Pensée computationnelle: pour un néopapertisme durable car sceptique

Pierre Dillenbourg

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.