

INTÉGRATION PÉDAGOGIQUE DES TIC
REVUE INTERNATIONALE DE L'AUPTIC

NO 6 — 2025

DEVELOPPER UNE PENSEE INFORMATIQUE ET ALGORITHMIQUE OUTILS, DISPOSITIFS, TRACES, PUBLICS, USAGES ET RESPONSABILITES

Coordination :
Walter Nuninger, Olivier Neuhaus et Mario Ramalho

AUPTIC
• education

« AUJOURD'HUI, LORSQUE L'ON PARLE DE DÉVELOPPEMENT D'UNE PENSÉE INFORMATIQUE POUR LES ÉTUDIANTS ET LES FORMATEURS, IL EST PRIMORDIAL D'INTERROGER LES CHOIX OPÉRATIONNELS POUR UNE INTÉGRATION CRITIQUE DES ÉVOLUTIONS TECHNOLOGIQUES DANS LES PRATIQUES ET DANS LES MÉTIERS »

WALTER NUNINGER, OLIVIER NEUHAUS ET MARIO RAMALHO

IPTIC

TABLE DES MATIÈRES

Walter Nuninger, Olivier Neuhaus, Mario Ramalho
Développer une pensée informatique et algorithmique : outils, dispositifs, traces, publics, usages et responsabilités
05-08

Patrick Etcheverry, Thierry Nodenot, Pantxika Dagorret Marta Toribio-Fontenla, Jennifer Lesbats Christophe Marquesuzaà
Développer une pensée algorithmique: une approche basée sur des design patterns algorithmiques et des ateliers de katas
09-26

Sabrin Housni, Gaëtan Temperman, Audrey Kumps, Karim Boumazguida, Laetitia Dragone, Sarah Descamps Bruno De Lièvre
Intégration de la pensée informatique dans la formation des professionnels de l'éducation : analyse croisée de performance et du degré de certitude
27-40

Neuhaus Olivier, Mario Ramalho
L'autoconfrontation croisée entre étudiants pour former au débogage informatique
41-55

VARIA

Jérôme Albert Schumacher, Esther Duteil
Des usages du numérique dans l'éducation musicale
56-70

COMITÉ DE RÉDACTION

Stéphanie Boéchat-Heer, Haute école pédagogique BEJUNE
Pierre-François Coen, Université de Fribourg
Natacha Noben, Université de Liège
Aurélien Fiévez, Haute spécialisée de Suisse occidentale (HES-SO)
Corinne Ramillon, Haute école pédagogique du Valais

COMITÉ SCIENTIFIQUE

Romaine Carrupt, Haute école pédagogique du Valais
Bernadette Charlier, Université de Fribourg
Pascal Detroz, Université de Liège
André Giordan, Université de Genève
Patrick Giroux, Université du Québec à Chicoutimi
François Larose, Université de Sherbrooke
Marcel Lebrun, Université catholique de Louvain
Emmanuel Sylvestre, Université de Lausanne
Walther Tessaro, Université de Genève
Nathalie Younès, Université Clermont Auvergne

ADMINISTRATION

Claire-Lise Gremion, révision
Noé Coen, mise en page

ISSN 2624-8085

ADRESSE DE LA REVUE

redaction@iptic.ch
www.iptic.ch

AUPTIC
• education

DEVELOPPER UNE PENSEE INFORMATIQUE ET ALGORITHMIQUE : OUTILS, DISPOSITIFS, TRACES, PUBLICS, USAGES ET RESPONSABILITES

Walter Nuninger, Université de Lille, France,
Walter.nuninger@univ-lille.fr

Olivier Neuhaus, Haute Ecole Fédérale en Formation Professionnelle, Suisse,
olivier.neuhaus@hefp.swiss

Mario Ramalho, Ecole de Métiers de Fribourg, Suisse,
mario.ramalho@edufr.ch

Préface

Ce court texte introduit le numéro thématique autour de l'apprentissage de la pensée informatique et algorithmique. Il résume l'orientation des textes retenus ; soit 30% des soumissions suite à un processus de double relecture en aveugle soutenu par 15 relecteurs (53% Suisse, 40% France, 7% Canada). Les coordinateurs tiennent à les remercier pour leur rigueur, les demandes d'amélioration aux propositions retenues et les conseils aux auteurs des travaux non retenus pour le numéro. Les coordinateurs remercient les auteurs pour l'intégration des remarques, propositions et améliorations de leurs propositions. Chaque article a fait l'objet de 3 à 5 relectures par le comité scientifique suivant :

Lionel Alvarez, *Université de Fribourg, Suisse*
Engin Bumbacher, *Haute Ecole Pédagogique Vaud, Suisse*
Morgane Chevalier, *Haute Ecole Pédagogique Vaud, Suisse*
Isabelle Collet, *Université de Genève, G-RIRE, Suisse*
Cédric Fluckiger, *Université de Lille, CIREL (ULR 4354), France*
Patrick Giroux, *Université du Québec à Chicoutimi, CRIFPE, Canada*
Sébastien Jolivet, *Université de Genève, IUFE et TECFA, Suisse*
Olivier Neuhaus, *Haute école fédérale en formation professionnelle HEFP, Suisse*
Walter Nuninger, *Université de Lille, CRIStAL (UMR 9189), France*
Yvan Peter, *Université de Lille, CRIStAL (UMR 9189), France*
Maud Plumettaz-Sieber, *Université de Genève, TECFA, Suisse*
Maude Pupin, *Université de Lille, CRIStAL (UMR 9189), France*
Mario Ramalho, *Ecole des métiers de Fribourg, Suisse*
Yann Secq, *Université de Lille, CRIStAL (UMR 9189), France*
Françoise Tort, *École Normale Supérieure de Paris Saclay*, France

1. Le numérique au cœur des compétences aujourd'hui et demain

La stratégie européenne « Europe 2020 » visait une croissance intelligente, durable et inclusive, soulignant l'importance de l'employabilité, de la mobilité territoriale et de l'intégration sociale. Dans ce contexte, l'éducation et la formation tout au long de la vie doit viser 8 compétences clefs dans le cadre de référence européen (2018/C 189/01) dont les compétences numériques incarnées par le référentiel européen DigComp 2.2 (Vuorikari, et al., 2022). Davies et al. (2011) présentent les compétences professionnelles du futur. Motivées par la digitalisation de la société, elles sont devenues essentielles depuis la crise sanitaire de 2019 qui a transformé l'usage du numérique dans les pratiques professionnelles et l'apprentissage. Lire, écrire, compter et coder relèvent des compétences classiques (Bardeau et Danet, 2014), mais l'expertise numérique, soit développer une pensée informatique et algorithmique (Shute et al., 2017) est devenu un impératif pour tous : il s'agit d'appréhender les outils et solutions numériques pour être en mesure de les utiliser, d'interagir avec le collectif et pour créer de la valeur tout en faisant évoluer sa pratique au regard des évolutions et innovations numériques actuelles et à venir, telles que l'intelligence artificielle, en étant conscient des risques des interactions collectifs.

Le plan d'action européen à l'horizon 2030 « *la voie à suivre pour la décennie numérique* » (UE, 2022) confirme ces objectifs pour le maintien en emploi, la formation des adultes et l'inclusion sociale. La pensée informatique est un « pouvoir-agir » pour atteindre un résultat spécifique grâce à la résolution de problèmes. Aujourd'hui enrichies, les compétences professionnelles futures comprennent d'une part les compétences transformatives liées aux définis sociétaux majeurs (changement climatique, crises globales) en lien avec la capacité d'innovation, le leadership visionnaire et la gestion des conflits et d'autre part, les compétences technologiques qui intègrent, pour les spécialistes, l'utilisation et le développement de technologies avancées comme l'analyse de données, l'intelligence artificielle et l'informatique quantique (Stifterverband für die Deutsche Wissenschaft e.V., 2021).

2. Apprentissage de l'informatique

L'apprentissage de l'informatique, de l'école primaire à l'université et son utilisation dans l'entreprise et au quotidien, révèlent des dilemmes et des écarts d'interprétation (Drot-Delange et al., 2019). La pluralité des attentes se manifeste dans l'utilisation de l'informatique comme outil d'apprentissage et comme objet de savoir (Fluckiger, 2019). Pour développer les compétences numériques retenues et les niveaux cibles en fonction des publics dans les parcours de formation, le curriculum peut couvrir les matériels (hardware), les logiciels (software) mais vise avant tout les concepts de base de l'informatique (affectation des variables, lecture et écriture, alternative et itération (PIAF, 2021)) permettant la représentation des données et le traitement des flux de contenus. Il s'enrichit du langage de description des algorithmes (pseudo-code) et de la modélisation fonctionnelle (Integrated Computer Aided Manufacturing, IDEF0) de l'ingénierie logicielle pour l'assemblage. L'accent est mis sur la conception et la résolution de problème ce qui conduit à l'appropriation des méthodes de développement (notamment par les tests unitaires) pour la performance (code sans erreur et maintenable (Martraire et al., 2022)). La réalisation opérationnelle est illustrée à l'aide de langages de programmation spécifiques et adaptés aux traitements visés. La réalisation s'adapte au contexte de formation (temps et ressources disponibles) et repose sur des choix pédagogiques (exemples structurés de difficultés croissances, apprentissage par problème…) et des environnements et solutions matérielles en lien avec les orientations des filières (Baron et al., 2015).

Pour développer la pensée informatique et algorithmique, des solutions de simulation avec ou sans ordinateur sont utilisées (Vincent et Collectif Tangente, 2018). Les systèmes no-code ou low-code et le langage naturel permettent de s'affranchir des langages de programmation. L'immersion dans des environnements virtuels 3D offre de nouvelles possibilités en termes d'accessibilité et de coût (Alvarez, 2023).

L'intégration de l'intelligence artificielle dans les dispositifs pour soutenir l'apprentissage individuel et collectif questionne les compétences et les pratiques (Kloos et Alario-Hoyos, 2021). Premièrement, les environnements d'apprentissage instrumentés permettent d'analyser la performance des réalisations (production numérique). Deuxièmement, le feedback semi-automatisé pour les développeurs évolue en « tuteurs intelligents » ou Code Companion (ChatBot enrichis par les Large Language Models) qui soutiennent véritablement la réalisation et l'appropriation des concepts selon le principe « apprendre en faisant » (Alario-Hoyos et al., 2024). Ainsi, même si le parti pris de ce numéro n'est pas d'aborder l'impact

de l'intelligence artificielle dans les articles retenus, les perspectives sont bien d'évaluer la pertinence des dispositifs pour l'apprentissage et la pratique professionnelle au regard des enjeux sociétaux en termes de pensée informatique et algorithmique (Kloos et al., 2024).

3. Focus sur trois dispositifs et perspectives

Ce numéro thématique présente des dispositifs mis en place dans différents champs de formation, quels que soient les niveaux et les publics cibles.

Le premier article d'Etcheverry et al. (France) se penche sur une approche ludique pour soutenir l'apprentissage de l'informatique avec des ateliers *katas* pour aborder les concepts de base de l'algorithmique auprès d'un public d'étudiants de baccalauréat. L'intérêt de l'approche réside dans sa conception, la ludification et l'utilisation d'un environnement d'apprentissage pour motiver un apprentissage progressif et réflexif.

Le second article d'Housni et al. (Belgique) s'intéresse au développement de la pensée informatique dans le cadre de la formation au Master de Sciences de l'Éducation, public qui sera amené à accompagner des débutants sur ces mêmes compétences. Le mérite du dispositif est de mettre en situation d'apprenant les futurs formateurs à la Pensée Informatique et Algorithmique (PIA) et de les questionner sur leur sentiment de réussite. Les perspectives sont d'explorer alors leur sentiment de capabilité à aborder de tels concepts avec les futurs élèves.

Enfin, le troisième et dernier article par Neuhaus et Ramalho (Suisse) présente une expérimentation utilisant l'analyse vidéo mise en place pour développer les compétences de débogage informatique des futurs professionnels du domaine. L'intérêt est d'allier la confrontation des pairs et l'outil vidéo pour aider la co-identification des étapes essentielles du processus de validation et de débogage propre au métier de développeur.

4. Conclusion

Les articles retenus révèlent des stratégies pédagogiques innovantes, centrées sur l'apprenant et mises en œuvre de manière opérationnelle avec différents publics en milieu écologique (il n'y a donc pas de groupe permettant la comparaison relativement aux questions de recherche). Dans une approche d'amélioration continue, les choix et résultats concrets sont questionnés. Ils méritent une attention qui, malgré les limites liées aux contextes d'expérimentation, permettent aux auteurs d'apporter leur contribution pour soutenir la pensée informatique et algorithmique. Ainsi, les dispositifs proposés influent bien sur la performance des étudiants, ouvrent des perspectives pour renforcer la résolution de problème, la production numérique et la maîtrise des outils existants et à venir. En centrant sur l'apprenant, la ludification, les approches collaboratives et le regard des pairs, ils sont des atouts indéniables. Cependant, il convient d'identifier les leviers moteurs des évolutions individuelles et collectives, robustes aux contextes et aux constitutions de groupes. Comme pour toute approche active, le constat est bien de renforcer l'accompagnement en situation, par l'humain ou en s'appuyant sur les outils d'analyse de code et la rétroaction semi-automatisée afin de soutenir l'apprentissage mais aussi la pratique professionnelle (Nuninger, 2025).

Quel est alors l'impact de l'intelligence artificielle sur les dispositifs présentés ? Nous avons choisi de ne pas inclure dans ce numéro thématique de proposition autour de l'intelligence artificielle : les évolutions sont rapides et modifient les modes d'apprentissage, les dispositifs et les pratiques d'enseignement. Ce numéro n'aborde donc pas les compétences technologiques propres au traitement des données en nombre, aux grands modèles de langages (LLMs) et à leur exploitation : sont-elles incluses suffisamment dans DigComp au titre de la capacité à produire numériquement en exploitant des solutions existantes ou demandent-elles une dimension supplémentaire spécifique qui s'appuie sur le numérique mais nécessite d'autres connaissances, la reproductibilité, une veille législative et la prise en compte des enjeux durables ? Pour la formation, l'enjeu réside bien aujourd'hui dans l'analyse de la performance et de la robustesse des solutions basée sur les LLMs en ayant conscience des possibilités nouvelles, des risques et de la responsabilité sociétale et éthique des systèmes d'intelligence artificielle notamment pour l'éducation (Directive 2001/95).

Nous vous souhaitons beaucoup de plaisir à la découverte de ces articles.

5. Bibliographie

Alario-Hoyos, C., Kemcha, R., Kloos, C. D., Callejo, P., Estévez-Ayres, I., Santín-Cristóbal, D., Cruz-Argudo, F., et López-Sánchez, J. L. (2024). *Tailoring Your Code Companion : Leveraging LLMs and RAG to Develop a Chatbot to Support Students in a Programming Course.* IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), 1-8. https://doi.org/10.1109/TALE62452.2024.10834365.

Alvarez, J. (2023). *Serious Game, un "carcan ludique" ? Jeux vidéo, travail, instrumentalisations*. Editions Loco.

Bardeau F. et Danet N. (2014). *Lire, Écrire, Compter, Coder*. Edition Fyp Presence.

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

Davies, A., Fidler, D. et Gorbis, M. (2011). *Future Work Skills 2020*. Institute for the Future for University of Phoenix Research Institute. https://legacy.iftf.org/futureworkskills/

Drot-Delange, B., Pellet, J.-P., Delmas-Rigoutsos, Y. et Bruillard, É. (2019). Pensée informatique : points de vue contrastés. *STICEF, 26*(1), 1-24. https://doi.org/10.23709/sticef.26.1.1

Fluckiger, C. (2019). *Une approche didactique de l'informatique scolaire*. Presses universitaires de Rennes.

Kloos, C.D, et Alario-Hoyos, C. (2021). *Educational Pyramids Aligned: Bloom's Taxonomy, the DigCompEdu Framework and Instructional Designs*. Actes de World Engineering Education Forum/Global Engineering Deans Council (WEEF/GEDC), Madrid, Spain, 110-117, doi: 10.1109/WEEF/GEDC53299.2021.9657335.

Kloos, C. D., Alario-Hoyos, C., Estévez-Ayres, I., Callejo-Pinardo, P., Hombrados-Herrera, M. A., Muñoz-Merino, P. J., Moreno-Marcos, P. M., Muñoz-Organero, M., et Ibáñez, M. B. (2024). *How can Generative AI Support Education ?* IEEE Global Engineering Education Conference (EDUCON), 1-7. https://doi.org/10.1109/EDUCON60312.2024.10578716

Martraire, C., Thiéfaine, A, Bartaguiz, D., Hiegel, F. et Fakih, H (2022). *Software craft : TDD, Clean Code et autres pratiques essentielles*. Dunod.

Nuninger, W., (2025). *Performance Indexes for Assessing a Learning Process to Support Computational Thinking with Peer Review*. Actes of the 17th Int. Conf. on Computer Supported Education – Vol. 2, ISBN 978-989-758-746-7, ISSN 2184-5026.

PIAF Consortium (2021). *Pensée Informatique et Algorithmique dans l'enseignement Fondamental*. https://piaf.loria.fr/contributions/referentiel-des-competences/

Shute, V. J., Sun, C., et Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Stifterverband für die Deutsche Wissenschaft e.V. (2021). *Future Skills 2021*. https://www.stifterverband.org/medien/future-skills-2021

Union Européenne-UE (2022). *Programme d'action pour la décennie numérique à l'horizon 2030*. https://eur-lex.europa.eu/eli/dec/2022/2481/oj

Vincent, J.-M., et Collectif Tangente. (2018). *L'informatique débranchée : Le numérique sans ordinateur – Activités de découverte du primaire au lycée*. Pole.

Vuorikari, R., Kluzer, S. et Punie, Y. (2022). *DigComp 2.2 : The Digital Competence Framework for Citizens - With new examples of knowledge, skills and attitudes*. Office of the European Union. doi:10.2760/490274, JRC128415. C.

DÉVELOPPER UNE PENSÉE ALGORITHMIQUE UNE APPROCHE BASÉE SUR DES DESIGN PATTERNS ALGORITHMIQUES ET DES ATELIERS DE KATAS

Patrick Etcheverry, Université de Pau et des Pays de l'Adour, France,
patrick.etcheverry@iutbayonne.univ-pau.fr

Thierry Nodenot, Université de Pau et des Pays de l'Adour, France,
thierry.nodenot@iutbayonne.univ-pau.fr

Pantxika Dagorret, Université de Pau et des Pays de l'Adour, France,
pantxika.dagorret@iutbayonne.univ-pau.fr

Marta Toribio-Fontenla, Université de Pau et des Pays de l'Adour, France,
marta.toribio@iutbayonne.univ-pau.fr

Jennifer Lesbats, Université de Pau et des Pays de l'Adour, France,
jennifer.lesbats@iutbayonne.univ-pau.fr

Christophe Marquesuzaà, Université de Pau et des Pays de l'Adour, France,
christophe.marquesuzaa@iutbayonne.univ-pau.fr

Cet article présente un retour d'expérience sur une approche pédagogique visant à développer la pensée algorithmique chez des étudiants de première année du Bachelor Universitaire de Technologie (B.U.T.) en informatique. L'expérimentation repose sur l'introduction de design patterns algorithmiques, utilisés pour guider la réflexion lors de la résolution de problèmes algorithmiques récurrents. Afin de favoriser leur appropriation, l'approche s'appuie sur un atelier de katas, combiné à des activités de défense collective, d'évaluation par les pairs et de défi individuel. L'expérimentation explore également l'impact de la collaboration en groupe et de la ludification sur l'engagement et la progression des étudiants. Les résultats montrent une amélioration des compétences algorithmiques, bien que des difficultés persistent dans l'adaptation des design patterns à des contextes variés. L'article propose enfin des perspectives visant à accompagner plus finement la structuration de la pensée algorithmique des étudiants, notamment à travers le développement d'outils d'analyse de code permettant un retour réflexif sur leurs productions algorithmiques.

Mots-clés : Design patterns algorithmiques, Katas, Analyse réflexive, Ludification, Collaboration.

1. Introduction

En France, le BUT qui a remplacé le Diplôme Universitaire de Technologie (DUT) en 2021, est une formation professionnalisante en trois ans, visant à préparer les étudiants à des métiers spécifiques. Parmi les 24 spécialités proposées, le BUT Informatique joue un rôle central dans la formation au métier de développeur. Ce cursus, alliant théorie et pratique, met particulièrement l'accent sur des compétences fondamentales, parmi lesquelles figurent la programmation et l'algorithmique. Selon le programme national en vigueur (MESRI, 2022), l'enseignement de l'algorithmique constitue une étape clé pour aider les étudiants à développer une pensée structurée et critique, essentielle à la résolution de problèmes complexes en informatique. Cependant, plusieurs évolutions récentes rendent cet enseignement plus difficile.

Tout d'abord, les programmes ont considérablement réduit le volume horaire alloué à l'algorithmique. Ainsi, dans le programme de 2005 (MESRI, 2005), cette discipline était enseignée tout au long de la première année durant environ 150 heures de cours et travaux dirigés. Cette répartition offrait aux étudiants le temps nécessaire pour assimiler progressivement les notions fondamentales. En revanche, dans le programme actuel (2022), ces notions sont concentrées sur environ 100 heures au premier semestre. Cette réduction des temps d'apprentissage limite les opportunités pour les étudiants de réfléchir en profondeur aux concepts algorithmiques et d'acquérir une pensée algorithmique solide.

Par ailleurs, le profil des étudiants entrant en BUT a également évolué. Depuis l'introduction, en 2011-2012, des spécialités au lycée telles que *Informatique et Sciences du Numérique* (ISN) - devenue *Numérique et Sciences Informatiques* (NSI) - pour les Bacs Généraux et *Systèmes d'Information et Numérique* (SIN) pour les Bacs Technologiques, davantage d'élèves sont sensibilisés à l'informatique dès le secondaire. Toutefois, cette sensibilisation privilégie souvent le volet programmation, avec un accent sur la manipulation concrète du code, au détriment de la réflexion algorithmique, nécessaire en amont.

Ainsi, nous observons en BUT Informatique une réelle difficulté à susciter l'intérêt des étudiants pour la réflexion algorithmique, jugée souvent peu utile car :

- ils réussissent déjà à écrire des programmes, seuls ou au lycée, sans phase de réflexion algorithmique ;
- ils peinent à évaluer la viabilité d'une solution algorithmique sans la programmer ;
- ils ne sont pas sensibilisés à l'analyse de la solution produite, pour en assurer la robustesse.

Les étudiants restent ainsi attirés par l'écriture de code, car la machine, par sa capacité à exécuter rapidement des solutions, offre une validation immédiate : soit le code fonctionne, soit il produit une erreur. Cependant, cette approche par essai-erreur occulte souvent des failles dans les solutions algorithmiques sous-jacentes, comme la gestion inadéquate des cas limites, des duplications inutiles de code ou encore des performances non optimales.

Face à cette situation, le défi est de valoriser la pensée algorithmique dans l'enseignement de l'algorithmique et de la programmation. Il s'agit de faire comprendre aux étudiants l'intérêt d'une réflexion algorithmique pour améliorer une solution et réduire le temps de sa mise au point. La démarche ne cherche pas à dissocier l'algorithmique de la programmation, car les étudiants ressentent un besoin fort de concrétiser leurs idées via du code. Il s'agit plutôt de montrer que des temps investis à la réflexion algorithmique peuvent enrichir et optimiser leurs productions.

L'article propose un retour d'expérience sur une activité réalisée avec des étudiants de première année de BUT Informatique, menée en complément des enseignements classiques (Cours Magistraux, Travaux Dirigés et Pratiques). Elle vise à ancrer des raisonnements algorithmiques appliqués à des problèmes fréquemment rencontrés en programmation, comme l'échange de deux valeurs, le parcours de tableaux ou la recherche d'un élément dans une liste.

Nous nous sommes appuyés sur le concept des katas de programmation, initialement conçus pour perfectionner des compétences de codage par la répétition d'exercices. Nous avons adapté cette méthode en y intégrant des temps de réflexion sur la dimension algorithmique au travers de design patterns algorithmiques. Pour maintenir l'engagement des étudiants lors de ces phases d'apprentissage, souvent perçues comme rébarbatives, nous avons ajouté une dimension collaborative et ludique.

L'article est organisé de la façon suivante : la section 2 présente les fondements pédagogiques de la pensée algorithmique ainsi que les travaux en lien avec l'usage des design patterns et des katas en pédagogie. La section 3 détaille le dispositif mis en place, incluant les activités pédagogiques et la méthodologie employée.

La section 4 décrit l'expérimentation menée et les indicateurs analysés. La section 5 expose les résultats obtenus et discute leur portée. Enfin, la section 6 conclut en soulignant les contributions et les perspectives d'amélioration du dispositif.

2. Cadre théorique

Nous présentons ici les fondements théoriques liés à la pensée algorithmique, à ses composantes essentielles et aux stratégies pédagogiques susceptibles d'en favoriser l'apprentissage.

2.1 Pensée informatique et pensée algorithmique

La pensée **informatique** regroupe un ensemble de compétences permettant d'aborder la résolution de problèmes de manière efficace. Parmi elles, *l'abstraction* joue un rôle fondamental en permettant de représenter les données et processus sous une forme simplifiée et manipulable (Wing, 2006 ; Barr et Stephenson, 2011). La *modélisation* constitue une étape clé de cette démarche, en structurant un problème sous une forme exploitable pour un traitement informatique (Denning, 2009 ; Selby et Woollard, 2013).

Au-delà de la mise en œuvre d'une solution, la pensée informatique intègre également des mécanismes d'*amélioration continue* et *d'évaluation* qui permettent d'ajuster, d'optimiser et d'évaluer les solutions en fonction des contraintes rencontrées (Wing, 2006 ; Brennan et Resnick, 2012).

La pensée **algorithmique**, bien que faisant partie intégrante de la pensée informatique, se distingue par son focus sur les stratégies de résolution de problèmes à travers des séquences d'instructions bien définies. Un élément central de cette approche est la *décomposition*, qui consiste à diviser un problème complexe en sous-problèmes plus faciles à résoudre (Wing, 2006 ; Grover et Pea, 2013). Cette capacité de fragmentation est complétée par le *séquençage*, qui organise ces étapes dans un ordre logique garantissant la cohérence et l'efficacité de la solution finale (Brennan & Resnick, 2012).

Une fois la structure définie, la *modélisation logique* permet de formaliser la solution d'un problème à l'aide de structures de contrôle comme des itérations ou des conditions (Selby et Woollard, 2013). L'*analyse* de la solution évalue sa robustesse face aux cas limites et sa capacité d'adaptation à de nouvelles contraintes (Grover et Pea, 2013).

Ainsi, la pensée algorithmique constitue un levier essentiel dans la formation des étudiants, leur permettant d'acquérir des compétences transférables dans divers contextes informatiques et au-delà (Drot-Delange et al., 2019). Son intégration dans l'enseignement, notamment via des approches pédagogiques adaptées, apparaît donc comme une nécessité (Brennan et Resnick, 2012).

Les contributions proposées dans cet article s'inscrivent précisément à l'intersection de la pensée informatique et de la pensée algorithmique.

2.2 Deux grandes familles de problèmes algorithmiques

L'apprentissage de l'algorithmique se structure autour de deux familles de problèmes, chacune jouant un rôle clé dans le développement des compétences des étudiants.

Problèmes classiques et récurrents. Ces problèmes, tels que l'échange de deux valeurs ou le parcours d'une collection, apparaissent fréquemment et s'appuient sur des méthodes de résolution établies. Grâce à leur récurrence, les étudiants identifient des schémas de raisonnement réutilisables et structurent progressivement leur approche algorithmique (Fluckiger, 2019). L'introduction de modèles de solutions éprouvés constitue un cadre structurant, évitant aux étudiants de repartir de zéro face à chaque nouveau problème (Resnick, 2013).

Problèmes complexes et solutions originales. Ces problèmes nécessitent une analyse approfondie et mobilisent une réflexion algorithmique avancée. Wing (2006) et Grover et Pea (2013) soulignent leur importance pour développer une pensée critique et créative, essentielle dans la résolution de problématiques ouvertes.

Ces deux catégories de problèmes ne s'opposent pas, mais se complètent. Les problèmes pouvant être résolus avec des modèles de solution réutilisables offrent un socle solide, aidant les étudiants à affronter des défis plus complexes (Brennan et Resnick, 2012).

2.3 Dissociation entre réflexion algorithmique et syntaxe

Un des défis majeurs dans l'enseignement de l'algorithmique réside dans la dissociation entre réflexion abstraite et contraintes syntaxiques. Cette séparation permet aux débutants de se focaliser sur le raisonnement algorithmique sans être freinés par les erreurs de syntaxe ou d'exécution (Fluckiger, 2019). Le lecteur trouvera dans Branthôme (2023) des travaux complémentaires centrés sur les questions de syntaxe lors du passage d'une représentation graphique d'algorithme (ou de programme) en lignes de code.

Selby et Woollard (2013) insistent sur l'importance d'une telle dissociation pour aider les étudiants à structurer leur raisonnement et renforcer une compréhension durable des concepts algorithmiques. Ils apprennent à structurer leur réflexion de manière progressive tout en allégeant leur charge cognitive (Grover et Pea, 2013 ; Labouche, 2024).

De son côté, Wing (2006) souligne qu'en appréhendant d'abord les concepts abstraits, les étudiants développent des compétences transférables à divers langages de programmation. Dans cette optique, des outils visuels comme Scratch ou Blockly offrent un environnement propice à l'apprentissage des fondements de l'algorithmique (Brennan et Resnick, 2012).

2.4 Des temps dédiés à la réflexion algorithmique

2.4.1 L'importance de l'analyse préalable

Une analyse préalable approfondie permet non seulement de mieux cerner les exigences et contraintes d'un problème, mais aussi d'anticiper les difficultés potentielles et d'optimiser la conception de la solution. Décomposer les objectifs, identifier les contraintes et anticiper les cas limites favorisent une approche méthodique et rigoureuse (Jonassen et al., 2006).

Papadakis et Kalogiannakis (2020) soulignent que cette analyse préalable joue un rôle essentiel dans la consolidation des apprentissages et le transfert des compétences vers de nouveaux contextes. Encourager les étudiants à formaliser explicitement leur réflexion avant de coder leur permet de structurer leurs idées et de mieux en comprendre les enjeux.

2.4.2 La confrontation des idées pour affiner les solutions

L'échange avec les pairs ou les enseignants stimule la pensée algorithmique et renforce l'autorégulation cognitive. Vygotsky (1978) met en avant le rôle des interactions sociales dans l'apprentissage, en particulier via la zone proximale de développement. Domínguez et al. (2022) montrent que la collaboration améliore la performance des étudiants en programmation en favorisant l'analyse critique et la diversification des stratégies.

2.4.3 La réflexion post-solution pour renforcer les apprentissages

Analyser *a posteriori* une solution permet d'en évaluer la robustesse et d'identifier des améliorations possibles. Schön (1983) parle de « réflexion sur l'action » comme un levier pour développer la pensée critique et la transférabilité des connaissances. Borge et White (2016) soulignent que le feedback post-solution renforce l'apprentissage en engageant les étudiants dans un questionnement sur leurs propres raisonnements.

2.4.4 La valeur des schémas de solutions réutilisables

Les design patterns popularisés par Gamma et al. (1995) formalisent des solutions éprouvées et réutilisables, facilitant la structuration des raisonnements. Leur intérêt réside dans leur réutilisabilité (adaptables à divers contextes), leur structuration de la pensée (aide à l'analyse des problèmes) et leur indépendance du langage (applicables à différents paradigmes de programmation).

