Détermination de profils relatifs à la mobilisation de schème lors de la résolution de puzzles de programmation

Marielle Léonard[1,2], Maxime Bouton[1] et Yvan Peter[1]

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2] Univ. Lille, ULR 4354 - CIREL - Centre Interuniversitaire de Recherche en Éducation de Lille, F-59000 Lille, France
prénom.nom@univ-lille.fr

Résumé. En nous appuyant sur la théorie des champs conceptuels développée par Vergnaud, et plus précisément sur le concept de schème, nous étudions les compétences d'élèves de collège à résoudre des puzzles de programmation en langage Scratch. À cette fin, nous construisons une combinaison d'indicateurs qui nous permet de déterminer automatiquement, à partir de traces d'interaction, des profils relatifs à l'existence de la mobilisation d'un schème opérationnel pour résoudre un puzzle de programmation.

Mots clés : Programmation par blocs , Apprentissage de la programmation , Puzzle de programmation , Schème , Analyse de traces

Résumé. This article presents our study of secondary school pupils fluency during programming puzzles solving using the Scratch language. We analyse their interaction traces through the scope of Vergnaud's theory of conceptual fields (scheme concept). We built a combinaison of indicators that can determine fluency profiles automatically.

Mots-clés : Block-based programming , programming learning , Puzzle , Scheme , Learning analytics

1 Introduction

Depuis 2016 en France, la programmation informatique est présente dans les programmes scolaires de l'ensemble de la scolarité primaire et secondaire. Au collège, c'est la programmation par blocs avec des langages comme Scratch qui est préconisée. Pour ce niveau collège, nous nous intéressons à la résolution de problèmes de programmation de type *puzzle* dans des micromondes [7]. Il s'agit de problèmes fermés, où la tâche à réaliser est prescrite et pour lesquels une solution proposée peut être évaluée de manière exacte, comme valide ou non.

Notre objectif est d'étudier les compétences des élèves à résoudre ce type de problème à partir de leurs traces d'interaction avec l'environnement de programmation. Nos questions de recherche sont formulées de la manière suivante :

- RQ1 : Quels indicateurs construits à partir de traces d'interaction rendent compte de l'existence de compétences à résoudre des puzzles dans un environnement de programmation par blocs ?
- RQ2 : Quels profils relatifs au degré de compétence à résoudre des puzzles dans un environnement de programmation par blocs peut-on définir à partir de ces indicateurs ?

De nombreuses acceptions du terme de compétences existent [3]. Pour notre part, nous nous appuyons sur des éléments de la théorie des champs conceptuels de Vergnaud, que nous présentons dans une première partie. [11]. Nous complétons par la consultation de travaux relatifs à l'analyse de traces d'interaction. Nous décrivons ensuite notre cadre expérimental ainsi que la construction de nos outils d'analyse : visualisation, indicateurs. Nous présentons les résultats obtenus avant de conclure et d'aborder les perspectives.

2 Compétence et schème dans la théorie des champs conceptuels

Les puzzles de programmation relèvent d'une approche constructiviste de l'apprentissage, développée par Piaget [8]. Cette approche a été reprise et approfondie notamment par Gérard Vergnaud, auteur de la théorie des champs conceptuels [11]. Cette théorie vise à comprendre la conceptualisation, en particulier dans le cas des activités cognitives complexes, dont la programmation informatique fait partie.

Selon Vergnaud, « Sans conceptualisation, il n'y a pas d'activité opératoire possible et pas de compétence »[10]. Par compétence, Vergnaud désigne « "la forme opératoire de la connaissance", qui permet de faire et de réussir »[13]. Pour Vergnaud, le concept de compétence présente une limite, dans la mesure où il « renvoie d'abord au résultat de l'activité et insuffisamment à l'organisation de l'activité elle-même. »[12]. Malgré cela, il convoque ce concept à plusieurs reprises, mentionnant par exemple que « A est plus compétent que B, s'il s'y prend d'une meilleure manière. Le comparatif "meilleure" suppose des critères supplémentaires : rapidité, fiabilité, économie, élégance, compatibilité avec la manière de procéder des autres, etc... »[13] Concernant l'organisation de l'activité, elle est prise en charge par le concept de schème, défini comme une « organisation invariante de la conduite pour une classe donnée de situations » [11].

