;

– Shute, Sun et Asbell-Clarke, se proposant de démystifier le *computational thinking* (Shute *et al.*, 2017) ;

1 Le mot *heuristique* s'entend ici au sens de : mouvement de découverte.

– Hickmott, Prieto-Rodriguez et Holmes, explorant le lien avec l'apprentissage des mathématiques (Hickmott *et al.*, 2017) ;

– Hsu, Chang et Hung, se centrant sur des questions d'apprentissage et d'enseignement (Hsu *et al.*, 2018) ;

– Lockwood et Mooney, pour l'enseignement secondaire (Lockwood et Mooney, 2018) ;

– Kirwan, Costello et Donlon, se limitant à l'apprentissage en ligne (Kirwan *et al.*, 2018).

On peut également citer un numéro spécial de la revue *International Journal of Child-Computer Interaction* consacré à la pensée informatique et à l'apprentissage du code pour les jeunes enfants (Howland *et al.*, 2018).

Des recherches visent à circonscrire la pensée informatique, d'une part, comme une somme de concepts ou de compétences. C'est le cas par exemple des travaux de Csizmadia *et al.* (2015), pour qui la pensée informatique recouvre les compétences d'abstraction, de généralisation, d'évaluation, de pensée algorithmique et de décomposition. Malgré tout, ces compétences très générales restent difficilement observables lors de l'activité des élèves (Drot-Delange et Tort, 2018).

D'autre part, des recherches s'appuient sur les travaux existants pour tenter de repérer les invariants et évaluer le consensus des experts sur telle ou telle conception de la pensée informatique. C'est le cas de l'approche présentée dans (Selby et Woollard, 2013). Les auteurs constatent le nombre croissant de chercheurs ou d'acteurs du monde éducatif mobilisant la notion de pensée informatique. Ils mènent une revue de littérature pour retenir les concepts associés à la pensée informatique qui font consensus et ont une définition robuste.

Trois concepts semblent répondre à ces critères : processus de pensée, abstraction et décomposition. Les auteurs retiennent ensuite les concepts suivants, qui présentent cependant moins de constance que les précédents : pensée algorithmique, évaluation et généralisation.

Il est intéressant aussi de noter ce qu'ils ne retiennent pas, la pensée logique ou d'autres termes similaires : pensée mathématique, pensée heuristique en sont des exemples... Pourtant certains auteurs les incluent dans la pensée informatique, en considérant qu'elle serait l'équivalent du raisonnement logique effectué par les humains, à l'image de l'intelligence artificielle ou des activités proposées par le courant de l'informatique débranchée (Drot-Delange, 2013). S'ils décident de ne pas retenir cette

notion de pensée logique, c'est qu'elle est selon eux sujette à des interprétations trop ouvertes, qui empruntent à d'autres disciplines telles que les sciences de l'ingénieur ou les mathématiques. Selby et Woollard (2013) considèrent que ces emprunts n'aident pas vraiment à définir la pensée informatique.

Le manque de consensus sur la notion même de pensée informatique est un constat largement partagé. Comme le remarquent Shute et ses collègues (Shute *et al.*, 2017), la définition même évolue avec l'accumulation des connaissances sur cette pensée informatique. La multiplicité de ces revues de la littérature atteste que son opérationnalisation ne va pas de soi, montrant la difficulté d'une approche finalement plutôt abstraite et théorique. D'ailleurs, Grover et Pea (2013) se demandent pourquoi l'article princeps de Wing (2006) a eu tant de succès et a servi de cri de ralliement (« *rallying cry* ») aux éducateurs, chercheurs en éducation et aux administrateurs. La pensée informatique serait une sorte d'étendard et il conviendrait maintenant d'analyser ce que font ceux qui s'y rallient. Une démarche ascendante est ainsi privilégiée : il ne s'agit pas de savoir si la notion est correctement appliquée, mais d'étudier comment ce qui est fait concourt à en préciser la définition.

