DOCTEUR EN SCIENCES

Évaluer la compréhension en informatique à partir des représentations erronées

Henry, Julie

Award date:
2022

Awarding institution:
Universite de Namur

Mesurer la compréhension en informatique à partir des représentations erronées

Julie Henry

Jury

Pr. Margarida Romero
Université Côte d'Azur, France

Pr. Kim Mens
Université de Louvain-La-Neuve, Belgique

Pr. Anne-Sophie Collard
Université de Namur, Belgique

Pr. Benoit Frenay
Université de Namur, Belgique

Pr. Wim Vanhoof (Président)
Université de Namur, Belgique

Pr. Bruno Dumas (Promoteur)
Université de Namur, Belgique

Thèse présentée en vue de l'obtention
du grade de Docteur en Sciences Informatiques

Sous la supervision du Pr. Bruno Dumas

Université de Namur
NAmur Digital Institute (NADI)
Faculté d'Informatique
Centre de recherche PReCISE

Graphisme de couverture : ©Presses Universitaires de Namur

Rue Grandgagnage 19
B - 5000 Namur (Belgique)

ISBN : 978-2-39029-163-3
Dépôt légal : D/2022/1881/11

RÉSUMÉ

Dans la plupart des filières informatiques, les cours de programmation constituent à la fois le point d'entrée et l'obstacle principal pour les étudiants en première année : un obstacle jugé, par avance et par réputation, infranchissable ou sans intérêt par beaucoup d'entre eux. En se basant sur les représentations qu'ils ont de la programmation et de ses concepts-clé, des potentiels candidats optent pour un autre choix de filière avant même d'avoir essayé ou sont poussés vers la sortie prématurément, ratant peut-être leur vocation. Face à ce problème, il est nécessaire de se pencher sur des aspects didactiques aidant à l'enseignement de la programmation : les novices, les représentations erronées, les modèles mentaux et les outils d'évaluation que sont les *concept inventories* (CIs).

Les CIs sont conçus pour identifier rapidement les représentations erronées des apprenants et offrir ainsi aux enseignants du matériel pour améliorer leur enseignement. Dans sa thèse, l'auteure évalue le potentiel de ces outils en étudiant les rôles qu'ils peuvent jouer dans l'enseignement de l'informatique, et dans l'enseignement de la programmation en particulier. Pour ce faire, un travail exploratoire a été mené, mettant en place des CIs dans trois usages spécifiques : identifier les représentations erronées présentes chez les apprenants et/ou les modèles mentaux qu'ils utilisent, mesurer leur fréquence au sein d'un groupe d'apprenants et mesurer le gain de compréhension des apprenants sur une période donnée.

Cinq CIs ont été développés, composés soit de questions à choix multiple, soit d'énoncés textuels avec échelle de Likert : deux sur le concept de variable, un sur le concept d'instruction conditionnelle, un sur le concept de fonction et un sur le concept de *finite state machine*. Ces CIs ont été construits pour répondre à des besoins concrets, issus de contextes différents : un cours d'introduction à la programmation en première année de bachelier, un cours de programmation orientée-objet en deuxième année de bachelier et un cours de modélisation des systèmes embarqués en master.

Les contributions de cette thèse se situent à la fois dans les processus de développement d'outils d'évaluation via la proposition de CIs originaux, mais également dans les résultats de leur administration dans le cadre des cours cités plus haut, qui ont amené notamment à un profilage des apprenants.

Mots clés : didactique, enseignement de la programmation, enseignement de l'informatique, psychologie de la programmation, représentations erronées, modèles mentaux, *concept inventories*

ABSTRACT

In most computer science programs, programming courses are both the entry point and the main obstacle for first-year students : an obstacle that many of them consider, in advance and by reputation, to be insurmountable or irrelevant. Based on the representations they have of programming and its key concepts, potential candidates opt for another choice of program before even trying or are pushed out prematurely, perhaps missing their vocation. Faced with this problem, it is necessary to look at didactic aspects that help in teaching programming : novices, misconceptions, mental models and evaluation tools such as concept inventories (CIs).

CIs are designed to quickly identify learners' misconceptions and thus provide teachers with material to improve their teaching. In her thesis, the author evaluates the potential of these tools by studying the roles they can play in computer science education, and in programming education in particular. To do so, an exploratory work was conducted, implementing CIs in three specific uses : identifying the misconceptions of learners and/or the mental models they use, measuring their frequency within a group of learners, and measuring the gain of understanding of learners over a given period.

Five CIs were developed, consisting of either multiple choice questions or textual statements with Likert scales : two on the concept of variable, one on the concept of conditional instruction, one on the concept of function and one on the concept of finite state machine. These CIs were built to meet concrete needs, coming from different contexts : an introductory programming course in the first year of the bachelor's degree, an object-oriented programming course in the second year of the bachelor's degree and a course on modeling embedded systems in the master's degree.

The contributions of this thesis are both in the processes of development of evaluation tools through the proposal of original CIs, but also in the results of their administration in the framework of the above-mentioned courses, which led in particular to a profiling of the learners.

Keywords : didactic, programming education, computer science education, psychology of programming, misconceptions, mental models, concept inventories

“Marriage is like a coffin and each kid is another nail”.
Homer, philosophe
The Simpsons, Saison 14

*“Well, **she**’s got all the money in the world,*
*but there’s one thing **she** can’t buy... A dinosaur”.*
Homer, toujours philosophe
The Simpsons, Saison 3

Pour Richard et Milo

FIGURE 1 – Extrait de Goupil ou Face [goupil-ou-face.fr], de Lou Lubie

FIGURE 2 – Extrait de Goupil ou Face [goupil-ou-face.fr], de Lou Lubie

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

Des activités d'initiation au "codage" à destination des enfants émergent de partout. "Coder" serait une des compétences essentielles du 21e siècle. Ajoutez à cela la pénurie qui existe, en Belgique et ailleurs dans le monde, dans les métiers dits "du numérique" (incluant les métiers de l'informatique) et vous comprendrez pourquoi l'enseignement de l'informatique revient autant sur le devant de la scène depuis quelques années.

Réfléchir à l'impact d'un enseignement de l'informatique - dès le plus jeune âge, dans un contexte scolaire ou non - est, entre autres, le travail d'un didacticien. Pour que cet enseignement contribue notamment à résoudre les problèmes socio-économiques que sont la fracture numérique et les métiers en pénurie, il importe que cet enseignement soit "bien fait". Notamment, il ne faudrait pas le limiter à l'enseignement de la programmation. De plus, il faudrait des enseignants formés "didactiquement" à cette discipline. Enfin, il faudrait que soient considérés des aspects tels que la problématique du genre ou encore les stéréotypes et préjugés qui existent autour des métiers de l'informatique.

Est-ce que tous les informaticiens passent leurs journées à "Coder" ? Heureusement, non. Les images véhiculées par le métier de programmeur occultent complètement la richesse des métiers accessibles lorsqu'on suit une filière informatique. Et ces images ne sont pas forcément positives. Le stéréotype du geek reste encore très présent : un mâle blanc portant des lunettes, un t-shirt de *gamer*, une coupe de cheveux peu soignée et ayant de l'acné. Il aime les technologies et les super-héros. Les jeunes ajoutent d'autres stéréotypes [1] :

1. Toute ressemblance avec des personnes existant ou ayant existé serait purement fortuite.

— L'informatique est un métier d'hommes (par provocation, les stéréotypes suivants seront exprimés au masculin)
— L'informaticien a réussi des études difficiles
— Il est doué en mathématiques
— Il passe ses journées devant un écran
— Il travaille souvent seul
— Il n'est pas créatif
— Il se nourrit de façon peu équilibrée
— Il s'expose à des problèmes de santé : vue, migraines, dos, addiction
— Il a peu de contacts sociaux

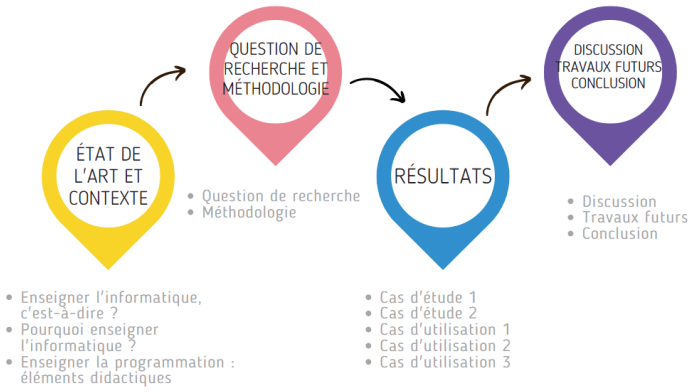
Ces stéréotypes expriment les représentations qu'ont les jeunes du métier de programmeur, essentiellement. Pourtant, programmeur n'est qu'un métier parmi les métiers de l'informatique. Mais c'est le plus connu (voire le seul connu pour certains) et donc celui pris pour illustration. Les jeunes trouvent ce métier (et avec lui, tous les métiers de l'informatique) peu attractif, mais ils savent qu'il y a des opportunités d'emploi. Cela lui fait gagner quelques places dans le classement des métiers qui ont la cote. Un autre argument souvent avancé pour justifier l'intérêt de ce métier est le salaire : selon la rumeur, ça paye bien (voire très bien). Ces raisons qui poussent quelques candidats (pas encore assez nombreux) dans les filières informatiques ne sont pas les bonnes et pourraient expliquer en partie les hauts taux d'échec observés.

En bref, limiter l'enseignement de l'informatique à sa composante "programmation" n'aiderait pas à recruter et, pire encore, ne jouerait pas en la faveur d'une augmentation du nombre de filles dans les filières informatiques. C'est pourtant un objectif sociétal actuel : plus de mixité dans des métiers où sont développées les technologies de demain.

