Introduire la concurrence en début de secondaire ?

Libert, Cedric; Vanhoof, Wim

Published in:
L'informatique, objets d'enseignements enjeux épistémologiques, didactiques et de formation

Publication date:
2020

Introduire la concurrence en début de secondaire ?

Cédric Libert et Wim Vanhoof

Université de Namur, Belgique
{cedric.libert,wim.vanhoof}@unamur.be

Résumé Dans la littérature, plusieurs auteurs encouragent l'enseignement de la concurrence, car beaucoup d'évolutions en informatique font appel à ce concept. Nous rapportons dans cet article le résultat d'une expérience menée auprès de 101 adolescents et adolescentes de 12 à 15 ans à qui nous avons dispensé deux séances de 100 minutes sur la concurrence par passage de messages. Nous y commentons l'évolution des élèves face au concept de concurrence et face à la réalisation d'un problème de synchronisation.

Keywords: concurrence · secondaire · programmation concurrente · passage de messages

1 Introduction

En Fédération Wallonie Bruxelles, les cours de programmation dispensés en secondaire se concentrent sur l'enseignement des paradigmes impératif et orienté objet. On considère généralement ces paradigmes comme plus simples à comprendre [2,3], notamment grâce à leur nature séquentielle. D'un autre côté, la programmation concurrente, permettant de programmer l'exécution de plusieurs tâches simultanées, fait généralement partie de cours plus avancés, après le secondaire. Certains auteurs prétendent toutefois que la programmation concurrente peut être au moins aussi intuitive pour les étudiants que la programmation séquentielle [11,12]. De plus, dans son rapport de 2013 sur l'enseignement de l'informatique, l'ACM recommande que le cours introductif à la programmation, destiné à des novices, s'inscrive dans un modèle où la concurrence est la norme, et l'aspect séquentiel un cas particulier [1, p.44]. En effet, l'évolution de trois grands domaines de l'informatique -matériel, logiciel et données- implique la concurrence.

Pour vérifier la possibilité d'enseigner la programmation concurrente en secondaire, nous avons mené une expérimentation auprès de 101 étudiants de secondaires en Fédération Wallonie Bruxelles. Nous y avons élicité leurs préconceptions lexicales sur le terme "concurrence" et examiné comment ils tentent de résoudre un problème de synchronisation, au moyen d'un test initial [7]. Nous avons ensuite donné aux élèves deux séances de 100 minutes de cours sur la concurrence dans le cadre du projet School-IT [9], qui consiste à développer,

tester en classe et regrouper des matériaux d'enseignement liés à différentes facettes de l'informatique. Finalement, nous leur avons fait passer un test final, correspondant au test initial.

Dans cet article, nous vérifions l'évolution de ces analyses préalables après ces deux séances de cours en comparant les réponses aux questionnaires pré-test et post-test. Nous présentons d'abord le contexte scientifique de cette recherche : quelle est la conception de certains auteurs sur la pertinence d'enseigner la concurrence à des novices et quel est l'état de la recherche sur les préconceptions des apprenants. Nous décrivons ensuite l'expérience que nous avons menée avec les élèves et comment nous avons traité les données récoltées. Puis, en nous basant sur ces données, nous discutons des résultats quantitatifs et qualitatifs des questionnaires que nous avons soumis aux étudiants avant et après les deux séances de cours.

2 Contexte

La concurrence est un concept de programmation qui permet de programmer plusieurs actions pour qu'elles aient lieu en même temps [8] dans la poursuite d'un même but. Elle est présente intrinsèquement dans le monde, où des évènements simultanés ont lieu en permanence. La programmation concurrente capture cette caractéristique en s'affranchissant de l'obligation de séquentialité et en permettant l'exécution simultanée de plusieurs instructions.