Sweller et al. (2011) expliquent, via la théorie de la charge cognitive, que l'apprentissage est plus efficace lorsque les étudiants s'appuient sur des exemples structurés plutôt que de créer une solution *ex nihilo*. Ce processus favorise la reconnaissance de schémas et la transposition des solutions à de nouveaux contextes, réduisant ainsi la complexité cognitive de la résolution de problèmes.

Grover et Pea (2013) insistent sur la nécessité d'identifier et d'exploiter des structures récurrentes en algorithmique. Cela inclut :

- les structures de contrôle fondamentales (itérations, conditions, modularisation) ;
- les stratégies de raisonnement algorithmique, permettant de transposer des solutions apprises à de nouvelles variantes d'un problème.

Brennan et Resnick (2012) soulignent l'importance des motifs récurrents (*computational patterns*) qui facilitent une compréhension transférable. Par exemple, l'échange de deux valeurs suit une séquence standardisée d'affectations applicable à divers types de données, illustrant ainsi l'intérêt d'un enseignement fondé sur des modèles réutilisables.

2.5 Le kata comme approche d'apprentissage par la répétition

L'acquisition des compétences algorithmiques, basée sur la capacité à reconnaître et mobiliser des schémas de solution génériques répondant à des problèmes récurrents, n'est pas une activité spontanée, elle nécessite un entraînement structuré.

2.5.1 L'apprentissage par la répétition : un levier cognitif reconnu

En sciences cognitives, les travaux d'Anderson (1983) sur l'acquisition des connaissances procédurales montrent que la répétition favorise l'apprentissage.

Selon l'auteur, l'apprentissage débute par une application consciente des règles avant d'aboutir à une automatisation progressive. Ericsson (2007) montre également que la maîtrise d'une compétence repose sur un entraînement délibéré où les erreurs sont progressivement corrigées. Enfin, Logan (1988) explique que la répétition favorise un accès rapide aux solutions mémorisées, facilitant ainsi la transition entre une réflexion analytique et une application automatisée.

2.5.2 Katas de programmation : un dispositif structurant pour l'apprentissage actif

Un kata de programmation se compose d'une série d'exercices courts et ciblés, répétés sous des variations progressives. Introduits en programmation par Thomas et Hunt (1999), il reprend le principe des arts martiaux où la répétition systématique permet une assimilation progressive. Appliqués à l'algorithmique, les katas peuvent renforcer la capacité à résoudre des problèmes récurrents pour lesquels il existe des solutions génériques pré-établies.

En dépassant une approche purement empirique, les katas peuvent favoriser une appropriation durable des concepts et offrent un entraînement adapté pour apprendre à maîtriser la résolution de problèmes récurrents, tout en renforçant l'autonomie et la flexibilité des étudiants face à des situations algorithmiques variées.

2.6 Ludification appliquée à la pensée algorithmique

La ludification consiste à intégrer des mécanismes inspirés du jeu pour favoriser l'engagement des apprenants. Cependant, si ces approches peuvent renforcer l'implication, elles ne garantissent pas une motivation universelle et doivent être appliquées avec discernement. Viau (1997) identifie trois leviers fondamentaux de la motivation en contexte scolaire : la perception de valeur (« *cette activité est-elle utile pour moi ?* »), la perception de compétence (« *suis-je capable de réaliser cette activité ?* ») et la perception de contrôlabilité (« *le résultat dépend-il de moi ?* »). Le dispositif que nous présentons en section 3 tente d'agir sur ces trois axes, par des défis progressifs et un système d'évaluation adaptatif. Il met en œuvre les leviers suivants :

1. *Introduction d'éléments narratifs* (Dicheva et al., 2015) pour renforcer l'engagement des étudiants dans les exercices en plus de leur expliquer les étapes clés d'une réflexion algorithmique.
2. *Défis progressifs* (Kapp, 2012) pour que les exercices proposés maintiennent un équilibre entre challenge et sentiment de compétence. La complexification des problèmes algorithmiques ne doit pas engendrer une démotivation des étudiants.

3. *Limitation temporelle* pour stimuler la concentration et l'engagement (Landers et Landers, 2014), mais pas de stress contre-productif. En aucun cas la difficulté des exercices conjuguée à la pression temporelle ne doit constituer un obstacle insurmontable pour l'étudiant.
4. *Compétition mais aussi entraide entre groupes* pour encourager les étudiants à se dépasser (Huang et Soman, 2013). Au-delà de la compétition, la mise en avant d'objectifs collectifs sera un levier pour renforcer la dynamique de collaboration et l'apprentissage entre pairs.
5. *Récompenses et reconnaissance* pour matérialiser la progression des étudiants et maintenir la motivation (Hamari et Koivisto 2015). Il s'agira plus de tenir compte des progrès individuels que de la seule réussite/performance (Deci et Ryan, 2000).

3. Dispositif pédagogique proposé

Le dispositif propose un ensemble d'activités amenant les étudiants à se mettre en situation de répondre à un besoin formulé à l'aide d'une série d'exercices sous forme de jeux de katas de programmation. Ces activités et exercices sont conçus pour aider les étudiants à maîtriser les design patterns algorithmiques. La scénarisation des activités est basée sur les principes de la ludification, l'ensemble du dispositif étant implémenté sur la plateforme Moodle.

3.1. Des design patterns algorithmiques pour structurer la réflexion

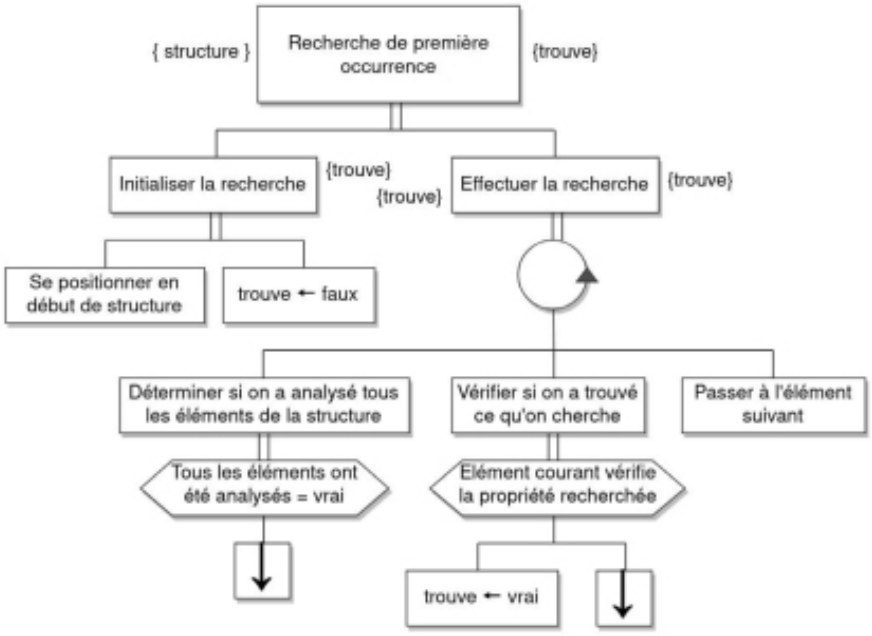
Afin d'intégrer les principes énoncés dans la section précédente dans un cadre pédagogique structuré, nous avons formalisé des design patterns algorithmiques adaptés aux problèmes classiques et récurrents en informatique. Dans la lignée des design patterns en génie logiciel (Gamma et al., 1995), ces modèles fournissent des solutions génériques applicables à des familles de problèmes tels que le parcours de collections d'éléments, la recherche d'une valeur dans un ensemble, ou encore la saisie sécurisée d'une donnée.
Parmi les design-patterns enseignés, citons notamment :

- *Échange de deux valeurs* : décrit les étapes génériques pour intervertir les valeurs de deux variables.
- *Saisie-vérification* : permet d'assurer qu'une valeur saisie respecte une contrainte (ex. : un numéro de mois compris entre 1 et 12).
- *Parcours complet avec traitement systématique* : applique une action à tous les éléments d'une collection (ex. : mettre en majuscule tous les caractères d'un mot).
- *Parcours complet avec traitement conditionné* : applique une action uniquement aux éléments vérifiant une propriété donnée (ex. : incrémenter de 1 toutes les valeurs négatives d'un tableau).
- *Recherche de première occurrence* : identifie la présence d'un élément correspondant à un critère dans une collection (ex. : détecter si une liste contient au moins un nombre négatif).

Le formalisme utilisé pour représenter ces design patterns repose sur des éléments visuels clés qui reprennent les concepts fondamentaux de l'algorithmique (*cf.* section 2.1) :

- Décomposition du problème (| |) pour diviser la tâche en hauteur en sous-problèmes.
- Séquençage des étapes (—) pour structurer en largeur la progression logique de la résolution.
- Modélisation des structures de contrôle (conditions et itérations) sous forme de symboles spécifiques.

Figure 1 : design pattern algorithmique pour détecter la présence d'un élément vérifiant une propriété particulière dans une collection d'éléments.



La Figure 1 donne l'aperçu d'un design pattern applicable pour résoudre un problème visant à détecter, dans un ensemble d'éléments, la présence d'au moins un élément vérifiant une propriété donnée : vérifier la présence d'un nombre négatif dans un tableau, vérifier la présence d'une majuscule dans une chaîne de caractères, etc.

Ce design pattern propose des étapes de résolution qui sont toujours les mêmes pour résoudre ce type de problème : se positionner au début de la structure (au début du tableau / au début de la liste / au début du fichier…), supposer que la recherche ne va pas aboutir puis répéter les actions suivantes :

- déterminer si on a analysé tous les éléments (du tableau/de la liste/du fichier…) en vérifiant si on est en fin de structure. Si c'est le cas la recherche prend fin ;
- déterminer si l'élément courant vérifie la propriété recherchée (le nombre est négatif/le caractère est une majuscule). Si c'est le cas, on infirme l'hypothèse de départ en indiquant que la recherche a abouti et on stoppe la recherche ;
- tenter de passer à l'élément suivant (à la cellule suivante dans le tableau, à l'élément suivant dans la liste, à la ligne suivante dans le fichier…).

Les étapes de résolution doivent donc être adaptées en fonction de la structure dans laquelle on effectue la recherche (une chaîne de caractères, un tableau d'entiers, un fichier…) et de la propriété recherchée (un caractère en majuscule, un nombre négatif, un mot de 5 lettres…).

La Figure 2 illustre la manière d'exploiter le design pattern de la figure 1 pour produire un algorithme (en pseudo-code) permettant de déterminer si une valeur négative se trouve :

- dans une structure de type tableau (algorithme de gauche) ;
- dans une structure de type liste (algorithme de droite).

Figure 2 : exploitation du design pattern algorithmique de la figure 1 pour rechercher la présence d'une valeur négative dans un tableau (à gauche) ou dans une liste (à droite).

```
// Initialiser la recherche
1.1 Se positionner en début de structure
    i ← 0
1.2 Initialiser la variable résultat
    trouve ← Faux

// Effectuer la recherche
2. REPETER
    2.1 Déterminer si on a analysé
        tous les éléments de la structure
        SI i ≥ nbCases
            ARRETER
        FIN SI

    2.2 Vérifier si on a trouvé ce qu'on
        cherche :
        SI tab[i] < 0
            trouve ← Vrai
            ARRETER
        FIN SI

    2.3 Passer à l'élément suivant :
        i ← i + 1
FIN REPETER

// La variable "trouve" contient le
résultat
```

```
// Initialiser la recherche
1.1 Se positionner en début de structure
    allerDebut(liste)
1.2 Initialiser la variable résultat
    trouve ← Faux

// Effectuer la recherche
2. REPETER
    2.1 Déterminer si on a analysé
        tous les éléments de la structure
        SI parcoursEnCours(liste) = faux
            ARRETER
        FIN SI

    2.2 Vérifier si on a trouvé ce qu'on
        cherche :
        SI elementCourant(liste) < 0
            trouve ← Vrai
            ARRETER
        FIN SI

    2.3 Passer à l'élément suivant :
        avancer(liste)
FIN REPETER

// La variable "trouve" contient le
résultat
```

3.2 Pratique des design patterns algorithmiques par le jeu des katas

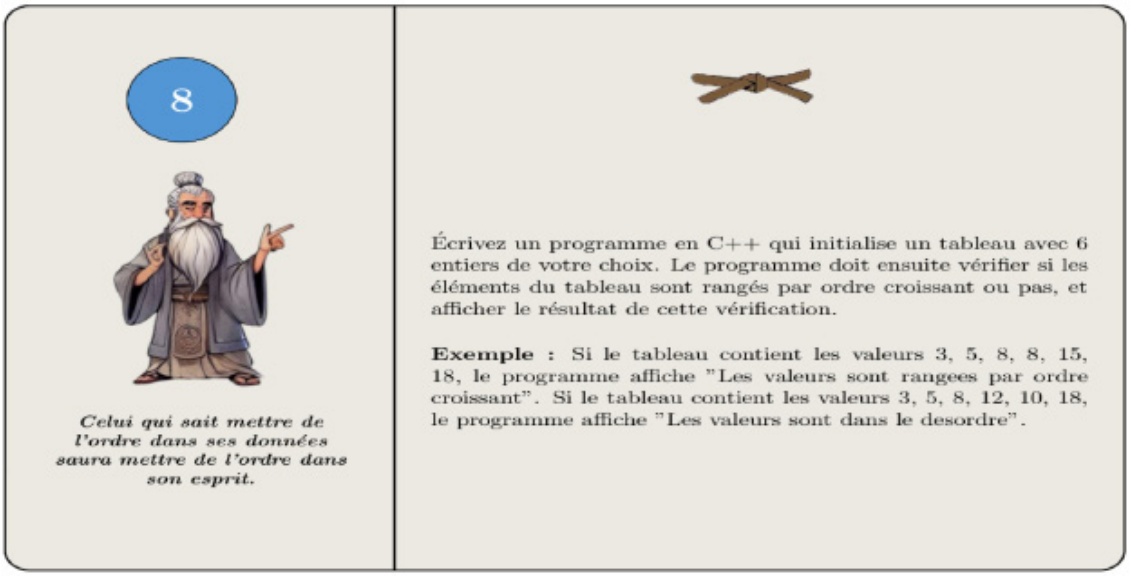
Les katas de programmation offrent un cadre structuré favorisant l'assimilation des design patterns par la répétition ciblée et la progression graduelle (*cf.* section 2.5). En s'exerçant de manière répétée sur des problèmes similaires mais sous différentes variantes, les étudiants intègrent progressivement des schémas de résolution proposés par ces design patterns.

Dans notre dispositif, un kata est une séquence d'exercices algorithmiques progressifs visant à ancrer une compétence algorithmique spécifique. Nous avons choisi d'utiliser les katas pour renforcer l'appropriation des design patterns algorithmiques, mais d'autres utilisations des katas auraient pu être faites, par exemple pour cibler la décomposition itérative ou conditionnelle qui sont d'autres compétences algorithmiques fondamentales.

L'atelier de katas reprend plusieurs mécanismes inspirés du jeu avec l'objectif d'engager les étudiants dans leur apprentissage.

L'activité est décrite par des éléments narratifs immersifs qui se concrétisent par un mentor fictif qui accompagne les étudiants à travers des proverbes inspirants ou humoristiques.

Figure 3 : un exemple d'exercice présenté par Maître Algoshi dans un kata



L'atelier de katas a par ailleurs une durée limitée pour inciter les étudiants à structurer leur réflexion et à prendre des décisions rapidement démontrant ainsi un certain niveau de maîtrise. Bien que potentiellement stressante, cette contrainte peut aussi stimuler l'engagement (*cf.* analyse des résultats en section 5).

Les exercices de katas sont aussi structurés par niveaux de difficulté croissante ; chaque exercice résolu en débloque un plus complexe, encourageant l'amélioration continue. Pour chaque atelier de katas, la progression des étudiants se matérialise par des badges attestant un niveau de maîtrise croissante des design pattern ; ces badges se matérialisent par un système de ceintures et de niveaux inspiré des arts martiaux (Tableau 1).

Tableau 1 : progression dans les niveaux de maîtrise

Niveau	Couleur de ceinture	Badge	Note max atteignable
1	Ceinture blanche-jaune		7/20
2	Ceinture jaune		8/20
3	Ceinture jaune-orange		9/20
4	Ceinture orange		11/20
5	Ceinture orange-verte		13/20
6	Ceinture verte		15/20
7	Ceinture bleue		16/20
8	Ceinture marron		18/20
9	Ceinture noire		20/20

Les exercices de katas sont réalisés par des groupes auto-constitués pour encourager les étudiants à s'entraider et à se dépasser au sein d'un groupe. Dans l'esprit de la ludification, il y a cependant un aspect compétitif entre groupes au travers des badges/ceintures.

Figure 4 : groupes d'étudiants et niveau de maîtrise correspondant

Niveau de maîtrise	Groupe	
Ceinture marron		Groupe 01 - Les Champions du Kata
		Groupe 09 - Les Elus du Grand Esprit Algorithmique
		Groupe 13 - Les Protecteurs des Secrets Algorithmique
		Groupe 18 - Les Tigres du Code Structuré
Ceinture bleue		Groupe 07 - Les Guerriers du Katana Algorithmique
		Groupe 16 - Les Sentinelles du Code Sacré

3.3 Design patterns algorithmiques : évaluation du niveau acquis

L'apprentissage des design patterns algorithmiques ne repose pas uniquement sur leur mise en pratique intensive au sein d'un kata. Comme vu en section 2.4.3, la réflexion post-solution joue un rôle clé dans la consolidation et l'ancrage durable des apprentissages. L'évaluation de la progression de chaque étudiant sur le niveau de compétences acquis pendant l'atelier de katas prend différentes formes.

1. **Défense collective** : chaque groupe justifie et explicite ses solutions en mobilisant les principes des design patterns algorithmiques étudiés. Cet exercice favorise la prise de recul sur les raisonnements algorithmiques menés durant les exercices de katas. Il incite les étudiants à verbaliser et argumenter leurs décisions (*cf*. sections 2.4.2 et 2.4.3).
2. **Évaluation par les pairs** : en parallèle, les étudiants analysent les productions d'autres groupes et évaluent/critiquent les choix algorithmiques effectués. Comme souligné en section 2.4.3, cette analyse favorise le développement de compétences métacognitives et renforce la capacité à structurer un raisonnement algorithmique.
3. **Défi individuel** basé sur trois principes :

a. Évaluation de la mobilisation d'un design pattern : chaque étudiant résout un problème similaire à ceux abordés durant l'atelier de katas, mais avec une variante. Il s'agit de vérifier s'il identifie et adapte le design pattern approprié sans assistance.

b. Développement de l'autonomie : Contrairement aux exercices de katas réalisés en groupe, cet exercice impose une analyse individuelle de l'énoncé, une identification de la structure du problème et une adaptation du design pattern.

c. Validation de la compréhension approfondie : l'assimilation des design patterns passe par la capacité à reconnaître leur champ d'application et à ajuster leur mise en œuvre en fonction des spécificités d'un problème donné.

Ces différentes évaluations donnent lieu à une note et aident l'enseignant à mesurer le niveau des acquis.

4. Expérimentation

4.1 Objectif de l'expérimentation

Nos travaux de recherche visent à explorer des stratégies pédagogiques permettant d'aider les étudiants débutant en informatique à surmonter leurs difficultés à développer une pensée algorithmique et à la mobiliser efficacement dans la production de programmes informatiques.

L'objectif est de tirer des leçons d'une expérience que nous avons menée avec des étudiants pour améliorer leur réflexion avant et après la phase de codage.

L'expérimentation menée vise à répondre à la question suivante : la mise en place d'un atelier de katas, combinée à une évaluation variée (défense collective, évaluation par les pairs, défi individuel), permet-elle d'améliorer les compétences en algorithmique des étudiants tout en renforçant leur engagement grâce au côté ludique de l'atelier ?

4.2 Profil des étudiants participant à l'expérimentation

L'expérimentation a été menée auprès de 62 étudiants en semestre 1 du BUT Informatique, représentatifs de la diversité des profils rencontrés en première année d'enseignement supérieur (promotion de 56 hommes et 6 femmes, issus de parcours variés) :

- Baccalauréats généraux scientifiques : 45 étudiants (mention TB : 4, B : 6, AB : 30, sans mention : 5).
- Baccalauréats technologiques : 12 étudiants (mention B : 5, AB : 5, sans mention : 2).
- Baccalauréats professionnels : 2 étudiants (mention B : 1, AB : 1).

Ces parcours hétérogènes se traduisent par des différences de bagage informatique et de raisonnement algorithmique. Sur les 4 dernières promotions, 80 à 90 % des étudiants ont fait de l'algorithmique via l'option NSI (Numérique et Sciences Informatiques) pour les bacs généraux et SIN (Systèmes d'Information et Numérique) pour les bacs technologiques. Mais l'hétérogénéité des enseignements au lycée entraîne une compréhension variable des distinctions entre algorithmique et programmation en début d'année.

4.3 Niveau des étudiants avant l'expérimentation

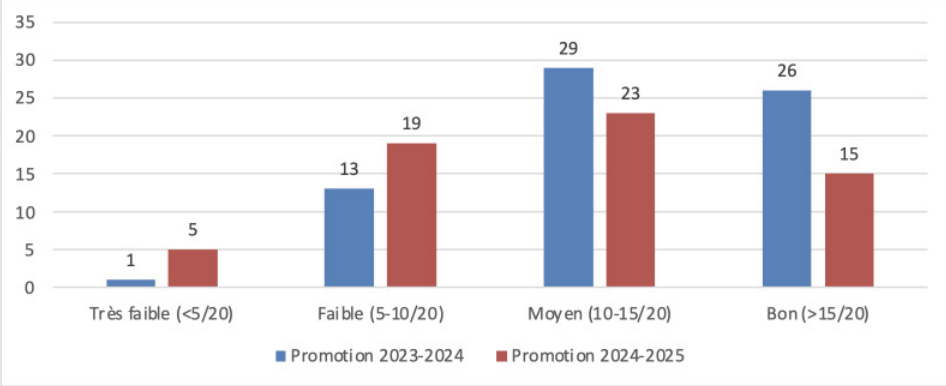
Avant l'expérimentation, un examen de diagnostic a été mené pour évaluer les compétences algorithmiques des étudiants sur des problèmes simples impliquant des structures de contrôle élémentaires (conditions, boucles) et des types de données de base (entiers, flottants, booléens, caractères). Cet examen de diagnostic est réalisé chaque année quelques semaines après le début du module d'enseignement de l'algorithmique en semestre 1.

L'analyse des résultats traduit une forte disparité des niveaux dans la promotion actuelle, avec près de 40 % d'étudiants ayant un niveau faible ou très faible. Pour situer ces résultats, nous avons comparé les performances avec celles de la promotion précédente (Tableau 2 et Figure 5).

Tableau 2 : résultats de l'examen de diagnostic

Indicateurs	Promotion 2023-2024	Promotion 2024-2025
Effectif total	69	62
Note minimale	3,5/20	2,0/20
Note maximale	19/20	17,5/20
Moyenne	12,53/20	10,56/20
Médiane	13/20	11,5/20
Écart-type	3,82	4,31

Figure 5 : examen diagnostic – Répartition des notes et nombre d'étudiants correspondants



Tendances observées :

- Baisse du niveau moyen : 10,56 contre 12,53 pour la promotion précédente, avec une médiane en recul (11,5 contre 13).
- Hausse du nombre d'étudiants en difficulté : 39% des étudiants affichent un niveau faible ou très faible (contre 20% auparavant).
- Réduction du nombre d'excellents profils : 24% des étudiants ont obtenu plus de 15/20, contre 38% pour la promotion précédente.
- Hétérogénéité plus marquée : l'écart-type est plus élevé (4,31 contre 3,82), soulignant un fossé plus important entre les étudiants en difficulté et les meilleurs éléments.

Ces écarts doivent bien sûr être interprétés avec prudence. Bien que l'examen ait été conçu pour évaluer des compétences équivalentes d'une année sur l'autre et ait été corrigé par le même enseignant, plusieurs facteurs externes peuvent expliquer ces différences, notamment la composition de la promotion (parcours scolaires, filières du baccalauréat) ou des variations dans les programmes d'enseignement secondaire. Ainsi, ces résultats offrent un contexte essentiel pour analyser l'expérimentation, sans constituer une base absolue de comparaison entre promotions.

4.4 Déroulé de l'expérimentation

L'expérimentation s'est déroulée sur trois semaines, alternant travail en autonomie et séances encadrées. Après une phase de préparation, les étudiants ont participé à un atelier de katas en groupe, suivi d'une défense collective et d'une évaluation critique. L'expérimentation s'est conclue par un défi individuel permettant d'évaluer la progression de chacun.

4.4.1 Semaine 1 : constitution des groupes et préparation de l'atelier de katas

La première semaine a débuté par une séance encadrée (1h30) présentant les objectifs, critères d'évaluation et organisation de l'expérimentation. Les étudiants ont formé librement des groupes de trois à quatre, puis se sont inscrits sur Moodle, utilisé comme support central.

Une séance en autonomie (1h30) a été consacrée à la révision des design patterns algorithmiques et des exercices associés. Chaque groupe a également défini une répartition des rôles et une stratégie de travail pour optimiser son travail lors de l'atelier de katas.

4.4.2 Semaine 2 : atelier de katas et préparation à la défense collective

Une séance encadrée de 3h a été consacrée à l'**atelier de katas**, en salle informatique. Tous les groupes d'étudiants ont été confrontés à un même kata composé de neuf exercices algorithmiques de difficulté croissante dont la résolution nécessitait de mobiliser un ou plusieurs design patterns algorithmiques. Pour chaque exercice, la démarche suivie par les étudiants est toujours la même :

- Analyse du problème : identification des caractéristiques clés.
- Choix et adaptation du design pattern algorithmique : discussion collective et réflexion sur papier.
- Codage : implémentation en C++ sous Visual Studio Code, avec compilateur et débogueur.
- Validation et dépôt : vérification collective avant soumission sur Moodle pour débloquer l'exercice suivant.

Le lendemain, une séance en autonomie (1h30) a été dédiée à la préparation de la défense collective et de l'évaluation critique :

- relecture et analyse des productions pour structurer l'argumentation ;
- répartition des prises de parole pour assurer une participation équilibrée ;
- première analyse des productions des autres groupes disponibles sur Moodle, en vue de l'évaluation critique.

4.4.3 Semaine 3 : défense collective, évaluation critique et défi individuel

Une séance encadrée de 1h30 a été consacrée à la défense collective et à l'évaluation critique, menées simultanément.

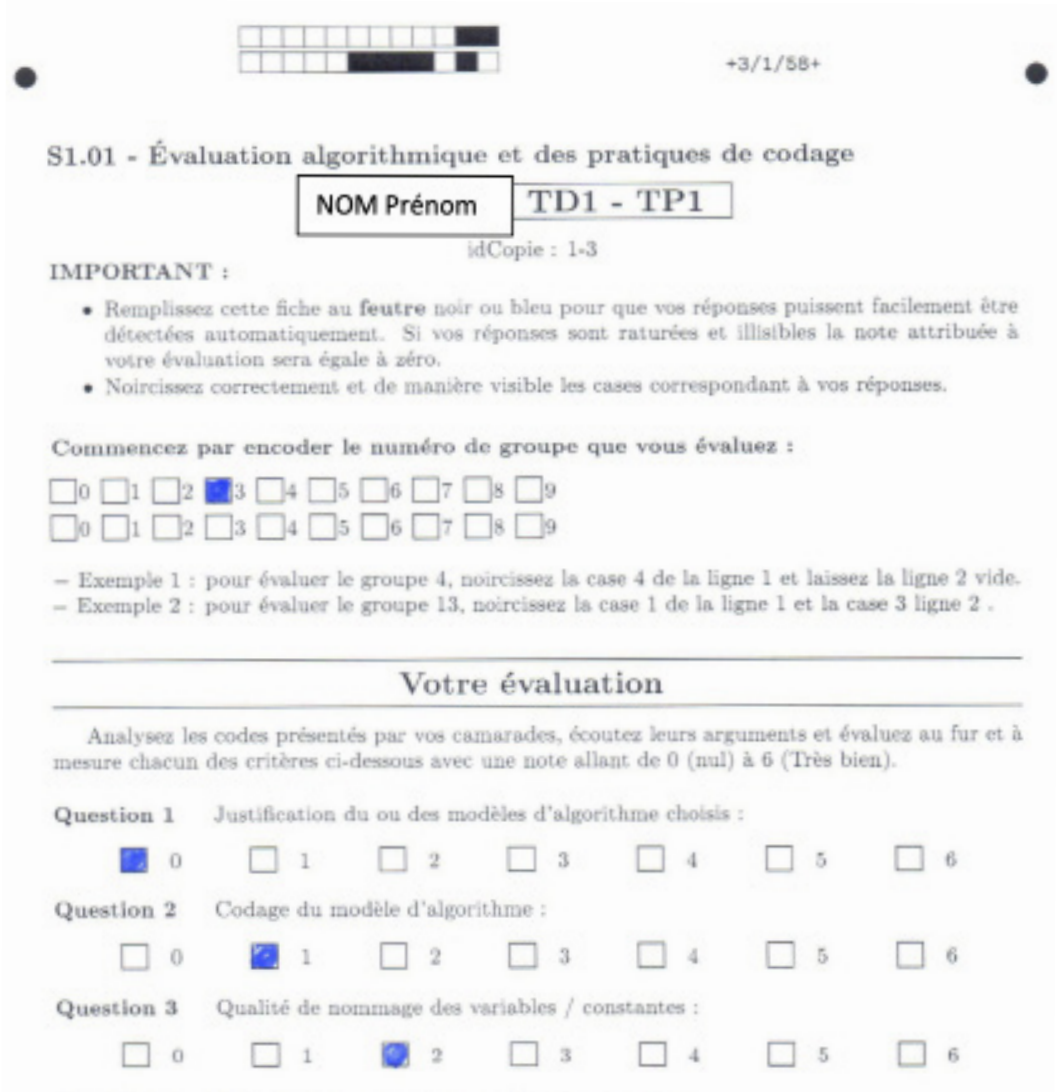
Défense collective. Chaque groupe a présenté et justifié deux solutions algorithmiques :

- L'exercice le plus avancé traité lors de l'atelier de katas.
- Un exercice imposé par l'enseignant pour tester leur capacité à argumenter une solution imparfaite.

Les groupes ont exposé leur raisonnement en projetant leur code depuis Moodle et en explicitant le design pattern algorithmique utilisé.

Évaluation par les pairs. En parallèle, les étudiants ont évalué les justifications algorithmiques de leurs pairs à l'aide de fiches *Auto Multiple Choice* (AMC) afin de faciliter le recueil et l'analyse des réponses (Figure 6).

Figure 6 : extrait d'une analyse critique réalisée par les pairs via des fiches AMC



+3/1/58+

S1.01 - Évaluation algorithmique et des pratiques de codage

NOM Prénom | TD1 - TP1

idCopie : 1-3

IMPORTANT :

- Remplissez cette fiche au **feutre** noir ou bleu pour que vos réponses puissent facilement être détectées automatiquement. Si vos réponses sont raturées et illisibles la note attribuée à votre évaluation sera égale à zéro.
- Noircissez correctement et de manière visible les cases correspondant à vos réponses.

Commencez par encoder le numéro de groupe que vous évaluez :

☐0 ☐1 ☐2 ☑3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

– Exemple 1 : pour évaluer le groupe 4, noircissez la case 4 de la ligne 1 et laissez la ligne 2 vide.
– Exemple 2 : pour évaluer le groupe 13, noircissez la case 1 de la ligne 1 et la case 3 ligne 2 .

Votre évaluation

Analysez les codes présentés par vos camarades, écoutez leurs arguments et évaluez au fur et à mesure chacun des critères ci-dessous avec une note allant de 0 (nul) à 6 (Très bien).