De plus, la question se pose : est-ce utile que tout le monde sache "coder" ? Le slogan "Apprendre à coder pour ne pas être codé" est souvent cité. Pourtant, aujourd'hui, des outils "no code" et des aides automatiques à la programmation facilitent la conception et la création d'applications et questionnent la programmation "à l'ancienne". **Plus que de savoir "coder", il faut surtout comprendre** le fonctionnement et les limites des systèmes programmés. Par exemple, quid si le système de recommandation de Netflix ne récupérait pas vos données d'utilisation ? Autre exemple, l'IA AlphaGo a été programmée pour jouer au jeu de Go, mais est-elle capable de jouer aux échecs ?

L'importance accordée à la programmation dans les écoles, dès le plus jeune âge, est malheureuse et continue au-delà du cursus obligatoire. En effet, dans la plupart des filières informatiques, **les cours de programmation constituent à la fois le point d'entrée et l'obstacle principal** pour les étudiants en première année : un obstacle jugé, par avance et par réputation, infranchissable ou sans intérêt par beaucoup d'entre eux. En se basant sur les représentations qu'ils ont de la programmation et de ses concepts-clé, des potentiels candidats optent pour un autre choix de filière avant même d'avoir essayé ou sont poussés vers la sortie prématurément, ratant peut-être leur vocation.

Bien que consciente des effets de la mise sur un piédestal de la programmation, l'auteure place ce sous-domaine de l'informatique au cœur de ses recherches pour que son enseignement se fasse dans les meilleures conditions possibles. Cette thèse s'intérèsse donc aux aspects didactiques aidant à cet enseignement, et de façon plus large aidant à l'enseignement de l'informatique. L'auteure **évalue**, entre autres, **le potentiel d'un outil d'évaluation conçu pour identifier rapidement les représentations erronées que possèdent les apprenants** (et avec elles, leur compréhension) et offrir aux enseignants du matériel pour améliorer leurs actions d'enseignement. Pour ce faire, un **travail exploratoire** a été mené avec des outils répondant à des besoins contextuels différents et visant des utilisations différentes : identifier les représentations erronées présentes chez les apprenants, mesurer leur fréquence et leur évolution sur une période donnée.



Ce document est structuré en quatre parties. Pour guider la lecture, chaque chapitre débute par un visuel (cfr ci-dessus) et une courte recontextualisation.

La **partie 1** constitue l'état de l'art et pose le cadre dans lequel la recherche s'inscrit. Le **chapitre 2** propose une (re)définition de ce que sont (et seront dans les années à venir) l'informatique et l'enseignement de l'informatique en Europe et en Belgique (contexte spécifique de cette thèse) à travers la littérature scientifique, des textes politiques, des référentiels de compétences et des études menées dans le cadre de cette thèse. L'objectif est de déterminer ce qui existe, ce qui existera et ce qui pourrait exister. Ce chapitre identifie un problème en Belgique francophone : les cours d'informatique sont quasi inexistants et leur contenu est généralement largement insuffisant au regard de ce qui est attendu de l'Europe et fait par ailleurs dans le monde.

Cette problématique amène au **chapitre 3** où se pose la question du pourquoi enseigner l'informatique. Trois motivations sont décrites, avec respectivement des objectifs de développement personnel (la pensée informatique), des objectifs économiques (le manque de main-d'oeuvre) et des objectifs citoyens (la citoyenneté numérique). Ces trois motivations sont détaillées et argumentées à travers la littérature scientifique, des référentiels de compétences et des études menées dans le cadre de cette thèse. Notamment, l'importance des représentations qu'ont les

jeunes de l'informatique est déjà soulignée. En outre, ce chapitre pose les premières pierres d'une réflexion quant à une éducation au numérique visant à former des citoyens réflexifs et critiques.

Enfin, le **chapitre 4** clôture l'état de l'art en se concentrant sur le sous-domaine de l'informatique le plus enseigné dans les écoles de par le monde : la programmation. Puisque la programmation est une des uniques portes d'entrée pour introduire l'informatique dans les écoles en Belgique francophone, il est question ici de soulever des éléments aidant à l'enseignement de cette discipline considérée difficile depuis des années. Il ne s'agit pas de faire une enième revue de la littérature concernant les difficultés des apprenants ou les facteurs de réussite, mais bien de se pencher sur des aspects didactiques tels que la notion de novices en programmation, les représentations erronées et les modèles mentaux en vue de constituer le socle théorique sur lequel repose cette thèse.

La **partie 2** définit la question de recherche au cœur de cette thèse et la méthodologie mise en place pour y apporter des éléments de réponse. Le **chapitre 5** positionne cette recherche en tant que recherche en didactique de l'informatique. Les contributions sont décrites et argumentées au regard de leurs apports dans ce domaine. Enfin, la question de recherche est posée, portant sur le(s) rôle(s) que peuvent jouer des outils d'évaluation dans l'enseignement de l'informatique et de la programmation, en cas d'étude. Le **chapitre 6** s'intéresse notamment au processus de développement et d'administration de l'outil d'évaluation.

La **partie 3** concerne les résultats de cette recherche doctorale et est composée de cinq chapitres. Les **chapitres 7 et 8** présentent les cas d'étude : un cours d'initiation à la programmation et un cours de programmation orientée-objet. Ceux-ci constituent les contextes bien spécifiques dans lesquels ont été développés des outils d'évaluation portant sur des concepts-clé en programmation. Notamment, le deuxième cas d'étude propose une alternative au processus de développement décrit dans la littérature. Les outils développés sont alors utilisés dans trois situations (**chapitres 9 à 11** - les deux cours de programmation cités, plus un cours de modélisation des systèmes embarqués), avec deux objectifs distincts, à savoir mesurer un gain de compréhension chez les apprenants et comparer des populations d'étudiants (en ce qui concerne leur apprentissage).

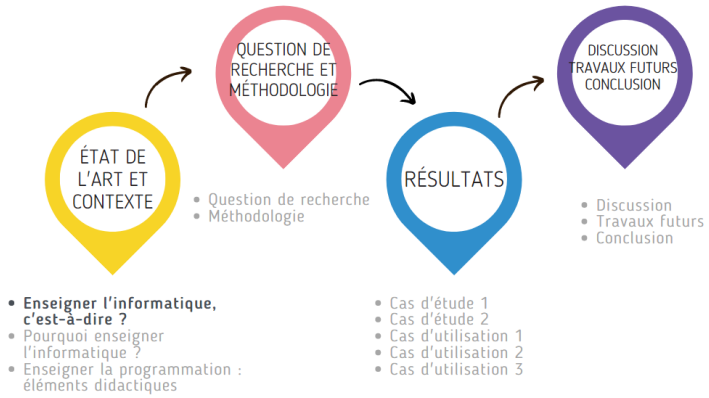
La **partie 4** fait le point sur cette thèse. Le **chapitre 12** revient sur les éléments de réponse apportés aux questions de recherche. Le **chapitre 13** se tourne vers les perspectives qui existent pour donner suite à cette recherche. Enfin, le **chapitre 14** conclut ce travail.

Première partie

État de l’art et contexte

CHAPITRE 2

ENSEIGNER L'INFORMATIQUE, C'EST-À-DIRE ?



Avant toute chose, il s'agit de définir ce qu'est (et ce que devrait/devra être) l'enseignement de l'informatique du point de vue des apprenants (élèves et étudiants - public-cible de cette recherche doctorale), mais également du point de vue de ceux qui sont sensés dispenser cet enseignement, à savoir les enseignants.

Là où les technologies de l'information et de la communication tenaient le rôle majeur dans l'éducation au numérique de tout citoyen, l'informatique commence à se faire une place. Mais quelles sont les compétences informatiques attendues des enseignantes et des apprenants, notamment par l'Europe (cfr Sections 2.2, 2.3 et 2.7) ? Quelles sont les compétences réellement développées chez les apprenants en Belgique francophone (cfr Sections 2.5 et 2.6) ? Enfin, les enseignants sont-ils capables d'aider à prendre en charge le développement de ces compétences (cfr Section 2.7) ?

2.1 "Il y a informatique... et informatique"

En français, les termes *computer science* et *computing* se traduisent tous deux par informatique. Pourtant, en 1989, la *Task Force on the Core of Computer Science* [Denning et al., 1989] insiste pour distinguer ces termes : *computing* est plus large que *computer science* (CS) qu'il englobe au même titre que l'ingénierie informatique (*computer engineering*). "*CS and engineering (aka computing) is the systematic study of algorithmic processes - their theory, analysis, design, efficiency, implementation, and application that describe and transform information. The fundamental question underlying all of computing is, What can be (efficiently) automated*" [Denning et al., 1989, p. 16]. Guzdial va plus loin en ajoutant les sciences de l'information, les technologies de l'information et de la communication (TICs) et le génie logiciel (*software engineering*) au *computing* [Guzdial, 2015].

En 2015, le mouvement éducatif *CS for All* voit le jour aux États-Unis. [1] Il vise à inclure une part d'informatique dans l'éducation ou le développement professionnel de chacun. Guzdial confirme que c'est bien l'informatique dans son sens le plus étroit (aka CS) qui est considérée par ce mouvement. Il s'agit d'enseigner ce qui représente, selon lui, le coeur de la discipline *computing* et qui permet d'en comprendre les différents composants (sans forcément devenir un expert dans chacun de ces composants) [Guzdial, 2015]. Mais quel est cet essentiel à enseigner ? S'agit-il de la programmation ? C'est fort probable. En effet, l'enseignement de l'informatique est souvent (et parfois exclusivement) lié à l'enseignement de la programmation (ou du code) : une vision restreinte qui n'est pas sans conséquence (cfr Section 2.4 et Chapitre 3).