Cette propriété, en fonction du paradigme auquel elle est intégrée, se réalise sous la forme d'un modèle particulier, qui décrit comment les instructions parallèles peuvent communiquer. Dans cet article, nous utiliserons un modèle de concurrence en particulier qui est le modèle acteur. Dans ce modèle, le système programmé est constitué d'acteurs qui agissent en même temps et peuvent envoyer des messages à d'autres acteurs ou recevoir des messages d'autres acteurs.

L'idée d'enseigner la programmation concurrente relativement tôt dans le cursus n'est pas neuve. Dès 1997, Feldman et Bachus décrivent en quoi il serait possible de l'expliquer à des néo-programmeurs [4]. En 1998, Lynn Andrea Stein, quant à elle, plaide pour qu'on passe du modèle de programmation Turing/Von Neumann, qui considère qu'un programme est une fonction et donc que son exécution est réductible à un calcul, à un modèle d'interaction où un programme est une communauté d'agents qui communiquent [10].

C'est dans ce contexte qu'en 2001, Yifat Ben-David Kolikant [5] analyse les préconceptions de ses étudiants, dans une approche constructiviste de l'apprentissage. Elle réalise cette recherche dans le cadre d'un cours de programmation concurrente dispensé à des élèves de fin du secondaire (17-18 ans), en orientation informatique avancée. Elle souhaite, grâce à ses conclusions de recherche, adapter son cours aux problèmes rencontrés par ses élèves. Elle y traite principalement du problème de synchronisation entre les différentes actions concurrentes. Les étudiants ont ainsi dû résoudre le problème suivant :

Pour vendre un ticket dans un cinéma, un vendeur réalise l'algorithme suivant :

```
Pour chaque client
    Chercher la meilleure place libre: C
    Si C existe (s'il restait encore au moins une place libre)
       Marquer C comme occupée
       Imprimer le ticket correspondant à C
```

Les étudiants doivent décrire un système et un algorithme qui permettent à deux vendeurs de vendre des tickets en même temps. Ils doivent en sus expliquer en quoi leur solution permet d'éviter que deux tickets identiques soient vendus à des clients différents.

Les chercheurs ont ensuite classé les réponses en cinq catégories : trois (C1, C2, C3) qui incluent une gestion centralisée de la concurrence ; deux (D1, D2) qui proposent une solution centralisée au problème :

— C1 : une entité centrale gère l'entrelacement des actions ;
— C2 : l'étudiant fait des suppositions sur l'ordre d'exécution, de sorte qu'il n'existe qu'un seul scénario possible ;
— C3 : certaines ressources communes sont privatisées au sein d'entités concurrentes ;
— D1 : les vendeurs communiquent de façon implicite, via une mémoire partagée ;
— D2 : les vendeurs communiquent de façon explicite, via des messages.

De ces réponses, Kolikant conclut que :

— les étudiants trouvent plus de solutions centralisées que décentralisées
— ceux qui ont suggéré des solutions décentralisées utilisent principalement la communication pour bloquer une tâche tant qu'une autre n'est pas réalisée
— beaucoup simplifient le problème, pour le résoudre plus facilement
— un système concurrent distribué sur plusieurs ordinateurs est plus intuitif pour eux que la concurrence sur un seul ordinateur

Quelques années après, en 2007, Lewandowski et al. [6] a mené une expérience similaire, avec les mêmes exercices, sur des étudiants de première année d'étude supérieure en sciences informatiques. Il modifie le problème pour l'adapter au manque de connaissance de ses étudiants et ne leur demande ainsi pas de (pseudo-)code ni de définition précise du matériel utilisé, mais uniquement une description en anglais de leur système. Il y retrouve les même catégories élicitées par Kolikant et met au jour deux problèmes saillants : les étudiants éprouvent des difficultés à repérer et éviter les *race conditions*, c'est-à-dire le caractère non-déterministe de leur solution ; et certains simplifient à outrance la tâche pour en résoudre une version plus simple. Il conclut de son analyse que manifestement les étudiants sur lesquels il a réalisé son expérience ont des préconceptions très variées, de sorte qu'un enseignant en programmation concurrente pourrait introduire son cours par la résolution d'un problème du type "vente de tickets" et la comparaison des différentes solutions proposées par les étudiants et soulever les mauvaises conceptions sur l'exécution concurrente. Finalement, il remarque que la distribution des solutions de ses étudiants est différente de celle obser-

vée par Kolikant. En particulier, plus de la moitié des solutions proposées sont centralisées (C1, C2, C3) contre un tiers chez Kolikant.