Vergnaud catégorise les situations (ou tâches) en deux classes :

- des classes de situations pour lesquels le sujets dispose des compétences nécessaires au « traitement relativement relativement immédiat de la situation ». Dans ce cas, la conduite du sujet est largement automatisée et organisée autour d'un schème unique.

— des classes de situations pour lesquels le sujet ne dispose pas de toutes les compétences nécessaires. Dans ce cas, le sujet entre dans un processus d'accommodation, lors duquel des schèmes sont essayés, déconstruits, réorganisés.[11, p. 136]

Nous retenons que la durée nécessaire au traitement de la situation est la variable déterminante pour distinguer ces deux classes de situation et que les compétences sont associées à la mobilisation d'un schème opérationnel pour traiter la situation.

Dans notre contexte de résolution de puzzles de programmation, nous cherchons donc un ou plusieurs indicateurs construits à partir de traces d'interaction qui permettent de rendre compte de la durée irréductible à l'exécution de la tâche (traitement immédiat révélateur de la mobilisation d'un schème unique) et du temps éventuellement nécessaire à un processus d'accommodation. Notre objectif est alors de déterminer automatiquement des profils, que nous appelons profils relatifs à la mobilisation de schème, fondés sur le traitement immédiat ou non du puzzle de programmation.

3 Analyse des traces d'interaction

Nous relevons des travaux sur lesquels nous appuyer pour construire des indicateurs à partir de traces d'interaction collectées lors d'une activité d'apprentissage et déterminer des profils par rapport au traitement d'une situation.

Pelanek et al. dressent un panorama des aspects à prendre en considération lors de la modélisation d'un domaine et de la modélisation d'un processus d'apprentissage, en fonction du contexte et de l'objectif de la modélisation [5]. Dans le tableau de synthèse présenté par ces auteurs, nous relevons qu'en plus de la validité des réponses, le temps nécessaire pour résoudre la tâche et l'historique des réponses soumises sont indiqués pour mesurer l'aisance (*fluency*) lors d'un processus d'apprentissage.

Dans un autre article, Pelanek explore le temps pour résoudre une tâche, associé à la nature des réponses fausses [6]. Il construit un indicateur pour normaliser le temps de réponse par rapport à la difficulté de la tâche. Cet indicateur reste cependant dépendant de l'échantillon considéré. Par ailleurs, il catégorise les erreurs de manière binaire en *commune* et *non commune*. À cette fin, il définit un seuil d'occurrences par rapport au total des réponses fausses. En combinant ces deux indicateurs, il établit quatre *profils de performance* : réponse correcte, réponse fausse dans un temps très court, erreur commune sans que le temps soit court, autre erreur.

Jiang et al. établissent également, en utilisant une méthode de *clustering*, des profils de comportement lors de la résolution d'un problème de programmation [4]. Ils caractérisent quatre profils à partir des programmes intermédiaires soumis : ceux qui abandonnent rapidement (*quitters*), ceux qui sont loin d'une solution valide lors des premiers essais, s'en approchent plus ou moins sans aboutir (*approachers*), ceux qui sont loin d'une solution valide lors

des premiers essais puis s'en approchent jusqu'à trouver (*solvers*), ceux qui soumettent un programme valide dès les premiers essais (*knowers*)

De ces travaux nous retenons l'idée de combiner deux indicateurs pour caractériser le traitement immédiat ou non d'une situation, et donc l'existence d'un schème opérationnel. Ces deux indicateurs sont le temps passé sur la tâche et nombre de tentatives (prise en compte des états intermédiaires). Nous retenons aussi l'idée de normaliser l'indicateur construit par rapport à des variables que nous souhaitons écarter temporairement de notre analyse.