S'agissant des références, les différents auteurs cités s'accordent pour remonter aux travaux de Seymour Papert, soulignant le côté enseignement et apprentissage de la pensée informatique, puis à l'article de Jeannette Wing de 2006. L'apprentissage par le jeu (*game-based learning*) et le constructivisme sont les théories principales qui servent de base aux articles sur la pensée informatique (Kalelioglu *et al.*, 2016 ; Kirwan *et al.*, 2018).

On peut noter l'intérêt des grandes associations américaines, comme CSTA (*Computer Science Teachers Association*) et ISTE (*International Society for Technology in Education*),et aussi des grandes entreprises telles que Google, avec le site *Exploring Computational Thinking*, (https://edu.google.com/resources/programs/exploring-computational-thinking) qui se veut un support pour les enseignants qui souhaitent intégrer la pensée informatique dans leurs pratiques d'enseignement), ou Microsoft qui propose une formation sur la pensée informatique et son utilisation dans différents contextes éducatifs sur le site *Computational Thinking and its importance in education* (https://preview.education.microsoft.com/en-us/course/a41b9507/overview). Notons également l'article de Barr et

Stephenson (2011) qui a proposé une définition opératoire de la pensée informatique.

L'importance de la pluridisciplinarité est souvent mise en exergue et les champs concernés par la pensée informatique sont nombreux. Ainsi, Mazzone (2018) voit en Andy Warhol un modèle pour la pensée informatique et la création artistique, dans la génération et la production de formes, tant en peinture que sur les premiers ordinateurs Amiga : une simulation de l'abstraction des processus et des méthodes de production qui nous sont familiers dans l'art électronique computationnel d'aujourd'hui.

L'utilisation de concours informatique, comme le concours Castor (Drot-Delange et Tort, 2018), est mentionnée comme un vecteur important pour développer la pensée informatique. La question de la transférabilité, toujours complexe, est souvent évoquée. Mais plutôt que parler de transfert, on peut remarquer que l'informatique, avec ses instruments et ses formes de travail, s'est imposée dans les recherches au cœur de la plupart des disciplines. C'est un mode de pensée, via des instrumentations spécifiques, qui a été repris et développé dans de nombreux champs disciplinaires.

Enfin, dans les mises en œuvre de la pensée informatique, beaucoup regrettent le manque de recherches expérimentales ou quasi-expérimentales dans les classes ainsi que la rareté des évaluations.

3. Quels choix de formation ?

Comme nous venons de le voir, la notion de pensée informatique est sujette à caution, mais, même si chacun la redéfinit partiellement selon ses propres filtres, elle est largement utilisée. Peut-être que cette notion, victime de son succès, a perdu sa signification avec la généralisation de son emploi. En tous cas, cela montre l'intérêt de discuter de son régime de fonctionnement pour la formation.

Pour ce faire, un jeu à trois termes se présente (informatique, pensée informatique et programmation), avec des définitions de la pensée informatique plutôt fluctuantes qui s'inscrivent dans les objectifs ou finalités des formations.

3.1. La pensée informatique et l'informatique ne se résument pas à la programmation

Le fait que l'informatique et la pensée informatique ne se résument pas à la programmation semble bien admis. La pensée informatique pourrait d'ailleurs presque s'en passer. Ainsi, dans le cadre d'une étude européenne (Bocconi *et al.*, 2016), il est constaté que certains chercheurs considèrent qu'il n'y a pas nécessité, pour développer la pensée informatique, de programmer un ordinateur. Ce résultat pourrait s'obtenir en mettant en œuvre une démarche de résolution de problèmes mobilisant des stratégies comme des algorithmes, l'abstraction ou le débogage. Toutefois, pour d'autres, la programmation est une étape incontournable, car c'est l'apprentissage des concepts centraux en informatique qui permettront de développer cette forme de pensée (Voogt *et al.*, 2015).