Notre article s'inscrit dans la lignée de ces deux contributions en fournissant aux élèves le même problème à résoudre. Toutefois, nous nous démarquons sur trois points. D'une part, les étudiants sur lesquels nous avons mené notre recherche sont plus jeunes et n'ont, pour beaucoup, jamais programmé. D'autre part, nous avons adapté le problème à résoudre, à la manière de Lewandowski, en leur demandant de décrire leur solution en français, et nous avons ajouté d'autres exercices qui permettent d'en savoir plus sur les conceptions lexicales du terme "programmation concurrente". Finalement, notre but n'est pas de déterminer une classification des solutions proposées (comme l'a fait Kolikant), ni de déterminer si les étudiants sont capables de reconnaitre les problèmes de concurrence dans le cas où il y a des ressources partagées (comme l'a fait Lewandowski), mais de déterminer à quel point des élèves de 12 à 15 ans sont capables de comprendre et de résoudre des problèmes de concurrence, et comment.

3 Méthodologie

Notre recherche s'articule en trois phases : un questionnaire pré-test, pour évaluer les préconceptions des étudiants ; deux séances de 100 minutes de cours données dans chacune des classes ; un questionnaire post-test, identique au questionnaire pré-test, pour évaluer l'évolution des conceptions.

Le public de notre étude est constitué de 101 élèves répartis sur trois écoles, en première (27), deuxième (20) et troisième (54) année, soit 73 adolescents et 28 adolescentes de 12 à 15 ans.

3.1 Pré-test et post-test

Le pré-test et post-test se sont déroulés de façon identique, respectivement en début de la première séance de cours et en fin de la seconde. Les élèves avaient 20 minutes pour répondre au questionnaire. Puisqu'il s'agissait d'une inquiétude récurrente, nous leur avons précisé d'emblée qu'il ne s'agissait pas d'une évaluation certificative, mais d'une recherche scientifique. 35% de ces étudiants avaient déjà programmé avant, principalement avec des systèmes de programmation visuelle tels que Scratch ou Micro:bit.

Dans la première partie du questionnaire, nous demandons aux étudiants s'ils ont déjà programmé et d'expliquer, le cas échéant, ce qu'ils ont déjà fait. Cette explication est importante, car elle nous permet de nous rendre compte de ce que les étudiants considèrent comme étant de la programmation.

Dans la deuxième partie, nous cherchons à évaluer les préconceptions lexicales de la notion de programmation concurrente, à travers 3 questions. Nous leur demandons, d'abord, de définir la programmation concurrente. Ensuite, nous leur montrons sept systèmes (Android, jeu vidéo, calculatrice, programme Scratch, distributeur de boissons, Facebook, Whatsapp) et nous leur demandons d'indiquer, pour chacun, s'ils utilisent la programmation concurrente (oui, non, je

ne sais pas, je ne connais pas ce système). Finalement, ils doivent expliquer, pour l'un des systèmes où ils ont indiqué "oui", pourquoi ils considèrent que c'est concurrent. Ils doivent faire de même pour un système où ils ont indiqué "non".

Finalement, dans la troisième partie, nous leur soumettons le problème des vendeurs de tickets tels que décrit dans la section 2. Nous leur indiquons que, dans la configuration actuelle, deux vendeurs peuvent être amenés à vendre le même ticket à deux clients différents et nous leur demandons d'imaginer quatre solutions qui permettraient de résoudre le problème en modifiant le système. Nous leur signalons qu'ils sont invités à donner leurs solutions en français et pas dans un langage de programmation.

