Résolution de défis et pensée informatique

Béatrice Drot-Delange, Françoise Tort

▶ **To cite this version:**

Béatrice Drot-Delange, Françoise Tort. Résolution de défis et pensée informatique. 10èmes rencontres scientifiques de l'ARDIST, Mar 2018, Saint-Malo, France. hal-01851772

HAL Id: hal-01851772
https://hal.science/hal-01851772

Submitted on 30 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Feuille de style à utiliser pour les **communications isolées**

Résolution de défis et pensée informatique

Béatrice Drot-Delange
Université Clermont Auvergne, ACTé, beatrice.drot-delange@uca.fr

Françoise Tort
ENS Paris-Saclay, STEF, francoise.tort@ens-paris-saclay.fr

Résumé

Les défis du concours informatique Castor sont conçus pour que les élèves mettent en œuvre les compétences de la pensée informatique. Or, en France, les curricula d'informatique ne font pas référence à de telles compétences. Cette communication vise à découvrir quelles compétences sont repérées et évaluées par les enseignants lors de l'analyse de l'activité des élèves en train de résoudre les défis du concours. Les résultats montrent qu'il s'agit bien des compétences de la pensée informatique. Toutefois, les discours des enseignants renvoient davantage à l'activité plus générale de résolution de problèmes.

Mots-clés

Didactique de l'informatique, concours Castor, programmation, littératie informatique, enseignant.

Problem solving tasks and computational thinking

Abstract

The tasks proposed in Bebras contests are designed to bring students to develop computational thinking skills. However, in France, curricula in informatics do not refer to computational thinking. This article aims to discover what skills are spoted and assessed by teachers when they analyze how students solve Bebras tasks. The result shows that it is mainly computational thinking. However, teacher's discourses are much more about the problem solving activity itself.

Key-words

Computer science education, Bebras constest, programming, digital literacy, teacher.

INTRODUCTION

Le concours Castor a pour objectif de promouvoir la science informatique auprès des plus jeunes, dans un contexte où elle est peu, voire pas, présente dans les programmes scolaires (Tort et Dagiene, 2012). Créé en 2004 en Lituanie, il est organisé aujourd'hui dans plus de 20 pays. En France, il est proposé en collaboration par l'INRIA, l'ENS Paris-Saclay et l'association France IOI qui se chargent de l'édition des contenus et de la mise en œuvre technique. Il est ouvert aux élèves des cycles 3, 4 et 5 - c'est à dire de la classe de CM1 à l'école primaire (11-12 ans) jusqu'à la classe de terminale au lycée (17-18 ans). L'inscription des candidats est à l'initiative des enseignants, qui choisissent le plus souvent de faire participer leurs classes entières. Ils organisent et encadrent la passation du concours sur le temps scolaire, dans leurs établissements.

Les défis sont conçus de manière à ce que leur résolution nécessite de mobiliser les compétences de la pensée informatique (Dagiene et Sentance, 2016). Il s'agit des compétences d'abstraction, de généralisation, d'évaluation, de pensée algorithmique et de décomposition telles qu'elles ont été définies dans Csizmadia et *al.* (2015). Ces compétences sont utilisées par certains pays tels que le Royaume-Uni pour fournir un guide de sélection des ressources du concours Castor à l'enseignant[1].

La pensée informatique est présentée explicitement dès l'introduction des programmes en informatique de ce pays comme un objectif de cet enseignement pour comprendre et changer le monde (*Departement for education*, 2013). Mais à la lecture des programmes, on constate d'une part que la pensée informatique n'y est pas définie, d'autre part que l'élaboration de la relation entre pensée informatique et items du programme reste à la charge de l'enseignant. Fournir aux enseignants des ressources explicitant ces relations serait donc un élément facilitateur de leur travail.

En France, la notion de pensée informatique n'est pas structurante des programmes scolaires, d'autres références sont utilisées (Baron, Bruillard, Drot-Delange, 2015 ; Bruillard, 2017). Pourtant, les défis du concours Castor français embarquent les mêmes principes qu'au niveau international. Dès lors, quelles compétences sont repérées et évaluées par les enseignants dans les défis du concours et la manière dont les élèves les résolvent ?

[1] Voir « UK Bebras Computational Thinking Challenge. Answers 2016 », en ligne : http://www.bebras.uk/uploads/2/1/8/6/21861082/uk-bebras-2016-answers.pdf - Consulté le 27/10/2017

METHODOLOGIE

Le protocole complet est détaillé dans Drot-Delange et Tort (2018). Nous présentons les éléments utiles pour la présente étude.