S'il est admis que pensée informatique et programmation sont distinctes, les travaux de recherche sur la pensée informatique prennent le plus souvent comme appui des contextes de programmation. Ceci pourrait conforter l'idée que c'est la même chose, ou qu'au moins il est nécessaire de programmer pour développer une pensée informatique. Ainsi certaines activités, comme la programmation en Scratch, permettraient de développer la pensée informatique. Cette idée est présente dans le rapport de l'Académie des sciences (2013), dans lequel il est mentionné que « l'apprentissage de la programmation permet de découvrir les rudiments de la pensée informatique » (p. 24). Mais rares sont les recherches qui visent à produire des résultats sur ces liens.

S'agissant d'informatique, l'accord semble se faire sur le rôle forcément limité, même s'il est essentiel, de la programmation. Il est possible de faire de la musique sans faire de solfège, mais il est (presque) impossible d'en faire sans jouer d'un instrument. Il semble que le parallèle existe pour l'informatique : on peut faire de l'informatique sans apprendre de formalisme, au moins dans un premier temps, mais il n'est pas possible d'en faire sans passer, d'une façon ou d'une autre, par une activité de programmation.

Pour autant, Jeff Atwood, un grand nom de l'informatique appliquée — fondateur du site site Stack Overflow (https://stackoverflow.com) — dans un article intitulé « *Please don't learn to code* » (Atwood, 2012), insiste sur l'erreur qui consiste à considérer la programmation comme une fin en soi. À la place, il recommande de rechercher avant tout à comprendre

comment les choses autour de nous fonctionnent à leur niveau élémentaire (et à mieux communiquer avec les autres êtres humains).

Mais comment expliciter la partie de l'informatique qui n'est pas programmation ?

3.2. L'informatique ne se résume pas à la pensée informatique

Cette question conduit à faire ressortir un des enjeux de l'enseignement de l'informatique. Il ne s'agit pas « seulement » de développer des compétences plus ou moins abstraites et transférables de résolution de problèmes ; il s'agit aussi d'expliquer le fonctionnement du monde qui nous entoure.

En ce sens, mettre de côté « informatique » pour garder uniquement « pensée informatique » nie ces enjeux. Une approche où une forme de pensée informatique serait enseignée de façon complètement déconnectée de l'informatique elle-même ne répondrait pas au besoin de formation de futurs citoyens capables de s'exprimer de façon fondée sur des enjeux sociétaux potentiellement techniques et complexes.

Pour faire une comparaison avec l'enseignement des sciences naturelles, il y a bien sûr, derrière cet enseignement, une volonté de parler de la démarche scientifique en général et de cette façon structurée de construire des connaissances et de faire des découvertes. Ce n'est pas sans parallèle avec un enseignement de l'informatique qui viserait à montrer comment des problèmes complexes (organisation d'immenses masses de données, gestion de multiples processus en parallèle, ou méthodes d'apprentissage automatique) ont été approchés, décomposés et traités par les informaticiens — enseignement pendant lequel on peut spécifiquement s'attarder sur des techniques générales de résolution de problèmes particulièrement opportunes. Mais tout comme la démarche scientifique traitée hors discipline scientifique semble largement désincarnée, on peut considérer que la pensée informatique sans l'informatique est privée de sa « substantifique moelle ».

Faire le lien avec les questions scientifiques en arrière-plan est fondamental pour donner du sens aux activités et aux apprentissages. C'est d'ailleurs cette approche qu'adopte le concours Castor, où chaque question est agrémentée d'un paragraphe « C'est de l'informatique » qui explique comment les stratégies mises en œuvre dans la résolution de la

question sont similaires à celles mises en œuvre dans de réelles problématiques informatiques.

Notons toutefois que le *computational thinking* est présenté explicitement dès l'introduction des programmes en informatique du Royaume-Uni comme un objectif de cet enseignement pour comprendre et changer le monde (*Department for Education*, 2013). L'élaboration de la relation entre pensée informatique et items du programme reste à la charge de l'enseignant (Drot-Delange et Tort, 2018).