CE QU'IL FAUT EN RETENIR
Les termes *computer science* (CS) et *computing* se traduisent tous deux par informatique. Cependant, *computer science* est une composante de *computing*, au même titre que les technologies de l'information et de la communication (TICs). Dans le cadre de cette recherche, c'est l'enseignement de l'informatique dans son sens le plus étroit (aka CS) qui est considéré, notamment pour garantir une compréhension des composantes du *computing* (entre autres, les TICs) à tous les citoyens.

2.2 Ce qu'en pense l'Europe

En première ligne pour former les citoyens se retrouvent les enseignants. Le **Comité Syndical Européen de l'Éducation** (CSEE) défend depuis longtemps l'intérêt d'un investissement public suffisant et durable dans les infrastructures de formation du personnel éducatif et des établissements d'enseignement, afin que tous les enfants aient accès à une éducation numérique adéquate. Déjà en 2006,

1. Computer Science For All [obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all], consulté le 16/01/2022

le CSEE [2] déclare que "les gouvernements doivent mettre à disposition les fonds nécessaires pour organiser la formation et l'éducation des enseignants. La formation initiale de tout enseignant devrait comprendre une formation aux TICs, à un niveau de compétences et de savoir-faire, sous l'angle à la fois technique et pédagogique de l'utilisation des TICs" (p.29). Selon une enquête menée à la même époque, [3] les TICs sont officiellement intégrées dans les programmes de formation des enseignants pour une majorité des pays membres de l'Europe.

Dix ans plus tard, le CSEE publie un document d'orientation politique [4] qui vise à offrir une vision et une analyse précises de la profession enseignante au 21e siècle et de l'utilisation des TICs. Pour "intégrer l'utilisation des technologies aux programmes scolaires, les enseignants se doivent d'acquérir un grand nombre de compétences et aptitudes" (p.6). Ainsi, les TICs sont décrites comme une aide aux enseignants "dans leur vie professionnelle quotidienne, à la fois sur le plan pédagogique et pratique. C'est la raison pour laquelle chacun d'entre eux devrait posséder une connaissance globale des logiciels et du matériel informatique, notamment les environnements d'apprentissage numérique, les systèmes de contrôle des apprenants, les applications de gestion, l'utilisation des navigateurs pour parcourir la toile, et être capable de participer aux réseaux sociaux et de se familiariser avec des outils et des applications spécifiques à une matière" (p.7). Au niveau des programmes scolaires, il est attendu que soient intégrés des sujets tels que "la manipulation, la présentation et la publication des médias, la conception de sites web, la saisie de données, la gestion des bases de données, la collecte d'informations, les environnements collaboratifs et le partage de fichiers" (p.7). Il est dit "essentiel que chaque enseignant soit formé en vue d'acquérir ces compétences et aptitudes" (p.7). La formation initiale et la formation continue des enseignants sont jugées cruciales pour suivre l'évolution rapide des technologies et des compétences numériques.

Toujours en 2016, le **Service de l'éducation du Conseil de l'Europe** travaille sur l'élaboration de nouvelles orientations politiques et stratégiques [5] qui visent à aider les jeunes à acquérir les compétences dont ils ont besoin pour participer activement et de manière responsable à la société à l'ère du numérique. Il est ainsi suggéré d'introduire dans les programmes scolaires une éducation à la citoyenneté numérique, à savoir "le maniement efficace et positif des technologies numériques (créer, travailler, partager, établir des relations sociales, rechercher, jouer, communiquer et apprendre), la participation active et responsable (valeurs, aptitudes, attitudes, connaissance) aux communautés (locales, nationales, mondiales) à tous les niveaux (politique, économique, social, culturel et interculturel), l'engagement dans un double processus d'apprentissage tout au long de la vie (dans des structures

2. Rapport d'activité du CSEE 2004-2006 [hetice.ulg.ac.be/sites/default/files/csee_act_rep_vol3_fra.pdf], consulté le 16/01/2022

3. La formation des enseignants en Europe [csee-etuce.org/images/attachments/etuce_PolicyPaper_fr.pdf], consulté le 16/01/2022

4. La profession enseignante au 21e siècle [csee-etuce.org/en/resources/policy-papers/1789-the-21st-century-teaching-profession-and-the-use-of-ict-2016], dossier consulté le 16/01/2022

5. Digital citizenship education - Volume 1: Overview and new perspectives [book.coe.int/en/human-rights-democratic-citizenship-and-interculturalism/7452-digital-citzenship-education-volume-1-overview-and-new-perspectives.html], dossier consulté le 16/01/2022

formelles, informelles et non formelles) et la défense continue de la dignité humaine". Par ailleurs, une aide est également évoquée pour les enseignants qui devront mettre en place cette éducation. Trois ans plus tard, le Conseil de l'Europe publie le *Digital Citizenship Education Handbook*[6] et cite le cadre européen **DigComp** (cfr Section 2.3) comme référence pour les compétences numériques attendues de tous les citoyens.

En 2019, le réseau **Eurydice** a publié une évaluation[7] des politiques éducatives numériques des systèmes éducatifs dans 43 pays européens. L'importance des compétences numériques et de la formation au numérique des enseignants y est soulignée. Deux tiers des systèmes éducatifs européens mentionnent les compétences numériques dans leurs cadres de compétences et les rendent ainsi obligatoires dans les programmes de formation initiale : "les compétences numériques spécifiques aux enseignants font partie des compétences essentielles que ceux-ci doivent posséder" (p.11). Si la définition de compétences numériques spécifiques diffère d'un pays à l'autre, le rapport précise que "les enseignants doivent savoir comment intégrer les technologies numériques dans leur enseignement (...) et être capables de les utiliser avec efficacité" (p.11). En outre, les compétences numériques sont décrites comme devant être continuellement mises à jour pour répondre à des technologies en constante évolution et à l'évolution de la société en général. En ce qui concerne l'éducation numérique des enfants, la Belgique francophone est, en 2018, un des rares pays à ne pas développer les compétences numériques dans l'enseignement primaire et secondaire. La réforme du programme d'étude qu'est le **Pacte pour un Enseignement d'Excellence** (cfr Section 2.7) est mentionnée comme une solution à ce manque.

Enfin, en septembre 2020, la **Commission européenne** (CE) publie la mise à jour du **Plan d'action pour l'éducation numérique (2021-2027)**.[8] Ce plan qui vise à soutenir l'utilisation des technologies et le développement des compétences numériques dans l'éducation fait suite au Plan (2018-2020) mis en œuvre dans le cadre "Éducation et formation 2020"[9] (adopté en 2009). Le nouveau plan s'articule autour de deux priorités stratégiques : le développement d'un écosystème d'éducation numérique hautement performant et le renforcement des aptitudes et compétences numériques pertinentes pour la transformation numérique. Pour y parvenir, la CE a notamment l'intention de renforcer l'expertise et la pédagogie des enseignants dans l'utilisation des outils numériques, mais également de mettre à jour le cadre européen des compétences numériques (DigComp 2.1 - cfr Section 2.3). Cette mise à jour devrait notamment avoir un impact sur la formation des enseignants. En effet, il est attendu de ceux-ci qu'ils soient capables de développer chez leurs apprenants :

— "des aptitudes et compétences numériques de base dès le plus jeune âge" ;
— "une culture numérique, y compris pour lutter contre la désinformation" ;

6. Digital Citizenship Education Handbook [rm.coe.int/168093586f], dossier consulté le 16/01/2022
7. Digital Education at school in Europe, consulté le 16/04/2021 [eacea.ec.europa.eu/national-policies/eurydice/content/digital-education-school-europe_en]
8. Digital Education Action Plan (2021-2027) [ec.europa.eu/education/education-in-the-eu/digital-education-action-plan_fr], consulté le 16/01/2022
9. Cadre "Éducation et formation 2020", consulté le 16/01/2022 [ec.europa.eu/education/policies/european-policy-cooperation/et2020-framework_fr]

— "une bonne connaissance et compréhension des technologies à forte intensité de données, telles que l'intelligence artificielle" ;
— "des compétences numériques avancées, qui produisent davantage de spécialistes numériques" ;
— des compétences en *computer science*.

CE QU'IL FAUT EN RETENIR
Jusqu'en 2019, les compétences numériques attendues d'un enseignant se limitaient surtout à l'intégration des TICs dans son enseignement et à leur utilisation efficace dans ses tâches professionnelles. Désormais, les compétences numériques souhaitées sont avancées, spécialisées et visent à développer chez chacun une "citoyenneté numérique" (cfr Section 3.3), telle que définie par le Conseil de l'Europe [a]. En outre, l'informatique (aka CS) est citée dans les discours européens comme un des éléments essentiels dans l'éducation numérique de tout enfant et il incombe aux enseignants d'être en mesure de mettre en oeuvre cette éducation.

a. Manuel d'éducation à la citoyenneté numérique [rm.coe.int/prems-047719-fra-2511-handbook-for-schools-web-16x24/168098f322], consulté le 16/01/2022

2.3 DigComp, l'informatique selon l'Europe

Développé par la CE, le cadre **DigComp** [10] a connu différentes versions : 1.0 en 2013, 2.0 en 2016, 2.1 en 2017 et la version 2.2 en 2022 actuellement en cours de spécification, qui devrait être publiée dans le courant de l'année 2022. Dans sa version 2.0, DigComp comprend 21 compétences regroupées en cinq champs de compétences (cfr Table 2.1). Une lecture rapide de ces compétences (et de leurs descriptions) met en évidence la place prioritaire laissée aux TICs et la réduction des compétences informatiques à la programmation, l'algorithmique (en ce qui concerne la résolution de problème) et à certains aspects de sécurité, même s'il serait possible, pour un enseignant motivé qui voudrait aller plus loin, de développer des aspects techniques dans chacun des cinq champs de compétences.