Nous avons filmé 24 élèves de lycée, en train de résoudre individuellement des défis. Il s'agit de trois défis relevant d'une activité de programmation : « Course de grenouille », « Labyrinthe » et « Robot peintre » (cf. Figure 1).

Nous avons sollicité trois enseignants de mathématiques et un enseignant de génie électronique[2], tous en charge de l'enseignement d'Informatique et Sciences du Numériques et faisant passer régulièrement le concours Castor à leurs élèves, de la seconde à la terminale. Les enseignants ont visualisé les vidéos pour identifier les connaissances mobilisées par l'élève, les stratégies de résolution mises en œuvre et préciser s'ils jugeaient ces stratégies surprenantes, originales ou attendues. Chaque enseignant a visualisé de 5 à 8 vidéos, pour une durée cumulée de 35 à 45 minutes, et en a produit une analyse écrite.

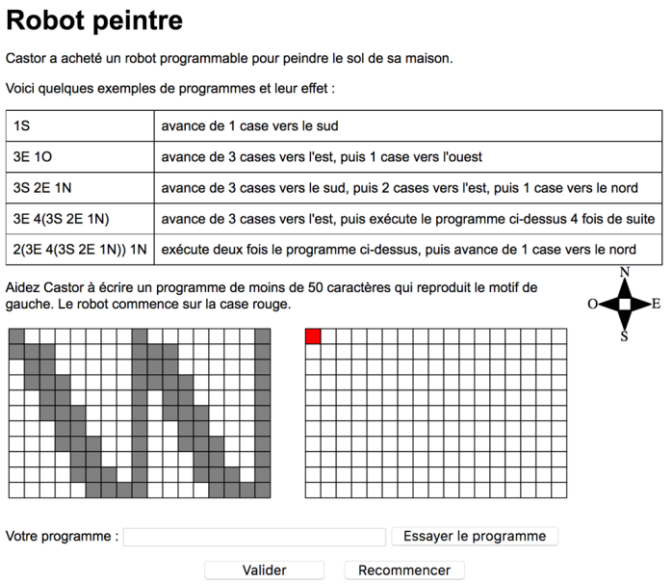

Figure 1 : Un défi du concours Castor 2013, objet de la présente étude.

[2] L'enseignant de génie électronique intervient dans la filière Sciences et Technologies de l'Industrie et du Développement Durable (STI2D).

Nous avons réalisé une analyse thématique de ces productions écrites à l'aide du logiciel NVivo 10. Nous avons procédé d'abord de manière ascendante en respectant le plus possible les expressions et termes utilisés par les enseignants. Ensuite, nous avons regroupé les nœuds obtenus en catégories correspondant aux cinq compétences génériques de la pensée informatique. Le tableau 1 présente le principe de codage que nous avons appliqué. Il est inspiré de Dagiene, Sentance et Stupuriené (2017) mais détaille plus précisément les activités codées.

Tableau 1 : Principe de codage des activités observées par les enseignants avec les compétences de la pensée informatique.

Abstraction	Manipuler une représentation abstraite (tel que graphique, tableau) ; construire une telle représentation ; simplifier un problème en ôtant les détails inutiles.
Pensée algorithmique	Créer et exécuter un algorithme ; écrire un programme ; procéder séquentiellement.
Décomposition	Décomposer le problème en objectif intermédiaire, en sous-tâches, en sous parties à traiter.
Évaluation	Tester, vérifier, contrôler par rapport à une référence ; sélectionner/comparer des éléments ; respecter des contraintes.
Généralisation	Identifier un problème connu ; repérer des similarités, des patrons ; réutiliser une solution, l'adapter la situation.

RESULTATS

Le codage des analyses de vidéos avec les compétences de la pensée informatique donne la répartition suivante, pour l'ensemble des 113 fragments codés (cf. Figure 2).

Les références à la capacité d'abstraction prédominent. Dans le détail, les enseignants parlent, outre de « comprendre une consigne » (15 fragments codés), de « visualiser mentalement » (9 fragments) au sens d'anticiper le résultat de l'exécution des instructions de déplacement sur une grille, ou, en sens inverse, de trouver les instructions correspondant à un déplacement. Ils constatent cette capacité chez un élève : « *Manifestement, cet élève a bien compris la situation et est capable de visualiser pas à pas le déroulement du programme sans l'exécuter* », ou, au contraire, s'étonnent de l'insuccès des élèves : « *Je suis surpris qu'il ne parvienne pas à exécuter le déplacement mentalement* ». Ils opposent cette capacité au fait de faire exécuter plusieurs fois le programme pour en voir le résultat : « *L'élève a besoin d'exécuter le programme et corrige au fur et à mesure. Cela révèle peut-être un manque de capacités d'abstraction* ».