3.3. La pensée informatique accompagne l'introduction de l'informatique dans les programmes

Ainsi, ce sont les modalités d'introduction de l'informatique dans les différents systèmes éducatifs qu'il convient d'observer. Pour les pays nordiques, Bocconi, Chioccariello et Earp (2018) décrivent une approche associant programmation et pensée informatique. Dans (Heintz *et al.,* 2016), les auteurs étudient comment dix pays différents ont abordé la question de l'introduction de l'informatique dans leur éducation primaire et secondaire. Selon eux, la pensée informatique est rarement mentionnée explicitement, mais les idées associées sont souvent incluses sous une forme ou une autre.

Il semble que l'intérêt de la notion de pensée informatique est d'amener à réfléchir aux finalités à donner à un enseignement de l'informatique dans un contexte scolaire. Une piste prometteuse semble être la réflexion sur les idées fondamentales à l'aide des critères proposés par Bers (2017) : une idée est fondamentale, selon elle, si elle personnellement utile à l'apprenant, si elle est épistémologiquement liée à d'autres disciplines, si elle trouve ses racines dans les connaissances intuitives qu'un enfant a pu développer sur une période longue.

Si on retrouve certaines dimensions développées par Schwill (Hartmann *et al.*, 2012), Bers introduit le critère de l'intérêt pour l'individu lui-même, certes décrété *a priori*, et ses connaissances personnelles, comme constitutif du caractère fondamental d'une idée. Cela semble constituer une passerelle intéressante avec les développements en didactique de l'informatique, comme ceux de l'informatique « située » (Guzdial, 2010 ; (Knobelsdorf et Tenenberg, 2013), ou le modèle de reconstruction didactique de l'enseignement de l'informatique (Bruillard, 2017), ancrés dans les pratiques informatiques quotidiennes des individus.

3.4. Des choix de formation prenant en compte plusieurs enjeux

Finalement, l'implication la plus claire de la pensée informatique sur l'enseignement de l'informatique est d'en repenser les enjeux et les modalités. Sur ce dernier point, toutefois, la pédagogie de projet semble s'imposer dans l'enseignement obligatoire, principalement en raison de l'organisation de nos systèmes éducatifs et de la place que peut s'octroyer une nouvelle discipline scolaire (*ibidem*). Cela reste néanmoins à confirmer.

On peut distinguer plusieurs enjeux d'un enseignement généralisé de l'informatique, prenant l'aspect « pensée computationnelle » dans un sens transdisciplinaire et/ou métacognitif.

– L'enjeu scientifique : la nécessité de comprendre le fonctionnement du monde numérique, à l'instar de la nécessité de comprendre le fonctionnement du monde physique.

– L'enjeu sociétal : la nécessité d'être à même d'évaluer de façon informée et fondée les impacts de l'informatique sur la société et de contribuer à façonner son développement.

– L'enjeu transdisciplinaire : l'acquisition des outils à la fois mentaux et techniques au service d'autres sciences ou contribuant à des stratégies plus générales de résolution de problèmes.

Une question se pose avec insistance : quels que soient les choix pour les curricula, quels enseignants seront à même de les mettre en œuvre ? Quelle formation pourrait être proposée à ces enseignants ? Même si la pensée informatique reste encore mal définie, on peut certainement déplorer le manque d'enseignants ayant les dispositions et les compétences pour l'enseigner.

4. De la pensée informatique aux compétences de base

Si l'on se limite, pour simplifier le cadre conceptuel de la pensée informatique, à ce qui concerne la résolution de problèmes de manière effective et efficiente, algorithmiquement, avec ou sans machine, différentes facettes ont été mises en exergue : décomposition, abstraction, conception d'algorithmes, débogage, itération et généralisation (Shute *et al.*, 2017).

Prenons l'exemple du débogage. Shute et ses co-auteurs le définissent comme la détection et l'identification des erreurs, leur correction, quand

un programme ne fonctionne pas comme il devrait. Dans (Bers *et al.,* 2014), il est considéré comme au cœur de la pensée informatique, constituant une mesure du développement de la pensée informatique chez les enfants. Ces auteurs identifient les différentes étapes par lesquels passent les enfants pour déboguer : d'abord reconnaitre un écart à l'état final attendu ou que quelque chose ne fonctionne pas comme attendu, puis choisir de maintenir l'objectif initial ou se tourner vers une alternative, ensuite formuler des hypothèses sur la cause du problème et enfin tenter de résoudre le problème. Il s'agit d'un apprentissage lié à l'informatique et plus généralement à l'ingénierie.