La version 2.1 décline les compétences de DigComp 2.0 en huit niveaux d'aptitude définis d'après la complexité de la tâche demandée, l'autonomie attendue et le domaine cognitif considéré. Les niveaux 5 à 8 (catégorisés *Advanced* à *Highly specialized*) répondent partiellement aux attentes du Plan d'action pour l'éducation numérique (2021-2027) concernant les "compétences numériques avancées, qui produisent davantage de spécialistes numériques". Les "aptitudes et compétences numériques de base dès le plus jeune âge" seraient, dans le cas de DigComp, définis par les niveaux 1 et 2, à savoir qu'il est attendu que les apprenants agissent sous guidance totale, puis partielle.

Enfin, la version 2.2 complète les versions précédentes en fournissant, entre autres, des "exemples de connaissances, de compétences et d'attitudes aidant les

10. Digital Competence Framework [ec.europa.eu/jrc/en/digcomp], consulté le 07/09/2022

Champ de compétences	Compétences
Information and data literacy	1.1. Browsing, searching and filtering data, information and digital content 1.2. Evaluating data, information and digital content 1.3. Managing data, information and digital content
Communication and collaboration	2.1. Interacting through digital technologies 2.2. Sharing through digital technologies 2.3. Engaging in citizenship through digital technologies 2.4. Collaborating through digital technologies 2.5. Netiquette 2.6. Managing digital identity
Digital content creation	3.1. Developing digital content 3.2. Integrating and re-elaborating digital content 3.3. Copyright and licences 3.4. Programming
Safety	4.1. Protecting devices 4.2. Protecting personal data and privacy 4.3. Protecting health and well-being 4.4. Protecting the environment
Problem Solving	5.1. Solving technical problems 5.2. Identifying needs and technological responses 5.3. Creatively using digital technologies 5.4. Identifying digital competence gaps

TABLE 2.1 – Les 21 compétences de DigComp 2.0

citoyens à s'engager avec confiance, de manière critique et en toute sécurité dans les technologies numériques, ainsi que dans les technologies nouvelles et émergentes telles que les systèmes pilotés par l'intelligence artificielle".

CE QU'IL FAUT EN RETENIR
Si dans ses discours, l'Europe évoque l'informatique (aka CS) et vise la "citoyenneté numérique", sa vision des compétences informatiques à enseigner à chaque citoyen était jusqu'à présent limitée : programmation, algorithmique et sécurité. Pourtant, les compétences en ce qui concerne les TICs sont citées en nombre dans DigComp. Idéalement, ces compétences devraient être précédées de compétences en informatique issues de sous-domaines (cfr Section 2.4) plus nombreux que les trois précédemment cités. Un pas est fait dans la bonne direction avec la mise à jour du référentiel (version 2.2) qui fait la part belle à un sous-domaine supplémentaire : l'intelligence artificielle.

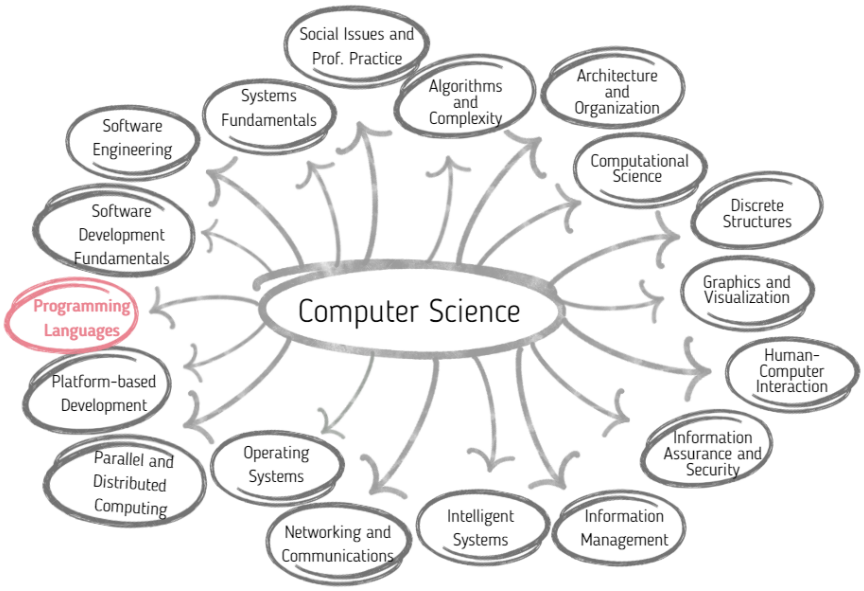


FIGURE 2.1 – Les 18 sous-domaines de l'informatique identifiés par l'ACM et l'IEEE

2.4 "Enseigner l'informatique, c'est-à-dire enseigner la programmation ou le code"

"*CS = programming*" [Denning et al., 2017, Denning et al., 1989]. "*We represent algorithms as the most basic objects of concern and programming and hardware design as the primary activities*" [Denning et al., 1989, p. 9]. Depuis plus de 30 ans, l'informatique est directement (et quasi exclusivement) apparentée à la programmation. Pour déconstruire cette idée fausse, l'*Association for Computing Machinery* (ACM) et l'*Institute of Electrical and Electronics Engineers* (IEEE) ont déployé des efforts considérables [Tedre, 2014], notamment en recommandant la présence, dans les programmes d'enseignement, de l'ensemble des 18 sous-domaines [11] de l'informatique (cfr Figure 2.1). Des recommandations auxquelles ont notamment été attentifs les programmes américains faisant référence dans le domaine lorsqu'il s'agit d'enseigner l'informatique aux jeunes de 6 à 18 ans (*K12*) [12] (cfr Figure 2.2).

Pourtant, l'idée fausse persiste, nourrie par les programmes d'enseignement en informatique qui proposent la programmation en cours majeur lors de la première année de cursus, mais aussi par le succès du mouvement *Learn to Code*. Les clubs de codage parascolaires, les *coding schools* et autres initiatives promettant un apprentissage accessible à tous et un accès rapide à de nombreux emplois bien rémunérés font régulièrement la Une des médias, donnant des arguments aux non-avertis pour

11. ACM & IEEE (2013). Computer Science Curricula 2013 [ieeecs-media.computer.org/assets/pdf/CS2013-final-report.pdf], consulté le 20/12/2021
12. K-12 CS Framework [k12cs.org/] et CSTA K–12 CS Standards [csteachers.org/Page/standards], consultés le 20/12/2021

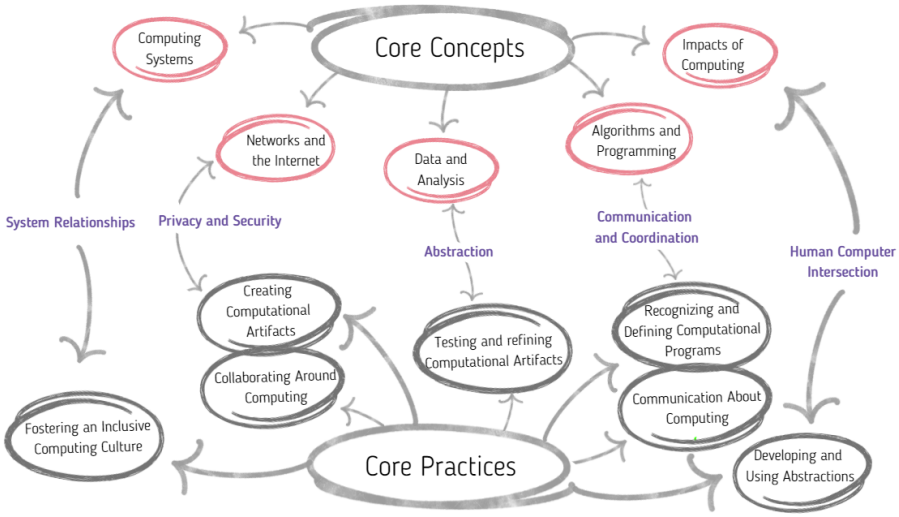


FIGURE 2.2 – *K-12 CS Framework Concepts and Practices*

concentrer l'enseignement de l'informatique sur la programmation, voire sur le code (*coding*).

Dans sa définition, Hartree distingue bien *programming* et *coding* : "*The process of preparing a calculation for a machine can be broken down into two parts, 'programming' and 'coding'. 'Programming' is the process of drawing up the schedule of the sequence of individual operations required to carry out the calculation, and 'coding' is the process of translation these operations in the particular form in which they are read by the machine*" [Hartree, 2012, p. 111]. Dans les années 1950, parce que le *coding* est considéré comme chronophage et fastidieux, la programmation devient l'activité prédominante. L'évolution des langages de programmation a même conduit à considérer la programmation comme une tâche de communication : "*Programming (...) is basically a process of translating from the language convenient to human beings to the language convenient to the computer*" [McCracken, 1957]. Le programme est alors défini comme une séquence automatique et la programmation, comme une spécification mathématique. Il est établi qu'une traduction linguistique (le *coding*) est nécessaire entre les deux et cette action est englobée dans la programmation : "*This sequence [of basic operations] is called the program and the process of preparing it is called programming*" [Wrubel, 1959, p. 4].

CE QU'IL FAUT EN RETENIR
"Informatique = programmation" et "programmer = coder" sont des raccourcis fréquents et qu'il est difficile de combattre. Pourtant, l'informatique est riche de 18 sous-domaines. Il est donc espéré que les référentiels à venir en Europe iront au-delà de ces idées fausses, notamment en s'inspirant des référentiels anglophones basés sur les recommandations de l'ACM et de l'IEEE.