Question 1 Justification du ou des modèles d'algorithme choisis :
☑ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

Question 2 Codage du modèle d'algorithme :
☐ 0 ☑ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

Question 3 Qualité de nommage des variables / constantes :
☐ 0 ☐ 1 ☑ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

Cette évaluation portait sur la pertinence des choix algorithmiques et sur la qualité du code produit. L'enseignant a également noté les productions selon les mêmes critères pour comparer les évaluations des étudiants avec une référence experte. Pour limiter la variabilité des jugements, et du fait que cet exercice d'évaluation est difficile pour des étudiants débutants, les évaluations ont été réalisées sur une échelle de 0 à 6 (plutôt que 0 à 20).
L'expérimentation s'est conclue par un **défi individuel** (1h30, sur papier).

- Objectif : produire un algorithme (sans code) en mobilisant un design pattern algorithmique de manière autonome.
- Niveau d'exercice : l'exercice était choisi entre le niveau 1 et le niveau (max +1) atteint par le groupe de l'étudiant lors de l'atelier de katas.

Le défi individuel poursuivait ainsi un double objectif. D'une part, il permettait d'évaluer le niveau acquis par chaque étudiant en matière de design patterns algorithmiques, en vérifiant sa capacité à identifier et mobiliser le schéma de résolution adapté à un problème donné. D'autre part, il constituait un exercice d'auto-évaluation, où chaque étudiant devait choisir un exercice correspondant à son niveau de maîtrise. Ce choix impliquait un équilibre stratégique : opter pour un exercice plus simple augmentait les chances de réussite mais limitait la note maximale atteignable, tandis qu'un exercice plus ambitieux offrait une meilleure valorisation des compétences mais exposait à un risque d'échec plus élevé.

5. Résultats et discussion

5.1 Impact potentiel de l'atelier de katas

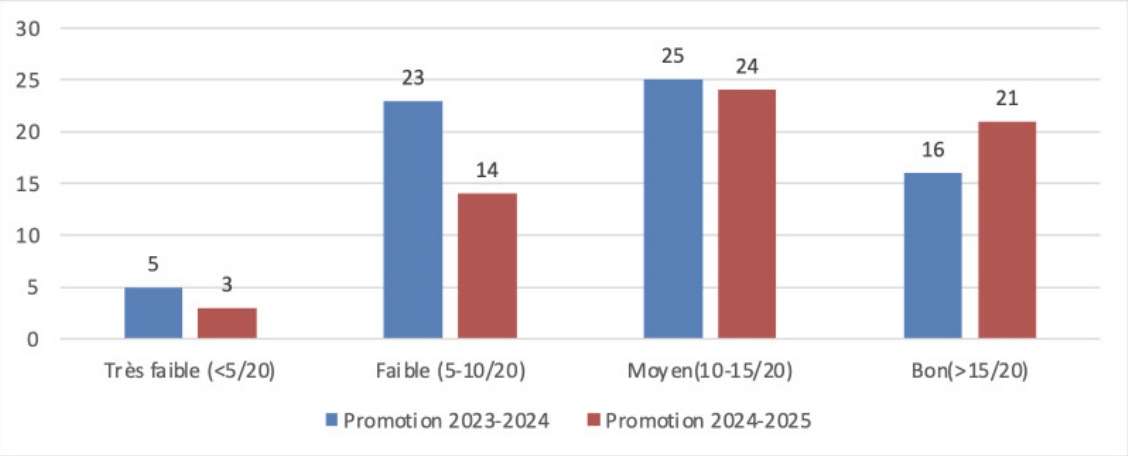
L'évaluation des compétences algorithmiques des étudiants a été réalisée à travers un examen final, couvrant l'ensemble des compétences du module d'algorithmique, y compris celles travaillées dans l'atelier de katas. Pour mesurer l'impact du dispositif, nous avons comparé les résultats de la promotion ayant suivi l'expérimentation (avec atelier de katas) à ceux de la promotion 2023-2024 (sans atelier de katas).

Les données (Tableau 3 et Figure 7) révèlent une légère amélioration des résultats de la promotion actuelle, initialement plus faible en début de semestre (+0,60 point de moyenne).

Tableau 3 : résultats de l'examen final

Indicateurs	Promotion 2023-2024	Promotion 2024-2025
Effectif total	69	62
Note minimale	2,0/20	3,5/20
Note maximale	19,75/20	20/20
Moyenne	12,01/20	12,61/20
Médiane	12,75/20	13/20
Écart-type	4,35	4,27

Figure 7 : examen final – Répartition des notes et nombre d'étudiants correspondants



La proportion d'étudiants ayant obtenu une note supérieure à 15/20 est plus élevée (+10 points de pourcentage), tandis que celle des étudiants en difficulté (<10/20) a diminué (de 41% à 28%). Ces résultats suggèrent que l'atelier de katas et ses trois activités associées (défense collective, évaluation critique, défi individuel) ont pu contribuer à une meilleure assimilation des compétences algorithmiques. Toutefois, ces effets doivent être interprétés avec prudence car d'autres facteurs, tels que la dynamique de groupe et les ajustements pédagogiques, ont pu influencer ces résultats.

5.2 Retours des étudiants

Une enquête anonyme a été menée en fin d'expérimentation pour recueillir les perceptions des étudiants sur les design-patterns algorithmiques et les activités associées. Les résultats détaillés de cette enquête sont consultables à l'adresse suivante :

https://anonym-guest.github.io/student-algo-skills-analysis/

Retours sur l'atelier de katas. L'atelier de katas a été jugé pertinent par 78% des étudiants, confirmant une forte adhésion au dispositif, bien qu'une minorité (5%) reste sceptique. Par ailleurs, 67% estiment que cet atelier a amélioré leur compréhension des design patterns algorithmiques, et autant reconnaissent l'importance de la répétition dans l'apprentissage. 69% souhaiteraient voir ce type d'atelier étendu à d'autres parties du programme, soulignant son potentiel pédagogique.

Retours sur l'intérêt des design patterns algorithmiques. Les design patterns algorithmiques sont perçus comme efficaces pour résoudre des problèmes classiques par 64% des étudiants, bien que 18% les jugent rigides ou inutiles. Un quart des répondants (25%) éprouve des difficultés à les adapter à des contextes spécifiques, mettant en évidence la nécessité d'un accompagnement supplémentaire sur cette phase d'adaptation. Par ailleurs, 76% des étudiants déclarent se sentir plus confiants pour identifier et utiliser le design pattern approprié, traduisant une meilleure compréhension des concepts et une capacité accrue à analyser les caractéristiques des problèmes rencontrés.

Retours sur la défense collective, l'évaluation par les pairs et le défi individuel. La défense collective a été jugée bénéfique par 55% des étudiants pour analyser et justifier leurs choix algorithmiques, tandis que 31% n'ont pas perçu cet apport et 14% n'ont pas donné d'avis. L'évaluation par les pairs a renforcé la compréhension de l'importance du choix du bon design pattern pour 49% des étudiants, mais 31% n'ont pas ressenti cet impact et 20% n'ont pas répondu, laissant planer un doute sur leur ressenti. Le défi individuel a été perçu comme un bon moyen de tester sa maîtrise algorithmique par 69% des étudiants. En revanche, 14% ont éprouvé des difficultés, probablement liées à la contrainte temporelle, et 17% n'ont pas exprimé d'opinion sur son intérêt.

Retours sur la dimension ludique du dispositif. L'ajout d'éléments narratifs (Maître Algoshi) et du système de ceintures a été perçu positivement par 64% des étudiants, renforçant leur engagement. La contrainte temporelle a stimulé 58% des étudiants, mais 31% la jugent problématique ou neutre, suggérant un ajustement pour éviter de pénaliser les moins rapides. Le travail en groupe a favorisé la compréhension pour 64% des étudiants, mais 20% ont trouvé l'approche collaborative peu efficace pour des questions sur la dynamique de groupe.

5.3 Contributions à la pensée algorithmique

Les résultats de l'expérimentation permettent d'identifier plusieurs leviers ayant contribué au développement de la pensée algorithmique chez les étudiants. L'analyse des performances des étudiants ainsi que leurs retours dans l'enquête montrent que certains dispositifs mis en place ont eu un impact plus marqué que d'autres.

L'intérêt des design patterns algorithmiques comme cadre structurant. Les design patterns algorithmiques ont aidé les étudiants à structurer leur réflexion, selon 78% des répondants. Leur usage a favorisé l'élaboration rapide de solutions mieux organisées, en incitant à raisonner en termes de concepts réutilisables plutôt que de traiter chaque problème individuellement. Cependant, certains étudiants ont rencontré des difficultés d'adaptation face à des problèmes nécessitant des ajustements spécifiques. Cette difficulté suggère que, bien que ces patterns offrent un cadre de réflexion pertinent, une pratique répétée est essentielle pour développer une capacité d'analyse et d'adaptation plus fine.

L'atelier de katas : un catalyseur pour la mise en pratique rapide des concepts algorithmique. L'atelier de katas a permis aux étudiants d'appliquer rapidement les design patterns algorithmiques dans le cadre des exercices en groupe. 73% des répondants ont jugé cet atelier bénéfique pour renforcer leur maîtrise des concepts. Le travail collaboratif et le rythme soutenu ont favorisé le développement d'une agilité dans le choix et l'adaptation des patterns. La verbalisation des raisonnements a également joué un rôle clé dans l'approfondissement de la pensée algorithmique. Les groupes les plus performants ont discuté et justifié collectivement leurs choix avant de coder, tandis que ceux ayant privilégié une approche plus intuitive ont parfois eu du mal à structurer leur solution.

Défense collective et évaluation critique : ancrer la réflexion par la justification et l'analyse. La défense collective a été perçue comme bénéfique par 55% des étudiants, leur permettant de prendre du recul sur leurs choix algorithmiques. De même, 49% des répondants ont trouvé que l'évaluation critique les aidait à mieux comprendre l'importance de choisir un design pattern adapté. Ces résultats montrent que ces activités ont contribué à structurer la réflexion algorithmique, en incitant les étudiants à analyser et justifier leurs décisions. Toutefois, la complexité de l'exercice souligne le besoin d'un accompagnement plus explicite sur les critères d'évaluation d'un algorithme pour renforcer ces compétences analytiques.

Le défi individuel : une mise à l'épreuve des compétences acquises. Le défi individuel a constitué un test d'auto-évaluation essentiel, permettant aux étudiants de mesurer leur maîtrise réelle des concepts algorithmiques. 69% des participants ont jugé cet exercice utile pour évaluer leurs compétences de manière autonome. Sans le soutien du groupe ni l'appui d'outils numériques, les étudiants ont dû raisonner uniquement sur la logique algorithmique. Ce défi introduisait aussi une dimension d'auto-évaluation, chaque étudiant devant choisir un niveau d'exercice adapté à son niveau. Cette approche les a incités à réfléchir sur leur progression, en trouvant un équilibre entre prise de risque et sécurisation de la réussite.

Vers un apprentissage algorithmique plus structuré et progressif. L'analyse des résultats met en évidence l'utilité des quatre piliers clés (design patterns algorithmiques, exercices de katas, défense collective incluant une analyse critique, défi individuel) choisis pour développer la pensée algorithmique des étudiants. L'articulation de ces dispositifs favorise un raisonnement algorithmique structuré et progressif. Toutefois, des améliorations restent possibles, notamment en renforçant l'accompagnement pédagogique sur l'évaluation des solutions et l'analyse critique des raisonnements algorithmiques.

6. Conclusion et perspectives

L'expérimentation menée a mis en évidence plusieurs bénéfices significatifs dans le développement de la pensée algorithmique des étudiants. L'introduction des design patterns algorithmiques, associée à des activités favorisant la réflexion avant et après la phase de codage, a structuré leur apprentissage et renforcé leur compréhension des concepts fondamentaux. Les résultats montrent une amélioration de leur capacité à raisonner et à produire des solutions algorithmiques viables.

Cependant, des lacunes persistantes ont été relevées, notamment en matière d'identification et d'adaptation des design patterns, en particulier dans la gestion des cas limites. Pour aller plus loin, nous envisageons d'accompagner davantage les étudiants dans leur réflexion algorithmique pendant l'activité de codage. Dans cette optique, nous développons Code Explorer (https://github.com/patrick-etcheverry/code-explorer), un projet basé sur Tree-sitter qui permet de représenter un code sous forme d'arbre syntaxique (*Abstract Syntax Tree*) puis de faire des requêtes pour l'analyser (https://tree-sitter.github.io/tree-sitter). L'objectif est d'offrir une représentation des codes produits par les étudiants en les décrivant avec un vocabulaire algorithmique : séquences, structures conditionnelles ou itératives, conditions d'arrêt, accumulateurs, compteurs, etc.

À terme, ce système pourrait devenir un compagnon pédagogique, capable de fournir des retours personnalisés aux étudiants, notamment pour :

- Analyser la pertinence des choix algorithmiques : repérer les oublis (initialisation, condition d'arrêt mal positionnée…) et guider l'étudiant vers de meilleures pratiques.
- Reformuler une solution problématique : permettre aux étudiants de prendre du recul en visualisant leur code sous un angle algorithmique avant de corriger leur conception.
- Proposer des améliorations : suggérer des modèles d'algorithmes plus adaptés (ex. remplacement d'une recherche séquentielle par une recherche dichotomique lorsque pertinent).

Ces travaux partent du constat que les étudiants demeurent fortement attirés par l'écriture de code, parfois au détriment de la réflexion algorithmique. L'enjeu est donc de leur fournir des outils favorisant une démarche d'analyse critique, les incitant à concevoir d'abord un raisonnement algorithmique structuré avant de passer à l'implémentation. Ainsi, Code Explorer s'inscrit dans une démarche visant à réconcilier algorithmique et programmation en ramenant régulièrement l'étudiant vers une réflexion algorithmique et une démarche d'analyse critique lui permettant de progresser sur sa réflexion algorithmique.

7. Bibliographie

Anderson, J. R. (1983). *The architecture of cognition*. Harvard University Press.

Barr, V., et Stephenson, C. (2011). Bringing computational thinking to K-12. *Communications of the ACM, 54*(6), 60–65. https://doi.org/10.1145/1929887.1929905

Borge, M., & White, B. (2016). Toward the development of socio-metacognitive expertise: An approach to developing collaborative competence. *Cognition and Instruction, 34*(4), 323–360. https://doi.org/10.1080/07370008.2016.1215722

Branthôme, M. (2023). *Apprentissage de la programmation informatique : analyses et ressources pour accompagner la transition collège-lycée* [Thèse de doctorat, Université de Bretagne occidentale - Brest]. https://theses.hal.science/tel-04397329

Brennan, K., et Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Computer, 52*(6), 28–30. https://doi.org/10.1145/1516046.1516054

Dicheva, D., Dichev, C., Agre, G., et Angelova, G. (2015). Gamification in education: A systematic mapping study. *Educational Technology & Society, 18*(3), 75-88. http://www.jstor.org/stable/jeductechsoci.18.3.75

Drot-Delange, B., Pellet, J.-P., Delmas-Rigoutsos, Y., et Bruillard, É. (2019). Pensée informatique : points de vue contrastés. *STICEF, 26*(1), 1-24. https://doi.org/10.23709/sticef.26.1.1

Ericsson, K. A. (2007). Deliberate practice and the modifiability of body and mind: Toward a science of the structure and acquisition of expert and elite performance. *International Journal of Sport Psychology, 38*(1), 4–34.

Fluckiger, C. (2019). *Une approche didactique de l'informatique scolaire*. Presses Universitaires de Rennes. https://journals.openedition.org/rfp/8523

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. https://doi.org/10.3102/0013189X12463051

Hamari, J., et Koivisto, J. (2015). Why do people use gamification services? *International Journal of Information Management, 35*(4), 419-431. https://doi.org/10.1016/j.ijinfomgt.2015.04.006

Huang, W. H., & Soman, D. (2013). *Gamification of education*. Research Report Series: Behavioural Economics in Action. Rotman School of Management, University of Toronto. https://mybrainware.com/wp-content/uploads/2017/11/Gamification-in-Education-Huang.pdf

Kapp, K. M. (2012). *The gamification of learning and instruction: Game-based methods and strategies for training and education*. Pfeiffer.

Labouche, J.-Y. (2024). Algorithmique débranchée pour le cycle 3. *Les nouvelles technologies pour l'enseignement des mathématiques, (92)*. https://revue.sesamath.net/spip.php?article1661

Landers, R. N., et Landers, A. K. (2014). An empirical test of the theory of gamified learning: The effect of leaderboards on time-on-task and academic performance. *Simulation & Gaming, 45*(6), 769-785. https://doi.org/10.1177/1046878114563662

Logan, G. D. (1988). Toward an instance theory of automatization. *Psychological Review, 95*(4), 492–527. https://doi.org/10.1037/0033-295X.95.4.492

Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation (MESRI). (2005). *Programme national - DUT Informatique*. https://www.enseignementsup-recherche.gouv.fr/sites/default/files/content_migration/document/informatique_157826.pdf

Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation (MESRI). (2022). *Annexe 2 - Licence professionnelle BUT Informatique*. https://www.enseignementsup-recherche.gouv.fr/sites/default/files/annexe-2-licence-professionnelle-bachelor-universitaire-de-technologie-informatique-29016.pdf

Papadakis, S., et Kalogiannakis, M. (Eds.). (2020). *Mobile learning applications in early childhood education*. IGI Global. https://doi.org/10.4018/978-1-7998-1486-3

Selby, C., et Woollard, J. (2013). Computational thinking: The developing definition. *University of Southampton*. https://eprints.soton.ac.uk/356481/

Sweller, J., Ayres, P., et Kalyuga, S. (2011). *Cognitive load theory*. Springer New York. https://doi.org/10.1007/978-1-4419-8126-4

Thomas, D., & Hunt, A. (1999). *The pragmatic programmer: From journeyman to master*. Addison-Wesley Professional.

Viau, R. (1997). *La motivation en contexte scolaire*. De Boeck Supérieur.

Vuorikari, R., Kluzer, S. et Punie, Y. (2022). *DigComp 2.2 : The Digital Competence Framework for Citizens - With new examples of knowledge, skills and attitudes*. Office of the European Union. doi:10.2760/490274, JRC128415. C.

Vygotsky, L. S. (1978). *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press. https://doi.org/10.2307/j.ctvjf9vz4

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

INTÉGRATION DE LA PENSÉE INFORMATIQUE DANS LA FORMATION DES PROFESSIONNELS DE L'ÉDUCATION : ANALYSE CROISÉE DE PERFORMANCE ET DU DEGRÉ DE CERTITUDE

Sabrin Housni, Université de Mons, Belgique,
sabrin.housni@umons.ac.be

Gaëtan Temperman, Université de Mons, Belgique,
gaetan.temperman@umons.ac.be

Audrey Kumps, Université de Mons, Belgique,
audrey.kumps@umons.ac.be

Karim Boumazguida, Université de Mons, Belgique,
karim.boumazguida@umons.ac.be

Laetitia Dragone, Université de Mons, Belgique,
laetitia.dragone@umons.ac.be

Sarah Descamps, Université de Mons, Belgique,
sarah.descamps@umons.ac.be

Bruno De Lièvre, Université de Mons, Belgique,
bruno.delievre@umons.ac.be

Résumé

Notre étude de l'effet de l'intégration de la pensée informatique (PI) dans l'enseignement supérieur a été menée auprès de 122 étudiants en Master Sciences de l'Éducation, spécialisés en Technologie de l'Éducation. Le dispositif comprend six séances de deux heures mêlant théorie, pratique et réflexion conduisant à une évaluation de la PI des étudiants par le « Computational Thinking Test » (CTT) et du sentiment de maitrise des étudiants objectivé par le degré de certitude. Les résultats des étudiants montrent une progression significative des deux variables. L'analyse révèle également une corrélation entre les scores au CTT et le sentiment de maîtrise.

Mots-clés : Pensée informatique, Évaluation, Performance, Degré de certitude, Computational Thinking Test.

1. Introduction

La pensée informatique (PI) est une compétence jugée importante pour s'adapter à l'avenir (Hsu et al., 2018) et pour mieux comprendre et interagir avec les algorithmes qui influencent nos vies (Romero et Vallerand, 2016 ; Dillenbourg, 2018). Elle est indispensable partout où il y a transformation de l'information et innovation des systèmes d'information (Ben Henda, 2016). Elle semble également nécessaire pour comprendre le monde complexe qui nous entoure (Dillenbourg, 2018) et pour répondre au besoin sociétal croissant dans le domaine des métiers digitaux (Amarelle, 2018).

Dans cette perspective, l'école joue un rôle clé dans cet apprentissage. Comme le discutent Bugmann et Karsenti (2018), en intégrant les compétences numériques, elle favorise le développement de la PI dès les premiers stades de l'éducation. Il est essentiel que les enseignants maîtrisent les savoirs associés et développent ces compétences pour pouvoir les enseigner efficacement (Tchounikine, 2017).

Ainsi, en Belgique francophone, l'intégration du nouveau référentiel « Formation Manuelle, Technique, Technologique et Numérique » (FMTTN) dans le cadre de la mise en œuvre d'un nouveau tronc commun renforce cette idée. Ce référentiel indique que la pensée informatique et algorithmique (PIA) doit être développée dès l'enseignement primaire (Fédération Wallonie-Bruxelles, 2022).

Dans ce contexte, notre étude s'intéresse au développement de la PI chez les étudiants du Master en Sciences de l'Éducation, spécialisés en Technologie de l'Éducation à l'Université de Mons en Belgique francophone. Nous cherchons à évaluer dans quelle mesure le cours intitulé « Développement pédagogique de la pensée informatique », basé sur le postulat de l'isomorphisme pédagogique (Develay, 1994), contribue à son développement. Cette recherche vise à proposer des perspectives pour l'intégration de la PI dans les programmes académiques en sciences de l'éducation, afin de renforcer la formation des futurs enseignants et de soutenir l'effort global d'intégration de ces compétences dans le système éducatif.

Cet article se structure comme suit : dans un premier temps, une définition de la PI est présentée, suivie d'une revue des recherches empiriques consacrées à son développement et à son évaluation. Ensuite, nous détaillons la méthodologie de notre recherche, suivie de l'analyse des résultats obtenus. Enfin, l'article se conclut par une discussion sur les implications pour la conception de dispositifs éducatifs visant à développer la PI dans les programmes d'enseignement supérieur.

2. Ancrage théorique

2.1 Définition de la pensée informatique

Le concept de PI, également appelée Computational Thinking, fut utilisé pour la première fois par Papert (1981). Bien qu'il n'existe pas de définition unanimement acceptée de ce concept dans la littérature (Drot-Delange et al., 2019 ; Lockwood et Mooney, 2017 ; Moreno-León et al., 2018), notamment en raison de son évolution en fonction des avancées dans le domaine (Shute et al., 2017), de nombreux auteurs s'appuient sur les définitions proposées par Wing (2006) comme point de départ de leurs réflexions. En effet, Wing (2006) a popularisé le concept de PI en préconisant son apprentissage pour les nouveaux étudiants universitaires. L'auteure définit la PI comme l'habileté à résoudre des problèmes de manière algorithmique. Elle précise que la PI devrait être considérée comme une compétence de la vie quotidienne dont tout le monde a besoin, plutôt qu'une simple expertise en programmation utile uniquement aux informaticiens (Hsu et al., 2018). La PI a donc d'abord été définie comme un processus humain de résolution de problèmes qui utilise la décomposition et nécessite une réflexion à plusieurs niveaux d'abstraction (Wing, 2006), soit les « processus de réflexion impliqués dans la formulation des problèmes et de leurs solutions afin que les solutions soient représentées sous une forme qui puisse être efficacement mise en œuvre par un agent de traitement de l'information » (Wing, 2010, p.20). Aho (2012) propose une perspective plus réductrice en décrivant la PI comme un processus mental qui consiste à formuler des problèmes de manière à pouvoir les résoudre étape par étape à travers des algorithmes.La PI se définit également de manière plus globale pour favoriser le développement de solutions numériques plus universelles comme la capacité de « savoir décomposer un problème en sous-problèmes plus simples ; savoir réfléchir aux tâches à accomplir pour résoudre un problème en termes d'étapes et d'actions (ce que l'on appelle un algorithme) ; savoir décrire les problèmes et les solutions à différents niveaux d'abstraction, ce qui permet d'identifier des similitudes entre problèmes et, par suite, de pouvoir réutiliser des éléments de solutions ; ainsi que, dans l'approche que [l'auteur] adopte, savoir écrire et tester ces algorithmes avec [un] langage de programmation [tel que] Scratch » (Tchounikine, 2017, para. 4).

D'un point de vue pragmatique, la PI est en mesure dès lors de rationaliser notre pensée en structurant les séquences d'instructions permettant de résoudre un problème de manière automatique ou d'accomplir une tâche de manière systématique (algorithme).

Enfin, Roman-Gonzalez (2015) indique que la PI implique la capacité à formuler et à résoudre des problèmes en s'appuyant sur les concepts fondamentaux de l'informatique et en utilisant les lexiques et la syntaxe des langages de programmation dont les éléments fondamentaux de l'affectation des variables, les séquence de base de lecture et d'écriture, les alternatives (conditions) et itérations (boucles).

2.2 Enseigner la pensée informatique

La littérature souligne un manque de clarté quant aux méthodes d'enseignement de la PI (Hsu et al., 2018). L'apprentissage de la programmation informatique est un processus itératif alliant activités de programmation, analyse et résolution d'un problème (Zapata-Cáceres et al., 2020). Ainsi, cela amène à résoudre des problèmes en appliquant une logique structurée en vue de construire des solutions : développer une PI va donc au-delà du simple apprentissage de la programmation (Jimenez et al., 2024).

Dans cette optique, une approche centrée sur l'algorithmique et la programmation apparaît pertinente pour développer la PI. Elle favorise le travail sur des réifications tout en offrant un support concret – qu'il s'agisse d'une activité, d'un écran ou d'un objet de programmation tangible – qui soutient la réflexion et l'action (Tchounikine, 2017). Elle implique de mettre l'algorithme et les structures de base sous-jacentes (notions de variables, de séquence de base de lecture et d'écriture, les alternatives et itérations) au cœur de l'enseignement pour permettre aux apprenants de savoir créer un algorithme avec ou sans ordinateur et utiliser un langage de programmation pour l'expérimenter (Tchounikine, 2017). Cette approche est également privilégiée en Belgique francophone, comme le souligne le nouveau référentiel FMTTN : « En abordant des notions de logigramme et de programmation, au travers d'exercices simples de lecture et d'écriture, l'élève développe sa pensée informatique et algorithmique » (REF FMTTN, p.25).

L'apprentissage de l'algorithmique et de la programmation peut entrainer plusieurs difficultés pour les apprenants (Fessard et al., 2019). Pour surmonter ces défis, différents choix pédagogiques sont possibles. Classiquement, la littérature oppose les activités débranchées et les activités branchées de programmation (Frison et al., 2018).

Les activités débranchées consistent notamment à développer des algorithmes sans utiliser d'ordinateurs, permettant aux apprenants de comprendre les concepts fondamentaux liés à cet apprentissage (Drot-Delange, 2013). Celles-ci constituent une première étape avant de passer à la programmation effective (Frison et al., 2019). Les activités branchées consistent, quant à elles, à mettre en œuvre ces algorithmes, soit de concevoir les programmes opérationnels en s'appuyant sur un environnement informatique ou un outil informatique (Romero et al., 2018), tels que des logiciels de programmation ou des objets tangibles programmables (Sigayret et al., 2021).

La littérature consultée ne converge pas nécessairement sur l'efficacité d'un dispositif par rapport à un autre (Romero et al., 2018 ; Fessard et al., 2018 ; Sigayret et al., 2021). Au contraire, ceux-ci ne sont pas nécessairement en opposition et peuvent cohabiter (Tchounikine, 2017). Ils semblent donc complémentaires selon le moment de l'apprentissage ou les objectifs pédagogiques visés. Les activités débranchées réduisent la charge cognitive liée à l'utilisation d'une machine, permettant aux apprenants une meilleure attention et une meilleure interaction avec le groupe (Romero et al., 2018). Néanmoins, les logiciels de programmation offrent une expérience interactive lors de laquelle les apprenants peuvent expérimenter différentes approches de résolution et recevoir un feedback immédiat, favorisant ainsi l'apprentissage par l'essai-erreur (Sigayret et al., 2021). De même, les robots pédagogiques tangibles permettent de combiner la programmation avec une composante physique, ce qui peut renforcer la compréhension pratique des concepts abstraits de programmation (Sullivan et Heffernan, 2016).

Dillenbourg (2018) souligne qu'il existe désormais une variété d'environnements numériques d'apprentissage : des plateformes logicielles et des robots éducatifs pour l'apprentissage de la programmation et de la PI. Cette diversité de dispositifs permet aux enseignants d'adopter des approches variées en fonction des besoins spécifiques des apprenants et des objectifs pédagogiques qu'ils visent.

Ainsi, c'est l'articulation de diverses méthodes qui semble enrichir l'apprentissage de la PI en offrant aux apprenants des opportunités variées de développer et d'appliquer leurs compétences en programmation et en résolution de problèmes. Pour objectiver ces apprentissages, il est nécessaire d'évaluer systématiquement le développement de ces compétences.

2.3 Évaluer la pensée informatique

Bien que la manière de mesurer le niveau de développement de la PI ne fasse pas consensus dans la littérature scientifique (Román-González et al., 2017), celle-ci fait l'objet de nombreuses initiatives que nous nous proposons de catégoriser au regard du modèle des 4P : performances, processus, perceptions et profils (Temperman, 2013).

Les performances des apprenants peuvent être évaluées à partir de questionnaires administrés en pré et post-test portant sur l'évaluation de la PI (Brackmann et al., 2017 ; Romero et al., 2018 ; Marfisi-Schottman et al., 2020). Les processus mis en œuvre dans l'apprentissage peuvent être analysés via un modèle d'observation des étapes de résolution de problèmes (Chevalier et al., 2020) ou encore sur base d'une grille d'observation qui permet d'évaluer la PI à travers la réalisation d'activités par les apprenants (Bellegarde & Boyaval, 2020 ; Chaker et Njingang Mbadjoin, 2020). Le lien entre les performances et les processus peut être questionné via une analyse croisée en considérant la progression des apprenants (via des questionnaires pré et post expérimentations) et les comportements en cours d'apprentissage comme les traces d'activité de programmation (Grover et al., 2014 ; Peter et al., 2019 ; Léonard et al., 2021). Enfin, La perception des compétences des apprenants est également évaluée, permettant de croiser leur ressenti sur leurs capacités avec leur progression effective.

L'éventail de ces approches montre qu'il est important de réfléchir à la complémentarité entre les différentes stratégies d'évaluation de la PI, tout en tenant compte des contraintes éventuelles de l'environnement d'apprentissage et du contexte dans lequel sont récoltées les données.

3. Méthodologie

La méthodologie de cette étude se compose de trois sections principales : le contexte de la recherche et la description de l'échantillon, la description du dispositif pédagogique mis en place, et enfin les questions de recherche accompagnées de la présentation de l'instrument utilisé.

3.1 Contexte de la recherche et description de l'échantillon

Les 122 participants à cette étude sont des étudiants inscrits en début de cycle du Master en sciences de l'éducation, avec une spécialisation en technologie de l'éducation, à l'Université de Mons (90 étudiantes et 32 étudiants). Le cours intitulé « Développement pédagogique de la pensée informatique », introduit durant l'année académique 2021-2022, vise deux objectifs complémentaires : d'une part, faire découvrir aux étudiants les concepts de la PI (tels que les séquences, les itérations, les alternatives, etc.), et d'autre part, leur permettre d'identifier, d'utiliser et d'évaluer différents outils pédagogiques pour développer cette pensée. L'objectif est donc de familiariser les étudiants avec diverses activités pédagogiques, leur permettant de mobiliser les concepts et outils pour les engager dans une réflexion pédagogique et critique sur l'intégration de ces concepts et outils dans leur propre pratique éducative.