Plus simplement, la pensée informatique tourne autour du fait d'être capable de formuler un problème, de le comprendre et de le résoudre. Avec une telle définition, on peut s'interroger sur l'absence de référence à Polya (1965), très populaire au cours des développements de l'intelligence artificielle dans les années 1970 et 1980. Mais, comme on l'a vu, remonter à Papert dans l'histoire de la pensée informatique oriente sur les questions d'enseignement et d'apprentissage, et même sur les apprentissages de base, les apprentissages pour tous. Cela conduit aux questions de littératie, liées aux compétences de base comme la lecture et l'écriture.

C'est une position défendue par diSessa (2000), dans sa vision de la culture informatique universelle. Selon lui, les dispositifs informatisés (ordinateurs) forment la base pour une nouvelle littératie qui va modifier la manière avec laquelle les humains pensent et apprennent, et chacun sera un créateur aussi bien qu'un consommateur de formes expressives dynamiques et interactives. Cette perspective est également défendue par Jacob et Warschauer (2018) et par Wing : « *My grand vision is that computational thinking will be a fundamental skill—just like reading, writing, and arithmetic—used by everyone by the middle of the 21st century* » (Wing, 2017).

Au-delà des traditionnelles littératies informationnelles (*information literacy*) et numériques (*digital literacy*), on peut faire le lien avec la translittératie, qui s'exerce sur une multitude de supports et de médias, étudiée dans le projet ANR Translit (https://anr.fr/Projet-ANR-12-CULT-0004). C'est également la manière contemporaine de voir les questions d'organisation, avec des synthèses entre des traditions documentaires et les offres informatiques, conduisant à la science de l'organisation (Glushko, 2016).

La notion d'écriture est centrale et les effets de l'écriture sont bien exposés par Jack Goody (1977). L'apprentissage du calcul se fait aussi par la manipulation d'objets et par la lecture et la transformation d'écritures. Ces écritures fournissent une représentation de l'ordre des nombres et des quantités, les opérations s'effectuent par l'intermédiaire des écritures. On a beaucoup écrit sur l'externalisation de la mémoire avec les technologies informatiques, et il est intéressant de revenir au projet MyLifeBits (décrit sur les sites https://www.microsoft.com/en-us/research/project/mylifebits/ et https://fr.wikipedia.org/wiki/MyLifeBits), caractéristique des phénomènes d'externalisation, de conservation et d'organisation de ces mémoires. Les technologies informatiques actuelles n'offrent-elles pas une nouvelle forme d'externalisation de la pensée, en tous cas d'une certaine forme de pensée, les étapes successives d'un raisonnement étant traduites dans des écritures, extension de la logique à d'autres formes de raisonnement ?

Ainsi la question est celle de l'évolution du lire-écrire-compter dans un univers où les dispositifs informatisés et autres objets connectés sont omniprésents, ambiants même. On peut proposer le triptyque lire-écrire-computer (Bruillard, 2012). Si le développement de cette alphabétisation s'inscrit dans un courant d'émancipation humaine, il ne devrait pas conduire à oublier qu'il convient également de contrôler les machines, de limiter leurs moyens d'action. La vague actuelle d'engouement pour l'intelligence artificielle peut être préoccupante, si elle conduit à laisser trop de pouvoir aux machines et aux organisations qui les utilisent, sans développer la littératie liée à la pensée informatique à la hauteur de l'ambition forte de cette éducation.