4 Cadre expérimental

Les traces d'interaction que nous analysons sont issues de 658 sessions d'élèves de collège sur les parcours du concours en ligne de programmation Algorea[3] de l'année 2022. Chaque parcours est composé de 6 problèmes déclinés en 4 versions de difficulté croissante, ce qui fait 24 puzzles par parcours. Nous disposons donc de traces d'interaction de sujets avec l'environnement de programmation pour 144 puzzles différents (3 tours du concours dans 2 catégories), qui recouvrent les notions de base de l'algorithmique (séquence d'instructions, boucle, structure conditionnelle, variable, procédure sans paramètre). La figure 1 montre l'interface de programmation avec un exemple de puzzle.

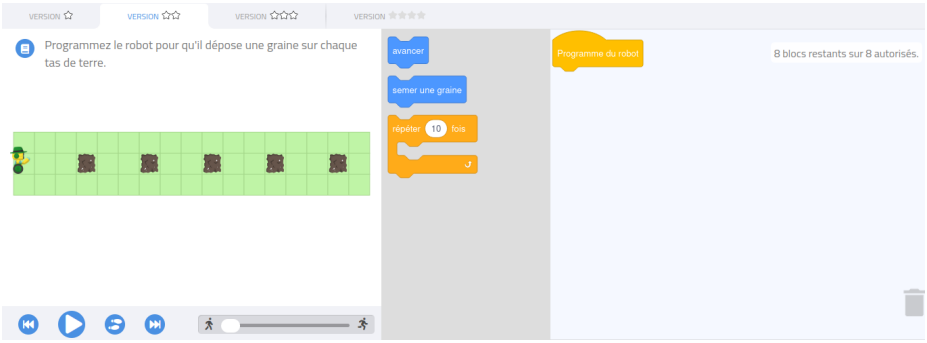

Fig. 1. Puzzle dans l'environnement de programmation Algorea

Lorsque l'utilisateur exécute un programme, celui-ci est évalué automatiquement par le système et un enregistrement est généré. L'évaluation du programme prend en compte les états intermédiaires et l'état final de la grille (le robot virtuel a-t-il bien rempli la mission qui lui est assignée en respectant les contraintes du problème ?) et les contraintes au niveau de

3. Concours de programmation en ligne organisé annuellement par l'association France-ioi (https://algorea.org)

l'édition du programme (limite du nombre d'instructions). Chaque enregistrement collecté contient l'identifiant de session, l'identifiant du problème, la version du problème, le programme exécuté, le résultat de l'évaluation du système (correct ou non) et l'horodatage associé à la soumission. Des enregistrements sont générés en plus pour certains événements importants comme un changement de puzzle. Après nettoyage des données, nous disposons de 74727 enregistrements utiles pour notre étude qui correspondent à 12340 confrontations d'un sujet avec un puzzle.

En complément, nous disposons également d'enregistrements vidéo de sessions pour une vingtaine de sujets (enregistrement de l'écran et de la voix). Ceux-ci nous permettent de disposer d'informations complémentaires pour appuyer nos choix méthodologiques.

5 Construction des outils pour l'analyse des données

5.1 Génération d'une visualisation sous forme de frise chronologique

Afin de disposer d'un premier outil pour explorer nos données, nous générons une vue synthétique de la session du sujet sous forme de frise chronologique (Figure 2). Ce type de visualisation est présent dans la littérature, mais plutôt pour l'étude du processus de résolution d'un problème particulier ([9], [2]).

Fig. 2. Visualisation de la session d'un sujet sous forme de frise chronologique (zoom sur les puzzles 4 et 5 du parcours, entre les $6^{ème}$ et $18^{ème}$ minutes)

L'axe horizontal est un axe temporel qui porte les différents évènements survenant pendant la session : changements de puzzle et exécutions. Chaque problème est représenté par une couleur dédiée, chaque version de problème (puzzle) par une nuance et une hauteur : plus clair et plus bas pour la version la plus facile jusqu'à plus foncé et plus haut pour la version la plus difficile de chaque problème. Chaque exécution d'un programme invalide est représentée par un trait noir, chaque soumission d'un programme valide par un trait blanc.