A l'instar des travaux de Tchounikine (2017), il est important de souligner que former des professionnels de l'éducation à l'enseignement de la PI n'implique pas nécessairement de les former à l'informatique. Il s'agit plutôt de leur fournir des éléments permettant de comprendre ce qu'est l'enseignement de cette forme de pensée et comment l'aborder pédagogiquement. Néanmoins, tout enseignement nécessite une maîtrise de l'objet d'enseignement (Tchounikine, 2017), ce qui renforce la pertinence du cours proposé et justifie son intégration dans le cursus des futurs enseignants et formateurs d'enseignants.

3.2 Présentation du dispositif technopédagogique

Dans ce dispositif de formation, nous nous appuyons sur le postulat de l'isomorphisme pédagogique tel que défini par Develay (1994, p.23) : « On utilise souvent le terme d'isomorphisme ou d'homomorphisme pour désigner le postulat selon lequel c'est en faisant vivre et analyser aux formés des situations proches, au niveau des attitudes, des méthodes voire des contenus, à celles que ceux-ci auront à faire vivre à leurs élèves que le formateur aide durablement ses formés à intégrer l'ensemble des procédures cognitives et affectives mises en jeu ». Cela soutient l'idée qu'il s'agit « d'appliquer aux formés les méthodes et techniques que l'on souhaite leur voir appliquer auprès des élèves » (Develay, 1994, p.23).

Pour mettre en pratique ce postulat et atteindre les deux objectifs visés – la familiarisation et l'engagement critique – le cours s'articule autour de séances synchrones et asynchrones. Chaque session offre, dans un premier temps, l'occasion de découvrir de manière active différents outils branchés et débranchés, puis, dans un second temps, de prendre du recul sur les démarches cognitives mises en œuvre.

Le dispositif pédagogique, réparti sur trois mois, inclut six sessions synchrones de deux heures chacune, dont les contenus sont détaillés dans le Tableau 1. Ces sessions sont complétées par des tâches asynchrones d'une durée équivalente. Le dispositif combine des apports théoriques, des ateliers pratiques et des échanges réflexifs. Pour les ateliers, des sous-groupes de 3 à 4 étudiants sont constitués afin de faciliter la manipulation directe des outils. La présence de 2 à 3 encadrants pédagogiques est nécessaire pour accompagner l'ensemble du groupe.

Ce dispositif peut être qualifié d'hybride (Charlier et al., 2006), car il alterne trois séances à distance via Teams, une application de communication collaborative et trois séances en présentiel, permettant ainsi l'utilisation de matériel tangible. Chaque séance synchrone est suivie de tâches d'application et de découverte des logiciels, à réaliser de manière asynchrone et autonome. Pour guider leur réflexion pédagogique, les étudiants disposent d'un syllabus détaillé leur permettant d'approfondir les concepts abordés.

Tableau 1. contenus et modalités des 6 séances du dispositif pédagogique

Séance(s)	Modalités	Contenu
1	Distanciel	• Présentation des concepts théoriques fondamentaux de la PI ; • Discussion sur les origines et le développement historique du concept ; • Analyse réflexive du contexte actuel et des initiatives en Belgique francophone ; • Découverte du logiciel Algoblocs (Piva, C. ; Villeubanne, France).
2	Distanciel	• Introduction aux environnements de programmation par blocs. Par exemple : Scratch (Resnick, M. ; Cambridge, Etat-Unis), Minecraft Education (Mojang Studios, avec le soutien de Microsoft ; Washington, Etat-Unis) ; • Découverte des principes informatiques et pédagogiques sous-jacents ; • Analyse réflexive autour des études empiriques en sciences de l'éducation associées à la programmation par blocs.
3	Distanciel	• Introduction aux principes de robotiques pédagogiques avec l'environnement MakeCode (Microsoft ; Washington, États-Unis) et la carte Micro-bit (British Broadcasting Corporation ; Londres, Royaume-Uni) ; • Découverte des principes informatiques et pédagogiques sous-jacents ;
4	Présentiel	• Présentation de Lego Spike (LEGO Education ; Billund, Danemark) • et de leur fonctionnement ; • Exploration des principes informatiques appliqués à la robotique ; • Analyse des principes informatiques, pédagogiques et des études empiriques en robotique éducative.
5	Présentiel	• Introduction aux activités débranchées et à leur importance ; • Stratégies pour passer des activités débranchées aux activités branchées ; • Analyse des principes informatiques, pédagogiques et des études empiriques sous-jacents.
6	Présentiel	• Introduction aux outils d'évaluation de la PI ; • Méthodologies d'évaluation basées sur des données de performance, de processus et de perception.

3.3 Collecte des données

Pour évaluer le cours, nous nous appuyons sur les travaux réalisés par les étudiants dans le cadre des tâches asynchrones intermédiaires, ainsi qu'un examen final. Cette évaluation vise à vérifier l'atteinte des objectifs du cours, en particulier la compréhension des concepts associés à la PI, leur intégration dans les pratiques pédagogiques, ainsi que la capacité des étudiants à aborder ces concepts avec des apprenants.

Avant la première séance et après la dernière séance, nous administrons un test de mesure de la PI via un Google Forms. L'adaptation française du *Computational Thinking Test* ou *CTT*, validé par Román-González et al. (2017), réalisée dans le cadre de l'étude CoCreaTIC (Romero et al., 2018) est utilisée. Notre choix s'est porté sur cet outil, car il a fait l'objet de validation (Zapata-Caceres et al., 2020 ; El-Hamamsy et al., 2022 ;

Lockwood et Mooney, 2018) et d'adaptations linguistiques en le traduisant (Çetin et al., 2020 ; Zhang et Wong, 2023 ; Romero et al., 2018). De plus, il permet l'administration rapide d'un pré-test et d'un post-test, offrant ainsi une évaluation de l'évolution des apprenants à deux moments distincts à partir d'un calcul de gains relatifs. Selon D'Hainaut (1975), le gain relatif permet d'évaluer l'efficacité d'une action pédagogique où chacun des participants possède un niveau initial qui lui est propre et dont la mesure s'obtient en calculant le rapport de ce que l'apprenant a gagné à ce qu'il aurait pu gagner au maximum. Selon ce même auteur, en sciences humaines, il y a un apprentissage effectif, et donc un effet positif de l'apprentissage, lorsque le seuil des 30 % est franchi.

D'un point de vue édumétrique, le CTT se compose de 28 items répartis de manière égale entre les 7 concepts du questionnaire (4 items par concept) : directions et séquences de base, boucle « for », boucle « while », condition « if » simple, condition « if/else » complexe, condition « while », fonction tels que définis par Román-González (2015). Ainsi, « Tous les items qui composent le test impliquent, dans une mesure plus ou moins grande, les processus cognitifs des quatre piliers de la PI : décomposition, reconnaissance des formes, abstraction et algorithmes » (Brackmann et al., 2017, p.66).

En outre, un item, évaluant le sentiment de maîtrise des étudiants, les invite à évaluer leur propre niveau de réussite au test : « Sur une échelle de 0 à 10, estimez votre niveau de réussite à ce test ». On peut associer ce sentiment de maîtrise au principe de degré de certitude qui constitue un marqueur de l'autorégulation (Focant, 2007). Cette perception évalue la capacité d'un individu à juger de manière précise la justesse de ses propres réponses (Focant, 2007 ; Dumont, 2021). En proposant cet item, on obtient une valeur numérique qui indique la probabilité qu'une réponse soit considérée comme correcte par le répondant (Leclercq et Plunus, 1996, cités par Dumont, 2021, p. 84).

Un individu est considéré comme un bon autorégulateur s'il manifeste une grande confiance dans ses réponses lorsqu'elles sont correctes et reconnaît leurs erreurs lorsque celles-ci sont avérées (Focant, 2007). Pour éviter les ambiguïtés dans l'évaluation de la certitude, Leclercq (1986, cité par Dumont, 2021, p. 88) préconise de ne pas utiliser de termes imprécis tels que « peu sûr », « moyennement sûr » ou « très sûr », car ces expressions peuvent être interprétées de manière variable selon les individus. Dans le domaine des sciences de l'éducation, cet indicateur est souvent employé pour garantir la validité des interprétations des résultats et sert également de facteur discriminant dans les évaluations objectives (Bernier, 1971 ; Magis et al. 2011).

3.4 Questions de recherche et traitement des données

A partir de ces différentes mesures, nous pouvons formuler les trois questions de recherche suivantes :

> **Question n°1** : *Le dispositif pédagogique permet-il une progression de la performance au Computational Thinking Test ?*
>
> **Question n°2** : *Le dispositif pédagogique modifie-t-il le degré de certitude des étudiants au Computational Thinking Test ?*
>
> **Question n°3** : *Quels sont les liens entre les scores obtenus aux Computational Thinking Tests et le degré de certitude ?*

Sur le plan statistique, l'ensemble des données a été traité avec le logiciel SPSS version 29.

Les questions 1 et 2 sont traitées à travers des analyses descriptives et inférentielles, en tenant compte de la normalité des données. Ces analyses visent à évaluer la progression des scores au CTT (Q1) ainsi que l'évolution de la perception du degré de certitude (Q2).

Pour approfondir notre analyse, nous avons réalisé une analyse de régression linéaire multiple (Q3) en intégrant, comme variables d'entrées indépendantes, les différents scores obtenus pour les 7 concepts du CTT et comme variable de sortie à modéliser/estimer le degré de certitude de la réussite au CTT. L'ambition est d'identifier parmi les scores aux différents concepts au CTT, ceux qui peuvent caractériser le degré de certitude.

Au niveau statistique, nous avons choisi d'appliquer la méthode de sélection par retrait progressif (méthode descendante) afin que le logiciel d'analyse statistique mette en évidence le modèle qui offre le degré de prédiction le plus élevé, en éliminant successivement les variables les moins pertinentes et non

significatives (par ex. sélections parmi les modèles candidats). Pour utiliser de manière appropriée cette méthode, il faut à la fois veiller à obtenir un nombre restreint de prédicteurs pertinents et significatifs tout en s'intéressant au degré de prédiction fourni par la valeur du R2 ajusté.

Enfin, nous avons considéré l'indice de tolérance qui nous donne la possibilité d'identifier un éventuel problème de colinéarité dans le modèle proposé. En nous référant à Stafford & Bodson (2006), il est généralement admis qu'une valeur de tolérance inférieure à .20 risque d'entraîner des problèmes d'estimation des coefficients associés aux variables affectées de colinéarité. De manière diachronique, nous avons réalisé cette analyse d'abord pour les données liées au pré-test, ensuite pour celles liées au post-test.

4. Résultats et discussion

Notre analyse des résultats s'articule autour de nos 3 questions de recherche.

4.1 Réponse à la question 1 : Le dispositif pédagogique permet-il une progression de la performance au Computational Thinking Test ?

Le tableau 2 montre les résultats descriptifs des 122 participants, mesurés avant (pré-test) et après (post-test) notre dispositif pédagogique. Il inclut également la moyenne des gains relatifs pour chaque concept évalué par le *CTT* (Román-González, 2015).

Nos résultats suggèrent que les apprenants ont montré des tendances positives dans l'amélioration des performances évaluées entre le pré-test et le post-test. L'analyse du tableau 3 indique une progression générale des apprenants, soulignant l'efficacité de l'intervention pédagogique pour renforcer les compétences en programmation. Quel que soit le concept considéré, le niveau au post-test est systématiquement supérieur au niveau initial. Les apprenants semblent avoir le plus progressé pour le concept conditions « if » simples avec une moyenne allant de 2,81 (70,30%) au pré-test à 3,11 (77,80%) au post-test. La moyenne des gains relatifs pour ce concept s'élève à 27,60%. Nous notons également une progression notable dans la maîtrise des boucles « for » dont le score moyen passe de 3,46 (86,50%) au pré-test à 3,70 (92,50%) au post-test, avec une moyenne des gains relatifs de 26,90%, indiquant que le dispositif pédagogique a bien renforcé ce concept. Par ailleurs, le score moyen des structures conditionnelles « if/else » complexes augmente de 2,80 (70,00%) à 3,17 (79,30%), soit une moyenne des gains relatifs de 24,30%. À l'inverse, c'est au niveau du concept conditions « while » que les apprenants semblent avoir le moins progressé avec une moyenne des gains relatifs s'élevant à 14,10%. Cette faible progression peut s'expliquer par la maitrise difficile d'une structure de contrôle qui fait appel à la logique booléenne et qui permet d'exécuter une série d'instructions de manière répétée tant qu'une condition est vraie. L'articulation de ces compétences complémentaires demande probablement un enseignement plus spécifique.

Enfin, nous observons de manière globale que la moyenne des gains relatifs, exprimée en pourcentage, s'élève à 34,50%, passant de 76,70% au pré-test à 83,90% au post-test. Selon D'Hainaut (1975), cette progression indique un effet positif de l'apprentissage. Le niveau de compétence élevé observé chez les participants dès le pré-test suggère une maîtrise préalable des concepts évalués par le CTT, probablement acquise au cours de leur parcours antérieur. Nous ajoutons que le taux d'hétérogénéité diminue après notre dispositif pédagogique, passant de 20,41% au pré-test à 16,07% au post-test. Ceci indique qu'après le dispositif pédagogique, le niveau d'hétérogénéité est plutôt faible, se rapprochant du seuil de 15% (D'Hainaut, 1975 ; Ouellet, 1985 cité par Gerard, 2003), indiquant une disparité plus faible entre les aptitudes des sujets relatives aux concepts mesurés par le CTT à l'issue de l'apprentissage.

Tableau 2. analyses descriptives et inférentielles : Progressions au CTT entre le pré-test et le pst-test (N = 122)

	Pré-test		Post-test		Moyenne des gains relatifs (%)	p-value
	χ (%)	σ	χ (%)	σ		
Directions et séquences de base	3,68 (92,00%)	0,59	3,81 (95,00%)	0,43	14,50	**0,014**
Boucles « for »	3,46 (86,50%)	0,67	3,70 (92,50%)	0,54	26,90	**<0,001**
Boucles « while »	3,30 (82,50%)	0,75	3,47 (86,80%)	0,66	22,00	**0,043**
Conditions « if » simples	2,81 (70,30%)	0,92	3,11 (77,80%)	0,99	27,60	**0,002**
Conditions « if/else » complexes	2,80 (70,00%)	1,23	3,17 (79,30%)	1,04	24,30	**0,002**
Conditions « while »	2,52 (63,00%)	1,21	2,78 (69,50%)	1,13	14,10	**0,03**
Fonctions	3,09 (77,30%)	1,04	3,45 (86,3%)	0,93	23,60	**<0,001**
Score total	21,48 (76,70%)	4,38	23,49 (83,90%)	3,78	**34,50**	**<0,001**
Taux d'hétérogénéité	20,41%		16,07%			

Les hypothèses nulles associées à cette première question de recherche sont similaires pour chaque concept évalué par le CTT, ainsi que pour le score global : « *il n'y a pas de différences significatives entre les scores au pré-test et les scores au post-test* ». En raison de la non-normalité des données[1], pour chacun des 7 concepts mesurés par le CTT ainsi qu'au score global, nous avons privilégié un test non paramétrique afin de vérifier l'H0.

Les résultats du test de Wilcoxon (Tableau 2) révèle des différences significatives[2] pour chaque concept considéré dans le CTT : séquences de base ($p = 0,014$), boucles « for » ($p < 0,001$), boucles « while » ($p = 0,043$), conditions « if » simples ($p = 0,002$), conditions « if/else » complexes ($p = 0,002$), conditionnelles ($p = 0,03$), fonctions ($p < 0,001$). Quel que soit le concept, les étudiants ont significativement progressé entre le pré-test et le post-test. Les étudiants ont globalement un score au CTT significativement plus élevé après avoir participé au dispositif technopédagogique ($p < 0,001$).

4.1 Réponse à la question 2 : Le dispositif pédagogique modifie-t-il le degré de certitude des étudiants au Computational Thinking Test ?

Le degré de certitude au CTT a augmenté entre le pré-test et le post-test. Les apprenants (N = 122) passent d'un score moyen de 6,61/10 (soit 66,10% ; -) à 7,70/10 (soit 77,00% ; -). Cette perception est par ailleurs plus homogène au terme de l'apprentissage (coefficient de variation [CV] = 26,17% au prétest ; contre CV = 17,66 % au post-test). Ce résultat tend à montrer une cohérence entre la progression effective et la progression perçue.

L'H0 associée à cette deuxième question de recherche est : « il n'y a pas de différence significative entre le degré de certitude au pré-test et le degré de certitude au post-test ». En raison de la non-normalité des données[1], un test non paramétrique est privilégié afin de vérifier l'H0. Les résultats du test de Wilcoxon indiquent que les étudiants ont un degré de certitude significativement[2] plus élevé après avoir participé au dispositif technopédagogique ($p < 0,001$). Ils expriment une perception plus positive concernant leur capacité de réussite au test.

[1] La non-normalité a été confirmée pour l'ensemble des variables par le test de Kolomogorov-Smirnov ($p < 0.001$)

[2] Intervalle de confiance = 95%

4.3 Réponse à la question 3 : Quels sont les liens entre les scores obtenus aux Computational Thinking Tests et le degré de certitude ?

Nous mettons en évidence, l'existence d'un lien positif significatif entre la performance mesurée par le score total au CTT des apprenants et leur degré de certitude quant à leur réussite à la fin du test, aussi bien au pré-test ($r = 0,602$; $p < 0,001$) qu'au post-test ($r = 0,723$; $p < 0,001$). Ainsi, l'augmentation de la corrélation au post-test indique que cette relation se renforce entre le début et la fin de la formation. Ces observations suggèrent que la formation a un impact positif sur le plan métacognitif, en alignant davantage la perception des étudiants avec l'évaluation objective de leur réussite au test.

4.3.1 Liens entre les scores au pré-test et le degré de certitude au pré-test

Le tableau 3 présente le modèle issu de l'analyse de régression multiple. La variable prédite est le degré de certitude. Dans ce modèle, nous constatons que la puissance estimée à partir du R^2 est de 0,429.

En ce qui concerne ces variables explicatives, le modèle indique que cinq prédicteurs sont significatifs et un prédicteur à la limite de la significativité. Ils permettent d'expliquer 42,9% du degré de certitude au prétest ($R^2 = 0,429$; $R = 0,655$; $p < 0,001$). Nous pouvons mettre en évidence qu'un score élevé au niveau des séquences de base (**β** = 0,253 ; p=0,005), des boucles « for » (**β** = 0,151 ; p= 0,078), des conditions « if/else » complexes (**β** = 0,214 ; p= 0,014), des boucles « while » (**β** =0,168 ; p= 0,044) et des fonctions (**β** = 0,226 ; p=0,010) ainsi qu'un score faible au niveau des conditions « if » simples (**β** = -0,170 ; p= 0,046), permet de prédire le degré de certitude au pré-test.

Tableau 3. analyses de régression multiple : Modèle explicatif du niveau de réussite perçu au prétest

Prédicteurs		*Bêta*	*Taux de signification*	*Tolérance*
1	Directions et séquences de base	0,253	0,005	0,628
2	Boucles « for »	0,151	0,078	0,692
3	Conditions « if » simples	-0,17	0,046	0,695
4	Conditions « if/else » complexes	0,214	0,014	0,679
5	Conditions « while »	0,168	0,044	0,729
6	Fonctions	0,226	0,010	0,67

Parmi les différentes variables, l'examen des coefficients Bêta montre que ce sont les scores aux séquences de base, aux conditions « if/else » complexes et aux fonctions qui ont le plus de poids dans l'explication du degré de certitude. Le coefficient Bêta négatif observé pour les conditions « if » simples soulève une question. Pour expliquer ce résultat, nous pouvons avancer l'idée que les étudiants, au début de leur apprentissage, ne disposent pas d'une perception précise de leur propre niveau de réussite.

4.3.2 Liens entre les scores au post-test et le degré de certitude au post-test

Le tableau 4 présente la même analyse au terme de l'apprentissage. La lecture de celui-ci montre que quatre variables significatives permettent d'expliquer 55.1% du degré de réussite au posttest ($R^2 = 0,551$; $R=0,742$; $p <0,001$). Les scores élevés aux items boucle « for » ($\beta = 0,142$; $p= 0,001$), conditions « if » simples ($\beta = 0,196$; $p= 0,005$), conditions « if/else » complexes ($\beta = 0,256$; $p= 0,001$) et boucles « while » ($\beta = 0,331$; $p= 0,001$) expliquent la perception de réussite au posttest. D'un point de vue didactique, on observe que les items les plus complexes c'est-à-dire conditions « if/else » complexes et boucles « while » ont le plus de poids dans l'explication. Sur le plan métacognitif, ce résultat laisse à penser que les étudiants ont développé un meilleur jugement du niveau de difficulté du test et sont en mesure de mieux estimer, au terme de la formation, leur performance réelle.

Tableau 4. analyses de régression multiple : Modèle explicatif du niveau de réussite perçu au post-test

Prédicteurs		*Bêta*	*Taux de signification*	*Tolérance*
1	Boucles « while »	0,142	0,001	0,817
2	Conditions « if » simples	0,196	0,005	0,803
3	Conditions « if/else » complexes	0,256	0,001	0,78
4	Conditions « while »	0,331	0,001	0,656

A ce stade, nous considérons que certaines variables d'entrée deviennent prépondérantes dans la prédiction du degré de certitude.

4.3.3 Liens entre les scores au pré-test, le degré de certitude au pré-test, les scores au post-test et le degré de certitude au post-test

Pour aller plus loin dans notre réflexion, nous nous questionnons sur les variables ayant le plus d'influence dans la prédiction du degré de certitude au post-test. Notamment, au regard des performances initiales élevées, nous nous demandons si le degré de certitude est lié au parcours antérieur des étudiants, déterminé au regard du score au CTT et du degré de certitude au pré-test, ou au dispositif, déterminé par le score au post-test.

Le tableau 5 indique que cinq variables significatives permettent d'expliquer 70,8% du degré de certitude au post-test (R^2= 0,708 ; R=0,842 ; p <0,001). On peut observer que la variable qui a le plus de poids dans l'explication est le niveau de certitude par rapport à leurs compétences en PI au départ (β = 0,482 ; p = 0,001). L'examen du tableau montre que cette perception initiale interagit positivement avec la maîtrise de 4 dimensions du CTT dont une relative aux boucles (while) et trois relatives aux conditions (« if » simples, « if/else » complexes, while) pour expliquer leur compétence perçue au terme de l'apprentissage. Sur la base du tableau 5, on peut mettre en avant qu'il s'agit de quatre concepts plus complexes à mettre en œuvre d'un point de vue cognitif et où la maîtrise est la plus hétérogène chez les apprenants.

Tableau 5. analyses de régression multiple : Modèle explicatif du niveau de réussite perçu au post-test (2)

Prédicteurs		*Bêta*	*Taux de signification*	*Tolérance*
1	Boucles « while »	0,184	0,001	0,799
2	Conditions « if » simples	0,218	<0,001	0,799
3	Conditions « if/else » complexes	0,155	0,009	0,751
4	Conditions « while »	0,15	0,023	0,592
5	Degré de certitude au pré-test	0,482	<0,001	0,707

5. Perspective

Ces premiers résultats apportent des réponses aux défis posés par notre dispositif technopédagogique qui ouvrent la voie à futures recherches.

Au niveau de la conception du dispositif, nous avons intégré dans notre scénario pédagogique des activités de découverte et d'apprentissage des concepts clés favorisant le développement de la PI (Tchounikine, 2017). Malgré l'absence de groupe témoin, les résultats de cette expérimentation révèlent une progression significative pour tous les concepts mesurés par le CTT. Bien qu'un niveau global initial assez élevé soit constaté, nous observons une moyenne des gains relatifs de 34,5%, indiquant un effet positif de l'apprentissage (D'Hainaut, 1975).

L'examen des niveaux de maîtrise initiaux montre une bonne maîtrise dans la majorité des dimensions du CTT, avec des scores plus faibles pour les dimensions relatives aux conditions « if » simples, aux conditions « if/else » complexes et aux conditions « while ». Si nous observons une progression positive entre le niveau initial et le niveau final pour ces 3 dimensions, les niveaux de maitrise indiquent cependant que l'acquisition de ces concepts se révèle plus complexe pour les étudiants. Il est donc nécessaire de porter une attention particulière à ces concepts dans la régulation de notre scénario pédagogique, en intégrant des activités spécifiques visant à leur développement.

L'analyse de la perception de la réussite montre également une progression significative entre le niveau initial et le niveau final. Ces résultats, a priori cohérents avec la progression des performances, indiquent que les étudiants semblent plus sûrs d'eux dans les réponses qu'ils fournissent. Sur la base de notre analyse corrélationnelle, nous observons une plus grande cohérence entre ce qu'ils expriment via leur degré de certitude et ce qu'ils réalisent effectivement. Les étudiants ont ainsi une meilleure conscience du niveau de difficulté du test et sont en mesure de mieux estimer, au terme de la formation, leur performance réelle.

Au niveau du contenu de la formation, nous avons favorisé la découverte complémentaire de trois types de dispositifs tels que préconisés par Sigayret et al. (2021). Ces dispositifs incluent : (1) les activités débranchées, (2) les logiciels de programmation, et (3) les objets tangibles programmables tels que les robots pédagogiques. D'un point de vue organisationnel, nous avons combiné deux modalités : des séances en distanciel et des séances en présentiel. Ce choix semble faciliter une meilleure assimilation des concepts par les étudiants. Cette hypothèse devra être vérifiée en proposant, par exemple, des dispositifs avec des modalités pédagogiques différentes afin de comparer les effets des diverses approches pédagogiques utilisées dans ce dispositif. Au niveau de l'évaluation, nous nous sommes basés sur le CTT, un outil validé qui offre l'avantage d'une passation rapide et permettant d'objectiver le score en PI à des moments clés du dispositif. Nous pensons que l'intégration d'un item relatif à la perception de la compétence (degré de certitude) offre une piste didactique intéressante pour mieux comprendre la progression des étudiants sur le plan métacognitif. Nos résultats corroborent les observations de Fessard et al. (2019) qui mettent en évidence les difficultés de compréhension de la structuration d'un algorithme chez les apprenants. Nous observons également cette tendance avec des apprenants adultes dans le contexte d'une formation en technologie de l'éducation.

6. Conclusion

Cette étude a permis d'évaluer l'effet d'un dispositif technopédagogique sur le développement de la PI chez les étudiants du Master en Sciences de l'Éducation, spécialité Technologie de l'Éducation. Les résultats montrent une progression significative des performances au CTT, ainsi qu'une amélioration significative de leur perception de réussite. Ces résultats nous donnent la possibilité de répondre positivement à nos questions de recherche, objectivant que le dispositif pédagogique a non seulement renforcé les performances des étudiants pour l'ensemble des concepts mesurés par le CTT, mais a aussi influencé leur sentiment de maitrise objectivé par le degré de certitude.

Les activités proposées, tant en présentiel qu'à distance, ont favorisé un apprentissage des concepts de PI. L'amélioration observée dans les scores du CTT semble confirmer l'efficacité de notre approche hybride, combinant théorie, pratique et analyse réflexive qui passe par l'intégration d'activités variées et structurées pour optimiser l'apprentissage des concepts du Computational Thinking test.

Bien que contextualisée, cette recherche exploratoire présente certaines limites. Elle ne mesure pas spécifiquement l'effet de chaque modalité d'apprentissage proposée dans le scénario pédagogique et le plan expérimental ne comporte pas de groupe témoin. Néanmoins, nous estimons qu'elle peut être utile pour les formateurs d'enseignants en termes d'articulation des activités et de méthodologie pour évaluer les progrès des apprenants engagés dans leur formation. L'exploitation des traces d'apprentissage en cours de processus nous semble être une piste pertinente pour mieux comprendre d'un point de vue qualitatif le développement de la PI des apprenants.

7. Bibliographie

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal, 55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

Amarelle, C. (2018). Message de la conseillère d'Etat. Dans G. Parriaux, J.-P. Pellet, G.-L. Baron, E. Bruillard & V. Komis (Eds.), De *0 à 1 ou l'heure de l'informatique à l'école* (p. 17-19). Peterlang.

Bellegarde, K., et Boyaval, J. (2020 5-7 février). *Initier des jeunes élèves à la robotique/informatique : gestes professionnels et agir enseignant* [Communication]. DIDAPRO 8 - L'informatique, objets d'enseignements – enjeux épistémologiques, didactiques et de formation, Villeneuve d'Ascq, France. https://www.didapro.org/8/wp-content/uploads/sites/4/2020/02/Bellegarde_23.pdf

Ben Henda, M. (2016, 26-28 octobre). *L'enseignement du code informatique à l'école : Prémices d'un humanisme numérique congénital* [communication]. 5e rencontre annuelle d'ORBICOM, Paris, France. https://hal.science/hal-01516577/document

Bernier, J. J. (1971). *Le degré de certitude et le degré d'information comme facteurs de discrimination dans un test objectif* [thèse de doctorat, Université de Laval]. Érudit. https://www.erudit.org/fr/theses/laval/1971/

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., et Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (WiPSCE '17) *(p. 65–72).* https://doi.org/10.1145/ 3137065.3137069

Bugmann, J., et Karsenti, T. (2018). Apprendre à programmer un robot humanoïde : impacts sur des élèves de l'adaptation scolaire. *Formation et profession : revue scientifique internationale en éducation, 26*(1), 26-42. https://doi.org/10.18162/fp.2018.460

Çetin, İ., Otu, T., et Oktaç, A. (2020). Adaption of the computational thinking test into Turkish. *Turkish Journal of Computer and Mathematics Education, 11*(2), 343-360. https://dergipark.org.tr/en/download/article-file/1079231

Charlier, B., Deschryver, N., et Peraya, D. (2006). Apprendre en présence et à distance : une définition des dispositifs hybrides. *Distances et Savoirs, 4*(4), 469-496. https://tecfa.unige.ch/tecfa/teaching/bachelor_74111/Cours_2010_2011/semestre1/cours11/charlier_deschryver_peraya_2006.pdf

Chaker, R., et Mbadjoin, T. N. (2020). Robots scolaires et configuration spatiale de la classe : Effets sur les interactions sociales dans le cadre de séquences pédagogiques au CE2. *Didapro 8 – DidaS-TIC : L'informatique, objets d'enseignements – enjeux épistémologiques, didactiques et de formation* (p. 66-78), Lille, France. https://hal.science/hal-03585614/document

Chevalier, M., Giang, C., Piatti, A., et Mondada, F. (2020). Fostering computational thinking through educational robotics: A model for creative computational problem solving. *International Journal of STEM Education*, *7*(39). https://doi.org/10.1186/s40594-020-00238-z

D'Hainaut, L. (1975). *Des fins aux objectifs de l'éducation*. Labor Nathan.

Develay, M. (1994). Peut-on former les enseignants ? *Recherche & Formation, 16*, 148-151.

Dillenbourg, P. (2018). Pensée computationnelle : Pour un néopapertisme durable car sceptique. Dans G. Parriaux, J.-P. Pellet, G.-L. Baron, E. Bruillard et V. Komis (dir..), *De 0 à 1 ou l'heure de l'informatique à l'école* (pp. 17-19). Peter Lang.