RÉFÉRENCES

Académie des sciences (mai 2013). *L'enseignement de l'informatique en France. Il est urgent de ne plus attendre* (Rapport). Institut de France-Académie des sciences. https://www.academie-sciences.fr/pdf/rapport/rads_0513.pdf

Arsac, J. (1981). Annexe 1. Dans J. C. Simon (dir.*), L'éducation et l'informatisation de la société. Rapport au Président de la République* (p. 152-165). La Documentation française.

Atwood, J. (2012). Please don't learn to code [article de blog]. https://blog.codinghorror.com/please-dont-learn-to-code/.

Baron, G.-L. (1990). Note de synthèse. *Revue française de pédagogie, 92*(1), 57-77.

Barr, V. et Stephenson, C., (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54.

Bers, M. U. (2017). The Seymour test: Powerful ideas in early childhood education. *International Journal of Child-Computer Interaction*, *14*, 10-14. https://doi.org/10.1016/j.ijcci.2017.06.004

Bers, M. U. (2018). La programmation en tant que place de jeu développementale : la pensée informatique et la robotique dans la petite enfance. Communication présentée au colloque *Didapro 7 — DidaSTIC : de 0 à 1 ou l'heure de l'informatique à l'école*. https://hal.archives-ouvertes.fr/hal-01748783

Bers, M. U., Flannery, L., Kazakoff, E. R. et Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). *Developing computational thinking in compulsory education. Implications for policy and practice* (Rapport EUR 28295). Luxembourg : Publications Office of the European Union. https://doi.org/10.2791/792158

Bocconi, S., Chioccariello, A. et Earp, J. (janvier 2018). *The nordic approach to introducing computational thinking and programming in compulsory education.* (Report prepared for the Nordic@BETT2018 Steering Group). https://doi.org/10.17471/54007

Boulle, F. (1984). Informatique à l'école. Introduction et éléments d'histoire. *Bulletin de l'EPI*, *6*. https://www.epi.asso.fr/revue/dossiers/d06p005.htm

Bruillard, É. (2012). Lire-écrire-computer : émanciper les humains, contrôler les machines. Dans e-Dossier de l'Audiovisuel « Éducation aux cultures de l'information », INA. http://www.epi.asso.fr/revue/articles/a1209d.htm

Bruillard, É. (2017). Enseignement de l'informatique entre science et usages créatifs : quelle scolarisation ? Dans J. Henry, A. Nguyen et É. Vandeput (dir.), *L'informatique et le numérique dans la classe. Qui, quoi, comment ?* (p. 205–218). Belgique : Presses Universitaires de Namur.

Chessel Lazzarotto, F. (2018). Former à la programmation en primaire, une form'action : robots d'Evian 2015–2018. Dans G. Parriaux, J. P. Pellet, G. L. Baron, E. Bruillard, V. Komis (dir.), *De 0 à 1 ou l'heure de l'informatique à l'école. Actes du Colloque Didapro 7 – DidaSTIC* (p. 117-128). Bern, Suisse : Peter Lang.

Csizmadia , A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. et Woollard, J. (2015). Computational thinking A guide for teachers. Computing at School. https://www.computingatschool.org. uk/computationalthinking

Delmas-Rigoutsos, Y. (2018). Proposition de structuration historique des concepts de la pensée informatique fondamentale. Dans G. Parriaux, J. P. Pellet, G. L. Baron, E. Bruillard, V. Komis (dir.), *De 0 à 1 ou l'heure de l'informatique à l'école. Actes du Colloque Didapro 7 – DidaSTIC* (p. 31-60). Bern, Suisse : Peter Lang.

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. http://hdl.handle.net/10945/35508

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Department for Education. (2013). *The National Curriculum in England*. Disponible en ligne à www.education.gov.uk/nationalcurriculum.

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge : MIT Press.

Drot-Delange, B. (2013). Enseigner l'informatique débranchée : analyse didactique d'activités (p. 1-13). Dans *Actes du congrès Actualité de la Recherche en Education et en Formation (AREF 2013)* (p. 1-13). https://halshs.archives-ouvertes.fr/sic_00955208

Drot-Delange, B. et Tort, F. (2018). Résolution de défis et pensée informatique. Présenté aux *10e rencontres scientifiques de l'ARDIST*. https://hal.archives-ouvertes.fr/hal-01851772/document

Glushko, R.-J. (dir.). (2016). *The discipline of organizing: Professional edition* (4e éd.). Sebastopol, CA : O'Reilly.