3.2 Séances de cours

La première séance de cours s'entame, suite au test, sur une discussion avec les élèves à propos de la concurrence. Il en ressort qu'un système concurrent est un système où plusieurs évènements peuvent avoir lieu simultanément ou quasi-simultanément. Les élèves donnent ainsi différents exemples de systèmes et discutent de leur aspect concurrent ou pas.

S'ensuit une mise en situation où certains élèves reçoivent une fiche avec une partie du visage (œil, sourcil, bouche), certains reçoivent une fiche *cerveau*, et enfin d'autres reçoivent une fiche sens (vue ou audition). Ils reçoivent également des rectangles de papier qui représentent des messages. Nous leur expliquons que chacune des fiches décrit les actions qu'ils doivent réaliser en fonction du message qu'ils reçoivent. Nous appelons cette fiche le *comportement*. En examinant les *messages*, nous parvenons à une définition avec eux de ce qu'est un message simple : un destinataire et un contenu. Dans le cas qui nous concerne, le destinataire est un élément du visage et le contenu une action qu'il doit effectuer.

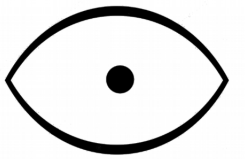
Nous réalisons ensuite une première exécution de ce système-visage : un premier stimulus est perçu par l'audition (un claquement de mains), qui envoie un message au cerveau, qui lui-même envoie des messages à différentes parties du visage. Après quelques secondes, un visage surpris se met en place. Nous réitérons l'opération, avec d'autres élèves et un autre stimulus, pour obtenir un autre visage, puis nous passons à une phase de discussion : forment-ils, ensemble, un système concurrent ? Qu'observent-ils ?

Après cette discussion, nous leur demandons, par groupe, de modifier les fiches de comportement des acteurs, en ajoutant le cas échéant des messages, pour créer un nouveau visage qui réagit à un autre stimulus, qu'ils doivent définir également. Ils peuvent, à cet effet, écrire directement sur les fiches qu'ils ont à leur disposition. À la fin de cette étape, nous testons ensemble le nouveau scénario élaboré par chacun des groupes, en identifiant, le cas échéant, les imprécisions dans la description des actions ou des messages.

Généralement, la première séance de 100 minutes s'arrête là. Au début de la séance suivante, nous leur demandons de réaliser un schéma qui représente le système-visage qu'ils ont exécuté la dernière fois, au moyen de flèches qui représentent les messages envoyés entre les différents agents.

Figure 1. Fiche de comportement pour l'œil.

ŒIL



Actions à faire pour chacun des messages reçus :

Message : **OUVRE**
À faire :
tourner l'œil du côté ouvert

Message : **FERME**
À faire :
tourner l'œil du côté fermé

Nous passons finalement à la partie de programmation sur les Micro:Bit. Nous leur expliquons d'abord le fonctionnement de l'appareil, de l'environnement de programmation et de l'envoi de messages. Nous leur demandons ensuite d'implémenter chacun sur son Micro:Bit la partie de visage qui lui est attribuée, en se basant sur le schéma qui a été réalisé. Puisqu'ils n'ont encore jamais programmé, cette phase est partiellement guidée. En fin de séance, nous mettons en commun les Micro:Bit pour qu'ils forment un visage et nous vérifions si chaque élément du visage réalise bien ce qui est demandé.

3.3 Analyse et traitement des données

Nous avons encodé les réponses des étudiants au pré-test et au post-test et les avons traitées de la façon suivante, en prélude à l'analyse.