5.2 Construction d'indicateurs pour la détermination de profils

En nous appuyant sur le cadre d'analyse de Vergnaud présenté dans la section 2, nous cherchons à déterminer automatiquement, à partir des traces d'interaction, si le sujet réussit un puzzle de programmation de manière

experte, en mobilisant un schème opérationnel (validation rapide et quasiment sans se tromper), ou s'il s'engage dans un processus d'accommodation.

Pour cela, nous avons besoin de calculer le temps passé sur chaque puzzle et le nombre d'essais réalisés avant d'aboutir à une solution valide ou à un abandon. En utilisant ces données, nous déterminons un seuil en temps et en nombre d'essais au-delà duquel nous considérons qu'un processus d'accommodation est engagé, qu'il aboutisse à une résolution du problème (profil accommodation) ou non (profil échec).

Nous fixons le seuil en nombre d'essais à trois, ce qui inclut la nuance d'un schème opérationnel qui fait l'objet d'une ou deux erreurs de mise en œuvre, et qui est ajusté aussitôt (profil ajustement). En effet, nous avons remarqué que dans ce cas, le programme soumis est corrigé très rapidement, puis à nouveau exécuté et validé par le système.

Comme argumenté dans la section 2, la vitesse de résolution du problème est essentielle pour déterminer le degré de compétence d'un sujet face à une situation. Cette vitesse dépend du nombre de blocs à déplacer pour obtenir le programme solution et de la dextérité du sujet. Un sujet plus jeune peut disposer d'un schème opérationnel mais une moindre dextérité avec la souris induit un temps de résolution plus long. Notre objectif est de construire un indicateur qui ne dépende plus ni du nombre de blocs à déplacer pour obtenir la solution de référence, ni de la dextérité du sujet.

Dans une première étape, nous calculons un temps moyen individuel pour placer un bloc dans l'éditeur (t_{bloc}). Le calcul de t_{bloc} prend seulement en compte la première soumission pour chaque puzzle abordé. En effet, cette première soumission correspond au schème mobilisé en première intention pour traiter la situation, qu'il soit opérationnel ou non. À l'inverse des soumissions suivantes où les comportements sont plus divers, nous observons (sessions vidéos en appui) très peu de latence entre les déplacements de blocs lors de cette première soumission.

$$t_{bloc} = \frac{\sum_i t_{prem}(i)}{\sum_i N_{prem}(i)} \tag{1}$$

t_{bloc} est calculé selon la formule 1 où $\sum_i t_{prem}(i)$ est la somme sur tous les puzzles du parcours des durées des premiers essais et où $\sum_i N_{prem}(i)$ est le nombre de blocs déplacés lors de l'ensemble des premiers essais.

Nous construisons ensuite un indicateur de rapidité r_i attaché à un puzzle i (formule 2). Celui-ci prend en compte le nombre de blocs à déplacer, en nous basant sur le nombre de blocs de la solution de référence $N_{ref}(i)$ pour ce puzzle i et sur t_i qui est le temps passé sur ce puzzle. Dans le cas d'un expert qui soumet la solution de référence en un seul essai, cet indicateur correspond au temps moyen nécessaire pour placer un bloc. À ce stade, cet indicateur ne tient pas encore compte de la dextérité du sujet.

$$r_i = \frac{t_i}{N_{ref}(i)} \tag{2}$$

Nous calculons enfin un indicateur d'accommodation a_i attaché à la résolution du puzzle i (formule 3). Pour cela, nous centrons l'indicateur r_i autour de 0 en prenant l'écart entre r_i et t_{bloc} et nous le réduisons en divisant par le temps moyen mis pour placer un bloc. Nous obtenons ainsi un indicateur qui est indépendant de la dextérité du sujet et du nombre de blocs de la solution de référence.