Goody, J. (1977). *The Domestication of the Savage Mind*. Cambridge University Press.

Grover, S. et Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43. http://journals.sagepub.com/doi/abs/10.3102/0013189X12463051

Guzdial, M. (2010). Does contextualized computing education help? *ACM Inroads, 1*(4), 4–6.

Hartmann, W., Näf, M. et Reichert, R. (2012). *Enseigner l'informatique (Traduit de l'allemand)*. Paris : Springer.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. *IEEE Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/FIE.2016.7757410

Howland, K., Good, J., Robertson, J. et Manches, A. (dir.) (2018). Computational thinking and coding in childhood [numéro thématique]. *International Journal of Child-Computer Interaction*.

Hickmott, D., Prieto-Rodriguez, E. et Holmes, K. (2017). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education, 4*(1), 48-69. https://link.springer.com/article/10.1007/s40751-017-0038-8

Hsu, T. C., Chang, S. C. et Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126,* 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

Jacob, S. R. et Warschauer, M. (2018). Computational Thinking and Literacy. *Journal of Computer Science Integration, 1*(1). https://doi.org/10.26716/jcsi.2018.01.1.1

Kalelioglu, F., Gülbahar, Y. et Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4*(3), 583-596. https://pdfs.semanticscholar.org/e4f9/4b4e16a87e4b815893388c63c096038a69b8.pdf

Kirwan, C., Costello, E. et Donlon, E. (2018). Computational Thinking and Online Learning: A Systematic Literature Review. Dans *Proceedings of the 17th European Conference on eLearning (ECEL 2018)* (p. 650-657). Reading, UK : Academic Conferences and Publishing International Limited.

Knobelsdorf, M. et Tenenberg, J. (2013). The context-based approach IniK in light of situated and constructive learning theories. Dans *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages. Proceedings of the 6th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP 2013)* (p. 103–114). Berlin : Springer.

Lockwood, J. et Mooney, A. (2017). Computational Thinking in Secondary Education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools. 2*(1).
http://ijcses.org/index.php/ijcses/article/view/26

Mazzone, M. (2018). Andy Warhol: Computational thinking, computational process. *Leonardo Music Journal*. https://doi.org/10.1162/LEON_a_01574

Papert, S. (1981). *Jaillissement de l'esprit. Ordinateurs et apprentissage*. Edition 1999. Flammarion. (Ouvrage original publié en 1980 sous le titre *Mindstorms - Children, computers and powerful ideas*. USA : Basic Books).

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM systems journal*, *39*(3.4), 720–729.

Parriaux, G., Pellet, J.-P., Baron, G.-L., Bruillard, E. et Komis, V. (dir.). (2018). De 0 à 1 ou l'heure de l'informatique à l'école. Actes du Colloque Didapro 7 – DidaSTIC. Bern, Suisse : Peter Lang.

Polya, G. (1965). *Comment poser et résoudre un problème*. Paris : Dunod. (Ouvrage original publié en 1945 sous le titre *How to solve it: A new aspect of mathematical method*. Princeton University Press).

Rogalski, J. (1987). Acquisition de savoirs et savoir-faire en informatique. *Cahiers de Didactique des Mathématiques, 43*.

Selby, C. et Woollard, J. (2013). *Computational thinking: the developing definition* (Rapport). https://eprints.soton.ac.uk/356481/

Shute, V. J., Sun, C. et Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142-158.
https://www.sciencedirect.com/science/article/pii/S1747938X17300350

Voogt, J., Fisser, P., Good, J., Mishra, P. et Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715-728.
https://doi.org/10.1007/s10639-015-9412-6

Wing, J. M. (2006). Computational thinking. Commun. ACM, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215.

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7-14.
https://doi.org/10.17471/2499-4324/922