Les références à la capacité d'évaluation sont également importantes. Il est le plus souvent question d'une démarche que les enseignants dénomment « essai-erreur » (15 fragments codés). L'élève efficace fait des prévisions et organise des tests pour « vérifier une solution » (14 fragments) : *« il a voulu faire un premier essai pour vérifier ses prévisions puis a conclu au deuxième essai* ». Le terme erreur ne semble d'ailleurs pas approprié : « *L'élève a acquis la démarche tentative/essai sur des instructions courtes, répétitions peu nombreuses. Une fois validée par le test, il n'a plus qu'à itérer autant que nécessaire*. » Une enseignante qualifie cette démarche de « *stratégie informatique : essai sur peu de code pour vérifier au fur et à mesure que cela fonctionne, permet de gérer plus facilement les bugs*. »

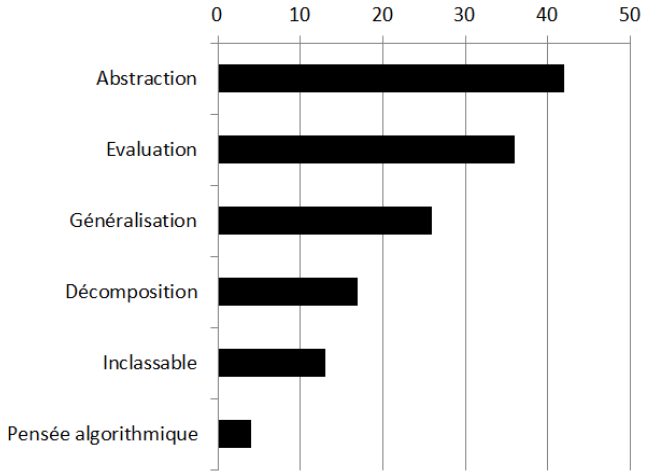

Figure 2 : Nombre de fragments de discours codés dans chaque compétence.

Cependant, ce procédé peut être « laborieux », voire inefficace, donnant lieu à des erreurs : « *L'élève procède par essais erreurs. Il fait de nombreux essais infructueux et ne semble pas pouvoir anticiper le résultat sans exécuter le programme*. ». Dans ce cas, l'efficacité passe par la capacité à identifier les erreurs (16 fragments) : *« Démarche normale pour quelqu'un qui visualise bien ce qu'il y a à faire et constatant une erreur ou pensant qu'il y a une erreur, s'aide d'un support (la souris) pour tester son codage »* et à les corriger : « *Essai/Erreur assez productif au final ; il rentre petit à petit dans le jeu et apprend de ses erreurs pour avancer*. ».

La capacité de généralisation se retrouve lorsque l'élève est capable de reconnaitre la situation proposée : « *L'élève connait la structure itérative et sait reconnaître une situation itérative* » ou sait comprendre la caractéristique du problème : « *il comprend la nécessité d'éviter la collision [entre deux robots]* ». Un enseignant parle également

de savoir adopter « *une vision globale du problème* » et ne pas se cantonner à réaliser un premier motif sans voir qu'il faudra le réutiliser dans une boucle plus large.

La capacité de décomposition est révélée lorsque l'élève procéde par étapes distinctes et que l'enseignant peut numéroter les étapes et en décrire l'objectif. L'enseignant peut également en regretter l'absence : « *N'essaie pas de coder d'abord un motif mais l'ensemble de la boucle sans comprendre* ». Une stratégie revient plusieurs fois, contraire à la décomposition, qui est basée sur la déduction. Il s'agit du « procédé pas à pas » (7 fragments codés). L'élève semble avoir peu de connaissances, et ne pas être en mesure de généraliser le problème : « *sans vision globale, il continue sa programmation pas-à-pas* ».

La référence à une pensée algorithmique est rare dans les analyses des enseignants. Elle ressort quand l'enseignant formule le programme que l'élève est en train d'écrire. Par exemple, pour le défi « Courses de grenouilles », un enseignant explique que l'élève « *identifie les ordres de déplacements pour les deux grenouilles (...) et enfin retarde l'arrivée pour synchroniser les deux programmes* ».