$$a_i = \frac{r_i - t_{bloc}}{t_{bloc}} \tag{3}$$

Si cet indicateur a_i est négatif, cela signifie que le sujet a résolu plus rapidement le puzzle i que la moyenne de ses premières soumissions. Dans ce cas, il y a absence d'accommodation, le sujet mettant en oeuvre un schème unique si l'on se réfère au cadre d'analyse de Vergnaud. Plus cet indicateur est élevé, plus le sujet a passé de temps d'accommodation sur le puzzle i relativement à sa dextérité et au nombre de blocs nécessaires pour résoudre ce puzzle. Combiné au nombre d'essais et à la validité de la dernière réponse soumise, cet indicateur d'accommodation nous permet, après fixation d'un seuil, de déterminer automatiquement des profils relatifs à la mobilisation de schèmes lors de la confrontation de sujets avec des puzzles de programmation.

6 Résultats et discussion

Fig. 3. Identification de profils sur la frise chronologique d'une session (zoom)

La visualisation de la session d'un sujet sous forme de frise chronologique nous permet d'observer plusieurs profils construits à partir de la distinction de Vergnaud à propos du traitement immédiat ou non d'une situation (Figure 3). Un profil est associé à un couple sujet/puzzle. Différents profils se succèdent pour un même sujet au cours d'une session. :

— Profil **expert** : le sujet résout le puzzle très rapidement en un seul essai. La résolution du puzzle est organisée par un schème unique, opérationnel dans la situation.
— Profil **ajustement** : le sujet résout le puzzle rapidement après une ou deux soumissions ratées. Il s'organise autour d'un schème opérationnel pour traiter la situation, mais avec des approximations lors de la mise en œuvre.

— Profil **accommodation** : le sujet passe un temps conséquent sur le problème et fait plusieurs soumissions ratées avant de soumettre un programme valide. Il ne dispose pas de schème directement opérationnel pour traiter la situation. Il entre dans un processus d'accommodation qui aboutit positivement.
— Profil **échec** : le sujet est en échec sur le problème. Comme pour le profil précédent, le sujet entre dans un processus d'accommodation, mais celui-ci ne permet pas d'aboutir à un programme valide pour résoudre le puzzle.

Nous identifions par ailleurs deux autres profils, non visibles sur cet exemple de frise :

— Profil **passé** : le sujet ouvre brièvement un puzzle sans lancer d'exécution de programme.
— Profil **indéterminé** : le sujet résout le puzzle en un seul essai mais avec un temps long, qui ne nous permet pas de conclure sur la mobilisation d'un schème opérationnel. Nous observons plusieurs cas sur nos enregistrements vidéos où le sujet suspend momentanément sa recherche pour faire autre chose, mais aussi des cas où le sujet manipule longuement des blocs dans l'éditeur avant de lancer une exécution et des cas où le sujet consulte le tutoriel disponible.

La question est, à ce stade, de définir un seuil pour l'indicateur d'accommodation construit dans la section 5.2. En effet, ce seuil détermine la répartition entre les profils expert et indéterminé d'une part, ajustement et accommodation d'autre part. Nous visualisons dans ce but la courbe de la fréquence des profils expert et ajustement en fonction du seuil considéré pour l'indicateur d'accommodation (Figure 4).

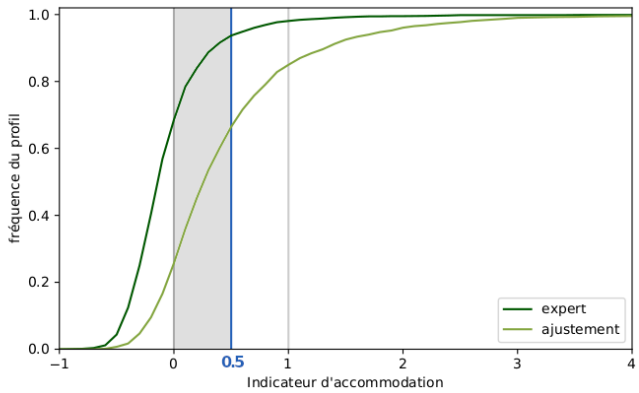

Fig. 4. Fréquence des profils expert (réussite en une seule tentative) et ajustement (réussite en deux ou trois tentatives) en fonction du seuil retenu pour l'indicateur d'accommodation