Drot-Delange, B. (2013). Enseigner l'informatique débranchée : analyse didactique d'activités. *Actualité de la Recherche en Éducation et en Formation* (p. 1-13), Montpellier, France. https://archivesic.ccsd.cnrs.fr/sic_00955208v1/document

Drot-Delange, B., Pellet, J.-P., Delmas-Rigoutsos, Y., et Bruillard, E. (2019). Pensée informatique : Points de vue contrastés. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation, 26*(1), 39-61. https://doi.org/10.23709/sticef.26.1.1

Dumont, M. (2021). *Pour un enseignement efficient de l'orthographe en formation initiale des enseignants. Analyse des effets de l'intégration de démarches métacognitives sur l'apprentissage et les perceptions des étudiants* [thèse de doctorat]. Université de Mons. HAL. https://hal.science/tel-03227928

El-Hamamsy, L., Zapata-Cáceres, M., Barroso, E. M., Mondada, F., Zufferey, J. D., et Bruno, B. (2022). The Competent Computational Thinking Test: Development and Validation of an Unplugged Computational Thinking Test for Upper Primary School. *Journal Of Educational Computing Research*, *60*(7), 1818-1866. https://doi.org/ 10.1177/07356331221081753

Fédération Wallonie-Bruxelles. (2022). *Formation manuelle, technique, technologique et numérique*. Fédération Wallonie-Bruxelles. http://www.enseignement.be/index. php?page=23827&do_id=17242&do_check=CNEJLFQGEC

Fessard, G., Wang, P., et Renna, I. (2019, 04-07 juin). Objet tangible ou simulation numérique : Deux simulations équivalentes pour l'apprentissage de la programmation ? Dans J. Broisin, E. Sanchez, A. Yessad et F. Chenevotot (dir.), *Actes de la 9ème Conférence sur les Environnements Informatiques pour l'Apprentissage Humain : Données numériques et prises en compte de l'apprenant dans les environnements informatiques pour l'apprentissage humain* (p.. 223-234), Paris, France. https://eiah2019.sciencesconf.org/data/pages/ActesEIAH2019_V4.0.pdf

Focant, J. (2007). Chapitre 2. La mesure des processus d'autorégulation : quelles méthodes ? quels enjeux ? Dans N. Nader-Grosbois (dir.), *Régulation, autorégulation, dysrégulation : Pistes pour l'intervention et la recherche* (p. 31-42). Mardaga.

Frison, P., Daoud, M., et Adam, M. (2018). Transition didactique de l'activité débranchée à la programmation avec AlgoTouch. Dans G. Parriaux, J.-P. Pellet, G.-L. Baron, E. Bruillard et V. Komis (dir.), *De 0 à 1 ou l'heure de l'informatique à l'école* (p. 273-289). Peter Lang.

Lockwood, J., et Mooney, A. (2017). *Computational thinking in education: Where does it fit? A systematic literary review* [working paper]. arXiv. https://arxiv.org/abs/1703.07659

Moreno-León, J., Román-González, M., et Robles, G. (2018). *On computational thinking as a universal skill: A review of the latest research on this ability.* IEEE Global Engineering Education Conference (CN), 1684-1689. https://doi.org/10.1109/ EDUCON.2018.8363437

Papert, S. (1981). Computers and computer cultures. *Creative Computing, 7*(3), 82-92. http://dailypapert.com/wp-content/uploads/2014/04/Computers-and-Computer-Cultures-March-1981.pdf

Peter, Y., Léonard, M., et Secq, Y. (2019, 4-7 juin). *Reconnaissance de Motifs et Répétitions : Introduction à la Pensée Informatique* [communication]. Conférence Environnements informatiques pour l'apprentissage humain, Paris, France. https://hal.archives-ouvertes.fr/hal-02151035

Román-González, M. (2015, 6-8 juillet). *Computational thinking test: Design guidelines and content validation* [communication]. Proceedings of the 7th International Conference on Education and New Learning Technologies (EDULEARN15) Barcelone, Espagne. https://www.researchgate.net/publication/290391277_COMPUTATIONAL_THINKING_TEST_DESIGN_GUIDELINES_AND_CONTENT_VALIDATION

Román-González, M., Pérez-González, J. C., et Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in human behavior*, *72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Romero, M., et Vallerand, V. (2016). *Guide d'activités technocréatives pour les enfants du 21e siècle* (Vol. 1). https://lel.crires.ulaval.ca/works/guidev1.pdf

Romero, M., Lille, B., Viéville, T., Duflot-Kremer, M., de Smet, C., & Belhassein, D. (2018, 27-29 août). Analyse comparative d'une activité d'apprentissage de la programmation en mode branché et débranché [communication]. *Educode - Conférence internationale sur l'enseignement du numérique et par le numérique*, Bruxelles, Belgique. http://hal.inria.fr/hal-01861732

Shute, V. J., Sun, C., et Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22,* 142-158. https://doi.org/10.1016/j.edurev.2017. 09.003

Sigayret, K., Tricot, A., et Blanc, N. (2021, 7-9 juin). *Pensée informatique et activités de programmation : quels outils pour enseigner et évaluer ?* [communication]. Conférence Environnements informatiques pour l'apprentissage humain, Fribourg, Suisse. https://hal.science/hal-03241689

Stafford, J., et Bodson, P. (2006). *L'analyse multivariée avec SPSS*. Presses de l'Université du Québec.

Sullivan, F. R., et Heffernan, J. (2016). Robotic Construction Kits as Computational Manipulatives for Learning in the STEM Disciplines. *Journal of Research on Technology in Education, 48*(2), 105-128. https://doi.org/10.1080/15391523.2016.1146563

Tchounikine, P. (2017). *Initier les élèves à la pensée informatique et à la programmation avec Scratch* (Version 2). http://lig-membres.imag.fr/tchounikine/PenseeInformatique Ecole.html

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

Wing, J. M. (2010). Computational thinking: What and why? *The Link Magazine*. http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Zapata-Caceres, M., Martin-Barroso, E., et Roman-Gonzalez, M. (2020, 27-30 avril). *Computational Thinking Test for Beginners: Design and Content Validation.* EDUCON, Porto. https://ieeexplore.ieee.org/document/9125368

Zhang, S., et Wong, G. K. W. (2023). Development and validation of a computational thinking test for lower primary school students. *Educational Technology Research And Development*, *71*(4), 1595-1630. https://doi.org/10.1007/s11423-023-10231-2

L'AUTOCONFRONTATION CROISÉE ENTRE ÉTUDIANTS POUR FORMER AU DÉBOGAGE INFORMATIQUE

Olivier Neuhaus, Haute Ecole Fédérale en Formation Professionnelle, Suisse, olivier.neuhaus@hefp.swiss

Mario Ramalho, Ecole de Métiers de Fribourg, Suisse, mario.ramalho@edufr.ch

Résumé

Le débogage du code d'un autre développeur est une activité centrale sur le terrain des professionnels de l'informatique. Cependant il n'est que rarement abordé en formation ou alors souvent de manière artificielle (par exemple déboguer son propre code). Afin de former des apprentis informaticiens en école plein temps au débogage informatique, un dispositif basé sur des exercices de dépannage réalisés par les élèves et filmés a été mis en place dans un centre de formation en Suisse. En s'appuyant sur la trace vidéo aussi bien du flux à l'écran que de la webcam réalisée durant l'exercice, une autoconfrontation croisée entre pairs permet de faire ressortir non pas le résultat mais bien la démarche ayant mené les étudiants à la détection puis la correction rapide d'un bug informatique dans un programme amenant ainsi à un réel partage de pratique entre les étudiants.

Mots-clés : Formation professionnelle ; Autoconfrontation croisée ; Analyse vidéo ; Programmation informatique ; Débogage

1. Introduction

En Suisse, le cursus de formation professionnelle permet, dès la fin de la formation scolaire obligatoire, d'apprendre un métier au sein d'une entreprise, en alternance avec des cours dispensés en centres de formation. Cette voie offre à des jeunes, dès la fin de leur parcours scolaire la possibilité de débuter une formation dans une entreprise en dual (alternance entre un emploi en entreprise et une formation en école professionnelle) ou en plein temps à l'école (offrant aussi bien la dimension pratique que théorique en école) dans l'optique de l'obtention d'un CFC (Certificat Fédéral de Capacité) ou AFP (Attestation Fédérale de Formation Professionnelle) leur ouvrant directement les portes du monde de l'emploi.

Pour les étudiants informaticiens[1], l'apprentissage de la programmation représente un enjeu important de la formation. Si cet apprentissage connaît depuis plusieurs années de grandes évolutions grâce à divers outils permettant par exemple de rendre visible son aspect abstrait, des compétences comme celles liées au dépannage ou à la correction de bug sont néanmoins toujours complexes à acquérir (Siegmund et al., 2014). Ce sont pourtant bien des « compétences essentielles pour la maintenance et les modifications de programmes » (Rogalski, 2015) pour les informaticiens en formation et futurs professionnels qui consacreront entre 20 et 40% de leur temps d'activité sur cette tâche (Michaeli et Romeike, 2019). Or, cette tâche est difficile à réaliser. Keringhan et Plauger (1995) présentent l'activité de débogage comme deux fois plus difficile à réaliser que celle de développement. En formation, si les bases de l'algorithmique et de la programmation sont généralement apprises en institut, aussi bien pour la formation duale qu'en école à plein temps, la faculté à prendre en main le code d'un autre professionnel, à l'analyser pour en faire une maintenance ou à le dépanner en cas de problème, sont généralement des tâches apprises au contact du terrain en entreprise ou apprises « à la dure » (Michaeli et Romeike, 2019) sans avoir abordé en classe les stratégies adéquates.

Pour les étudiants en école plein-temps, sans contact direct avec le terrain cette compétence est d'autant plus difficile à acquérir. Une approche prometteuse consiste à mettre ces étudiants en situation de dépannage durant les cours et à déployer ensuite une didactique basée sur l'alternance et situations (Gremion, 2022), les poussant à analyser leur propre style et à le comparer à un genre du métier (Clot et Faïta, 2000). Cependant, le constat qui ressort fréquemment des enseignants qui empruntent cette approche en classe est que, si certains étudiants parviennent rapidement à détecter et corriger le code qui pose problème, d'autres n'y parviennent toujours pas même après une durée plus longue. Selon notre propre expérience, le fait de créer des espaces d'échanges entre les élèves qui y parviennent rapidement et les autres n'est, dans ce contexte, que peu porteur, la tentation étant forte pour les plus rapides de non pas partager l'analyse qu'ils ont réalisée mais d'uniquement donner la solution en indiquant la ou les lignes qu'il s'agit de corriger. Ces échanges laissent ainsi cachée toute la question de l'analyse du code ayant mené au débogage, élément pourtant central de cette problématique.

C'est la raison pour laquelle le dispositif présenté dans la suite de cet article a été imaginé. Expérimenté à l'Ecole des Métiers de Fribourg, nous en présenterons les principes et les premiers résultats pour ouvrir sur les perspectives que de telles approches peuvent offrir dans le champ de la formation en algorithmique.

2. Contexte

2.2 La formation des informaticiens en Suisse

La formation professionnelle suisse est fréquemment citée en exemple à l'échelle internationale (Kriesi et al., 2022). En 2022, la voie de la formation professionnelle a été choisie par près de 60% des jeunes en fin de leur scolarité obligatoire (SEFRI, 2022), la modalité duale (avec une formation en alternance entre un emploi en entreprise et des cours en école) étant majoritairement celle empruntée au détriment d'une formation en école à plein temps.

Pour chaque métier, les ordonnances de formation sont gérées par le Secrétariat d'Etat à la Formation, à la Recherche et à l'Innovation (SEFRI) qui s'appuie sur les Organisations nationales du monde du Travail (ORTRA) constituées de représentants des milieux économiques et professionnels. Pour la formation des informaticiens, « ICT-Formation professionnelle Suisse » est l'ORTRA en charge de proposer une ordonnance de formation en lien, tant avec le niveau des étudiants, qu'avec les besoins du terrain et de l'économie. Cette ordonnance est révisée à une fréquence de 4 ans, donnant lieu à de nouveaux contenus à aborder en formation.

[1] Afin d'alléger le style d'écriture, nous avons décidé d'utiliser une écriture exclusivement au masculin. Cependant lorsque le masculin est utilisé, il désigne autant les femmes que les hommes. Ainsi, par exemple, les termes informaticiens ou enseignants dans cet article désignent autant les femmes que les hommes même s'ils ne sont pas rédigés sous la forme enseignant·e·s, informaticien·e·s.

Suite aux deux dernières révisions de l'ordonnance fédérale (RO 2020 5411)[2], l'apprentissage de la programmation et de l'algorithmique se présente en plusieurs modules de formation de niveau progressif. Les contenus sont principalement axés sur l'apprentissage a) des concepts de base de la programmation (boucles, tests, …), b) du concept de programmation orientée objet et c) de la programmation web.

2.2 La compétence de débogage informatique pour les étudiants

Comme précisé précédemment, afin d'assurer la qualité du code développé en vue d'une intégration, le fait de tester son algorithme est une pratique importante pour un informaticien et fait intégralement partie du plan de formation. En formation, celle-ci est abordée majoritairement lors de l'étape de développement (l'étudiant développe un code qu'il doit lui-même tester pour s'assurer de son bon fonctionnement). Or, sur le terrain, le travail de l'informaticien consiste très fréquemment à reprendre le programme d'un autre développeur pour corriger d'éventuels problèmes, améliorer les performances ou y ajouter de nouvelles fonctionnalités (Michaeli et Romeike, 2019).

Pour les étudiants en formation de type dual, les compétences mobilisées pour réaliser un débogage sont souvent acquises dans des projets concrets au sein de l'entreprise formatrice, en étant amenés à réaliser ces actions sur les programmes gérés par l'entreprise, en binôme avec un formateur de terrain dans un premier temps puis de manière autonome si l'étudiant a acquis l'expérience nécessaire.

Elles sont cependant plus difficiles à traiter dans le cadre d'une formation en école à plein temps puisqu'il faut concevoir les codes sources d'erreur à corriger qui soient utiles pour l'apprentissage. Pour ces étudiants (qui constituent notre corpus de recherche), l'implication vers des projets concrets est plus difficile. Au contact des entreprises formatrices qui les accueillent en fin de formation durant quelques mois de stages pour une première immersion sur le terrain, nous avons constaté que ce manque d'expérience sur des projets concrets et dans la reprise et correction du code d'un autre développeur dans le cursus en école était problématique et rendait plus difficiles leur premiers pas dans le monde professionnel.

Les rencontres réalisées avec certains stagiaires ont de plus relevé que, conscients de cette lacune, une gêne était ressentie face à ce manque de compétence dans cette activité de débogage pour laquelle ils ne se sentent pas à la hauteur des attentes du marché de l'emploi.

3. Cadre théorique

3. 1 La didactique de l'informatique

L'apprentissage de l'informatique et plus spécifiquement de la programmation n'est pas facile à aborder pour les nouveaux professionnels de la branche. Fluckiger (2019) présente comme première difficulté la complexité d'aborder en formation le même objet qu'est la machine informatique, à la fois comme objet de savoir que comme outil d'apprentissage.

En abordant l'apprentissage de l'informatique comme objet de savoir, une des premières difficultés réside dans le fait de traiter des concepts, majoritairement abstraits (Hartmann et al., 2011). Rogalski (2015) présente ainsi la difficulté pour le programmeur de franchir le pas « du faire à faire faire », programmant sa pensée en code que l'ordinateur devra ensuite exécuter.

Dès lors, afin de faciliter la prise en main de ces concepts abstraits, de nombreux outils tels que des robots (Nijimbere et al., 2015 ; Romero et DeBlois, 2022), des représentations graphiques (Gautier et Wrobel-Dautcourt, 2015), des jeux ou des activités débranchées (Vincent et Cohen, 2017) ont vu le jour. L'apport de ces approches réside avant tout dans le fait de donner à voir ce qui est fait par l'ordinateur. A chaque fois, l'importance du debriefing post-activité avec les étudiants permettant de revenir sur les concepts abordés, apparait néanmoins comme centrale (Plumettaz, 2024).

3.2 L'enseignement du débogage informatique

Personne n'écrit de logiciels parfaits, le travail de débogage informatique occupe ainsi une partie importante du temps d'un informaticien (Thomas et Hunt, 2024, p. 121).

L'entreprise informatique IBM (2024) présente le débogage comme un processus qui a pour but de « découvrir la cause des erreurs de codage, de prévenir les problèmes de fonctionnement des logiciels et d'en améliorer les performances générales ». Ce processus se décompose en six étapes différentes : a)

[2] *La dernière ordonnance de formation https://www.fedlex.admin.ch/eli/oc/2020/941/fr*

Reproduire les conditions dans lesquelles le bug est apparu, b) trouver le bug, c) déterminer l'origine du problème, d) corriger le bug, e) tester pour valider le correctif, f) documenter le processus.

En abordant la compétence de débogage informatique, la partie abstraite présentée dans le chapitre précédent réside non seulement dans le fonctionnement non-visible de l'exécution du code mais également dans le raisonnement du professionnel qui débogue et dont on ne peut observer que le résultat de son processus (par exemple une correction effectuée dans une ligne ou partie du code permettant de corriger l'erreur). Or, l'action de débogage consiste principalement à mener une analyse, faisant intervenir non seulement les connaissances des professionnels mais aussi les connaissances des caractéristiques globales du système informatique (Romero et DeBlois, 2022). McCauley et al. (2008, p. 76) présentent sept connaissances utilisées durant la phase de localisation d'un bug et du débogage : 1) connaissance du programme prévu, 2) connaissance du programme réel, 3) compréhension du langage d'implémentation, 4) expertise générale en programmation, 5) connaissance du domaine d'application, 6) connaissance des bugs possibles, 7) connaissance des méthodes de débogage.

Fitzgerald et al. (2008) relèvent également des compétences différentes à mobiliser pour effectuer des tâches de développement et de débogage, mettant en évidence une complexité plus grande pour la deuxième.

Dès lors, si des étudiants ou professionnels sont à l'aise dans l'activité de débogage, ils le seront généralement également dans celle du développement algorithmique. L'inverse n'est cependant pas nécessairement le cas (Fitzgerald et al., 2008). Aborder l'activité de débogage permet ainsi de développer indirectement les compétences de développement et de développer l'esprit critique des professionnels face à des logiciels existants (Vitali-Rosati, 2024, p. 119).

En formation, le débogage informatique est reconnu comme étant une compétence importante mais qui reste à la fois difficile à apprendre pour les programmeurs novices et à enseigner pour les professeurs d'informatique (McCauley et al., 2008). Ceci peut expliquer la raison pour laquelle elle est souvent négligée dans les cursus de formation (Yang et al., 2024). McCauley et al. mettent néanmoins en lumière le fait que les bonnes pratiques menant à un débogage de qualité peuvent être enseignées et relèvent les pistes suivantes pour y parvenir (2008, p. 86) : 1) utiliser un enseignement explicite, 2) partir d'exemples de programmes concrets, 3) donner la possibilité aux étudiants de comparer leurs résultats, 4) amener à penser en mode de test d'hypothèses en évitant d'aller directement vers l'envie de corriger le bug.

Ce sont autant d'enjeux dont notre dispositif expérimental devra tenir compte.

3.3 L'autoconfrontation croisée ou collective

Nous nous appuyons sur les travaux de Clot et al. (2000) pour mettre en place un dispositif sous forme d'autoconfrontation croisée entre les étudiants. Le principe consiste à confronter des professionnels à l'activité de leur pair.

Clot présente les apports de l'autoconfrontation croisée sur base vidéo de la manière suivante :

> Des controverses professionnelles peuvent alors s'engager, portant sur les styles des actions de chacun d'entre eux. Les écarts stylistiques étant évalués par rapport aux formes génériques propres au groupe professionnel, ce sont alors le genre professionnel et ses variantes que les sujets font entrer dans une zone de développement. (Clot et al., 2000, p. 5)

En comparaison avec une autoconfrontation simple qui consiste à mettre un professionnel en confrontation avec son activité, Clot relève qu'une approche croisée entre pairs apporte une modification du niveau d'analyse, à destination non plus d'un chercheur mais d'un collègue, l'amenant à penser, à sentir et à agir selon sa perspective à lui. Clot parle dès lors d'un dispositif ouvrant à une co-analyse du travail quand « on reprend l'analyse en commun d'un même enregistrement vidéo avec un autre expert du domaine, un collègue de travail de même niveau d'expertise » (2018, p. 142). Dans notre expérimentation, nous le présenterons en section 4.1, des groupes de 2 ou 3 étudiants ont été formés, raison pour laquelle nous pouvons aussi bien parler d'autoconfrontation croisée que collective.

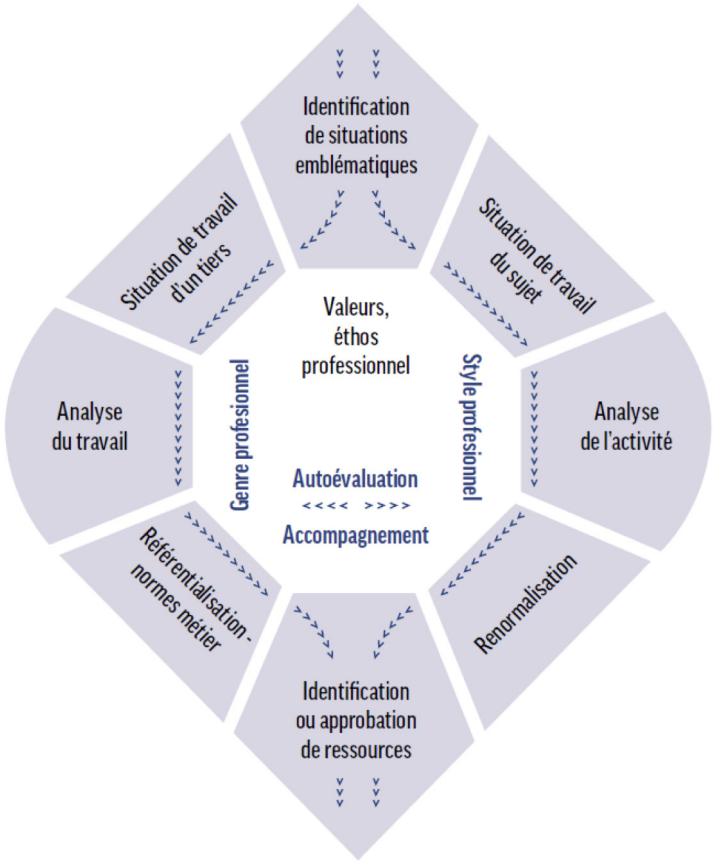
Le dispositif mis en place a ainsi principalement comme ambition de donner accès à la partie non visible de l'analyse menant au débogage informatique en confrontant les activités des pairs, ouvrant à des explications, des arrêts sur image et des compléments d'informations.

Le fait d'utiliser la trace vidéo comme support à cette activité permet de s'appuyer tant sur les fonctions de visibilisation que d'élucidation (Olry et al., 2021).

3.4 Didactique professionnelle par alternance et situation

Enfin, nous nous appuyons sur le dispositif DPAS (Modèle de didactique professionnelle par alternance et situations) de Gremion (2022) qui propose une approche didactique partant de situations emblématiques du métier et qui se divise en deux axes distincts : a) l'axe du genre professionnel de la didactique, b) l'axe du style professionnel de l'activité (Figure 1). L'apprentissage d'un métier passe par l'analyse de l'écart entre l'axe du genre et celui du style réalisé par le professionnel en formation : les parties autoévaluation et accompagnement se trouvant au centre de la Figure 1 dans le schéma adapté par Gremion et Maubant (à paraître, p. 64).

Figure 1 : Modèle DPAS (Gremion et Maubant, à paraître)



Ainsi, le fait de mettre les étudiants dans la situation emblématique du métier d'informaticien qu'est le débogage d'une application existante ne suffit pas à créer un apprentissage. Il s'agit pour les professionnels en devenir de mesurer l'écart entre leur style (présent dans la vidéo qu'ils ont eux-mêmes tournée lors de l'activité de débogage) et le genre professionnel (présent dans la vidéo d'un professionnel expérimenté, dans les normes professionnelles, dans les manuels, ...).

En confrontant les différents styles des apprentis durant l'activité d'analyse croisée des vidéos, la comparaison va au-delà d'une simple superposition de différentes pratiques mais permet de mesurer l'écart des pratiques de l'étudiant avec le genre du métier. La confrontation des différents styles permet ainsi de faire émerger un genre commun entre les étudiants (qui sera institutionnalisé lors de l'étape 6 en fin d'activité).

4. La recherche

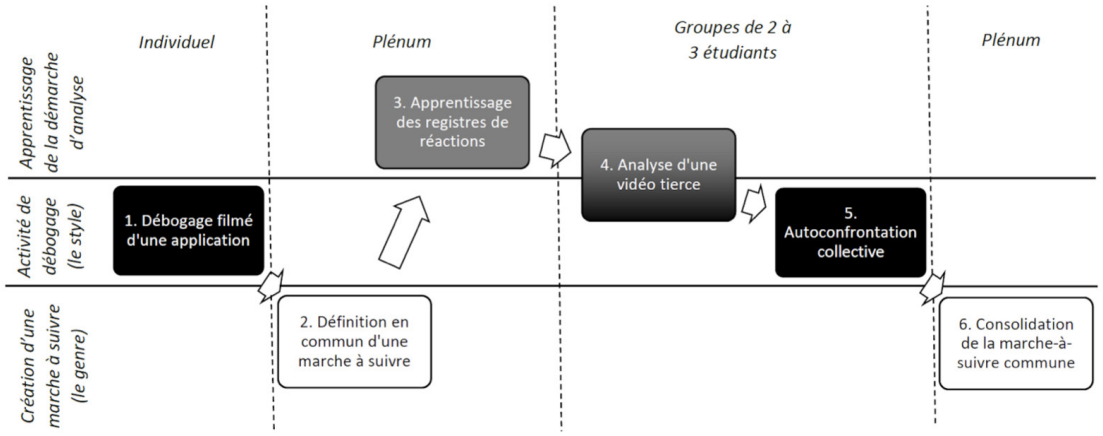
La recherche porte sur l'analyse du dispositif « Atelier du débogueur » mis en place. Ce chapitre permettra de présenter le dispositif, l'approche méthodologique ainsi que les résultats obtenus.

4.1 Dispositif pédagogique mis en place

Pour pallier tant les problèmes liés aux difficultés d'apprentissage du concept de débogage que les problèmes liés à la sensation d'incompétence ressentie par les étudiants dans ce domaine (soulignées dans la partie 2.2), un dispositif utilisant la trace vidéo et l'autoconfrontation croisée (Clot et al., 2000) a été mis en place depuis 2023 à l'Ecole des Métiers de Fribourg (Ramalho, 2023). Le choix de ce dispositif a été effectué en partant de l'hypothèse que la trace vidéo permette de relever, visionner voire revisionner plusieurs fois les étapes clés (et souvent cachées ou réalisées inconsciemment) se trouvant derrière l'activité et qu'une description orale de l'étudiant n'aurait pas forcément relevées. La vidéo permet ainsi au professionnel de voir, se voir et donner à voir son activité (Thievenaz, 2021). Notre dispositif appelé « l'atelier du débogueur » consiste à faire réaliser à toute une classe d'apprentis informaticiens le dépannage d'une même application tout en se filmant via webcam et en enregistrant le flux vidéo des actions menées sur l'écran. Des autoconfrontations croisées entre pairs sont ensuite organisées, afin de faire ressortir les étapes ayant mené à la détection puis à la correction du problème. Ce dispositif qui s'appuie sur d'autres expériences réalisées dans le cadre de la formation des enseignants professionnels (Neuhaus, 2024a) a pour visée de faire émerger la face cachée de l'activité de débogage.

L'atelier du débogueur se déroule en plusieurs étapes, individuelles, en plénum et en sous-groupes qui peuvent être séparées en 3 familles différentes : a) les étapes de film et d'analyse de l'activité de débogage, b) les étapes de création d'une marche à suivre commune, c) les étapes d'apprentissage de la démarche d'analyse. Ces étapes sont présentées dans la figure ci-dessous :

Figure 2 : Etapes de l'activité mise en place



Nous allons ci-dessous présenter en détail le contenu de ces différentes étapes.

1. pour débuter, les apprentis sont confrontés au débogage d'une application identique, contenant une erreur courante. Cette application est développée par l'enseignant dans le langage de programmation « Java », déjà connu des élèves. Volontairement l'erreur ajoutée par l'enseignant est cachée et ne laisse pas de « trace » (aucune information ou message d'erreur présent dans la console de développement) et nécessite ainsi une investigation plus grande pour la détecter et la corriger. Durant cette étape, il est demandé aux étudiants de se filmer individuellement via webcam et d'enregistrer simultanément le flux affiché à l'écran. Un temps de quinze minutes est mis à disposition des étudiants pour réaliser l'exercice ;
2. durant une séance collective, les étudiants définissent la liste des tâches essentielles à réaliser lors d'un processus de débogage. Le but de cette étape est de partir des conceptions et représentations des étudiants au lieu de s'appuyer immédiatement sur une liste de bonnes pratiques tirées d'un référentiel ;

3. l'apprentissage des registres de réactions à utiliser lors de l'analyse d'une vidéo se fait également de manière collective. Le risque en utilisant la trace vidéo pour réaliser l'analyse de l'activité d'un pair réside dans l'aspect évaluatif qui pourrait provoquer un sentiment de malaise, ce qui nécessite d'initialement se former et s'entraîner à l'exercice (Paul, 2022, p. 63). Muller et Lussi Borer (2018) relèvent trois registres de réactions différents lorsque des professionnels analysent une vidéo : descriptif, interprétatif et évaluatif. Le registre évaluatif est celui qui est le plus souvent spontanément mobilisé dans un premier temps ce qui risque de mener à un jugement rapide, peu constructif. Ces différentes étapes sont ainsi présentées aux étudiants par l'enseignant, en insistant sur l'importance de débuter l'analyse par une description de l'activité ;
4. avant d'entrer dans le traitement des vidéos des étudiants, une analyse est effectuée en commun sur une vidéo tournée par l'enseignant lui-même le montrant en situation de débogage d'une autre application. Les buts de cette étape sont multiples. Il s'agit tout d'abord pour les étudiants de s'approprier collectivement les registres de réactions présentées lors du point précédent mais également d'entrer pour la première fois dans l'analyse d'une activité de débogage, raison pour laquelle elle se trouve au milieu des deux zones dédiées dans la Figure 2 ;
5. enfin, des groupes de 2 ou 3 étudiants sont formés. Dans la mesure du possible ils sont volontairement constitués d'étudiants de niveaux différents (des étudiants ayant réussi à corriger l'application avec ceux qui n'y sont pas parvenus). Dans chaque groupe, la consigne est donnée de visualiser tour à tour la vidéo de chaque étudiant avant que ses collègues ne réagissent sous forme d'autoconfrontation collective (Clot, 2018) ; en fin d'activité, un moment collectif permet à la classe de coconstruire une marche à suivre optimisée pour le débogage, intégrant les apprentissages et les observations de chacun.

4.2 La question de recherche

La question de recherche présentée dans cet article est la suivante : Est-ce qu'un dispositif axé sur l'utilisation de la vidéo et de l'autoconfrontation collective tel qu'il est présenté dans l'atelier du débogueur aide les apprentis à faire ressortir les étapes principales menant au débogage d'une application et à se sentir au final davantage aptes à prendre en main et déboguer le programme d'un autre développeur ?

Notre hypothèse est que, confrontés à l'activité de leurs pairs, dans la même situation de débogage, les étudiants parviennent à identifier, non seulement la solution du débogage mais également les étapes clés qui ont mené à sa découverte.

De plus, ayant constaté sur le terrain, principalement lors du départ en stage de formation, des difficultés pour plusieurs étudiants face à la reprise de code d'un autre développeur, nous avançons l'hypothèse que, suite à cet exercice, les étudiants se sentiront davantage en confiance pour prendre en main le programme d'un autre professionnel dans le futur.

4.3 Le corpus de recherche

Le corpus de recherche est composé de 2 groupes-classe différents. Un de 2e année avec 17 étudiants et un autre de 3e année avec 19 étudiants. Les deux groupes sont constituées uniquement de garçons entre 16 et 18 ans et ont le même enseignant (corédacteur de cet article) pour les cours de programmation. Ce choix a été arrêté en fonction des classes auxquelles l'enseignant-chercheur enseignait durant la période de récolte de données.