Certaines stratégies identifiées par les enseignants sont apparues ne pas correspondre à l'une des capacités de la pensée informatique et ont été codées dans les « inclassables ». Il s'agit notamment du procédé par « tatônnement ». Il est parfois jugé normal « *quand on ne sait pas, on essaie, on tâtonne* », mais il peut aussi être jugé négativement en regard de l'efficience de la stratégie « *10 minutes de tâtonnement, essais, modifications pour parvenir au résultat* » et associé à la capacité à prendre une décision « *c'est la stratégie d'un élève qui ne sait pas faire, qui ose peu ; il découvre petit à petit comment résoudre le problème* ».

Dans les inclassables, on trouve aussi des capacités manipulatoires, liées à l'utilisation d'une interface : l'élève « *semble habitué à "manipuler", les connaissances sont donc plus générales et "pratiques" que liées à ce type d'exercice* » ou éventuellement liées à la pratique des jeux : « *Dans certains jeux pédagogiques l'utilisation des flèches pour coder un déplacement est courant* ».

DISCUSSION ET CONCLUSION

La question que nous nous posions était celle de savoir quelles compétences sont repérées par les enseignants d'Informatique et Sciences du Numérique lorsqu'ils analysent l'activité filmée d'élèves, en train de résoudre des défis du concours Castor. Dans quelle mesure ces compétences relèvent-elles de la pensée informatique ?

La pensée informatique est définie comme un ensemble de méta-compétences difficiles à observer. Nous avons fait l'hypothèse que l'analyse de l'activité des élèves par des enseignants pouvait faire émerger des éléments observables de ces compétences.

Notre méthodologie a permis d'identifier les compétences repérées par les enseignants, puis de les décontextualiser en les associant aux méta-compétences de la pensée informatique. Les résultats montrent que les cinq compétences de la pensée informatique sont bien repérées par les enseignants, même si elles n'apparaissent pas en ces termes dans leur discours.

Wing, à l'origine du regain d'intérêt pour la pensée informatique, a récemment précisé cette notion. Elle la définit comme le processus de pensée impliqué dans la formulation d'un problème et l'expression d'une solution calculable, par un humain ou un ordinateur (Wing, 2014). Si Wing insiste sur la formulation du problème, les défis du concours Castor incitent à la recherche de procédures de résolution.

Julo (1995) définit de telles procédures comme correspondant à l'ensemble des opérations élémentaires mis en œuvre pour atteindre le but proposé. L'une de ces procédures, observées et signalées par tous les enseignants, est celle de l' « essai/erreur » en informatique, qui reste à définir. Elle rejoint l'une des pistes identifiée par Julo pour découvrir la solution à un problème : par l'action, le tatônnement et les essais. Dans les défis du Castor, cette démarche est favorisée par la manipulation d'un artefact informatique qui fournit des indications en réponse aux actions de l'utilisateur. Elle s'explique aussi par le fait que les défis sont conçus de manière à ne pas nécessiter de pré-requis informatique, et autorisent le tatônnement dans la recherche d'une solution. Enfin, elle s'observe d'autant chez ces élèves qu'ils n'ont pas tous suivi un enseignement d'informatique et découvrent certaines notions pendant la résolution.

Décrire ces démarches de résolution nous semble une piste intéressante à suivre, afin d'aborder les compétences de la pensée informatique sous un angle plus « opérationnel » aidant à proposer des activités pédagogiques.

BIBLIOGRAPHIE

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

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

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. *Computing at Schools*.

Dagiene, V., & Sentance, S. (2016). It's Computational thinking! Bebras tasks in the curriculum. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (p. 28-39). Springer.

Dagiene, V., Sentance, S., & Stupurienė, G. (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. *Informatica*, *28*(1), 23-44.

Department for Education. (2013). *The National Curriculum in England*. Récupéré du site : www.education.gov.uk/nationalcurriculum.

Drot-Delange, B. et Tort, F. (2018). Concours Castor, ressource pédagogique pour l'enseignement de l'informatique ? *Colloque Didapro7-DidaSTIC. De 0 à 1 ou l'heure de l'informatique à l'école*, 7 – 9 février 2018. HEP Vaud, Lausanne.

Julo, J. (1995). Représentation des problèmes et réussite en mathématiques: un apport de la psychologie cognitive à l'enseignement. Presses universitaires de Rennes.

Tort, F. & Dagiene, V. (2012) Concours Castor : découvrir l'informatique autrement dans L'éducation aux cultures de l'information, E-Dossiers de l'audiovisuel, ina sup. En ligne : https://www.ina-expert.com

Wing, J. M. (2014). Computational Thinking Benefits Society. *Social Issues in Computing Blog*. New York: Academic Press.