De cette courbe, nous déduisons un intervalle de valeurs dans lequel nous situons le seuil recherché et concentrons nos analyses. La valeur 0 pour l'indicateur d'accommodation équivaut à une résolution avec un temps égal au temps moyen des premiers essais du sujet. Lorsque la valeur de l'indicateur d'accommodation vaut 0,5 cela signifie que le sujet met 1,5 fois plus de temps à résoudre le puzzle que le temps moyen de ses premiers essais.

Nous avons fait varier le seuil entre 0 et 0,5 et étudié l'incidence sur la répartition des profils. Nous avons finalement retenu un seuil à 0,5. La fixation du seuil à cette valeur comporte certes une part d'arbitraire. Elle permet cependant un équilibre satisfaisant entre :

— le résidu des validations en 1 essai (profil indéterminé) qui ne sont pas catégorisées dans le profil expert
— une distinction entre les profils ajustement et accommodation concordante dans une majorité de cas avec nos observations sur les enregistrements vidéo

En effet, nous avons confronté pour chaque couple sujet/puzzle le profil déterminé automatiquement à partir d'un seuil à 0,5 avec le profil attribué suite à une analyse de l'enregistrement vidéo (pour 73 cas sur 1277, soit 5,7%, le profil reste indéterminé ou non pris en charge par la catégorisation). Nous obtenons une concordance de 98% pour le profil échec, 92% pour les profils expert et accommodation, 70% pour le profil ajustement. Le profil indéterminé calculé à partir du seuil correspond, sur la base de l'analyse vidéo, pour 29% à un profil expert, 6% à un profil ajustement, 50% à un profil accommodation et pour 15% à un cas non pris en charge par la présente catégorisation (notamment suspension temporaire de l'activité).

En considérant ce seuil à 0,5, nous visualisons un histogramme des profils identifiés selon la valeur de l'indicateur d'accommodation (Figure 5). Ainsi, sur les 12340 confrontations de sujet à un puzzle de programmation, nous obtenons :

— 2819 occurrences (soit 22,8%) de profil *expert* : puzzle résolu en un seul essai et $a_i \leq 0.5$
— 1728 occurrences (soit 14,0%) de profil *ajustement* : puzzle résolu en 2 ou 3 essais et $a_i \leq 0.5$
— 4049 occurrences (soit 32,8%) de profil *accommodation* : puzzle résolu en 2 ou 3 essais et $a_i > 0.5$ ou résolution en plus de 3 essais.
— 2407 occurrences (soit 19,5%) de profil *échec* : puzzle non résolu, toutes les soumissions sont invalides.
— 1067 occurrences (soit 8,6%) de profil *passé* : puzzle ouvert mais pas de programme soumis.
— 270 occurrences (soit 2,2%) de profil *indéterminé* : un seul essai mais $a_i > 0.5$.

Cette étude montre qu'une part significative des réussites aux puzzles de programmation ne relèvent pas d'une résolution experte. En effet, parmi les 71,8% de confrontations d'un sujet à un puzzle qui aboutissent à une réussite (programme valide), seulement 51,3%, soit un peu plus de la moitié, relève d'un traitement relativement immédiat de la situation (profils expert et ajustement).