2.5 L'enseignement de l'informatique en Belgique francophone : la situation en 2022

ARTICLES PUBLIÉS INSPIRANT CETTE SECTION

- Joris, N., & Henry, J. (2014). L'enseignement de l'informatique en Belgique francophone : état des lieux. *1024 : Bulletin de la Société Informatique de France*, (2), 107-116.
- Henry, J., & Joris, N. (2016). Informatics at Secondary Schools in the French-Speaking Region of Belgium : Myth or Reality. *Proceedings of the International Conference on Informatics in Schools : Situation, Evolution and Perspectives (ISSEP)*, 13-24.

La définition de ce qu'est l'enseignement de l'informatique en Belgique francophone est envisagée à travers la lecture des programmes d'enseignement organisés dans l'enseignement ordinaire [13].

2.5.1 La structure de l'enseignement ordinaire

L'enseignement ordinaire en Belgique comporte trois niveaux : l'enseignement maternel (élèves de 3 à 6 ans), l'enseignement primaire (de 6 à 12 ans) et l'enseignement secondaire (de 12 à 18 ans). L'obligation scolaire va de 5 à 18 ans.

L'enseignement secondaire ordinaire se répartit en trois degrés couvrant deux années chacun : le premier degré (D1 - élèves de 12 à 14 ans, maximum 16 ans), le deuxième degré (D2 - élèves de 14 à 16 ans) et le troisième degré (D3 - élèves de 16 à 18 ans).

En ce qui concerne le premier degré, seul le premier degré commun (1C-2C) accessible aux élèves titulaires, en sortie de primaire, du certificat d'études de base sera cosidéré. Ce degré propose un tronc commun, une option obligatoire et une option au choix. Les deuxième et troisième degrés comprennent deux sections (transition T et qualification Q) et quatre formes d'enseignement (général G, technique T, artistique A et professionnel P). Les enseignements technique et artistique peuvent être organisés en section de transition (TT et AT) et en section de qualification (TQ et AQ). L'enseignement général est organisé en section de transition et l'enseignement professionnel, en section de qualification (cfr Figure 2.3). Ces deux degrés proposent une formation en quatre composantes : un tronc commun, une option obligatoire, une option au choix et un renforcement. L'importance accordée à chaque composante dépend de la forme d'enseignement et toutes les écoles ne proposent pas toutes les formes. Seules les formes d'enseignement G et TT (dont sont issus majoritairement les étudiants inscrits à l'université) seront considérées.

13. Consultables et téléchargeables via les plateformes officielles des Pouvoirs Organisateurs et Fédérations correspondantes.

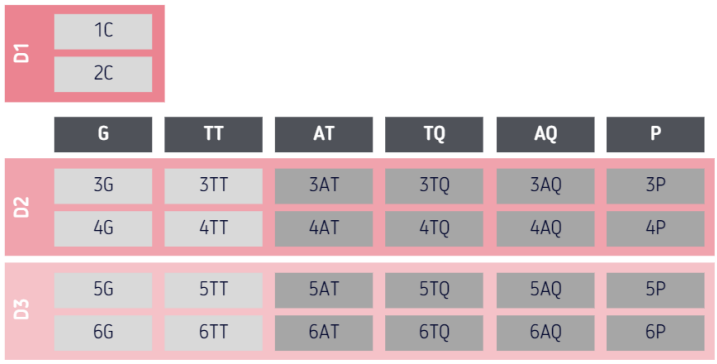


FIGURE 2.3 – Le système d'enseignement secondaire en Belgique

2.5.2 Notice de lecture d'un programme d'enseignement

Dans la région francophone de Belgique, les écoles sont regroupées en trois réseaux : le réseau de la Fédération Wallonie-Bruxelles (FWB), le réseau officiel subventionné (OS) et le réseau libre subventionné (LS). Parce que chaque réseau dispose de ses propres programmes, il est intéressant de considérer chaque réseau indépendamment des autres, de façon à pouvoir faire des comparaisons. En outre, cela nécessite d'homogénéiser la description des programmes d'enseignement en répondant, pour chacun, à trois questions évidentes :

- À quel public est destiné ce programme ?
- À quel(s) objectif(s) répond-il ?
- Quel est son contenu ?

En ce qui concerne le contenu, pour faciliter les comparaisons, un vocabulaire et un système d'abréviations ont été définis (cfr Table 2.2).

TABLE 2.2 – Système d'abbréviations utilisé dans cette section

Matériel informatique (hardware)	**HW**
Système d'exploitation	**SE**
App. de traitement de texte	**TXT**
Tableur	**TAB**
App. de présentation assistée par ordinateur	**PAO**
App. de traitement d'image	**TIM**
App. de traitement de son/de video	**TSV**
Base de données	**BDD**
Navigateur	**NAV**
Moteur de recherche	**MDR**

Recherche d'informations	**INFO**
Langage HTML	**HTML**
Langages XHTML/CSS	**XCSS**
Courrielleur	**EMAIL**
Réseau	**NET**
Codage de l'information	**CODA**
Logique mathématique	**MATH**
Algorithmique	**ALGO**
Programmation	**PROG**
Citoyenneté numérique	**CN**

2.5.3 L'informatique dans le réseau de la Fédération Wallonie-Bruxelles

Trois programmes d'enseignement en informatique sont proposés dans le réseau de la FWB, à savoir "Initiation à l'informatique/Informatique" (FWB-1), "Informatique" (FWB-2) et "Informatique de gestion" (FWB-3). Ils couvrent le D1, ainsi que l'ensemble des formes d'enseignement G et TT en D2 et D3.

Initiation à l'informatique/Informatique (FWB-1)

À quel public est destiné ce programme? Ce programme constitue une activité complémentaire (option obligatoire) au D1 et une option au choix dans les formes d'enseignement G et TT en D2-D3.

À quel(s) objectif(s) répond-t-il? Il est clairement annoncé que le cours d'informatique "doit avant tout être un cours de formation générale visant à atteindre des compétences transversales". Il n'est dès lors pas étonnant d'y retrouver des objectifs généraux tels que : "contribuer à l'épanouissement individuel de chaque élève" et "former l'élève au travail en équipe", entre autres. Sont cités au même titre "initier aux techniques et connaissances de base nécessaires à l'utilisation de l'informatique", "développer des attitudes critiques justifiées vis-à-vis de tout ce qui touche à l'informatique et au traitement automatique des données", "initier l'élève à la philosophie de l'informatique", "montrer la prééminence de l'esprit humain sur l'informatique", "être conscient des dangers, des limites de l'informatique et de la responsabilité de chacun dans ce domaine en faisant preuve de citoyenneté", ou encore "structurer sa pensée".

Quel est son contenu? Pour tenir compte des situations très différentes d'une école à l'autre (notamment du point de vue matériel), ce programme est composé de modules.

En D1, quatre modules sont proposés : (1) maîtriser la base de l'outil informatique, (2) produire et exploiter des documents, (3) exploiter des sources d'information numérique et (4) communiquer au moyen de la messagerie électronique.

En D2, l'apprenant est amené travailler trois modules : (1) présenter un document en traitement de texte, (2) concevoir une feuille de calcul au moyen d'un tableur et (3) utiliser un logiciel de présentation assistée par ordinateur.

Enfin, en D3, les modules doivent être considérés soit comme une synthèse des apprentissages abordés en D1 et D2 soit comme un approfondissement à ces mêmes apprentissages. Les modules proposés sont alors : (1) présenter une lettre simple en traitement de texte et utiliser le publipostage pour l'envoi, (2) exploiter les bases d'un tableur, (3) traiter une image numérique et (4) intégrer les logiciels de la suite bureautique.

Est spécifiée, pour chacun des modules précités, une liste de "savoir-faire" en guise de pistes didactiques. À titre d'exemples, on peut citer pour le D1 "utiliser le clavier", "utiliser les boutons gauche et droit de la souris à bon escient", "allumer/éteindre correctement l'ordinateur", "utiliser la fonction paragraphe", "utiliser les formats bordure et trame" et "lancer un navigateur internet". Au D2, les pistes énoncées sont "disposer un texte en style américain", "utiliser le dictionnaire des synonymes", "se déplacer dans la feuille de calcul de ligne en ligne", "produire un montage permettant de servir de base à une présentation orale", entre autres. Enfin, durant le D3, l'enseignant peut mettre en place des savoir-faire tels que "présenter une lettre simple en disposition bloc à la marge", "créer et exploiter des tableaux" ou "intégrer une image dans un traitement de texte".

Informatique (FWB-2)

À quel public est destiné ce programme ? Ce programme constitue une option au choix en D2-D3, dans l'enseignement TT.

À quel(s) objectif(s) répond-t-il ? Ce programme ne vise pas à "former des informaticiens, mais bien à utiliser l'informatique comme instrument privilégié pour développer les compétences et les savoirs requis prévus". Il s'agit de "faire acquérir aux élèves des mécanismes de pensée et des méthodes conformes, face à des situations de travail concrètes".

Quel est son contenu ? Le contenu est structuré par degré. En D2, 14 modules sont proposés. Six modules sont exclusivement abordés en troisième année (3TT) : "éléments d'histoire de l'informatique", "les composants d'un système informatique" , "les unités de mesure en informatique", "organisation des données informatiques", "utilisation de l'informatique dans une optique multimédia" et "conduite de projet multimédia". Quatre autres sont répartis entre les troisième et quatrième années : "le système d'exploitation", "logiciels de type courant" (le tableur, en 4TT), "représentation du nombre dans un système informatique" et "utilisation des concepts de base de la logique et de l'algorithmique". Enfin, quatre sont réservés à la quatrième année : "les règles déontologiques liées à l'usage de l'informatique", "maintien du système informatique en ordre de marche", "recherche d'informations sur Internet" et "conduite de projets de programmation".