Les étudiants étant en formation professionnelle d'informaticien, ils ont, avant cette expérimentation, déjà vécu entre 2 et 4 modules de formation sur le thème de la programmation. Chaque module dure 5 semaines, à raison de 1,5 jour bloc par semaine. Ils maitrisent ainsi les bases de la programmation (les tests, les boucles, les méthodes et fonctions, les bases de la programmation orientée objet) et ont déjà indirectement été amenés à mener une activité de débogage mais sur leurs propres programmes lors de la réalisation des exercices demandés dans les modules précédents.

4.4 Méthodologie

La méthodologie de recherche se dessine en plusieurs étapes qualitatives et quantitatives :

a. les différentes vidéos tournées par les étudiants avec leur webcam et la résolution de l'exercice de débogage ont été réceptionnées par l'enseignant et ont permis une analyse globale.
b. le moment d'autoconfrontation collective en sous-groupe a été filmé pour en garder une trace. Pour des raisons logistiques principalement (les autoconfrontations collectives ayant eu lieu en

même temps dans la classe), cette prise vidéo n'a cependant pas été réalisée sur tous les groupes mais uniquement sur certains.

c. deux sondages en ligne ont été réalisés pour sonder *a priori* et *a posteriori* le sentiment de capabilité dont font preuve les étudiants face aux activités de débogage et leur aisance avec l'outil de débogage.

Nous allons préciser ci-dessous la manière dont, pour ces trois étapes, les données ont été acquises, analysées et la manière dont les résultats ont été traités.

4.4.1 Le film des activités de débogage

Les vidéos ont été réalisées en captant aussi bien le flux affiché sur l'écran (montrant les différentes actions réalisées) que le flux capturé par la webcam. Si le but initial avec ces vidéos était d'obtenir un matériel sur lequel s'appuyer pour les autoconfrontations, après l'expérimentation, une visualisation de toutes les vidéos a été réalisée par l'équipe de recherche. Sans grille d'analyse, le but était de relever les étapes principales présentes lors du débogage.

Cette analyse a été réalisée tant en visualisant les activités faites par l'étudiant qu'en analysant les attitudes non-verbales et les mimiques présentes dans les vidéos par webcam. Nous nous appuyons sur les travaux de Thievenaz pour qui « les gestes, postures, regards et intonations peuvent, sous certaines conditions, être considérés comme porteurs de signification et d'intentionnalité ». (2019, p. 88). Ainsi l'analyse de la position du corps, des mains, du regard et des mimiques de la bouche sont autant d'indicateurs permettant d'aider la compréhension du cheminement cognitif de l'étudiant durant l'activité de debogage.

Par ailleurs, pour des raisons de droits à l'image, il est à relever que les vidéos ont été tournées par les étudiants eux-mêmes et restent en leur unique possession. Ainsi, après un visionnage global par l'équipe de recherche, elles ont été effacées et aucune sauvegarde n'en a été faite.

4.4.2 Les autoconfrontations collectives

Les moments d'autoconfrontation collectives ont été menés de manière autonome dans les groupes. La méthodologie d'autoconfrontation a été choisie pour favoriser la richesse des échanges et des apprentissages. En effet, dans notre corpus, 50% des apprentis n'ont pas réussi à découvrir le bug dans le temps imparti. Les groupes ont ainsi été formés en cherchant à mélanger des étudiants ayant trouvé la solution avec les autres. Pour en garder une trace permettant principalement d'illustrer le processus, l'enseignant-chercheur a filmé quelques interactions au hasard de son passage dans les rangs et a noté quelques remarques qui apparaissent comme représentatives du déroulé de l'expérience.

4.4.3 Sondages a priori et a posteriori

Des sondages ont été réalisés au début et à la fin de l'activité pour évaluer la perception des apprentis concernant leur aptitude au débogage. L'objectif était de mesurer leur sentiment de capabilité avant et après l'activité. Les questions posées étaient les suivantes :

a. Comment jugez-vous votre efficacité sur l'utilisation du débogueur (touche raccourcie, parcours de code, …) ?
b. Est-ce que vous vous sentez capables de déboguer un programme informatique d'un autre développeur ?

Les réponses à donner à ces deux questions étaient basées sur une échelle de Likert allant de 1 à 5 (1 = pas bon et 5 = très bon).

La première question est volontairement axée sur l'utilisation de l'outil de débogage. Celle-ci permettra de déterminer si la prise en main de l'outil représente un frein à la réalisation du débogage. La deuxième question s'articule sur la sensation générale de capabilité à déboguer le programme d'un autre développeur.

Les sondages tant *a priori* qu'*a posteriori* sont réalisés depuis une plateforme numérique sur laquelle les étudiants sont connectés. Les données permettent ainsi d'identifier les évolutions pour chaque étudiant entre le début et la fin de l'activité. Elles sont néanmoins anonymisées pour la présentation des résultats dans cet article.

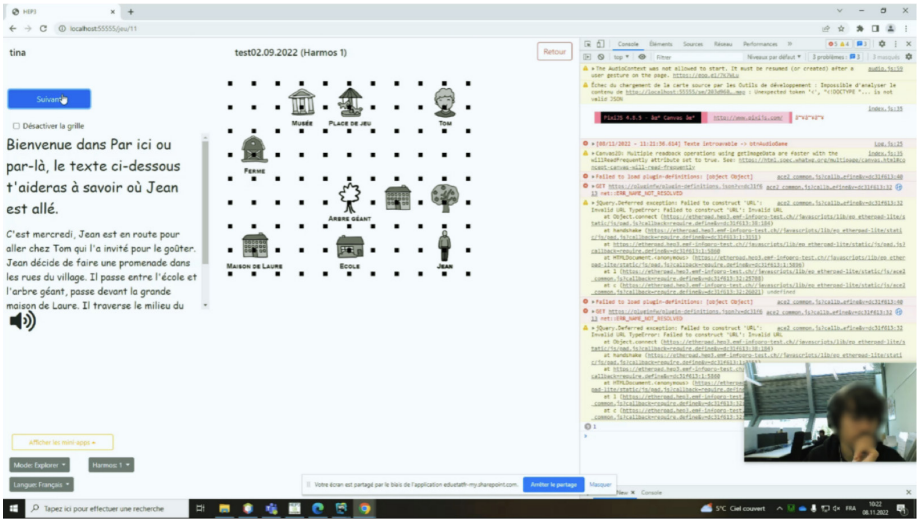
4.5 Résultats obtenus

Nous présenterons notre rendu en trois étapes : a) nous reviendrons sur les films réalisés par les apprentis sur l'activité de débogage ; b) nous présenterons les remarques relevées lors de de l'étape d'autoconfrontation collective ; c) enfin, nous présenterons les analyses des sondages *a priori* et *a posteriori* réalisés.

4.5.1 Débogage filmé par l'apprenti

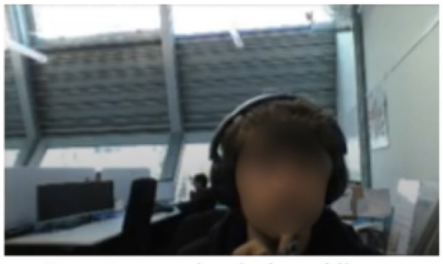
Cette vidéo a été utilisée prioritairement afin de permettre l'autoconfrontation entre les pratiques des étudiants. Cependant, cette expérimentation permet de relever certains éléments qui nous semblent importants par la superposition entre l'écran de l'étudiant et sa webcam (Figure 3).

Figure 3 : Exemple de film réalisé par les étudiants

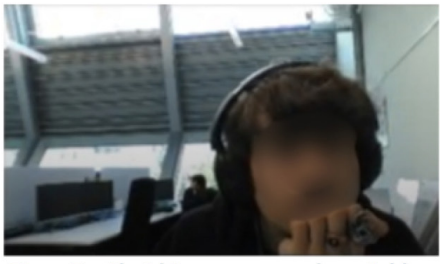


Le fait de demander aux étudiants de filmer simultanément l'écran et le visage permet une analyse complémentaire entre les actions entreprises par l'étudiant et son attitude non verbale (Figure 4).

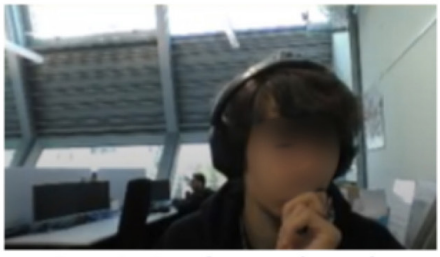
Figure 4 : Images représentant le déroulement du débogage pour un étudiant



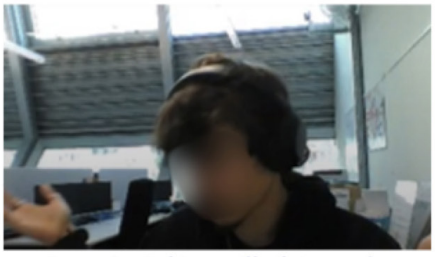
Etape 1 « Je recherche le problème »



Etape 2 « C'est bizarre ce nom de variable »



Etape 3 « Peut-être que c'est ça ! »



Etape 4 « Et bien voilà, j'ai trouvé »

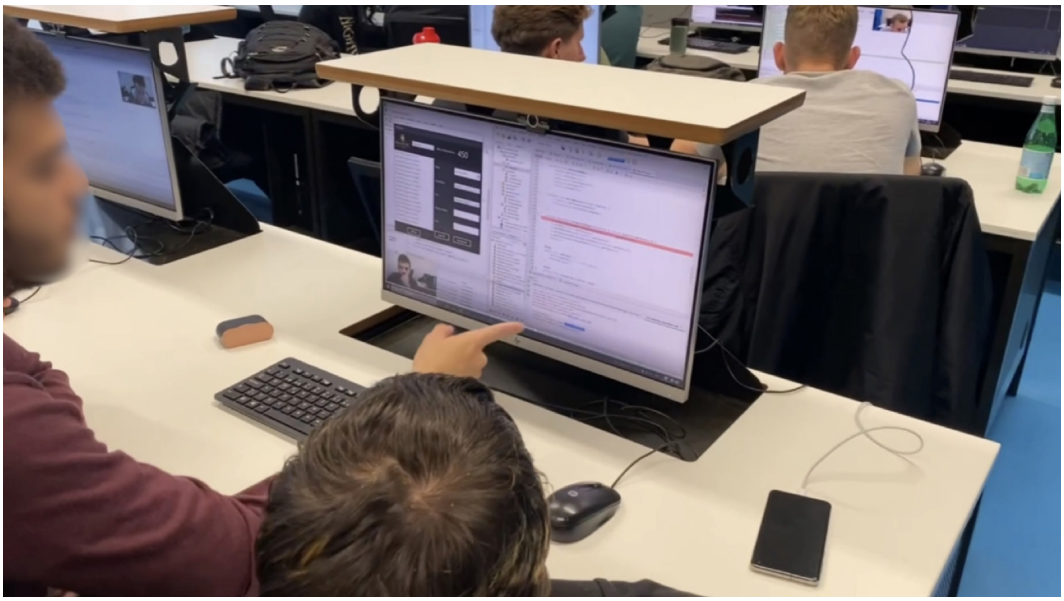
Nous pouvons relever des indicateurs intéressants visibles dans les quatre images ci-dessus, tant par la position des mains que la position corporelle. Ainsi la première image montre l'étudiant, l'index sur la bouche que nous interprétons comme un moment de réflexion. Le passage vers la 4e image montre les mains se plaçant sous le menton, devant la bouche pour terminer par un mouvement illustrant la satisfaction de la réussite.

Nous émettons l'hypothèse que ces indices non-verbaux permettent non seulement au développeur qui revisionne sa vidéo de se replonger au plus proche de son état et ses doutes durant l'activité mais également à son collègue visualisant la vidéo d'en prendre conscience plus facilement qu'avec les seules actions à l'écran.

4.5.2 Autoconfrontation collective

Durant cette étape, chaque étudiant montre sa vidéo à ses collègues de groupe avant que ceux-ci ne réagissent (Figure 5).

Figure 5 Autoconfrontation entre les étudiants



Cette étape n'a pas été systématiquement filmée et analysée pour tous les binômes. Cependant, lors de nos passages dans les rangs, nous avons constaté que les apprentis ont rencontré des difficultés à suivre rigoureusement le processus structuré de description, d'interprétation et d'évaluation. La partie évaluation était très rapidement abordée dans certains groupes. Néanmoins, sans en avoir fait un référencement systématique, nous avons relevé que les remarques échangées étaient bienveillantes et les conseils prodigués ont enrichi le groupe. Nous avons également relevé que de nombreux échanges portaient sur le processus abordé lors du débogage et non plus la simple correction de l'erreur. Certaines phrases représentatives ont été notées lors de nos passages telles que « Tu peux tester le programme en l'exécutant une première fois, avant de lancer le débogueur pour voir le problème », « Moi, j'ai plutôt lancé le débogueur, plutôt que de rechercher dans le code ce qui n'allait pas », « J'ai utilisé les touches de raccourci F7/F8 et relancé plusieurs fois l'application pour voir quel était le problème ».

Ces témoignages illustrent un partage d'expériences précieux qui contribue à l'amélioration des compétences de débogage de chacun. Nous pouvons relever que ces remarques abordent aussi bien le processus permettant de mener à bien cette activité que l'utilisation de l'outil de débogage. En permettant aux apprentis de confronter leurs méthodes et résultats, cette activité semble ainsi favoriser non seulement l'apprentissage collaboratif, mais aussi le développement de compétences critiques et analytiques essentielles au débogage.

4.5.3 Sondage sur la sensation de capabilité

Les résultats des sondages avant et après l'activité ont relevé une évolution importante des réponses des apprentis quant à leur capacité d'analyse et de prise en main d'un code tout comme dans la confiance affichée par ces derniers sur leur capacité à intervenir sur le terrain pour corriger le code d'un autre développeur.

Ainsi à la question a) « Comment jugez-vous votre efficacité sur l'utilisation du débogueur (touche raccourcie, parcours de code, ...) ? », les statistiques des réponses montrent l'évolution suivante :

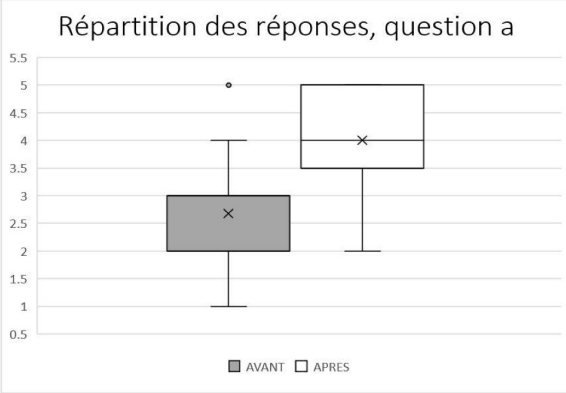
Tableau 1 Statistique des réponses à la question a (1 : pas du tout efficace ; 5 : très efficace)

		N	Moyenne	Médiane	[Min,Max]	Ecart type	Erreur standard
Score	Avant	28	2.6875	3	[1, 5]	0.9833	0.1858
	Après	29	4.0000	4	[2, 5]	0.8018	0.1489

La sensation d'efficacité avec l'outil de débogage est beaucoup plus élevée après l'activité et ceci en présentant moins de différences (écart type) entre les étudiants. Plus aucun étudiant ne se retrouve par ailleurs dans une sensation d'inefficacité totale.

Le graphique en Figure 6 illustre une sensation de capabilité qui se trouve globalement plus haute après l'activité mais avec un spectre général plus grand allant de 3,5 à 5.

Figure 6 : Schéma représentant les réponses à la question a



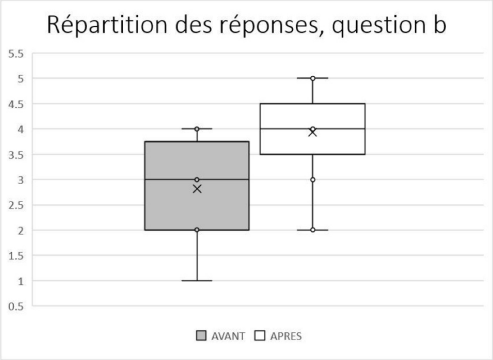
De la même manière, pour la question b) « Est-ce que vous vous sentez capables de déboguer un programme informatique d'un autre développeur ? » les résultats sont les suivants :

Tableau 2 : Statistique des réponses à la question b
(1 : pas du tout capable ; 5 : complément capable)

		N	Moyenne	Médiane	[Min,Max]	Ecart type	Erreur standard
Score	Avant	28	2.8214	3	[1, 4]	0.8929	0.1630
	Après	29	3.9310	4	[2, 5]	0.8422	0.1564

A nouveau, une évolution nette se dessine à ce sujet entre avant et après l'activité. Après l'activité, plus aucun étudiant ne se sent totalement incapable dans cette activité et la moyenne obtenue se rapprochant du score de 4 avec un écart type plus faible. Le graphique en Figure 7 montre de plus une normalisation des retours après l'activité avec des conceptions plus proches entre les étudiants.

Figure 7 Schéma représentant les réponses à la question b



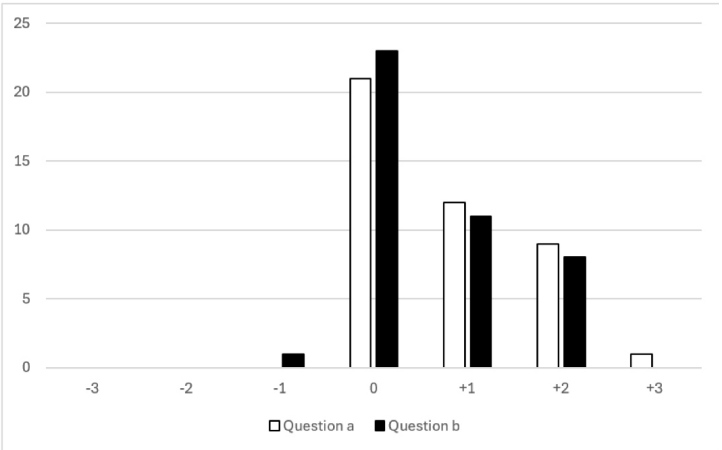
Enfin, afin de détecter si l'activité a été profitable individuellement pour tous les participants, nous avons également vérifié l'évolution individuelle de chaque étudiant. Le tableau ci-dessous présente le nombre d'étudiants ayant amélioré (de +1, +2, +3 ou +4) ou non sa réponse entre les deux questionnaires.

Tableau 3 : Evolution des réponses individuelles avant ou après l'activité

	-3	-2	-1	0	+1	+2	+3	+4
Question a	0	0	0	21	12	9	1	0
Question b	0	0	1	23	11	8	0	0

Le graphique suivant reprenant les mêmes données illustre une stagnation ou évolution positive pour presque tous les étudiants entre avant et après l'activité.

Figure 8 : Graphique de l'évolution des réponses individuelles avant ou après l'activité



Nous pouvons relever qu'à la question a) « Comment jugez-vous votre efficacité sur l'utilisation du débogueur (touche raccourcie, parcours de code, ...) ? », 48% des personnes n'ont pas vu de changement, 28% ont augmenté de 1 point et 21% ont augmenté de 2 points.

Pour la question b) « Est-ce que vous vous sentez capables de déboguer un programme informatique d'un autre développeur ? », 2% ont diminué de 1 point, 53% n'ont pas vu de changement, 16% ont augmenté de 1 point et 19% ont augmenté de 2 points.

4.6 Discussions

Les résultats présentés dans cet article sont encourageants quant à l'utilisation de tels dispositifs avec les apprentis informaticiens. Le dispositif mis en place permet non seulement aux étudiants d'aborder en cours des concepts difficiles, mais, comme le montrent les statistiques post activité, contribue également à donner aux étudiants une confiance leur permettant de se sentir à même de reprendre le code d'un autre développeur.

Cette première expérimentation ouvre néanmoins à des questions nouvelles sur le fonctionnement et les apports du dispositif, étape après étape. Ces questions ne peuvent néanmoins pas être répondues par la méthodologie adoptée dans cette recherche qui présente comme limite principale le fait que le dispositif d'analyse de l'étape d'autoconfrontation croisée est non systématique. Ainsi nous avons relevé certaines phrases d'étudiants au gré de nos passages dans les rangs mais pour mieux saisir l'apport de cette étape, il serait important que celle-ci soit également filmée et analysée de manière systématique. Cependant ces quelques phrases ont permis de mettre en évidence les étapes difficiles à enseigner en école et relevées dans notre cadre théorique en nous basant sur les prescriptions d'IBM (2024) et sur les connaissances à mobiliser relevées par McCauley et al. (2008). Comme évoqué par Fitzgerald et al. (2008), nous pouvons ainsi émettre l'hypothèse que le développement de cette compétence en analyse et débogage de code contribuera également à améliorer les compétences en développement. Cette tendance devra cependant être vérifiée dans de prochaines recherches.

Les résultats qui découlent de cette première expérimentation nous encouragent à poursuivre nos recherches de manière plus poussée. Nous pouvons évoquer en priorité les directions suivantes qui seront données à nos prochaines recherches : a) de telles expérimentations gagneraient à être non seulement reproduites à plusieurs reprises mais également en immersion réelle sur le terrain. Ainsi, pour les étudiants étant en stage en entreprise durant leur 4e année de formation, le fait de réaborder ces concepts à cette occasion pourrait être particulièrement fertile en termes formatifs, b) ensuite, si l'expérimentation a été réalisée avec des étudiants en formation en école à plein temps, une expérimentation en classe duale serait également à réaliser. En effet, celle-ci permettrait de potentiellement relever non pas des pratiques émergentes personnelles des candidats mais également de confronter les différentes approches ou habitudes des entreprises formatrices portées par les apprentis. Cette confrontation pourrait contribuer à apporter aux étudiants des approches différentes facilitant leur entrée dans le monde professionnel une fois la formation terminée, c) dans l'optique de la mise en place d'un portfolio à visée formative (Dévé et al., 2009) dans l'établissement, il pourrait être intéressant de demander aux étudiants de garder ces différentes vidéos réalisées au fil de leur apprentissage pour y revenir au fil de leur évolution, relevant ainsi de manière réflexive leur progrès (Gauthier, 2009). Ceci contribuerait à donner aux étudiants davantage de confiance envers leur capabilité à agir sur le terrain de manière adéquate.

5. Conclusions et perspectives

L'expérimentation réalisée sur dans le dispositif « l'atelier du débogueur » est riche de belles perspectives.

Comme relevé dans la discussion ci-dessus, nos résultats s'appuient sur un corpus relativement limité et une acquisition de données qui gagnerait à être davantage approfondie, ce qui sera fait dans de futures recherches. Cette première expérimentation laisse cependant entrevoir un impact positif de l'autoconfrontation croisée en utilisant la trace vidéo, tant sur la sensation d'efficacité des étudiants avec les outils de débogage que sur leur confiance face à cette tâche essentielle pour leur futur professionnel.

Par ailleurs, ces premiers résultats mettent en lumière certains indices qui encouragent à poursuivre les recherches dans cette direction. Nous en relevons deux principaux. D'une part l'autoconfrontation croisée sur les vidéos tournées par les apprentis informaticiens permet de faire remonter les parties invisibles de cette activité, contribuant au développement et à l'évolution de tous les étudiants. D'autre part, au sortir de cette activité, les étudiants se disent plus en confiance pour reprendre le code d'autres développeurs. Cette sensation de capabilité peut contribuer à rendre l'entrée plus facile sur le milieu professionnel.

Ainsi, par ce dispositif, les enseignants contribuent à donner confiance aux étudiants sur leur niveau de compétence, élément déterminant à la motivation scolaire (Viau, 2009). Si les référentiels de formation pour les informaticiens prévoient nombre de compétences essentielles à l'exercice du métier (chapitre 2.2), cette approche contribue à réduire l'écart entre les concepts vus en école professionnelle et les besoins du terrain. Par ailleurs, le fait d'aborder en école les compétences de débogage permettant selon certaines études d'améliorer les compétences en développement algorithmique des étudiants (Fitzgerald et al., 2008), nos résultats suscitent des espoirs de retombées indirectes en termes de formation des apprentis à plus large échelle.

Si cet article ne revient pas sur cette considération, il faut cependant résoudre en amont la question des droits nécessaires pour pouvoir tourner la vidéo. Dans cette perspective, une information claire doit être donnée aux étudiants sur les fins utilisées à cette vidéo. Il nous apparait également comme important que la liberté soit laissée à l'étudiant de ne pas se filmer s'il ne le souhaite pas. Il conviendra dès lors de leur donner une autre consigne. En ce sens, nous pouvons évoquer comme piste le fait de filmer uniquement le flux de l'écran mais sans l'utilisation de la webcam.

Forts de cette expérimentation, nous pouvons également envisager la mise en place de dispositifs similaires pour d'autres publics que celui de la formation professionnelle initiale, parmi lesquelles : a) des dispositifs basés sur l'analyse de traces vidéos pour d'autres tâches que le débogage, telles que l'utilisation d'outils bureautiques ; b) l'utilisation à des fins de formation continue pour des professionnels confirmés ; c) d'autres domaines ou métiers que celui de l'informatique gagneraient à être expérimentés par l'utilisation de la vidéo et de l'autoconfrontation croisée. Ceux de la formation des enseignants (Neuhaus, 2024a) et de la formation des chefs d'orchestre (Neuhaus, 2024b) ont déjà été expérimentés et portent des espoirs tant en formation de base qu'en formation continue.

La montée de l'intelligence artificielle donne une perspective supplémentaire. Nous pouvons déjà noter que les étudiants utilisent l'intelligence artificielle pour pouvoir comprendre et analyser leurs programmes au détriment d'autres outils ou forums tels que Stackoverflow[3] (Ramalho, 2024). De plus en plus d'étudiants utilisent l'IA notamment pour déboguer leur programme quand une trace est disponible. Cette alternative doit également être accompagnée par les enseignants pour améliorer les automatismes liés à cette aide de l'IA. Nous pourrions aussi imaginer avoir dans les prochains mois des outils d'intelligence qui permettraient d'analyser automatiquement les manipulations faites par les étudiants sur le code à déboguer. De même l'IA pourrait contribuer tant à la création de programmes qu'à l'ajout de bugs aléatoires dans ceux-ci. Il s'agit là de perspectives qui seront à prendre en compte en fonction des apports futurs de ces outils qui arrivent en nombre tant sur le terrain professionnel des informaticiens que dans le monde de la formation.

6. Bibliographie

Clot, Y. (2018). *La fonction psychologique du travail*. Presses Universitaires de France.

Clot, Y. et Faïta, D. (2000). Genres et styles en analyse du travail, concepts et méthodes. *Travailler*, *4*.

Clot, Y., Faïta, D., Fernandez, G. et Scheller, L. (2000). Entretiens en autoconfrontation croisée : une méthode en clinique de l'activité. *Perspectives interdisciplinaires sur le travail et la santé*, (2-1). https://doi.org/10.4000/pistes.3833

Dévé, V., Gagnayre, R. et d'Ivernois, J.-F. (2009). Le portfolio : définitions et perspectives pédagogiques à partir d'une analyse de textes canadiens et européens. *Education du Patient et Enjeux de Santé*, *27*(1). https://ipcem.org/img/articles/Port_Folio_pedagogie.pdf

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L. et Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93-116. https://doi.org/10.1080/08993400802114508

Fluckiger, C. (2019). *Une approche didactique de l'informatique scolaire*. Presses universitaires de Rennes.

Gauthier, P.-D. (2009, 1 janvier). *La réflexivité dans la démarche Portfolio*. https://doi.org/10.13140/2.1.4585.9844

Gautier, M. et Wrobel-Dautcourt, B. (2015). Patrie 3, chapitre 3 / artEoz, visualisation dynamique de programmes. Dans *Informatique en éducation: perspectives curriculaires et didactiques*. Presses universitaires Blaise Pascal.

Gremion, C. (2022). Usages et mésusages des pédagogies de l'alternance. *Phronesis*, *11*(1-2), 91. https://doi.org/10.7202/1087560ar

Gremion, C. et Maubant, P. (à paraître). Vers une mise en œuvre pédagogique de la didactique professionnelle par alternance et situation. Dans G. Ghisla, E. Boldrini, C. Gremion, F. Merlini et E. Wüthrich (dir.), *Didactique et situations, Approches et expériences pour la formation professionnelle*, (p. 63-73), CREME.

Hartmann, W., Näf, M. et Reichert, R. (2011). *Enseigner l'informatique*. Springer.