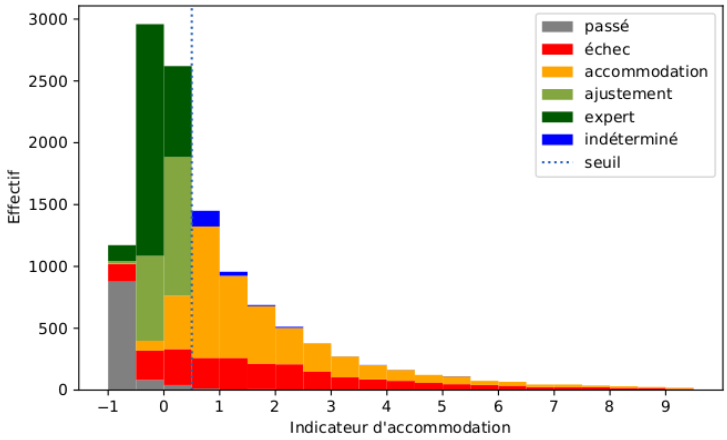

Fig. 5. Histogramme des profils avec un seuil de l'indicateur d'accommodation à 0.5 pour la détermination des profils expert et ajustement

Dans ce cas, la résolution révèle l'existence des compétences en jeu, et l'activité du sujet est organisée selon un schème unique. Ce résultat nous amène donc à inciter à la vigilance sur le fait d'utiliser un résultat brut (seulement réussite ou échec) lié à une série de puzzles pour évaluer des compétences en programmation, surtout lorsque le nombre d'exécutions du programme n'est pas limité. En revanche, la combinaison de la réussite au puzzle, du nombre d'essais et de l'indicateur d'accommodation proposé dans cet article, permet de rendre compte de manière plus fiable de la maîtrise des compétences en programmation sous-jacentes, à la condition que les compétences nécessaires à la résolution de chaque puzzle aient été correctement identifiées.

7 Conclusion & Perspectives

Dans ce travail nous avons cherché à déterminer le profil d'un sujet relatif à la mobilisation de schème lors de sa confrontation à un puzzle de programmation en langage Scratch. Notre contribution porte sur la mise en relation d'éléments de la théorie des champs conceptuels de Vergnaud avec l'analyse de traces d'interaction dans le but de quantifier le caractère immédiat ou non du traitement d'une situation et de caractériser automatiquement des profils dans le contexte de la résolution de puzzles de programmation en langage Scratch. À cette fin, nous avons calculé un indicateur d'accommodation attaché à un couple sujet/puzzle indépendant de la dextérité du sujet et de la solution de référence, que nous avons combiné avec le nombre d'essais et la validité ou non des programmes exécutés.

Associé à une identification précise des compétences en jeu dans chaque puzzle, cette détermination de profil sur la base de l'indicateur

d'accommodation et du nombre d'essais peut aider à automatiser la validation de compétences en programmation par blocs. Par ailleurs, nous avons implémenté dans un travail précédent un modèle d'apprentissage machine permettant une prédiction binaire (réussite / échec) de la réussite d'un sujet à un puzzle [1]. Nous avons adapté ce modèle pour la prédiction de nos profils avec une précision similaire au résultat précédent. Cette prédiction ouvre des perspectives de rétroactions personnalisées et d'adaptations de parcours.

Dans la mesure où nous n'avons pris en compte que la validité des réponses, le temps de réponse et le nombre d'essais, indépendamment du contenu des puzzles, ces résultats pourraient probablement être généralisés à des tâches pour lesquelles il est possible de définir une action élémentaire (ici placer un bloc dans l'éditeur). Nous notons que Pelanek a déjà entamé ce type de généralisation à des données de différentes natures [6].

Néanmoins, une limite de la démarche proposée dans cet article est qu'elle ne permet pas d'investiguer l'organisation invariante de l'activité, centrale dans le concept de schème. Nous déduisons seulement l'existence d'un schème unique organisateur de la conduite dans le cas des résolutions expertes, en nous fondant sur le cadre théorique de Vergnaud. Or selon l'auteur, « Le concept de compétence appelle une analyse de l'activité en termes de "schèmes". »[14]. Le présent travail n'est donc qu'une étape dans l'analyse de l'activité du sujet confronté à des puzzles de programmation. Se posent notamment des questions sur l'existence d'un apprentissage suivant la manière dont le sujet opère : est-ce que le sujet analyse le résultat de chaque exécution de son programme et en déduit les modifications à apporter pour aller vers une solution valide, procède-t-il par tâtonnement en exécutant une succession de programmes légèrement modifiés en espérant tomber par chance sur une solution valide, existe-t-il d'autres stratégies ? Quelles sont les conceptualisations sous-jacentes aux erreurs récurrentes ? Une autre limite tient à la difficulté à distinguer dans quelle mesure le degré d'accommodation est lié à un apprentissage ou à la difficulté propre d'un puzzle [5].