En ce qui concerne la définition de la concurrence demandée aux élèves, nous avons élicité les grandes catégories suivantes à partir de leurs définitions et des explications qu'ils ont données à la question de classification des systèmes : programmer des actions simultanées, plusieurs actions en même temps, concurrence économique, envoyer des messages, plusieurs actions (sans préciser plus), aucune idée. Nous n'avons pas retenu les 10% de réponses qui constituent des catégories singletons.

Pour ce qui est de l'exercice sur la vente de tickets, nous avons classifié chaque solution en l'une des cinq catégories de Kolikant (voir section 2). Nous avons également évalué, pour chacune des réponses, si elle est raisonnable, c'est-à-dire si elle propose une idée pour résoudre le problème présenté, même si l'étudiant n'explique pas en détail comment l'implémenter. Par exemple "empêcher les deux vendeurs d'utiliser le système en même temps" est considéré comme une solution

raisonnable pour résoudre le problème, bien qu'elle puisse conduire à d'autres problèmes et que l'étudiant ne spécifie pas une procédure qui permettrait d'atteindre cet objectif. Finalement, certaines ne sont simplement pas des solutions au problème soumis.

4 Résultats

4.1 Évolution de la compréhension terminologique

Grâce au post-test, nous avons remarqué qu'à la suite des deux séances de 100 minutes chacune sur la programmation concurrente, la perception du terme "programmation concurrente" a significativement changé : les définitions données (question 3) sont extrêmement différentes du pré-test, la catégorisation des systèmes concurrents ou non (question 4) témoigne d'un plus grand sentiment de connaissance et les justifications qu'ils donnent pour cette classification (question 5) sont cohérentes avec les nouvelles définitions données.

Concernant les définitions, le pré-test met en évidence deux grandes catégories de réponses chez les élèves : ceux qui avouent ne pas savoir (71) et ceux qui évoquent la concurrence dans le sens de compétition économique ou sportive (19). Les autres catégories sont relativement négligeables, mais nous notons tout de même que 3 élèves évoquent l'idée d'exécution d'actions simultanées. Après les deux séances de cours, dans le post-test, seuls 17 déclarent encore ne pas savoir ce qu'est la programmation concurrente. Parmi eux, 14 se trouvaient déjà initialement dans cette catégorie. Pour 36 étudiants, la programmation concurrente consiste à programmer un système où plusieurs actions peuvent avoir lieu simultanément. Et 23 autres évoquent également l'idée d'actions simultanées. Les étudiants qui se retrouvent dans ces deux catégories au post-test sont principalement ceux qui avaient déclaré initialement qu'ils ne savaient pas. A contrario, seuls 2 étudiants considèrent toujours la concurrence comme une forme de compétition économique ou sportive. Nous observons finalement que 10 étudiants considèrent que la programmation concurrente est définie par le fait qu'il y a un échange de messages entre entités, ce qui est dû au modèle de concurrence par passage de messages utilisé dans le cadre de ce cours. Nous concluons donc que les préconceptions terminologiques de départs ont été rapidement corrigées chez les étudiants, car un peu plus de la moitié d'entre eux considèrent au post-test que la programmation concurrente concerne des actions qui ont lieu en même temps.

Dans la question suivante, où il s'agit pour l'élève de déterminer pour chacun des systèmes mentionnés s'il est concurrent ou non, nous remarquons une diminution significative des réponses vides ou des réponses où ils indiquent ne pas savoir. Cela montre un possible sentiment de maitrise plus élevé chez les étudiants. Par ailleurs, au pré-test, les étudiants excluaient les systèmes qui n'étaient pas considérés comme commercialement en concurrence avec d'autres (la calculatrice et le distributeur de boissons). Ces systèmes demeurent, dans le post-test, peu sélectionnés par les étudiants comme étant concurrents, relativement aux autres. Toutefois, au regard des justifications données par les étudiants

Table 1. Nombre d'étudiants par type de définition de la programmation concurrente.