[3] *https://stackoverflow.com/ Stackoverflow est un forum web dédié aux développeurs sur lequel de nombreux professionnels et développeurs amateurs échangent sur des problèmes rencontrés lors de développements.*

IBM. (2024, 16 septembre). *Qu'est-ce que le débogage ? | IBM*. https://www.ibm.com/fr-fr/topics/debugging
Kernighan, B. W. et Plauger, P. J. (1995). *The elements of programming style* (2. ed., 16. print). McGraw-Hill.
Kriesi, I., Bonoli, L., Grønning, M., Hänni, M., Neumann, J. et Schweri, J. (2022). *Rapport de tendance - La formation professionnelle au niveau international et en Suisse - Tensions, Défis, Développement, Potentiels*. OBS HEFP Rapport de tendance 5. Zollikofen : Haute école fédérale en formation professionnelle HEFP. https://www.sfuvet.swiss/sites/default/files/2022-11/Trendbericht_2022_FR.pdf
McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L. et Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67-92. https://doi.org/10.1080/08993400802114581
Michaeli, T. et Romeike, R. (2019). Improving Debugging Skills in the Classroom – The Effects of Teaching a Systematic Debugging Process.
Muller, A. et Lussi Borer, V. (2018). Chap. 4 - Analyse de l'activité, environnements de vidéoformation et développement professionnel des enseignants. Différence intra et inter-activité: le possible, le réel et le virtuel. Dans C. Gaudin, S. Flandin, S. Moussay et S. Chaliès (dir.), *Vidéo-formation et développement de l'activité professionnelle enseignante* (p. 95 à 116). L'Harmattan.
Neuhaus, O. (2024a). *L'autoconfrontation collective entre pairs pour accompagner l'entrée dans le métier des enseignants professionnels* [thèse de doctorat]. Université de Genève. https://archive-ouverte.unige.ch/unige:178502
Neuhaus, O. (2024b, janvier). *L'évaluation croisée entre pairs pour former les chefs d'orchestres amateurs en Suisse* [présentation internationale]. Présentation internationale communication présentée au Symposium « Évaluer l'éducation artistique et culturelle, entre normalisation et différenciation » lors du 35e colloque international ADMEE, Braga, Portugal.
Nijimbere, C., Boulc'h, L. et Baron, G.-L. (2015). Patrie 3, chapitre 4 / Apprendre l'informatique par la programmation des robots. Cas de Legos Mindstorms NXT. Dans *Informatique en éducation: perspectives curriculaires et didactiques*. Presses universitaires Blaise Pascal.
Olry, P., Métral, J.-F. et Chrétien, F. (2021). L'usage des vidéos en didactique professionnelle : statuts technique, pragmatique, épistémique et acceptabilité sociale. *Savoirs*, *N° 55*(1), 59-75.
Paul, M. (2022). *Accompagner la problématisation des situations professionnelles: analyser ses pratiques autrement*. De Boeck Supérieur.
Plumettaz, M. F. N. (2024). *Institutionnalisation des savoirs informatiques lors du débriefing. Le cas de l'apprentissage avec le jeu Programming Game* [thèse de doctorat]. Université de Genève. https://archive-ouverte.unige.ch/unige:175760
Ramalho, M. (2023). *Didactique disciplinaire et de domaine professionnel - Atelier du débogueur* [Rapport de validation d'un module de formation à la HEFP]. Haute école fédérale en formation professionnelle HEFP.
Ramalho, M. (2024, 3 mai). L'utilisation de ChatGPT en formation, au-delà de l'instantanéité. *ICTJournal*. https://www.ictjournal.ch/articles/2024-05-03/lutilisation-de-chatgpt-en-formation-au-dela-de-linstantaneite
Rogalski, J. (2015). Partie 3, chapitre 5 / Psychologie de la programmation, didactique de l'informatique. Déjà une histoire… Dans *Informatique en éducation: perspectives curriculaires et didactiques*. Presses universitaires Blaise Pascal.
Romero, M. et DeBlois, L. (2022). Analyse du processus de construction de connaissances dans des activités de programmation à l'école. *Canadian Journal of Science, Mathematics and Technology Education*, *22*(2), 405-421. https://doi.org/10.1007/s42330-022-00210-9
SEFRI, S. d'État à la formation à la recherche et à l'innovation. (2022). *En 2022, près de 60% des jeunes optent pour une formation professionnelle initiale*. https://www.admin.ch/gov/fr/accueil/documentation/communiques.msg-id-89058.html
Siegmund, B., Perscheid, M., Taeumel, M. et Hirschfeld, R. (2014, 4 novembre). *Studying the Advancement in Debugging Practice of Professional Software Developers*. https://doi.org/10.13140/2.1.2073.7288
Thievenaz, J. (2019). La théorie de l'enquête de John Dewey : réexplorations pour la recherche en sciences de l'éducation et de la formation. *Recherche & formation*, *92*(3), 19-38.
Thievenaz, J. (2021). Éditorial Le recours à l'image en formation des adultes : voir, se voir, donner à voir. *Savoirs*, *N° 55*(1), 7-13.
Thomas, D. et Hunt, A. (2024). *Le programmeur pragmatique* (20 ans, édition anniversaire). Pearson France.
Viau, R. (2009). *La motivation en contexte scolaire* (2e éd). De Boeck université.
Vincent, J.-M. et Cohen, G. (2017). *L'informatique débranchée: le numérique sans ordinateur : activités de découverte du primaire au lycée*. Éditions Pole.
Vitali-Rosati, M. (2024). *Éloge du bug: être libre à l'époque du numérique*. Zones.
Yang, S., Baird, M., O'Rourke, E., Brennan, K. et Schneider, B. (2024). Decoding Debugging Instruction: A Systematic Literature Review of Debugging Interventions. *ACM Transactions on Computing Education*, *24*(4), 1-44. https://doi.org/10.1145/3690652

DES USAGES DU NUMÉRIQUE DANS L'ÉDUCATION MUSICALE

Jérôme Albert Schumacher. Haute école pédagogique du canton de Vaud, jerome-albert.schumacher@hepl.ch

Esther Duteil, Haute école de musique Genève-Neuchâtel esther.duteil@etu.hesge.ch

Résumé

Explorant les usages de 11 enseignant·e·s de musique recourant aux outils numériques, cette étude qualitative combine entretiens et observations in situ. Elle identifie des profils d'enseignant·e·s et des usages technopédagogiques liés à l'alternance, l'inclusion et l'accompagnement. L'analyse montre comment des outils variés (smartphone, partitions interactives, etc.) sont adaptés par bricolage, braconnage ou bidouillage pour soutenir l'apprentissage hybride. L'article souligne la valeur des ressources asynchrones pour l'autonomie et l'inclusion et envisage le potentiel de l'IA pour enrichir le feedback pédagogique.

Mots-clés : Education musicale, technologies numériques, pédagogie numérique, usages, formation hybride

1. Introduction

L'intégration des technologies numériques transforme les pratiques pédagogiques dans divers contextes éducatifs. Comprendre finement les usages réels qu'en font les enseignant·e·s est essentiel pour éclairer les enjeux de cette intégration. Cet article s'inscrit dans cette perspective en explorant les usages numériques d'enseignant·e·s de musique de Suisse romande. L'intérêt de se pencher sur ce domaine particulier n'est pas anodin : l'histoire de la musique est intrinsèquement liée à l'innovation technologique, depuis les premiers instruments jusqu'aux outils numériques les plus sophistiqués qui reconfigurent la création, la diffusion et l'écoute (Acquilino et Scavone, 2022). Ce champ disciplinaire est donc un terrain particulièrement fertile pour observer comment les enseignant·e·s s'approprient ces outils, les intègrent à leurs pratiques et, parfois, les détournent pour répondre à des finalités éducatives spécifiques.

Cet article présente les résultats d'une recherche menée par l'Institut de recherche en musique et arts de la scène de la Haute école spécialisée de Suisse occidentale et la Haute école de musique Genève-Neuchâtel. L'objectif principal est d'étudier et d'analyser les usages numériques d'enseignant·e·s de conservatoires et de hautes écoles de musique déclarant intégrer les technologies numériques dans leurs pratiques pédagogiques.

Une précédente recherche (Schumacher, 2025) avait été consacrée à l'autoévaluation des compétences numériques des professeur·e·s des hautes écoles de musique de Suisse romande. Il en était ressorti que les répondant·e·s estimaient posséder une bonne maîtrise des compétences liées à la pédagogie numérique ainsi qu'à l'utilisation et l'évaluation des informations numériques. Les compétences liées à la citoyenneté numérique (ex. : connaissance et maîtrise des droits d'auteur et des licences) étaient perçues comme plus faibles, mettant en exergue un besoin de formation institutionnel. Un autre volet de cette recherche avait été consacré à l'analyse des pratiques numériques déclarées par certain·e·s professeur·e·s dans le cadre d'entretiens semi-dirigés. L'analyse mettait en évidence une diversité de pratiques déclarées, dont notamment une utilisation privilégiée de l'ordinateur, des essais d'hybridation des enseignements et un emploi accentué des outils de communication proposés par les technologies numériques.

Ces résultats, bien qu'éclairants, n'étaient basés que sur des autoévaluations et des pratiques déclarées. Aussi, cette deuxième recherche a poussé la porte des studios de musique et s'est intéressée aux enseignant·e·s qui reconnaissent recourir aux technologies numériques dans leurs pratiques pédagogiques car, comme Latour (1987), nous pensons qu'il est essentiel d'observer « les mains, les yeux et le contexte matériel de ceux qui savent » (p. 83). Pour analyser ces usages observés, notre démarche s'appuie délibérément sur des cadres théoriques fondateurs issus de la sociologie et de l'anthropologie des techniques (notamment de Certeau, 1980 ; Lévi-Strauss, 1990 ; Perriault, 2008). Nous postulons que ces approches classiques de l'usage et de l'invention du quotidien offrent des clés de lecture particulièrement pertinentes pour comprendre l'appropriation contemporaine, souvent créative et détournée, des technologies numériques en contexte éducatif.

2. Contexte théorique : de l'usage aux usages du numérique

2.1 De l'usage …

Perriault (2008) définit l'usage, cette « entité insaisissable » (p. 13), comme « une utilisation stabilisée d'un objet, d'un outil, pour obtenir un effet » (p. 70). L'usage n'est pas instantané. Il se développe lorsque le rôle de l'objet s'est stabilisé dans la société. Par exemple, lorsque Thomas Edison publie son article dans la North American Review en 1878 au sujet du phonographe qu'il vient d'inventer, il propose une dizaine d'usages potentiels, allant de l'écriture des lettres à la musique en passant par l'archive familiale ou des fins éducatives (pour n'en citer que quelques-uns) (Maisonneuve, 2006).

La musique n'intervient qu'en quatrième position des usages évoqués par son créateur. Cependant, les usages sociaux du phonographe ont amené à ce que ce dernier produise avant tout un *effet* musical. En plus de préserver des enregistrements, s'adjoindra l'idée de les collectionner, de comparer les interprétations en ne se concentrant que sur la musique. Avec le temps, le phonographe supplantera le piano, instrument phare des salons du 20e siècle (d'où l'expression d'alors : *jouer du phonographe*). Cet exemple permet de mieux comprendre qu'au-delà de l'usage prescrit, se développe un usage informel, voire détourné prouvant qu'il est avant tout un construit individuel et social (Perriault, 2008). Un autre exemple de détournement d'usage est mentionné par Perriault (dans Maïdakoualé, 2021) : le début des années 1990 a marqué les prémices de la transmission d'émissions télévisées par satellite dans les grandes villes du Maghreb. Des personnes ne disposant pas des installations nécessaires ont alors détourné l'usage de leur couscoussier

pour le transformer en antenne parabolique, leur permettant de capter les émissions proposées par seize satellites. Ces « pratiques déviantes » (Perriault, 2008, p. 13), loin d'être de simples erreurs de manipulation relèvent davantage d'intentions, voire de préméditations. L'étude de l'usage implique qu'il est non seulement essentiel de comprendre comment les objets sont conçus, mais qu'il est tout aussi important d'observer comment ils sont utilisés par les individus dans leur vie quotidienne (de Certeau, 1980). Ce sont, en effet, « les processus d'utilisation qui doivent constituer un objet de recherche central et non l'artefact lui-même » (Rabardel, 1995, p. 27). Il faut donc observer ces habiletés là où elles sont produites « or ce lieu est toujours à l'échelle d'un ou de quelques individus » (Rabardel, 1995, p. 23). Cette nécessité d'observer les processus d'utilisation individuelle est particulièrement prégnante avec les technologies numériques, dont l'intégration dans le quotidien est rapide et les appropriations multiples.

2.1 … aux usages du numérique

Les outils numériques ont conquis notre style de vie et nous accompagnent au quotidien (Latour, 2000). Les rendez-vous se prennent en ligne, les vacances se réservent sur des sites internet dédiés, etc. L'exemple du smartphone est éclairant car ses usages sont multiples : il est tour à tour appareil photo, agenda électronique, GPS, mais aussi métronome et accordeur pour les musicien·ne·s, et même, comme Plantard (2016) le fait remarquer, objet de mode pour les plus jeunes générations.

Les auteurs reconnaissent que la notion d'usages numériques s'est développée en France dans le courant des années 1980 avec l'avènement du Minitel[1] et des expérimentations sociales qui en ont découlé (Jouët, 2000 ; Méadel, 2019 ; Paquienséguy, 2017) ; on y étudie notamment comment certain·e·s usager·ère·s recourent et s'approprient les messageries roses (Dakhlia et Poels, 2012 ; Jouët, 2000). Cette approche centrée sur l'appropriation par les usager·ère·s est particulièrement féconde pour analyser l'intégration des technologies en éducation, où les détournements et adaptations sont fréquents, comme l'illustre Plante (2016) avec l'exemple d'un jeu (Minecraft) détourné par des milliers d'enseignant·e·s pour atteindre des objectifs pédagogiques. Dans l'éducation musicale, où la technologie est historiquement liées à la création et à l'interprétation (Acquilino et Scavone, 2022), cette capacité d'appropriation est d'autant plus pertinente pour répondre aux spécificités de l'enseignement artistique. Ces processus d'appropriation, où les usager·ère·s s'approprient les outils numériques en les transformant et en les détournant (Méadel, 2019), peuvent être analysés à travers les notions complémentaires de *bricolage* (Lévi-Strauss, 1990), de *braconnage* (de Certeau, 1980) et de *bidouillage* (Larribeau, 2019). Sans pour autant évacuer les usages plus *traditionnels* des outils numériques (comme la simple consultation de ressources ou la communication), notre analyse se concentre sur ces formes d'appropriation créative qui témoignent d'une intégration plus profonde des outils dans la pratique pédagogique.

Pour Lévi-Strauss (1990), le *bricolage* est l'art d'utiliser « les moyens du bord, c'est-à-dire un ensemble de pièces et d'outils hétéroclites, récupérés ici et là, et dont la composition n'est pas liée à un projet spécifique mais est le résultat d'occasions diverses » (p. 33). Plantard (2016) ajoute : « c'est exécuter un grand nombre de tâches diversifiées dans un univers instrumental clos, avec un ensemble fini d'outils et de matériaux pour réaliser un projet déterminé. Tous les usagers du numérique bricolent avec les instruments qui les entourent » (p. 63). Ainsi, le·la *bricoleur·euse numérique* réalise des tâches avec les outils numériques qu'il·elle a en sa possession, s'accommodant des moyens du bord même s'il·elle ne détient peut-être pas le dernier ordinateur, la dernière version d'un logiciel ou même le bon programme. Il s'agit d'une adaptation fonctionnelle d'un outil existant à une nouvelle fin, sans en modifier la nature profonde.

Se référant à de Certeau (1980), Plantard (2016) définit le *braconnage numérique* comme étant « la forme collective d'intelligence pratique des instruments technologiques. Le braconnage tisse les liens avec les autres et modifie l'organisation et les interactions sociales » (p. 62). Le·la *braconneur·euse numérique* est un·e utilisateur·trice intégré·e dans un réseau social de pairs partageant les mêmes intérêts et avec lequel il·elle construit ses compétences par échanges et partages d'expériences, où chacun·e est tour à tour apprenant·e et enseignant·e. L'accent est mis sur l'appropriation sociale et le détournement intelligent au sein d'une communauté.

Le *bidouillage*, quant à lui, est défini par Larribeau (2019) comme étant une réponse à « l'envie de mettre en pratique une démarche collaborative, de reprendre le contrôle sur les objets, d'en découvrir les fonctionnalités imprévues […][qui] mobilisent donc un ensemble de compétences techniques, principalement informatiques » (p. 59). Le·la *bidouilleur·euse numérique* possède les compétences

[1] Le Minitel (Médium Interactif par Numérisation d'Information Téléphonique) est un outil de télématique (contraction de téléphone et informatique) développé et distribué en France entre le début des années 1980 et la fin des années 1990 (Gonzalez et Jouve, 2002).

technologiques suffisantes pour agir sur le programme via le code ; il·elle ne se contente pas d'opérations de surface, mais convoque ses connaissances de programmateur·trice pour influer directement sur le fonctionnement de la machine. Il s'agit ici d'une modification structurelle de l'outil lui-même.

Ces trois concepts ne sont pas mutuellement exclusifs mais représentent différentes logiques d'appropriation. Tandis que le bricolage relève d'une adaptation de surface et le braconnage d'une intelligence collective et *rusée*, le bidouillage implique une intervention plus profonde sur la technologie elle-même. Dans le contexte de cette étude, ils permettent de qualifier la manière dont les enseignant·e·s s'approprient de manière créative les outils numériques pour répondre à leurs besoins pédagogiques, qu'ils soient ou non prévus par les conceptrices ou concepteurs des technologies. Pour analyser la portée pédagogique de ces usages nous les mettrons en perspective avec les concepts d'*alternance*, d'*inclusion* et d'*accompagnement*, issus de la littérature sur la formation des enseignant·e·s (Runtz-Christan et Coen, 2021), qui serviront de grille de lecture pour interpréter les résultats.

3. Questions de recherche

Pour guider notre exploration des usages numériques des enseignant·e·s de musique observé·e·s *in situ* et en lien avec le cadre théorique exposé précédemment, cette recherche s'articule autour des trois questions suivantes :

- Comment l'enseignant·e de musique a-t-il·elle découvert les outils numériques et quelle relation déclare-t-il·elle entretenir avec eux ?
- Quels sont les usages numériques effectivement mis en œuvre par ces enseignant·e·s dans leur pratique pédagogique quotidienne ?
- Quelles sont les implications ou la valeur ajoutée de ces usages dans le cadre de l'enseignement de la musique, notamment au regard des concepts d'alternance, d'inclusion et d'accompagnement identifiés dans notre analyse ?

4. Méthode

Ce travail de recherche adopte une approche méthodologie qualitative. Cette dernière est considérée comme la plus adaptée pour découvrir et comprendre en profondeur les usages numériques en contexte, qui, la plupart du temps, sont individuels, situés et complexes (voir par exemple Perriault, 2008 ; Rabardel, 1995), permettant ainsi de saisir les processus d'appropriation technologiques par les acteur·trice·s eux·elles-mêmes. La collecte des données a combiné des entretiens semi-dirigés pour comprendre les trajectoires et les perceptions des enseignant·e·s, avec une phase d'observation participante pour saisir les usages en action. Les verbatims ont ensuite fait l'objet d'une analyse de contenu thématique (Bardin, 2007), alliant une approche inductive et déductive, à l'aide du logiciel MAXQDA24.

4.1 Participant·e·s et recrutement

Nous nous sommes entretenu·e·s avec onze enseignant·e·s de musique exerçant dans divers conservatoires et hautes écoles de musique de Suisse romande. Tous·te·s déclarent recourir régulièrement aux outils numériques dans leurs pratiques pédagogiques, ce qui constituait le critère d'inclusion principal pour cette étude. Ces personnes ont été recrutées par cooptation ainsi que par le biais d'annonces sur le réseau social Instagram.

Afin d'obtenir une diversité des perspectives et la palette d'usages la plus large possible, nous avons cherché à diversifier notre échantillon selon l'esthétique musicale enseignée (musique classique *vs* jazz et musiques actuelles) et le type de cours dispensé (enseignement instrumental *vs* enseignement des branches théoriques en groupe). Nous formulions l'hypothèse que ces différences contextuelles (esthétique, enseignement individuel *vs* collectif (Héroux et Lambert-Chan, 2017)) pouvaient influencer la relation au numérique et les usages développés.

Ce sont finalement sept hommes et quatre femmes qui ont accepté de participer à notre enquête. Bien que l'âge n'ait pas été un critère de sélection principal et n'ait pas été systématiquement recueilli, l'échantillon couvre implicitement plusieurs générations. Six répondant·e·s enseignent principalement dans le domaine classique, les autres relevant du cursus « jazz et musiques actuelles ». Cinq enseignant·e·s dispensent des cours collectifs théoriques et six des cours d'instrument individuels. Le tableau 1 présente le profil anonymisé de chaque répondant·e.

Tableau 1 : Profil des répondant·e·s

Sujet	Genre	Instrument enseigné	Esthétique	Institution d'enseignement
BH	Homme	Composition pour l'image	Classique	Conservatoire
JT	Homme	Musique assistée par ordinateur	Jazz et musiques actuelles	Conservatoire
BV	Femme	Piano	Classique	Conservatoire
PC	Femme	Solfège	Classique	Haute école de musique
CA	Homme	Histoire, harmonie, contrepoint, solfège	Classique	Haute école de musique
HM	Femme	Chant jazz	Jazz et musiques actuelles	Conservatoire
BC	Homme	Piano jazz	Jazz et musiques actuelles	Conservatoire
SM	Homme	Flûte jazz	Jazz et musiques actuelles	Conservatoire
HB	Homme	Théorie musicale	Classique	Haute école de musique
TM	Femme	Violon	Classique	Conservatoire
GL	Homme	Batterie	Jazz et musiques actuelles	Conservatoire

4.2 Collecte des données

La collecte des données s'est déroulée individuellement avec chaque participant·e et comportait deux phases complémentaires, pour une durée totale d'environ 90 minutes en moyenne.

- Un entretien semi-dirigé, guidé par une grille thématisée, visait à mieux comprendre la trajectoire personnelle de l'enseignant·e avec le numérique, la relation subjective qu'il·elle entretient avec les outils numériques, son parcours de découverte, ses usages déclarés, les raisons de ses choix et sa perception de leur impact pédagogique.
- Une phase d'observation participante ou de démonstration guidée a ensuite été menée dans son milieu pédagogique habituel (studio de musique, environnement numérique d'apprentissage) ou via le partage d'écran, afin de saisir les usages en action, d'observer l'interaction avec les outils en contexte pédagogique (parfois lors d'une interaction de type leçon avec un·e élève lorsque cela était possible et autorisé) et de confirmer les discours recueillis aux pratiques effectives. Certaines sessions ont été réalisées à distance (avec l'outil de téléconférence *Zoom*) alors que d'autres se sont déroulées en présentiel. Chaque entretien et séance d'observation/démonstration a été intégralement enregistré (format audio mp3 ou vidéo mp4 selon les cas) puis retranscrit verbatim avec l'aide de l'application Aiko. Le consentement éclairé des participant·e·s a été recueilli avant toute collecte, garantissant la confidentialité et l'anonymisation des données.

4.3 Analyse des données

Les quelques 500 pages de verbatims issus des entretiens, complétées par nos notes d'observation, ont été importées dans le logiciel d'analyse qualitative assistée par ordinateur MAXQDA24. Nous avons procédé à une analyse de contenu thématique, inspirée notamment de Bardin (2007) et Quivy et Van Campenhoudt (2006), combinant approche inductive (laissant émerger les catégories des données) et déductive (en lien avec notre cadre théorique et nos questions de recherche). Le processus s'est déroulé en plusieurs étapes : (a) lectures répétées du corpus ; (b) codage ouvert des segments pertinents pour identifier les thèmes émergents liés à la découverte du numérique, aux outils, aux usages, aux effets perçus, etc. ; (c) regroupement des codes en catégories et sous-catégories thématiques ; (d) développement d'un système de codes hiérarchisés et structurés dans MAXQDA24. Ce travail a permis d'identifier 27 usages technopédagogiques (cf. Tableau 2) et les 3 profils de découverte présentés dans les résultats (cf. Section 5.1). Par ailleurs, les enregistrements vidéo disponibles ont fait l'objet d'une analyse séquentielle spécifique (vidéographie), inspirée de Perriault (2008) et détaillée en section 5.3, visant à décrire finement l'interaction entre l'enseignant·e, l'élève (le cas échéant) et l'artefact numérique dans des situations pédagogiques concrètes, afin d'enrichir la compréhension des logiques d'usage (bricolage, braconnage, bidouillage) à l'œuvre.

5. Résultats

Cette section présente, en réponse à nos questions de recherche, les résultats de notre analyse. Dans un premier temps, nous décrivons comment les onze enseignant·e·s interrogé·e·s ont découvert les outils numériques (réponse à Q1). Dans un second temps, nous proposons une analyse des 27 usages technopédagogiques identifiés (réponse à Q2), que nous lions ensuite aux concepts pédagogiques d'alternance, d'inclusion et d'accompagnement (élément de réponse à Q3). Enfin, nous approfondissons la compréhension de certains usages via l'analyse de séquences vidéo (réponse à Q2 et Q3).

5.1 La rencontre avec le numérique : trois profils

Nous estimons que la rencontre avec le numérique peut avoir des implications directes sur les usages, notamment sur la manière dont les enseignant·e·s s'approprient les outils et les intègrent à leur pédagogie. Aussi, nous avons questionné les enseignant·e·s à propos de leur découverte des outils numériques. Trois profils principaux permettent de caractériser cette découverte : (1) la rencontre avec le numérique par intérêt et curiosité personnelle, relevant d'une démarche autodidacte ; (2) la rencontre avec le numérique dans le cadre des études, *via* une formation institutionnelle et (3) l'omniprésence du numérique dans sa vie, caractéristique des *digital natives*.

5.1.1 Profil 1 : La rencontre par intérêt et curiosité (Autodidactes)

Pour six enseignant·e·s (HB – CA – PC – BC – SM – BV), l'informatique a toujours éveillé une curiosité et un intérêt. Ces personnes ont vécu leur adolescence entre les années 1980 et 2000, moment de leur rencontre avec l'ordinateur. Leurs apprentissages ont essentiellement été de nature autodidacte. Elles ont ainsi développé des compétences relevant souvent du bricolage (Lévi-Strauss, 1990), parfois du braconnage (de Certeau, 1980) ou plus rarement du bidouillage (Larribeau, 2019). Les usages peuvent être qualifiés d'artisanaux : on crée du contenu dans un univers clos avec les outils qu'on a à sa disposition (bricolage), on échange avec des collègues sur les trucs et astuces, on se donne des conseils, on s'entraide (braconnage) ou, dans certains cas, on modifie la nature des outils pour les adapter aux besoins des élèves (bidouillage). On accepte un incertain technologique, on admet que l'effet escompté, parfois, ne se réalise pas :

- *J'ai grandi avec des ordinateurs où la moitié des trucs ne fonctionnait pas. Il fallait toujours bidouiller pour les faire fonctionner ; c'est pas encore aussi user-friendly le Windows des années 2000. Et donc, j'étais occupé à gérer beaucoup de problèmes. (HB)*
- *La plupart des pratiques que j'ai essayées de développer après viennent plutôt d'une réflexion personnelle du fait que j'ai toujours moi-même été intéressé par, disons, les outils numériques et, oui, j'ai pas l'impression d'avoir tellement eu l'occasion de les utiliser dans ma formation. (CA)*
- *Encore actuellement, je fonctionne par essai-erreur. (BV)*
- *Je me suis intéressé au système MIDI dès 1988 ; on avait de vieux ordinateurs et il fallait faire preuve de créativité. J'ai trouvé tout seul, enfin, j'ai découvert tout seul. (SM)*

Les usages développés par ces enseignant·e·s se trouvent encore grandement impactés par leur découverte du numérique. Ces dernier·ères n'ont pas peur de *mettre les mains dans le cambouis*, de chercher par eux·elles-mêmes et de proposer à leurs élèves une variété élargie d'outils numériques (logiciels, applications, développement de site web, etc.), faisant preuve d'une certaine autonomie et résilience technologique.

5.1.2 Profil 2 : La rencontre dans le cadre des études (Formation institutionnelle)

Quatre enseignant·e·s ont directement été confronté·e·s à une formation au numérique dans le cadre de leurs études (HM – GL – JT – BH). Dans l'ensemble, ces enseignant·e·s sont plus jeunes. L'orientation de leurs études musicales a probablement eu un impact (jazz et musiques actuelles pour HM et GL; composition pour BH et musique assistée par ordinateur[3] pour JT) puisque ces disciplines intègrent depuis de nombreuses années les outils numériques dans leurs curriculums. Nous observons que ces dernier·ère·s ont opéré un transfert des connaissances acquises dans le cadre de leur formation (la formation au numérique étant essentiellement orientée vers la performance et l'interprétation musicale) vers leurs pratiques pédagogiques. Il·elle·s recourent aux mêmes outils (ou des outils similaires), mais dans une finalité pédagogique :

[2] *Musical Instrument Digital Interface*

[3] *MAO*

- *J'ai étudié en Norvège pour mon master, et ça fait trente ans qu'il y a des masters en technologie musicale. Dans la technologie musicale, on est très en retard […] je voudrais que [la MAO] soit un peu crédibilisée. Il faudrait que ce soit vu aussi comme une branche. (JT)*
- *En fait, dans mes cours de composition, c'était déjà comme ça. On avait déjà un rapport avec la partition directement sur l'ordinateur pour l'édition, mais aussi pour la synchro avec l'image. La programmation aussi, parce que j'avais des cours de MAO. Donc, en fait, tous les outils que j'utilise avec mes élèves, c'est des outils que j'ai découverts quand j'étais en études, quasiment. (BH)*
- *Quand je crée de la musique, je vais utiliser les outils numériques parce que j'utilise un logiciel sur mon ordinateur, j'utilise un clavier MIDI, enfin tout un dispositif pour enregistrer. Mais tu vois aussi maintenant, quasiment tous les élèves ont des téléphones portables. Donc en fait, on utilise beaucoup, comme ça ils peuvent enregistrer leurs cours et réécouter et avoir des supports. Et je vois qu'en fait, ça devient vraiment un outil de travail hyper essentiel avec des choses toutes basiques comme le dictaphone pour s'enregistrer et se réécouter. Moi aussi je m'enregistre avec mon téléphone. (HM)*

Les usages *professionnels* développés dans le cadre de la formation initiale et/ou continue de ces enseignant·e·s ont été transférés dans le contexte de leur enseignement. Nous remarquons que les usages tournent essentiellement autour de l'audio (l'enregistrement pour le travail individuel, pour garder une trace de l'avancement du travail ou encore pour collectionner ses prestations). Les enseignant·e·s de cette catégorie restent dans leur zone de confort et de maîtrise, mobilisant des outils dont il·elle·s connaissent bien les effets liés aux usages, souvent dans une perspective d'efficacité et de support à la pratique instrumentale ou créative.

5.1.3 Profil 3 : L'omniprésence du numérique dans sa vie (Digital natives[4])

La dernière enseignante (TM), la plus jeune de l'échantillon, admet baigner dans le numérique depuis plusieurs années. Comme pour ses collègues du profil 2, elle effectue un transfert d'usages personnels liés aux technologies mobiles vers son enseignement :

- *C'est marrant parce que les outils numériques vont avec notre génération. Moi, je considère que photo, vidéo, tout ça, c'est des outils numériques. Et du coup, je les utilise naturellement depuis le début. Et puis, après, j'étais toujours très intéressée par tous les petits jeux, les applications. Dès que je découvre un truc, je le teste avec mes élèves parce que j'aime bien tout tester. Et j'allais dire un dernier truc. Oui, ma formation Suzuki[5] m'a poussée à utiliser les outils numériques. (TM)*

Les usages de cette enseignante sont essentiellement construits sur son intérêt intrinsèque pour les applications numériques qu'elle collectionne et teste elle-même et avec ses élèves via un smartphone. En effectuant une veille technologique, elle se tient au courant des nouveautés qu'elle pourrait proposer à ses élèves les plus jeunes. Ses usages se concentrent sur le smartphone : utilisation d'un cloud, applications, photo, audio. Il y a une orientation résolument ludique, exploratoire et axée sur la communication et la motivation dans ses choix et ses options.

5.2 Analyse des usages technopédagogiques

Au-delà des profils de découverte, l'analyse thématique des entretiens et des observations a permis d'identifier un répertoire de 27 usages technopédagogiques distincts, mobilisant divers appareils et outils. Les appareils numériques les plus utilisés par les répondant·e·s sont : l'ordinateur, le smartphone et la

[4] En employant le terme digital natives, les auteur·e·s souhaitent simplement différencier les profils des participant·e·s en fonction de leur familiarité générationnelle avec les technologies numériques, sans pour autant s'inscrire dans le débat critique (Bennett et al., 2008) opposant digital natives et digital immigrants (Prensky, 2001). Ce terme est ici utilisé comme une commodité de langage pour caractériser une expérience vécue d'immersion précoce dans l'environnement numérique et non pour valider les théories sous-jacentes sur d'éventuelles différences cognitives ou compétences innées.

[5] « La méthode Suzuki est une philosophie de l'apprentissage employée notamment pour celui de la musique. Elle a été développée au milieu du 20e siècle par Shinichi Suzuki. Les principes fondamentaux de cette méthode sont de commencer très tôt, de se baser sur le principe de l'apprentissage de la langue maternelle, de soigner le répertoire, de donner des cours en groupe et d'inviter les parents aux leçons individuelles » (Wikipédia, 2024).

tablette. Ils permettent de recourir à différents outils qui viennent s'y greffer : applications mobiles, sites internet (développement et consultation), enregistrements audio et vidéo, environnements numériques d'apprentissage, logiciels d'édition musicale, appareils connectés, logiciels dédiés, etc.