En D3, le programme est également composé de 14 modules, légèrement différents de ceux proposés en D2. Six modules sont organisés en cinquième année (5TT) : "fondamentaux de l'informatique", "logiciels de bureautique", "gestion pérenne et sécurisée des données", "hardware", "gestion responsable de l'outil informatique" et "base et arithmétique". Les modules "multimédia", "création de site Web" et "bases de la programmation" sont répartis sur l'ensemble du degré. Enfin, cinq modules sont prévus exclusivement pour la sixième année (6TT) : "transmission et réseaux", "gestion de base de données", "programmation orientée objet", "conduite de projet multimédia" et "conduite de projet de programmation".

Informatique de gestion (FWB-3)

À quel public est destiné ce programme ? Ce programme constitue une option au choix au D3, dans l'enseignement G.

À quel(s) objectif(s) répond-t-il ? Le programme spécifie à plusieurs reprises (et en gras) que "le but de ce cours n'est pas de former des informaticiens" et que "l'informatique doit rester un outil au service d'autres disciplines". Il n'est dès lors pas étonnant de ne retrouver que peu (voire pas) d'objectifs directement liés à l'informatique parmi les quatre cités : "prendre conscience de l'importance de - l'outil informatique, des nouvelles TIC (NTICs), du tableur, des bases de données - dans la vie professionnelle et la vie quotidienne", "établir des relations entre les notions d'économie et - l'outil informatique, les NTICs, le tableur, les bases de

données", "assimiler un vocabulaire et des techniques spécifiques" et "résoudre des situations-problèmes recourant - à l'outil informatique, aux NTICs, au tableur, aux bases de données".

Quel est son contenu ? Le contenu est structuré par année et par thème. Trois thèmes sont proposés en 5G, à savoir (1) les notions de systèmes d'exploitation, (2) les NTICs et (3) le tableur. Ce dernier thème sera approfondi en 6G où est également abordé un nouveau thème : (4) les bases de données.

Pour chacun des thèmes, le programme énonce une série de "contenus d'apprentissages obligatoires" et les "situations d'apprentissage associées". Ainsi, dans le thème (1), les contenus identifiés sont "la barre des tâches", "la personnalisation du bureau et du menu démarrer" et "l'entretien du matériel", entre autres. Les situations d'apprentissage précisent un environnement Windows : "démarrer Windows", "utiliser l'aide de Windows". Une fois de plus, on est loin de la méthodologie des invariants. De plus, certaines situations paraissent complètement désuètes : "formater une disquette", "sauver sur disquette". En ce qui concerne le thème (2), l'appellation NTICs désigne ici Internet, le courrier électronique et le scanner. Un choix d'intitulé quelque peu interpellant. Un choix de contenu qui l'est tout autant : "la recherche d'une information", "la création d'un site Web", "les principes de bases" d'un courrielleur et "les principales caractéristiques" du scanner font partie des contenus obligatoires. "Utiliser un moteur de recherche", "créer une première page Web", "envoyer un message" et "scanner un document" sont les situations d'apprentissage associées. Le thème (3) est à cheval sur deux années. C'est le thème le plus important du programme (en terme de nombres de périodes qui lui sont consacrées). Sont repris comme contenus "la construction d'une feuille de calcul", "les formules et fonctions", "la mise en forme d'une feuille de calcul", "la représentation graphique des données", entre autres. Les situations d'apprentissage consistent, par exemples, à "écrire une formule", "régler la hauteur des lignes", "recourir à l'assistant graphique". Enfin, le thème (4) fait la part belle à Access en citant comme contenu "l'impression à l'aide d'un état".

CE QU'IL FAUT EN RETENIR

Dans le réseau de la Fédération Wallonie-Bruxelles, l'enseignement de l'informatique offre une grande diversité dans les thèmes abordés : tous les thèmes listés à la Table 2.2 sont couverts par les différents programmes d'enseignement. Cependant, tout le monde n'a pas accès à cet enseignement (cfr Table 2.3) :

- Aucun thème n'est abordé dans le tronc commun ;
- La majorité des thèmes font partie d'une option au choix dans les deuxième et troisième degrés. Seuls quatre thèmes sont obligatoires au premier degré, à savoir matériel informatique, application de traitement de texte, moteur de recherche et courrielleur ;
- Les parcours d'enseignement en informatique les plus complets sont organisés en technique de transition.

TABLE 2.3 – Inventaire des thèmes informatiques enseignés dans le réseau de la FWB

	FWB-1 (G & TT)			FWB-2 (TT)		FWB-3 (G)
	D1	D2	D3	D2	D3	D3
HW						
SE						
TXT						
TAB						
PAO						
TIM						
TSV						
BDD						
NAV						
MDR						

	FWB-1 (G & TT)			FWB-2 (TT)		FWB-3 (G)
	D1	D2	D3	D2	D3	D3
INFO						
HTML						
XCSS						
EMAIL						
NET						
CODA						
MATH						
ALGO						
PROG						
CN						

	Thème enseigné dans le cadre d'une option obligatoire
	Thème enseigné dans le cadre d'une option au choix
	Thème non enseigné

2.5.4 L'informatique dans le réseau officiel subventionné

Dans le réseau OS, un seul programme d'enseignement est proposé (dans certaines Provinces [14]) : "Électronique et informatique".

(Électronique) et informatique

À quel public est destiné ce programme? Ce programme constitue, dans les Provinces où il est organisé, une option au choix aux D2 et D3 de l'enseignement TT.

À quel(s) objectif(s) répond-t-il? La partie "informatique" de ce cours comprend un cours théorique et des laboratoires (pratique). Les objectifs de la partie théorique sont les suivants : "connaître les grandes étapes de l'histoire de l'informatique", "nommer et distinguer les périphériques, composants de l'unité centrale et composants d'un ordinateur", "utiliser les unités de numérotation informatique", "réaliser un algorithme simple" et "expliquer et utiliser différents systèmes de codage". Les laboratoires visent pour leur part la "maîtrise des fonctionnalités de base et avancées d'un système d'exploitation", "l'utilisation d'un navigateur web et d'un moteur de recherche", "l'utilisation et la gestion d'une messagerie", "l'utilisation d'un traitement de texte et d'un tableur" et la "réalisation d'un algorithme d'analyse de problème".

Quel est son contenu? En ce qui concerne le cours théorique, les contenus décrits n'ajoutent pas grand-chose aux objectifs si ce n'est une répartition entre les deux années durant lesquelles ce cours est dispensé.

Par contre, pour les laboratoires, sont cités comme savoir-faire liés aux fonctionnalités de base d'un système d'exploitation "utiliser la souris", "utiliser la barre de défilement", "démarrer un programme", entre autres. "Gérer les fichiers depuis

14. Les programmes du réseau OS dépendent des Pouvoirs Organisateurs des différentes Provinces. Si la base est commune, il existe des variantes d'une Province à l'autre.

l'explorateur" et "supprimer et restaurer un fichier" sont des exemples de fonctionnalités avancées. "Comprendre les navigateurs Web" et "les moteurs de recherche" sont définis comme savoirs sans qu'aucune autre explication ne soit donnée. Il en est de même pour les autres points de matière à aborder : "premiers pas avec le traitement de texte", "premiers pas avec un tableur", "utiliser formules et fonctions", "travailler avec des fonctions", "mettre en forme une page", etc.

CE QU'IL FAUT EN RETENIR
Dans le réseau officiel subvenionné, certains thèmes ne sont pas proposés au sein des cours d'informatique : les applications de traitement du son et de la vidéo, les langages HTML, XHTML et CSS, mais aussi la citoyenneté numérique (cfr Table 2.4). En outre, l'unique programme d'enseignement en informatique n'est accessible qu'aux déuxième et troisième degrés de l'enseignement technique de transition et qui plus est, seulement dans certaines provinces. Dans ces conditions, il y a un grand risque qu'une majorité d'élèves n'aura pas l'opportunité de suivre un seul cours d'informatique sur l'ensemble de son cursus secondaire.

TABLE 2.4 – Inventaire des thèmes informatiques enseignés dans le réseau OS

	D2 (TT)	D3 (TT)
HW		
SE		
TXT		
TAB		
PAO		
TIM		
TSV		
BDD		
NAV		
MDR		

	D2 (TT)	D3 (TT)
INFO		
HTML		
XCSS		
EMAIL		
NET		
CODA		
MATH		
ALGO		
PROG		
CN		

	Thème enseigné dans le cadre d'une option obligatoire
	Thème enseigné dans le cadre d'une option au choix
	Thème non enseigné

2.5.5 L'informatique dans le réseau libre subventionné

Dans le réseau LS, l'informatique est considérée comme une discipline à part entière. Celle-ci est dès lors présente dans l'ensemble des établissements scolaires

du réseau. Deux programmes d'enseignement sont proposés : "Informatique - D2" et "Informatique - D3".

Informatique - D2

À quel public est destiné ce programme ? Ce programme constitue une option obligatoire au D2 de l'enseignement TT.

À quel(s) objectif(s) répond-t-il ? Il est clairement annoncé que "l'objectif de cette option n'est pas de former des informaticiens". L'enseignant doit, entre autres, être attentif à "former les élèves à mobiliser les savoirs opérationnels fondamentaux (définis par la suite), les concepts fondamentaux de l'informatique et leurs capacités cognitives ; former les élèves à collaborer ; former des élèves responsables". À noter que de nombreuses compétences reprises dans le programme relèvent autant, sinon plus, des cours généraux (compétences transversales) que des cours de l'option (compétences disciplinaires).