Catégorie de réponse	Pré-test	Posttest
Systèmes avec plusieurs actions simultanées	0	36
Actions en même temps	3	23
Compétition	19	2
Courant	2	0
Meilleure, grande, complexe	4	1
Aucune idée	71	17
IA	2	0
Plusieurs acteurs	0	1
Plusieurs programmeurs en même temps	0	3
Envoyer des messages	0	10
Ordinateur fait ce qu'on demande	0	1
Plusieurs choses se passent	0	5
Serveur	0	1
Compétition	0	1

dans la question subséquente pour expliquer leurs choix, nous en déduisons que la raison n'est plus liée à la compétition commerciale, mais à la réalisation ou non d'actions simultanées dans le système.

Table 2. Classification des systèmes par l'élève : ce système est-il concurrent ?

	Pré-test				Posttest			
	Concurrent	Pas concurrent	Ne sait pas	Rien	Oui	Non	Ne sais pas	Rien
Android	51	9	21	14	78	6	11	5
Jeu vidéo	49	7	26	13	88	6	3	4
Calculatrice	20	32	28	15	37	47	12	5
Programme Scratch	21	11	26	14	39	6	17	4
Distributeur	17	34	26	15	32	42	22	5
Facebook	49	12	21	13	79	9	9	4
Whatsapp	46	10	23	14	79	6	9	4
Totaux	253	115	171	98	432	122	83	31

L'exercice suivant met également en évidence ce plus grand sentiment de compétence. En effet, là où, dans le pré-test, 48 étudiants n'ont rien répondu quand il s'agit d'expliquer, pour deux des systèmes préalablement sélectionnés, pourquoi ils sont ou pourquoi ils ne sont pas concurrents, seulement 13 étudiants n'ont rien répondu dans le post-test. Mieux, 43 étudiants justifient, dans le post-test au moins un de leurs deux choix au moyen de la définition de la concurrence comme permettant l'exécution simultanée de plusieurs actions, contre aucun dans le pré-test. Par ailleurs, très peu d'étudiants (3) ont conservé la conception que la concurrence était liée à une forme de compétition économique. Nous notons finalement que 10 étudiants associent concurrence et envoi de messages,

comme nous l'avions déjà remarqué dans les définitions qu'ils ont données à la première question.

Table 3. Raisons pour avoir indiqué qu'un système est concurrent / n'est pas concurrent

	PRE		POST	
	Une réponse	Deux réponses	Une réponse	Deux réponses
Rien	71	48	45	13
Informatique	15	7	7	1
Concurrence économique	17	9	3	2
Complexe	7	2	1	0
En même temps	0	0	43	13
Plusieurs actions	0	0	26	3
Messages	0	0	8	0

De ces analyses, nous concluons que les élèves, après deux séances de 100 minutes de cours, étaient déjà capables de définir de façon assez juste ce qu'était la programmation concurrente et d'appliquer cette définition à des systèmes existants, dans le but de les classer en justifiant deux de leurs réponses.

4.2 Évolution des solutions au problème de la vente de tickets

Nous avons examiné les solutions proposées par les élèves pour résoudre le problème qui leur était soumis. En les comparant avec les réponses fournies dans le pré-test, nous remarquons deux points importants. D'une part, le nombre de solutions raisonnables fournies est significativement supérieur à ce qui se trouvait dans le pré-test. Nous passons ainsi de 90 solutions raisonnables à 114, et nous doublons le nombre d'étudiants qui proposaient au moins 3 de ces solutions : de 6 nous passons ainsi à 14.

Table 4. Solutions raisonnables

# solutions	# étudiants pré-test	# étudiants post-test
4	1	3
3	5	11
2	21	21
1	29	28
0	45	38

En revanche, lorsque nous observons les catégories de réponses fournies par les étudiants, nous remarquons qu'il y a relativement peu de changement entre le pré-test et le post-test. Il y a toutefois davantage de réponses qui citent explicitement l'envoi et la réception de messages. Cela s'est traduit par une augmentation

légère du nombre de réponses qui se trouvent dans la catégorie D2 (communication explicite par messages envoyés entre les entités).