D'où la nécessité d'analyses plus approfondies pour relier étroitement modélisation du domaine et modélisation de l'apprenant, ce qui revient, dans le cadre d'analyse de la théorie des champs conceptuels de Vergnaud à déterminer de quelle manière un sujet construit un champ conceptuel des bases de la programmation en langage Scratch. Nous visons cet approfondissement dans la suite de nos travaux, notamment en menant des analyses à des niveaux de granularité plus fins : structure des programmes soumis et nature des erreurs commises, interactions avec l'interface (notamment mouvements de souris).

Remerciements

Ce travail est soutenu par les projets Interreg Teach Transition (https://teachtransition.eu) et ANR IE-CARE (https://iecare.lip6.fr/. Nous remercions aussi l'association France-ioi pour la mise à disposition de la plateforme chticode.algorea.org.

Références

1. Bouton, M., Peter, Y., Léonard, M., El Mawas, N. : Tentative de prédiction de la réussite à un exercice de programmation (2022)
2. Chevalier, M., Giang, C., Piatti, A., Mondada, F. : Fostering computational thinking through educational robotics : a model for creative computational problem solving. International Journal of STEM Education **7**(1), 1–18 (2020), publisher : SpringerOpen
3. Crahay, M. : Dangers, incertitudes et incomplétude de la logique de la compétence en éducation. Revue française de pédagogie. Recherches en éducation (154), 97–110 (2006), publisher : ENS Éditions
4. Jiang, B., Zhao, W., Zhang, N., Qiu, F. : Programming trajectories analytics in block-based programming language learning. Interactive Learning Environments **30**(1), 113–126 (2022), publisher : Taylor & Francis
5. Pelánek, R. : Bayesian knowledge tracing, logistic models, and beyond : an overview of learner modeling techniques. User Modeling and User-Adapted Interaction **27**(3), 313–350 (2017), publisher : Springer
6. Pelánek, R. : Exploring the utility of response times and wrong answers for adaptive learning. In : Proceedings of the fifth annual ACM conference on learning at scale. pp. 1–4 (2018)
7. Pelánek, R., Effenberger, T. : Design and analysis of microworlds and puzzles for block-based programming. Computer Science Education **32**(1), 66–104 (Jan 2022)
8. Piaget, J. : La naissance de l'intelligence chez l'enfant. Delachaux et Niestlé Neuchatel-Paris (1935)
9. Toll, D., Wingkvist, A. : Visualizing Programming Session Timelines. In : Proceedings of the 11th International Symposium on Visual Information Communication and Interaction. pp. 106–107 (2018)
10. Vergnaud, G. : Pourquoi parler autant de conceptualisation. In : Actes du colloque Conceptualisation et surdité Synthèse des travaux, Suresnes (2003)
11. Vergnaud, G. : La théorie des champs conceptuels. In : Recherches en didactique des mathématiques, vol. 10/2.3. La Pensée Sauvage (1991)
12. Vergnaud, G. : Au fond de l'action, la conceptualisation. Presses Universitaires de France (2011), pages : 275-292 Publication Title : Savoirs théoriques et savoirs d'action
13. Vergnaud, G. : Forme opératoire et forme prédicative de la connaissance. Investigações em Ensino de Ciências **17**(2), 287–304 (2012)
14. Vergnaud, G. : Qu'est-ce que la pensée ? La nouvelle revue de l'adaptation et de la scolarisation **N° 63**(3), 277–299 (2013), publisher : I.N.S.H.E.A.