L'option est composée de deux parties : (1) informatique et (2) exploitation de logiciels informatiques. Sont citées comme compétences disciplinaires en informatique "connaître et comprendre l'architecture de base d'un ordinateur et de son environnement matériel", "connaître et utiliser les rôles du système d'exploitation et les différentes familles de logiciel", "comprendre les mécanismes de transformation et de manipulation de l'information", "maîtriser les concepts logiques de base de l'algorithmique" et "comprendre et manipuler les concepts du langage HTML".

La partie intitulée "exploitation de logiciels informatiques" voit énoncées les compétences suivantes : "exploiter les savoir-faire spécifiques aux logiciels", "comprendre et manipuler les technologies de base d'Internet et des réseaux" et "avoir un regard critique sur l'informatique dans la société".

Quel est son contenu ? Chacune des deux parties composant l'option proposent plusieurs thématiques.

En informatique, "le PC et son environnement (hardware et software)", "le codage de l'information", "le langage HTML" et "la programmation" sont les sujets à développer. Sont ainsi vus les différents composants d'un ordinateur et leurs interactions, mais aussi les différents types de logiciels et leur domaine d'application. L'enseignant doit tendre vers une vulgarisation de l'utilisation et des pertinences de choix des principaux matériels et logiciels "sans jamais entrer dans des considérations très techniques". Par ailleurs, la numérisation de l'information est introduite de façon simple (exemples concrets) pour permettre de comprendre comment l'ordinateur traite l'information (traitement formel). La théorie des couleurs fondamentales est brièvement expliquée. La création de pages Web est envisagée en deux étapes : l'apprentissage des principales balises d'HTML et la mise en forme des pages (blocs, ergonomie et initiation aux feuilles de style). Enfin la dernière section aborde les concepts fondamentaux de la programmation : primitives de lecture et d'écriture, notions de variables, constantes, types, instruction d'affectation, la logique booléenne, les structures de contrôle, etc.

La partie "exploitation de logiciels informatiques" se divise en six thématiques qui sont "l'apprentissage du clavier", "logiciel de traitement de texte", "logiciel tableur", "logiciel de présentation assistée par ordinateur", "Internet et réseaux" et "informatique et société". Comme son nom l'indique, cette partie de programme vise essentiellement le développement de compétences nécessaires à l'exploitation de fonctions de base des logiciels cités. Enfin, "l'informatique et la société" sensibilise aux sites à risques, explicite les différents types de licences d'utilisation, sensibilise aux métiers de l'informatique, etc.

Informatique - D3

À quel public est destiné ce programme? Ce programme constitue une option obligatoire au D3 de l'enseignement TT. Il fait logiquement suite au programme du même nom proposé en D2, tout en lui étant antérieur en écriture.

À quel(s) objectif(s) répond-t-il? Les objectifs annoncés sont identiques à ceux énoncés dans le programme en D2.

Quel est son contenu? L'option traite quatre sujets : (1) informatique, (2) programmation langages, (3) systèmes d'exploitation et logiciels et (4) multimédia. Chaque partie est divisée en différents thèmes, proposant chacun des contenus spécifiques. La répartition du contenu n'est pas identique dans les deux programmes et apparaît plus logique dans le programme en D2.

Ainsi, "représentation et traitement de l'information" reste un thème important du sujet (1), complété par les thèmes "informatique et société" et "Algèbre booléenne".

En (2), l'enseignant a le choix entre "paradigme procédural" et "paradigme orienté objet". Dans le premier cas, une analyse descendante, une programmation structurée et des savoirs "algorithmiques" sont privilégiés à la connaissance de la syntaxe. Les matières abordées sont la structure de données, les procédures et fonctions (sans distinguer ces deux concepts) et les algorithmes "réutilisables" (permuter le contenu de deux variables, déterminer le maximum de trois valeurs, trier un tableau par ordre croissant, etc). En ce qui concerne, le choix du "paradigme orienté objet", l'accent est mis sur la description de l'environnement de l'application et les mécanismes propres à la programmation orientée objet (classe, objet, instance, encapsulation, héritage, etc.). Dans les deux cas, la liberté est donnée (si le temps le permet) d'aborder des points de matières communs tels que la programmation basée sur les objets (résolution de problèmes par sélection d'objets disponibles), les "types de base considérés comme objets" et les objets des interfaces graphiques (boutons d'options, liste déroulantes, menus, barres d'outils, etc.).

Comme son nom l'indique, le sujet (3) permet d'aborder notamment des notions telles que la définition et les fonctions d'un système d'exploitation, les interfaces homme-machine, la gestion de fichiers, les réseaux (locaux et distants), les serveurs et les clients. Les logiciels (tableur et traitement de texte) sont, à ce stade, considérés comme vus en ce qui concerne les notions de base. Ils sont donc abordés dans

ce programme dans le cadre de documents plus complexes. Il est suggéré de ne pas enseigner "les trucs et ficelles spécifiques d'un logiciel donné". "Il ne s'agit pas d'enseigner le mode d'emploi d'un logiciel". Enfin, les bases de données sont également abordées en deux temps : le gestionnaire de base de données relationnel et le logiciel "client".

En (4), le "traitement des images" permet de revoir et d'approfondir ce qui a déjà été abordé en D2 : le codage et les couleurs. Les actions de base réalisables sur des images "bitmap" (rogner, retourner, inverser, etc.) et vectorielles (formes de base, courbes de bézier) sont abordées. Le "traitement du son" suit une logique semblable : révision du codage et traitement basique des fichiers "waves" (montage de séquences). Les autres matières prévues sont le logiciel de présentation assistée par ordinateur dans ses fonctions élémentaires, la "page Web" (langage XHTML et CSS, respect des normes, ergonomie, etc.) et le codage des séquences vidéo.

CE QU'IL FAUT EN RETENIR
Les programmes d'enseignement de l'informatique dans le réseau libre subventionné sont des options obligatoires dans la filière technique de transition. Dès lors, seuls les élève suivant cette filière seront initiés à l'ensemble des thèmes listés à la Table 2.2 (cfr Table 2.5).

TABLE 2.5 – Inventaire des thèmes informatiques enseignés dans le réseau LS

	D2 (TT)	D3 (TT)
HW		
SE		
TXT		
TAB		
PAO		
TIM		
TSV		
BDD		
NAV		
MDR		

	D2 (TT)	D3 (TT)
INFO		
HTML		
XCSS		
EMAIL		
NET		
CODA		
MATH		
ALGO		
PROG		
CN		

	Thème enseigné dans le cadre d'une option obligatoire
	Thème enseigné dans le cadre d'une option au choix
	Thème non enseigné

2.6 Le bagage en informatique des jeunes en sortie de secondaire ordinaire

ARTICLES PUBLIÉS INSPIRANT CETTE SECTION

- Henry, J., & Joris, N. (2015). Le bagage TIC des étudiants en Belgique francophone. État des lieux. In G-L. Baron, E. Bruillard, & B. Drot-Delange (eds.), *Informatique en éducation : perspectives curriculaires et didactiques*. Presses Universitaires Blaise-Pascal.
- Henry, J., & Joris, N. (2016). Informatics at Secondary Schools in the French-Speaking Region of Belgium : Myth or Reality. *Proceedings of the International Conference on Informatics in Schools : Situation, Evolution and Perspectives (ISSEP)*, 13-24.

Avec quel bagage en informatique les jeunes sortent-ils de l'enseignement obligatoire ? Une question à laquelle il est difficile de répondre compte tenu de la structure de l'enseignement en Belgique francophone. Des éléments peuvent toutefois être apportés : théoriquement, au regard des programmes d'enseignement (le bagage "attendu") et pratiquement, en menant une enquête auprès des jeunes eux-mêmes (le bagage "mesuré").

2.6.1 Le bagage "attendu" en informatique

La lecture des différents programmes d'enseignement en informatique et l'identification des thèmes qui les composent permettent de définir le bagage en informatique que chaque apprenant pourrait posséder à la sortie de son cursus obligatoire (primaire-secondaire). Ce bagage "attendu" apparait variable en fonction du réseau et de la forme d'enseignement secondaire (G - cfr Table 2.6 - ou TT - cfr Table 2.7).

Dans la forme d'enseignement G, le bagage "attendu" en informatique (BAI) est nul pour les apprenants dans les réseaux OS et LS. En ce qui concerne l'enseignement TT, si un apprenant choisit de suivre tous les cours d'informatique proposés par son école, il pourrait posséder un bon niveau en informatique en sortie de cursus. Néanmoins, les écoles sont libres d'organiser ou non les options en fonction de leur succès ou sur décision du Pouvoir Organisateur responsable. En 2016, dans le réseau de la FWB, seules 75 écoles proposant la forme d'enseignement G (sur 118 - 63%) et 10 écoles proposant la forme TT (sur 39 - 25%) organisaient des cours d'informatique. Dans le réseau OS, c'est le cas de seulement 10 écoles proposant la forme d'enseignement TT (sur 45 - 22%). Enfin, dans le réseau LS, seules 22 écoles proposant la forme TT (sur 124 - 17%) étaient concernées. Cela signifie qu'il était alors facile pour un apprenant de terminer son parcours d'enseignement obligatoire sans avoir suivi le moindre cours d'informatique.