Cette faible différence trouve sans doute son explication dans le fait que l'activité de programmation *per se* comprenait une large part de temps guidé. En effet, puisque deux tiers des étudiants n'avaient jamais programmé, il a fallu consacrer un certain temps à les accompagner dans l'implémentation du visage. Un autre facteur, conséquent au précédent, est qu'ils n'ont pas eu l'occasion, faute de temps, de créer eux-mêmes un système concurrent pour résoudre un problème.

Table 5. Nombre de solutions données par catégorie (voir section 2 pour la description des catégories)

Catégorie de solutions	solutions pré-test	solutions post-test
C1	13%	12%
C2	20%	21%
C3	28%	30%
D1	27%	20%
D2	12%	17%

Finalement, nous remarquons une différence assez nette avec la distribution des réponses observées par Kolikant (qui s'explique probablement par la différence d'âge et d'expérience de programmation), et une différence légèrement moins marquée avec celle rapportée par Lewandowski qui, comme nous, a réalisé son expérience sur des étudiants sans expérience de programmation (mais moins jeunes). Nous y remarquons une grande baisse des solutions de type décentralisée, déjà observée chez Lewandowski qui l'expliquait par une plus grande facilité des étudiants à ordonner des actions de façon centralisée.

Table 6. Distribution des types de réponses donnée

	Kolikant	Lewandowski	Libert
C1	21.5%	7%	7%
C2	5.9%	9%	13%
C3	5.9%	22%	18%
D1	25.2%	31%	12%
D2	25.2%		10%
Pas raisonnables	16.3%	31%	40%

4.3 Réactions des élèves à l'activité

Lors des différentes activités, nous avons relevé deux réactions intéressantes qui se sont produites dans certaines classes : l'identification d'un goulot d'étran-

glement qui limite les performances du système (*bottleneck*) et la création spontanée de procédures.

Ainsi, dans 3 des 7 classes, après l'activité d'exécution manuelle du visage, des élèves nous ont indiqué que le cerveau contribuait à retarder la transmission des messages, parce que tous les messages lui arrivent, qu'il doit en envoyer plusieurs et qu'il ne sait pas faire plusieurs choses en même temps. Cela se traduit physiquement par la difficulté qu'a l'élève qui exécute les instructions du cerveau à gérer les différents messages qui lui parvient et à envoyer les messages correspondants. Cette discussion a conduit les élèves à se demander s'il ne valait pas mieux envoyer directement les messages aux acteurs concernés, sans passer par un cerveau central. Cette remarque met en évidence que certains jeunes élèves sont capables de se rendre compte qu'un goulot d'étranglement limite les performances d'un système concurrent.

Dans 2 autres classes, lorsqu'ils devaient déterminer un nouveau comportement pour le visage, plusieurs groupes ont choisi de définir de nouveaux mouvements possibles pour différentes parties du visage. Et lorsque ce comportement était semblable dans plusieurs acteurs, ils ont défini des procédures. Ils ont ainsi nommé l'action qui devait en résulter, ils l'ont définie à part et y ont fait référence dans les différents acteurs concernés. Ainsi, le concept de procédure semble relativement aisé à faire émerger chez les élèves au sein d'un cours sur la concurrence.

5 Conclusion

Au regard des résultats obtenus, nous tirons les conclusions suivantes.

Tout d'abord, comme le souligne la partie terminologique du test, la notion de concurrence semble relativement rapide à assimiler pour eux : ils sont capables après deux séances de 100 minutes d'en donner une définition correcte et d'utiliser cette définition pour classifier des systèmes. En outre, les justifications qu'ils donnent pour classifier ces systèmes semblent cohérentes avec la définition donnée. En particulier, ils mettent rapidement de côté les préconceptions comme quoi la concurrence serait liée à une forme de compétition économique ou sportive.