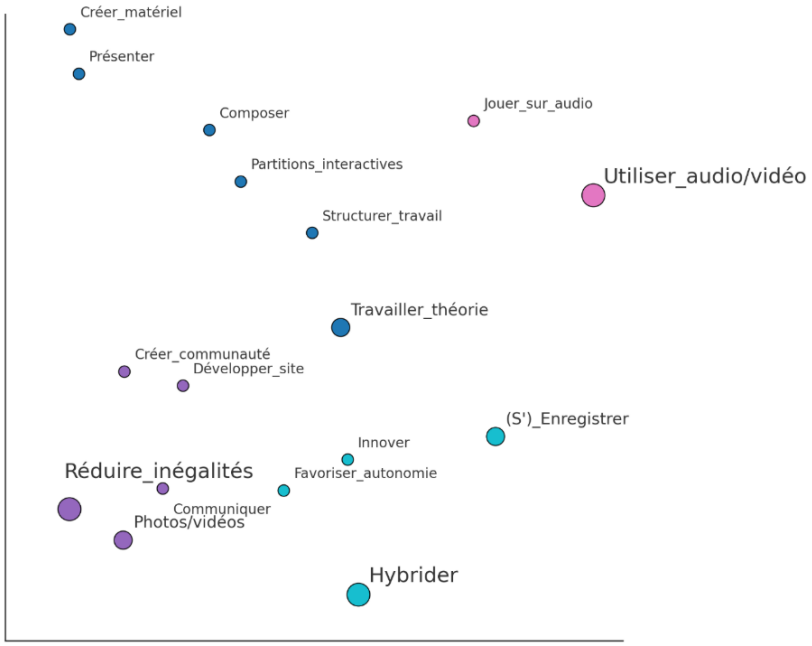
Le tableau 2 présente ces 27 usages ainsi que la fréquence d'apparition dans les entretiens par ordre décroissant (c'est-à-dire le nombre d'enseignant·e·s ayant mentionné chaque usage). Bien que certains usages puissent se recouper ou s'inscrire dans des catégories plus larges (par exemple, *prendre des photos* peut faire partie d'*utiliser l'audio et la vidéo pour apprendre*), cette liste a été conservée telle quelle pour refléter la richesse et la variété des termes employés par les participant·e·s pour décrire leurs pratiques.

Tableau 2 : Usages technopédagogiques identifiés et fréquence d'apparition

Usage	Fréquence
Utiliser l'audio et la vidéo pour apprendre	8
Hybrider l'enseignement	7
Réduire les inégalités	6
Enregistrer ses élèves, s'enregistrer	5
Prendre des photos, faire des vidéos	5
Développer un site web pour l'apprentissage	5
Travailler la théorie musicale	4
Favoriser l'autonomie	4
Utiliser l'audio pour jouer	4
Utiliser des playbacks	4
Utiliser des partitions interactives	4
Créer du matériel pédagogique	3
Économiser du papier	3
Éditer de la musique	3
Présenter (des notions)	3
Structurer le travail	3
Communiquer	3
Innover	3
Suivre le travail de l'élève	3
Gérer (les partitions, le travail)	2
Composer de la musique dans le cadre du cours	2
Garder la pulsation	2
Créer une communauté	2
Développer la créativité	2
Travailler l'instrument	2
Partager des pratiques pédagogiques	1
Individualiser	1

La figure 1 présente la Carte de codes générée par le logiciel MAXQDA24, illustrant les relations entre les principaux codes d'usages. Dans cette représentation, la taille des cercles est proportionnelle à la fréquence de chaque code (usage) et leur proximité sur le graphique suggère une co-occurrence fréquente dans les discours des enseignant·e·s, indiquant ainsi des liens thématiques forts. Les couleurs regroupent des usages conceptuellement proches. *Utiliser l'audio ou la vidéo pour apprendre* y occupe une place centrale, suggérant que les enseignant·e·s développent et fondent souvent leurs usages du numérique pédagogique *via* cette porte d'entrée. L'hybridation de l'enseignement semble aussi occuper une place importante dans les usages et joue un rôle clé dans l'innovation pédagogique, favorisant l'autonomisation de l'élève. Certain·e·s enseignant·e·s tendent vers une réduction des inégalités en développant plusieurs usages : communication avec les élèves, exploitation de traces (photos, vidéos, sites internet). Cette figure nous indique aussi que les outils numériques ne sont pas uniquement utilisés pour des aspects pratiques et créatifs (au niveau de la pratique musicale), mais qu'ils jouent aussi un rôle dans l'apprentissage des branches théoriques.

Figure 1 : Carte de codes



Afin d'analyser la portée pédagogique de ces usages et de répondre à notre troisième question de recherche et considérant que notre axe professionnel et de recherche est orienté vers la formation des enseignant·e·s, nous avons tenté d'organiser ces usages autour de concepts-clés liés à ce domaine (Runtz-Christan et Coen, 2021). Le tableau suivant propose ainsi une classification structurée de ces usages. Nous estimons qu'une telle mise en relation pourrait stimuler la réflexion autour de la formation des enseignant·e·s à l'usage des technologies numériques. Le tableau 3 présente la correspondance établie entre les usages les plus fréquemment mentionnés et les trois concepts-clés : l'alternance, l'inclusion et l'accompagnement.

Tableau 3 : Correspondance entre les usages les plus mentionnés et les concepts-clés de la formation des enseignant·e·s (Runtz-Christan et Coen, 2021)

Usage	Lien avec les concepts-clés
Hybrider l'enseignement ; Utiliser l'audio ou la vidéo pour apprendre ; Enregistrer ses élèves, s'enregistrer ; Utiliser des partitions interactives ; Communiquer ; Travailler l'instrument ; Travailler la théorie musicale	Alternance
Réduire les inégalités ; Partager les pratiques pédagogiques ; Créer une communauté	Inclusion
Prendre des photos, faire des vidéos ; Favoriser l'autonomie ; Structurer le travail ; Suivre le travail de l'élève ; Individualiser ; Gérer (les partitions, le travail)	Accompagnement

Une analyse croisée montre que ces concepts-clés ne sont pas exclusivement attachés à un profil d'enseignant particulier. Cependant, des tendances se dessinent : les profils 2 (*formé·e·s institutionnellement)* et 3 (*digital natives*) semblent particulièrement mobiliser les usages liés à l'alternance (par exemple, HM, JT, TM), en s'appuyant sur des outils qu'elles ou ils maîtrisent pour créer un *continuum* entre le cours et le domicile. L'inclusion est une préoccupation partagée, mais les stratégies varient : les Profils 1 (*autodidactes*) comme PC et BH créent des ressources de A à Z (sites web), tandis que les autres profils adaptent des outils existants. Enfin, l'accompagnement *via* le numérique est une pratique transversale, bien qu'illustrée de manière particulièrement créative par des enseignant·e·s des Profils 1 (BC, BV, PC) et 3 (TM).

5.2.1 L'alternance : Déplacer l'apprentissage hors les murs

Les usages analysés révèlent que l'intégration du numérique donne la possibilité de créer des activités de formation favorisant l'alternance entre différents temps et lieux d'apprentissage. Ces diverses activités permettent aux élèves de *déplacer* la leçon en-dehors du studio de musique. L'alternance se caractérise ici par une extension temporelle et spatiale de l'interaction pédagogique.

Dans sa forme habituelle, l'enseignement de la musique est structuré en deux étapes : le cours (qui est d'une durée générale de 30 à 60 minutes hebdomadaires) et le travail à domicile. Le paradoxe est que la part essentielle de l'apprentissage se déroule de façon indépendante, à la maison (Batézat-Batellier et Bourg, 2020). L'élève doit s'assurer qu'il·elle a bien compris ce qui était attendu de lui·d'elle et qu'il·elle soit en mesure d'y travailler durant la semaine. Dans le cas d'une alternance facilitée par le numérique, la formation se déroule dans plusieurs lieux (Balslev, 2021), notamment entre l'école de musique et le domicile de l'élève. Il y a une réduction du paradoxe identifié par Batézat-Batellier et Bourg (2020) puisque l'élève peut avoir accès à différentes formes de traces numériques (par exemple, visionner une vidéo contenant une partie de la précédente leçon ; consulter un site internet ; contacter son enseignant·e ; recevoir un tutoriel vocal de son enseignant·e, etc.). Pour TM, questionner l'alternance lui a permis de se questionner sur son métier d'enseignante : *Avant j'avais un enseignement où j'étais plus dans l'efficacité, genre il faut faire ça, ça dans la leçon. Je faisais des leçons avec beaucoup de séquences et maintenant j'essaie plus de faire en sorte qu'ils soient capables à la maison d'être efficaces*. HM a découvert les outils numériques dans sa formation et sa pratique de chanteuse de jazz et leur adaptation à l'enseignement sert de support à l'alternance : *Mes élèves, il faut qu'ils aient de la matière sonore pour s'aider, pour travailler chez eux à la maison […]. Je leur faisais des instrumentaux, j'accompagnais au piano et puis je leur envoyais sur leur téléphone*. Dans la même idée, BH conseille à ses élèves de travailler avec des partitions interactives pour prolonger le travail élaboré dans le cadre du cours : *Elles permettent d'accélérer ou de ralentir le tempo*. JT, quant à lui, demande à ses élèves d'installer sur leur smartphone un logiciel d'enregistrement et de création musicale pour poursuivre leur travail à la maison.

5.2.2 L'inclusion : Réduire les inégalités et créer du lien

Plusieurs usages numériques identifiés tendent vers une réduction des inégalités et une inclusion des élèves, essayant de garantir une « éducation pour tous » (Gremion, 2021, p. 144). PC a expérimenté, il y a une vingtaine d'années, la création de sites internet d'accompagnement au solfège pour les élèves ne bénéficiant pas d'un entourage familial connaissant la musique : *Le solfège et les problèmes qui se posaient fréquemment, c'est les enfants qui n'avaient pas de parents musiciens ou de grands frères, grandes sœurs musiciens et qui étaient un peu livrés à eux-mêmes pour travailler. Certains travaillaient avec ça tous les jours*. Pour BH, le recours aux outils numériques dans le cadre de ses enseignements de composition permet de démocratiser une pratique pouvant être jugée élitiste : *Ça rend la composition beaucoup plus accessible à tous. Sans ça, combien de personnes tu laisses sur le carreau ?* Plus traditionnellement, les outils numériques peuvent servir à différencier ou individualiser une partie de l'enseignement : *J'utilise encore beaucoup la communication numérique pour les élèves qui ont besoin d'un peu plus de suivi ou qui ont énormément de morceaux* (TM) ; *Sur mon site internet, chaque élève a une page qui lui est réservée* (SM). Pratiquer l'inclusion dans une institution peut aussi se manifester par le partage et l'entraide entre collègues, y compris ceux maîtrisant moins les outils numériques : *On a réuni tous les profs, y compris ceux qui n'étaient pas trop friands de technologie et on leur a dit : maintenant on a envie de faire un site avec tous les profs de culture musicale et ceux qui sont largués en informatique, pas de souci, on le fait et on vous explique après ; vous choisissez si vous voulez en faire partie ou pas. En fait, tous les profs quasiment se sont greffés dessus. Ils ne mettent pas forcément des devoirs, mais ils peuvent mettre des liens, par exemple des musiques qu'ils font ensemble en cours* (BH). En somme, ces usages montrent que le numérique est mobilisé pour réduire les inégalités d'accès au savoir musical, pour différencier l'enseignement et pour renforcer la cohésion au sein de la communauté éducative.

5.2.3 L'accompagnement : Soutenir l'autonomie et le cheminement de l'élève

Les usages numériques recourant à la prise de photo et/ou de vidéo ainsi que toutes les activités d'autonomisation de l'élève se réfèrent directement au concept d'accompagnement, qui permet à l'élève « de construire son propre trajet de façon à favoriser son autonomisation et son émancipation » (Rebetez, 2021, p. 6). Ici, l'accompagnement se distingue de l'alternance en se focalisant spécifiquement sur la mise à disposition de ressources personnalisées visant l'autocorrection et l'autonomisation de l'élève dans sa pratique individuelle. Les entretiens mettent en évidence que les usages développés avec la photo et/ou la vidéo encouragent l'élève à définir son propre cheminement : *Je fais ça aussi parfois avec certains élèves, pour les positions des mains, quand ils ont du mal à lire les numéros, alors je prends mon smartphone, dès qu'ils arrivent à les mettre bien, je les prends en photo et je les envoie, comme ça, c'est leur main et puis ils ont plus de… Ils connaissent mieux* (BC). D'autres dispositifs visent à fournir des ressources pour le travail autonome : *Les élèves peuvent aller sur mon site et voir, par exemple, mes listes de lecture sur YouTube, mes listes de lecture sur Spotify* (BV). *J'ai fait un petit site comme ça, avec des exercices qui étaient enregistrés, donc on pouvait cliquer sur l'exercice, l'écouter, le répéter en même temps* (PC). Ces démarches mettent en évidence que le numérique est considéré comme un outil mis à disposition de l'élève par son enseignant·e de musique. In fine, c'est à l'élève de choisir s'il·elle utilise ces dispositifs mis en place car « l'accompagnement est un choix de l'accompagné dans lequel il est libre d'aller où il le souhaite » (Rebetez, 2021, p. 7).

5.3 Analyse de situations filmées : Les usages en action

De manière générale, les observations confirment les usages rapportés dans les entretiens. Les outils les plus fréquemment observés en action sont le smartphone (utilisé comme appareil photo, enregistreur, métronome), la tablette (pour les partitions interactives) et l'ordinateur (sites web, exercices d'écriture et d'analyse). L'interaction est quasi systématiquement centrée sur la co-construction entre l'enseignant·e et l'élève autour de l'*artefact* numérique. L'enseignant·e agit comme un·e médiateur·trice, manipulant l'outil pour illustrer un propos, tandis que l'élève est invité à interagir avec le résultat. Ces usages, intégrés de manière fluide dans la leçon, relèvent majoritairement du bricolage et du braconnage et visent des objectifs très concrets : fournir un *feed-back* visuel et/ou auditif, donner un support pour le travail à domicile ou clarifier un point théorique.

Afin de prolonger la compréhension des usages technopédagogiques décrits dans le cadre des entretiens et d'illustrer concrètement les logiques d'appropriation, nous avons étudié quatre séquences extraites des enregistrements vidéo réalisés en présentiel et à distance. Chaque vidéogramme a été visionné intégralement avant d'isoler des extraits emblématiques permettant de décrire le contexte matériel, les gestes de l'enseignant·e, la nature de l'interaction et les effets pédagogiques observés *in situ*. Cette démarche d'analyse tend à rendre tangible ce que Perriault (2008) appelle « la logique de l'usage », c'est-à-dire la manière dont un artefact numérique se stabilise et se transforme, dans notre cas, en pratique éducative réelle, mobilisant les concepts de bricolage, braconnage et bidouillage.

Dans la première séquence filmée en studio de musique (Enregistrement_01), BC dédie son smartphone à la documentation photographique de la posture de la main gauche de son élève. À l'instant même où la position correspond à l'idéal, elle déclenche l'appareil et transmet immédiatement la photo à l'élève via messagerie : *je prends mon smartphone, dès qu'ils arrivent à les mettre bien, je les prends en photo et je les envoie*. Ce geste, simple en apparence, relève du bricolage au sens de Lévi-Strauss (1990) : l'appareil grand public devient un instrument pédagogique, personnalisant l'accompagnement et inscrivant visuellement l'information kinesthésique dans la mémoire de l'élève. Il·elle peut alors ultérieurement relire et comparer les clichés, favorisant l'autonomie et la métacognition dans le travail à domicile.

Dans la deuxième séquence étudiée (Enregistrement_03), SM utilise une montre métronome connectée pour faire résonner physiquement un tempo lent (noire = 60) dans le corps de l'élève. Une fois le rythme paramétré, SM encourage son élève à *caler ses battements et sensations* sur la vibration générée, notant que *si on la met pour des tempos longs, ça peut être pratique*. Cette configuration illustre le braconnage tel que défini par de Certeau (1980) : un usage collectif et créatif d'un instrument technologique, non pas pour son objet premier, mais pour renforcer la sensation corporelle de la pulsation. Par là même, l'apprentissage de la métrique musicale s'enrichit d'une dimension kinesthésique inédite et l'élève bénéficie exactement du même feedback haptique lorsqu'il·elle réutilise l'appareil chez lui·elle, soulignant l'importance de l'alternance entre temps de cours et pratique autonome.

La troisième séquence, filmée à distance (Enregistrement_04), présente BH manipulant une partition interactive sur tablette via le logiciel *MuseScore*. En pleine démonstration, elle ralentit le tempo pour expliciter un phrasé, puis enchaîne des boucles successives afin de focaliser l'attention de l'élève sur un passage délicat. Comme le souligne BH, *les partitions interactives permettent d'accélérer ou de ralentir le tempo*. Ce processus s'apparente au bidouillage selon Larribeau (2019) : l'enseignant·e adapte un outil de notation destiné initialement à l'édition musicale pour qu'il serve directement des finalités pédagogiques ciblées. L'élève peut ensuite accéder à la même interface à domicile, opérant un transfert spontané des compétences développées en cours vers son environnement personnel, facilitant là encore l'alternance et le travail autonome.

Enfin, lors d'un cours de théorie musicale à distance (Enregistrement_02), CA diffuse un document PDF de partition via *Zoom*, puis utilise l'outil d'annotation pour surligner en direct les motifs harmoniques et schématiser les enchaînements. Enregistrant l'ensemble de la leçon, CA indique que *les étudiants allophones apprécient de pouvoir ralentir la lecture et activer la traduction automatique*. Ici, l'intégration du numérique fait écho à la notion d'inclusion : la ressource vidéo annotée devient un objet réutilisable et modulable, offrant aux étudiant·e·s la possibilité de gérer leur rythme de révision et de contourner les barrières linguistiques. Cet usage combine une forme de bricolage (détournement de l'outil d'annotation) et d'accompagnement personnalisé.

Ces quatre illustrations confirment la prééminence de l'audio et de la vidéo comme supports d'apprentissage et soulignent l'importance de l'hybridation pour créer un *continuum* entre le temps de cours et l'exercice

en autonomie. Au-delà de la simple validation des usages déclarés, l'observation des situations filmées enrichit notre compréhension de la dynamique technopédagogique. Elle met en lumière la co-construction en continu entre enseignant·e et élève, où bricolage, braconnage et bidouillage se conjuguent pour transformer des artefacts numériques en instruments d'apprentissage.

6. Discussion

L'articulation entre les entretiens semi-dirigés et l'analyse des séquences filmées permet de proposer une lecture interprétative et plus fine des usages numériques en éducation musicale, en dépassant la seule dimension déclarative pour prendre en compte « la logique de l'usage » telle que l'a décrite Perriault (2008). À travers l'étude conjointe des verbatims et des vidéogrammes, et en réponse à nos questions de recherche sur les implications pédagogiques (Q3), trois dimensions principales se dégagent : l'inclusion par le numérique, l'hybridation des apprentissages et la nature créative des processus d'appropriation. Nous abordons également une perspective prospective sur les outils émergents comme l'IA.

6.1 Le numérique comme vecteur d'inclusion

Nos résultats suggèrent que les outils numériques peuvent effectivement constituer un levier pour favoriser l'inclusion en éducation musicale. Les entretiens ont montré comment les sites de révision de PC ou les ressources de BH démocratisent l'accès à la théorie et à la composition pour les publics variés, y compris ceux issus de milieux peu familiers avec la musique. L'observation des capsules *Zoom* annotées confirme cette tendance : CA a souligné que la possibilité pour les étudiant·e·s allophones de ralentir la vidéo et d'activer la traduction automatique améliore non seulement leur compréhension, mais suscite également un sentiment d'appropriation et de confiance accrue. Cette modalité, en transformant la séance en ressource pérenne et modulable, illustre pleinement l'objectif « d'éducation pour tous » (Gremion, 2021) et renforce la réduction des inégalités linguistiques et cognitives dans l'enseignement musical.

Par ailleurs, la prise de vue instantanée de la posture corporelle par BC répond à un besoin d'accompagnement personnalisé : en envoyant la photographie de la main de l'élève dès qu'elle atteint la position idéale, l'enseignante offre un support visuel mémorable, prolongeant la leçon au-delà du studio. Ce retour immédiat, loin de se limiter à une simple illustration, devient un levier d'inclusion et d'accompagnement pour les élèves qui peuvent comparer leurs propres clichés, s'auto-corriger et conserver une trace tangible de leurs progrès. Ces exemples montrent comment des usages numériques, parfois simples, peuvent contribuer à rendre l'enseignement plus accessible et adapté à chacun·e.

6.2 La formation hybride : un apprentissage qui transcende le studio de musique, répondant au besoin d'alternance

Les pratiques d'alternance entre présentiel et distanciel observées chez HM et TM montrent que l'apprentissage musical peut s'étendre naturellement dans différents espaces temporels et physiques (Balslev, 2021). L'envoi de tutoriels audio sur smartphone ou la mise à disposition d'extraits vidéo synchronisés participent à ce *continuum* pédagogique, favorisant la régularité et la motivation des élèves. Ces constats entrent en résonance avec la littérature internationale qui documente ces modalités d'apprentissage flexibles et distribuées, rendues possibles par les technologies mobiles et les plateformes en ligne (Acquilino et Scavone, 2022). L'expérience de SM, avec montre métronome connectée, renforce ce sentiment en ancrant la pulsation dans le corps grâce à une vibration tactile ; l'élève dispose d'un feedback identique en cours et à domicile. Cette modalité hybride, à la croisée des usages déclarés et observés, répond à la nécessité de réduire le paradoxe entre temps de leçon et travail autonome, souvent pointé dans l'enseignement instrumental (Batézat-Batellier et Bourg, 2020), en offrant un dispositif stable et reproductible.

L'hybridation ne se limite pas au transfert de fichiers : la création de ressources interactives sur tablette, pilotée par BH, permet de boucler les séquences de jeu et d'ajuster le tempo à la volée. Cette immersion progressive hors du cadre formel du studio illustre comment l'alternance peut être pensée non seulement en termes de lieu, mais aussi de temporalité et de modalités sensorielles, enrichissant ainsi les procédures d'apprentissage et favorisant potentiellement une plus grande autonomie de l'élève.

6.3 Bricolage, braconnage et bidouillage : des logiques d'appropriation créatives entre artisanat et institution

Les entretiens ont identifié trois profils d'enseignant·e·s (autodidactes, formé·e·s institutionnellement, *digital natives*), auxquels l'analyse vidéo apporte des illustrations saisissantes : le bricolage de BC, le

braconnage de SM et le bidouillage poussé par BH. Au-delà de l'illustration, la discussion de ces cas à la lumière des concepts de Lévi-Strauss (1990), de de Certeau (1980) et de Larribeau (2019) révèle la créativité déployée par les enseignant·e·s pour adapter les outils à leurs besoins pédagogiques spécifiques. Ces modalités d'appropriation montrent que la frontière entre l'outil et l'usage se redessine constamment : un smartphone, un fichier PDF ou une montre deviennent prétexte à des inventions pédagogiques, fruits d'une adaptation située, parfois en collaboration en temps réel entre enseignant·e et élève. Cela suggère une tension dynamique : d'une part, les pratiques *artisanales*, individuelles, relevant de l'initiative personnelle (souvent chez les autodidactes) ; d'autre part, des usages plus standardisés ou transférés de la formation (Profil 2), potentiellement encouragés ou limités par le cadre institutionnel. Le partage d'un site web entre collègues (BH) illustre une tentative de dépasser l'isolement artisanal vers une forme de mutualisation, relevant peut-être d'un *braconnage* institutionnel organisé. Comprendre ces différentes logiques est crucial pour penser la formation et l'accompagnement des enseignant·e·s.

6.4 Perspectives futures : intégrer l'intelligence artificielle et penser le feedback réflexif

L'étude relève que, malgré une appropriation étendue des technologies actuelles, les enseignant·e·s participant·e·s n'exploitent pas encore les potentialités de l'intelligence artificielle dans leurs pratiques pédagogiques. Or, l'IA pourrait théoriquement automatiser la détection de postures, l'analyse rythmique et la génération de feedbacks personnalisés en temps réel, ouvrant de nouvelles voies pour renforcer l'autonomie de l'apprenant·e. Les observations vidéo que nous avons collectées constituent déjà une base d'entraînement précieuse pour développer des algorithmes capables de reconnaître les gestes instrumentaux et de proposer des corrections instantanées.

Cependant, anticiper ces évolutions implique d'accompagner les enseignant·e·s non seulement dans la prise en main technique de tels outils, mais aussi et surtout dans la réflexion éthique et pédagogique sur leurs usages. Quel type de feedback l'IA peut-elle réellement fournir ? Comment s'articule-t-il avec le feedback humain et relationnel ? Comment éviter une standardisation excessive ou une déshumanisation de l'apprentissage ? Préserver la dimension humaine et réflexive de l'accompagnement devient alors un enjeu central.

7. Conclusion

En combinant entretiens semi-dirigés et observations *in situ*, cette recherche offre une vision approfondie et incarnée des usages numériques en éducation musicale. Les entretiens ont permis de cartographier une diversité d'usages – de la photographie des postures aux plateformes interactives – tandis que les observations *in situ* ont mis en évidence la manière dont ces usages se stabilisent, se détournent et se mutualisent en temps réel, suivant les logiques de bricolage (Lévi-Strauss, 1990), de braconnage (de Certeau, 1980) et de bidouillage (Larribeau, 2019).

Le numérique s'affirme ainsi comme un levier puissant potentiel d'inclusion, d'autonomie et d'hybridation des apprentissages, tout en posant des défis nouveaux en termes de formation continue et d'éthique technologique, comme l'a montré notre discussion. Parmi ces défis, l'anticipation de l'intelligence artificielle – capable de proposer des feedbacks instantanés et personnalisés – constitue une piste prometteuse mais nécessitant une réflexion critique pour la prochaine génération de technopédagogie.

Pour accompagner ces évolutions et favoriser une intégration réfléchie du numérique, il serait intéressant d'envisager le développement de :

- modules de formation hybride combinant ateliers pratiques et ressources numériques, axés sur l'analyse des usages et l'adaptation des outils plutôt que sur la seule maîtrise technique ;
- communautés de pratique pour partager, analyser et documenter les usages pédagogiques innovants et les défis rencontrés ;
- projets de recherche-action visant à évaluer l'impact des algorithmes d'IA sur les apprentissages musicaux et à co-construire des scénarios d'utilisation pertinents et éthiques.

En fin de compte, l'alliance des entretiens et de l'analyse vidéo trace la voie d'une technopédagogie réellement située, où chaque artefact numérique se redéfinit constamment au gré des usages, des contextes et des interactions entre enseignant·e·s et élèves.

Bibliographie

Acquilino, A. et Scavone, G. (2022). Current State and Future Directions of Technologies for Music Instrument Pedagogy. *Frontiers in Psychology, 13*.

Balslev, K. (2021). Alternance. Dans E. Runtz-Christan et P.-F. Coen (dir.), *Collection de concepts-clés de la formation des enseignantes et des enseignants en Suisse romande et au Tessin* (p. 13-15). Loisirs et pédagogie.

Bardin, L. (2007). *L'analyse de contenu.* Presses Universitaires de France.

Batézat-Batellier, P. et Bourg, A. (2020). Tablettes tactiles et apprentissage du piano. L'utilisation d'une application par de jeunes élèves lors de leur pratique instrumentale à la maison. *Distances et médiations des savoirs, 32*. https://doi.org/10.4000/dms.5782

Bennett, S., Maton, K. et Kervin, L. (2008). The digital natives debate: A critical review of the evidence. *British Journal of Educational Technology, 39*(5), 775-786.

Dakhlia, J. et Poels, G. (2012). Le minitel rose : Du flirt électronique… Et plus si affinités. Entretien avec Josiane Jouët. *Le Temps des Médias, 19*, 221-228.

De Certeau, M. (1980). *L'invention du quotidien. 1. Arts de faire.* Union Générale d'Edition.

Gonzalez, A. et Jouve, E. (2002). Minitel : Histoire du réseau télématique français. *Flux, 47*, 84-89.

Gremion, L. (2021). Inclusion scolaire. Dans E. Runtz-Christan et P.-F. Coen (dir.), *Collection de concepts-clés de la formation des enseignantes et enseignants en Suisse romande et au Tessin* (p. 144-147). Loisirs et pédagogie.

Héroux, I. et Lambert-Chan, L. (2017). L'évaluation des apprentissages instrumentaux en musique classique au Canada, et en Europe francophone. Dans D. Leduc et S. Béland (dir.), *Regards sur l'évaluation des apprentissages en arts à l'enseignement supérieur. Tome 1* (p. 95-114). Presses de l'Université du Québec.

Jouët, J. (2000). Retour critique sur la sociologie des usages. *Réseaux, 18*(100), 487-521.

Larribeau, A. (2019). Du bidouilleur amateur à l'informaticien, apprendre le détournement : Une étude de la socialisation du hacking informatique. *Sociologies pratiques, 38*, 59-70.

Latour, B. (1987). Les « vues » de l'esprit. *Réseaux, 5*(27), 79-96.

Latour, B. (2000). La fin des moyens. *Réseaux, 100*, 39-58.

Lévi-Strauss, C. (1990). *La pensée sauvage*. Pocket.

Maïdakoualé, I. (2021). Jacques Perriault, la logique de l'usage. Essai sur les machines à communiquer. *Communications, technologies et développement, 9*. https://doi.org/10.4000/ctd.4088

Maisonneuve, S. (2006). De la machine parlante au disque. Une innovation technique, commerciale et culturelle. *Presses de Sciences Po, 92*, 17-31.

Méadel, C. (2019). Une histoire de l'usager des technologies de l'information et de la communication (TIC). *Le Mouvement social, 268*, 29-44.

Paquienséguy, F. (2017). L'usage au fil des Tic. *Interfaces numériques, 6*(3), 464-481.

Perriault, J. (2008). *La logique de l'usage. Essai sur les machines à communiquer*. L'Harmattan.

Plantard, P. (2016). Usages des technologies numériques : Innovations et imaginaires. Dans P. Musso (dir.), *Industrie, imaginaire et innovation* (p. 57-68). Manucius.

Plante, P. (2016). Apprentissage, jeu sérieux et « détournement sérieux de jeu ». *Formation et profession, 24*(2), 72-74.

Prensky, M. (2001). Digital Natives, Digital Immigrants. *On the Horizon, 9*(5), 1-6.

Quivy, R. et Van Campenhoudt, L. (2006). *Manuel de recherche en sciences sociales*. Dunod.

Rabardel, P. (1995). *Les hommes et les technologies ; approche cognitive des instruments contemporains*. Armand Colin.

Rebetez, F. (2021). Accompagnement. Dans E. Runtz-Christan et P.-F. Coen (dir.), *Collection de concepts-clés de la formation des enseignantes et enseignants en Suisse romande et au Tessin* (p. 6-9). Loisirs et pédagogie.

Runtz-Christan, E. et Coen, P.-F. (2021). *Collection de concepts-clés de la formation des enseignantes et enseignants en Suisse romande et au Tessin*. Loisirs et pédagogie.

Schumacher, J. A. (2025). Le numérique pédagogique dans les Hautes écoles de musique de Suisse romande. Autoévaluation des compétences, Pratiques déclarées. *Revue musicale OICRM, 12*(1), 75-92.

Wikipédia. (2024). *Méthode Suzuki*. https://fr.wikipedia.org/wiki/M%C3%A9thode_Suzuki



IPTIC

NO 6 — 2025

DEVELOPPER UNE PENSEE INFORMATIQUE ET ALGORITHMIQUE OUTILS, DISPOSITIFS, TRACES, PUBLICS, USAGES ET RESPONSABILITES

Walter Nuninger, Olivier Neuhaus, Mario Ramalho
Développer une pensée informatique et algorithmique : outils, dispositifs, traces, publics, usages et responsabilités 05-09

Patrick Etcheverry, Thierry Nodenot, Pantxika Dagorret, Marta Toribio-Fontenla, Jennifer Lesbats, Christophe Marquesuzaà
Développer une pensée algorithmique : une approche basée sur des design patterns algorithmiques et des ateliers de katas 10-27

Sabrin Housni, Gaëtan Temperman, Audrey Kumps, Karim Boumazguida, Laetitia Dragone, Sarah Descamps, Bruno De Lièvre
Intégration de la pensée informatique dans la formation des professionnels de l'éducation : analyse croisée de preformance et du degré de certitude 28-41

Neuhaus Olivier, Mario Ramalho
L'autoconfrontation croisée entre étudiants pour former au débogage informatique 42-55

VARIA

Jérôme Albert Schumacher, Esther Duteil
Des usages du numérique dans l'éduction musicale 56-70

AUPTIC • education

9 772624 908000