Les différentes approches pour évaluer l'apprentissage de la programmation et de la pensée informatique en contexte scolaire

Kevin Sigayret, Nathalie Blanc, André Tricot

Les différentes approches pour évaluer l'apprentissage de la programmation et de la pensée informatique en contexte scolaire

Kevin SIGAYRET, Nathalie BLANC, André TRICOT

Université Paul Valéry Montpellier 3, Laboratoire EPSYLON EA 4556, Montpellier, France
Contact : kevin.sigayret@univ-montp3.fr

Introduction

- Depuis une vingtaine d'années, l'apprentissage de la programmation informatique (ré)intègre les cursus scolaires, en France comme à l'étranger. L'apparition de logiciels de programmation éducatifs visuels, tels que Scratch, rend plus accessible cette discipline complexe.
- Apprendre à programmer stimulerait certaines capacités cognitives essentielles au développement intellectuel des enfants et adolescents (*Psycharis & Kallia, 2017 ; Popat & Starkey, 2019*).
- Une des finalités principales de cet enseignement : Transmettre la « **pensée informatique** », un ensemble de processus utilisés pour résoudre des problèmes en s'appuyant sur des concepts issus de l'informatique (*Wing, 2006*), et dont la maîtrise constituerait un **enjeu sociétal majeur pour le XXIème siècle**. La pensée informatique regroupe les capacités cognitives de résolution de problèmes qui déterminent (entre autres) les compétences en programmation (*Denning, 2017*).
- Aucune définition ou conceptualisation précise des processus et compétences inclus dans cette pensée informatique ne fait véritablement consensus, malgré les nombreuses propositions qui ont été faites ces dernières années (*Selby & Woollard, 2013 ; Shute, Sun et Asbell-Clarke, 2017 ; …*)

Par conséquent, bien que les chercheurs s'accordent sur la nécessité d'enseigner la programmation et la pensée informatique dès le plus jeune âge, il existe encore **de nombreuses incertitudes sur la manière avec laquelle cet apprentissage peut être évalué**.

Objectif

Synthétiser les apports et limites des **outils quantitatifs** utilisés pour évaluer la pensée informatique en contexte scolaire (de l'école élémentaire jusqu'au lycée).

Méthode

Revue de la littérature basée uniquement sur le titre des articles ce qui permet d'exclure rapidement des milliers de références qui ne sont pas centrées sur une description précise des outils d'évaluation (source : base de données transdisciplinaire Google Scholar).

Mots-clés utilisés : *Computational Thinking Scale(s) – Computational Thinking Test – Computational Thinking Assess/Assessing/assessment* (et équivalents français)

Critères d'inclusion : articles en français ou en anglais présentant une description précise et/ou des données psychométriques attestant de la validité d'un outil, d'une approche ou d'une méthode permettant d'évaluer la maîtrise de la pensée informatique.

Critères d'exclusion : tests uniquement à destination d'une population adulte, outils d'évaluation qualitatifs, études préliminaires.

Au total, **22 articles** ont été retenus et permettent de classer les outils d'évaluation de la pensée informatique en 3 catégories distinctes.

Résultats & Discussion

Cette revue de la littérature a permis d'identifier trois approches différentes pour évaluer la maîtrise de la pensée informatique :

- ❑ **Les échelles auto-évaluatives** : la pensée informatique est parfois considérée comme l'agrégation de plusieurs compétences bien identifiées (pensée critique, pensée créative...). Limites liées à la subjectivité des réponses fournies et s'adressent plutôt à des élèves âgés (Lycée ou fin du Collège). Ex : *Computational Thinking Scales (CTS).*
- ❑ **Les outils d'analyse « objectifs » du code produit** par l'élève : ils permettent d'estimer un certain niveau de maîtrise de la pensée informatique à partir des blocs qui sont utilisés par l'apprenant lorsqu'il construit un programme sur un logiciel éducatif. Evaluation surtout formative/itérative qui permet aux élèves d'être évalués pendant le processus d'apprentissage et d'améliorer la qualité de leur production. Généralement dépendant d'un langage ou d'un logiciel particulier mais peuvent être utilisés à tous les âges. Ex : *Dr. Scratch*.
- ❑ **Les tâches de résolution de problèmes** : Utilisation de problèmes dont la résolution n'est possible qu'en utilisant certaines compétences liées à la pensée informatique. Evaluation alors centrée sur le résultat (« *le problème est-il résolu ?* ») ou sur les processus implicites (« *Comment l'élève résout-il le problème ?* »). Ces tâches permettent de diagnostiquer un certain niveau de performances des élèves. Généralement conçues pour tester la capacité à choisir la solution adéquate parmi plusieurs solutions proposées. N'évaluent pas la capacité de l'élève à générer lui-même sa propre solution à un problème. Ex : *Computational Thinking Test (CTT).*

Conclusion. Si plusieurs outils ont été validés pour mesurer la pensée informatique, les différentes approches proposées présentent certaines limites et aucune ne fait véritablement consensus. De nombreux chercheurs recommandent d'ailleurs de combiner plusieurs approches. **Bénéficier d'outils capables de distinguer les aspect notionnels (connaissance et compréhension des notions) et procéduraux (capacité à appliquer ces notions pour résoudre des problèmes) impliqués dans la pensée informatique demeure une priorité pour faciliter l'évaluation de ces compétences.**

Références

- Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60(6)*, 33-39.
- Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education, 128,* 365-376.
- Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583-602.
- Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. In *Paper presented at the 18th annual conference on innovation and technology in computer science education, Canterbury.*
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142-158.