Ensuite, pour relativiser un peu, nous constatons que les deux séances de cours ne suffisent pas à modifier fondamentalement la façon dont ils tentent de résoudre le problème de concurrence fourni. En effet, la répartition des réponses en différentes catégories est relativement semblable au post-test et au pré-test. Nous notons juste une augmentation du nombre de réponses raisonnables et une légère augmentation du nombre de réponses impliquant de la communication explicite entre les agents du système. Nous pensons qu'une ou deux séances supplémentaires permettant la réalisation d'un système concurrent de façon moins guidée permettrait d'améliorer leur capacité à résoudre des problèmes de ce type, car ils y seraient confrontés directement.

Finalement, nous concluons que, étant donné les résultats obtenus et les réactions des élèves face à cette activité sur la concurrence, rien n'indique que l'ap-

prentissage de la programmation concurrente ne soit pas adaptée à des jeunes de 12 à 15 ans. Ils sont capables, en très peu de temps, de comprendre le concept de concurrence, de l'utiliser, de participer à l'implémentation d'un système concurrent par passage de messages et de donner des solutions simples à un problème de synchronisation. Nous observons d'ailleurs que, bien qu'ayant un nombre de réponses raisonnables bien inférieur à ceux rapporté dans les précédentes études (60%, contre 69% et 83.7%), nous avons plus de 60% des étudiants qui ont été capables de fournir au moins une réponse raisonnable. Toutefois, ces résultats devraient être approfondis au moyen de séances supplémentaires qui permettraient aux élèves de se confronter réellement aux problèmes d'un système concurrent et de développer par eux-mêmes des solutions de plus grande taille.

Références

1. ACM/IEEE-CS Joint Task Force on Computing Curricula : Computer science curricula 2013. Tech. rep., ACM Press and IEEE Computer Society Press (2013)
2. Brabrand, C. : Constructive Alignment for Teaching Model-Based Design for Concurrency, pp. 1–18. Springer Berlin Heidelberg (2008)
3. Carro, M., Herranz, A., Mariño, J. : A model-driven approach to teaching concurrency. Trans. Comput. Educ. **13**(1), 5 :1–5 :19 (2013)
4. Feldman, M.B., Bachus, B.D. : Concurrent programming can be introduced into the lower-level undergraduate curriculum. SIGCSE Bull. **29**(3), 77–79 (1997)
5. Kolikant, Y.B.D. : Gardeners and cinema tickets : High school students' preconceptions of concurrency. Computer Science Education **11**(3), 221–245 (2001)
6. Lewandowski, G., Bouvier, D.J., McCartney, R., Sanders, K., Simon, B. : Commonsense computing (episode 3) : concurrency and concert tickets. In : Proceedings of the third international workshop on Computing education research. pp. 133–144. ACM (2007)
7. Libert, C., Vanhoof, W. : Analysis of students' preconceptions of concurrency. In : Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence. pp. 9–12. ACM (2019)
8. Sadowski, C., Ball, T., Bishop, J., Burckhardt, S., Gopalakrishnan, G., Mayo, J., Musuvathi, M., Qadeer, S., Toub, S. : Practical parallel and concurrent programming. In : Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education. pp. 189–194. SIGCSE '11, ACM (2011)
9. Smal, A., Frenay, B., Henry, J. : School-it : une «mallette numérique» pour éduquer à l'informatique (descriptif de poster). In : Didapro 7–DidaSTIC. De 0 à 1 ou l'heure de l'informatique à l'école (2018)
10. Stein, L.A. : Challenging the computational metaphor : Implications for how we think (1999)
11. Sutter, H. : The Free Lunch Is Over : A Fundamental Turn Toward Concurrency in Software. Dr. Dobb's Journal **30**(3) (2005)
12. Van Roy, P., Armstrong, J., Flatt, M., Magnusson, B. : The role of language paradigms in teaching programming. SIGCSE Bull. **35**(1), 269–270 (2003)