TABLE 2.6 – BAI dans chacun des réseaux, pour l'enseignement G

	FWB	OS	LS
HW			
SE			
TXT			
TAB			
PAO			
TIM			
TSV			
BDD			
NAV			
MDR			

	FWB	OS	LS
INFO			
HTML			
XCSS			
EMAIL			
NET			
CODA			
MATH			
ALGO			
PROG			
CN			

	Thème enseigné dans le cadre d'une option obligatoire
	Thème enseigné dans le cadre d'une option au choix
	Thème non enseigné

TABLE 2.7 – BAI dans chacun des réseaux, pour l'enseignement TT

	FWB	OS	LS
HW			
SE			
TXT			
TAB			
PAO			
TIM			
TSV			
BDD			
NAV			
MDR			

	FWB	OS	LS
INFO			
HTML			
XCSS			
EMAIL			
NET			
CODA			
MATH			
ALGO			
PROG			
CN			

2.6.2 Le bagage "mesuré" en informatique

Les BAI de chaque réseau donnent une idée de l'informatique qui pourrait être dispensée dans les écoles secondaires en Belgique francophone si les programmes d'enseignement étaient effectivement organisés. Cependant, parce que le manque de formation des enseignants est souvent pointé du doigt lorsqu'il s'agit d'enseigner l'informatique [Henry and Smal, 2018, Henry and Joris, 2013], la question se pose de savoir si les thèmes composant ces BAI sont réellement abordés dans les classes.

Pour déterminer le bagage "mesuré" en informatique (BMI), deux enquêtes ont été menées : une première, en 2013, auprès d'étudiants inscrits en première bachelier à l'Université (apprenants sortant de l'enseignement obligatoire) et une seconde, en 2016, auprès de professeurs d'informatique en place dans des écoles secondaires.

Enquête auprès des étudiants

Cette première enquête a été envoyée par courriel à 950 étudiants de première bachelier de l'Université de Liège (soit 17,9% des inscrits durant cette année académique). L'enquête comprenait un maximum de 12 questions portant sur le cursus des étudiants dans l'enseignement secondaire : réseau, forme d'enseignement et thèmes abordés en informatique (cfr Table 2.2).

Sur les 170 étudiants ayant répondu, 145 seulement correspondent aux critères fixés pour permette la comparaison avec les BAI, à savoir un enseignement secondaire s'étant déroulé en Belgique francophone et de forme G ou TT. Parmi ces 145 étudiants, 89 (61,4%) signalent ne pas avoir eu l'occasion, durant leur cursus, de suivre des cours d'informatique. Parmi eux, 36 étudiants (42,8%) ont fréquenté le réseau FWB, 2 (2,4%) le réseau OS, 46 (54,8%) le réseau LS et 5 (non considérés) plusieurs réseaux au cours de leur cursus.

Des 56 (38,6%) à avoir suivi des cours d'informatique, 17 étudiants recommencent leure première année et ont donc été retirés de l'échantillon pour éviter toute confusion avec les cours d'informatique proposés à l'université. Enfin, 6 étudiants ont changé de réseau durant leur cursus. Les données collectées ne permettant pas de tenir compte de ces parcours spécifiques, ces étudiants n'ont pas été considérés.

Sur les 33 étudiants constituant l'échantillon, ils sont 9 (27,3%) à avoir suivi un cursus dans une école du réseau FWB (sous-échantillon 1), 3 (9,1%) dans une école du réseau OS (sous-échantillon 2) et 21 (63,6%) dans une école du réseau LS (sous-échantillon 3). Tous ont étudié dans la forme d'enseignement G en D2 et D3.

La répartition entre les sous-échantillons est représentative de la population scolaire dans l'enseignement secondaire de transition (G et TT), en D2-D3 : le réseau LS accueille 65% des apprenants, le réseau OS 9,9% et le réseau FWB 25,1% [15].

Dans l'enquête, il a été demandé aux étudiants si 14 thèmes [16] de la Table 2.2 avaient ou non été abordés durant leur cursus. Le répondant pouvait émettre un doute ("je ne suis pas certain(e)"). Les résultats ainsi obtenus ont été comparés aux BAI de l'enseignement G, pour chacun des réseaux (cfr Table 2.6).

En ce qui concerne les résultats du sous-échantillon 1 (n = 9 - cfr Table 2.8), les programmes de la FWB semblent être partiellement utilisés par les enseignants. Six des 11 thèmes présents dans le BAI sont acquis par plus de 50% des étudiants. Il s'agit principalement de thèmes liés aux logiciels de bureautique, au navigateur et à la recherche d'informations. Les autres (matériel, système d'exploitation, bases de données, langage HTML et courrielleur) semblent avoir moins la cote. Des thèmes qui n'auraient pas du être abordés, car non prévus dans les programmes, le sont

15. Indicateurs 2013 - Population scolaire, par réseau, dans l'enseignement fondamental et secondaire [enseignement.be/index.php?page=23827do_id=10376do_check=], consulté le 17/01/2022

16. L'enquête a été administrée en parallèle de l'analyse des programmes d'enseignement et n'a donc pas bénéficé des apports de cette lecture, à savoir l'identification des 20 thèmes couramment proposés par les programmes. N'ont pas été cités dans l'enquête les thèmes application de traitement de son/de vidéo, les langages XHTML et CSS, le réseau, la logique mathématique et la citoyenneté numérique. Le thème moteur de recherche a été intégré au thème recherche d'informations.

malgré tout dans des faibles proportions (maximum 22%) : application de traitement d'image, réseau, codage de l'information et programmation.

TABLE 2.8 – Sous-échantillon 1 : BMI *vs* BAI pour l'enseignement G du réseau FWB - * *exprime le doute du répondant*

	oui	non	?*	BAI
HW	33,3	33,3	33,3	
SE	44,4	22,2	33,3	
TXT	**88,8**	11,1		
TAB	**88,8**	11,1		
PAO	**66,6**		33,3	
TIM	22,2	**55,5**	22,2	
TSV				
BDD	33,3	33,3	33,3	
NAV	**66,6**		33,3	
MDR				

	oui	non	?*	BAI
INFO	**77,7**		22,2	
HTML	22,2	**55,5**	22,2	
XCSS				
EMAIL	33,3	11,1	**55,5**	
NET				
CODA	22,2	33,3	44,4	
MATH				
ALGO		**66,6**	33,3	
PROG	11,1	**66,6**	33,3	
CN				

Les résultats du sous-échantillon 2 (n = 3 - cfr Table 2.9) laissent penser qu'il existe, dans le réseau OS, des cours d'informatique non liés à un programme d'enseignement officiel. Comme pour le réseau FWB, les thèmes les plus cités sont liés aux logiciels de bureautique : application de traitement de texte et tableur. Six thèmes sur les 14 évalués par les étudiants sont cités par plus de 66% d'entre eux comme n'ayant jamais été abordés.

TABLE 2.9 – Sous-échantillon 2 : BMI *vs* BAI pour l'enseignement G du réseau OS - * *exprime le doute du répondant*

	oui	non	?*	BAI
HW	33,3	33,3	33,3	
SE	33,3		**66,6**	
TXT	**66,6**		33,3	
TAB	**100**			
PAO	33,3	33,3	33,3	
TIM	33,3	**66,6**		
TSV				
BDD		**66,6**	33,3	
NAV	33,3	33,3	33,3	
MDR				

	oui	non	?*	BAI
INFO		**66,6**	33,3	
HTML	33,3	**66,6**		
XCSS				
EMAIL	33,3	**66,6**		
NET				
CODA		33,3	**66,6**	
MATH				
ALGO		**100**		
PROG		**100**		
CN				

Comme pour le réseau OS, les résultats du sous-échantillon 3 (n = 21 - cfr Table 2.10) semblent indiquer l'existence de cours non issus d'un programme officiel. 6 thèmes sur 14 ont été vus par plus de 50% des étudiants. De nouveau, il s'agit des thèmes les plus souvent enseignés, à savoir les logiciels de bureautique, le navigateur et la recherche d'informations, mais aussi le système d'exploitation. Plus de 60% des étudiants affirment n'avoir jamais eu l'occasion d'apprendre des thèmes tels que base de données, langage HTML, algorithme ou programmation.

TABLE 2.10 – Sous-échantillon 3 : BMI *vs* BAI pour l'enseignement G du réseau LS - * *exprime le doute du répondant*

	oui	non	?*	BAI
HW	47,6	19,0	33,4	
SE	**52,4**	9,5	38,1	
TXT	**95,2**		4,8	
TAB	**61,9**	14,3	23,8	
PAO	**66,7**	14,3	19,0	
TIM	23,8	42,9	33,3	
TSV				
BDD	4,8	**61,9**	33,3	
NAV	**52,4**	28,6	19,0	
MDR				

	oui	non	?*	BAI
INFO	**57,1**	14,3	28,6	
HTML	9,5	**85,7**	4,8	
XCSS				
EMAIL	28,6	47,6	23,8	
NET				
CODA	19,0	42,9	38,1	
MATH				
ALGO		**71,4**	28,6	
PROG	19,0	**61,9**	19,1	
CN				

CE QU'IL FAUT EN RETENIR

En 2013, selon les résultats obtenus via une enquête menée auprès de 145 primo-arrivants :

- Ils sont un peu plus de 60% à n'avoir pas eu l'occasion de suivre un cours d'informatique durant leur cursus obligatoire. À noter que, dans l'enseignement général, deux réseaux sur trois ne disposent d'aucun programme officiel en informatique.
- Ils sont un peu moins de 40% à disposer d'un bagage en informatique acquis à l'école. Il semble donc que des cours soient dispensés malgré l'absence de programme officiel.
- Ces cours sont souvent centrés sur les outils de bureautique.
- De façon logique, l'algorithmique et la programmation, non incluses dans le bagage "attendu" en informatique en ce qui concerne l'enseignement général, sont majoritairement non enseignées.

Enquête auprès des enseignants

Cette deuxième enquête a été envoyée, en 2016, par courriel à deux listes de diffusion créées dans le cadre de projets liés à l'informatique dans l'enseignement secondaire : CoP-PR-TIC et visaTICE. L'enquête comprenait un maximum de 18 questions portant sur les enseignants eux-mêmes (formation initiale, expérience), les écoles où ils travaillaient (réseau, forme d'enseignement, niveau) et le contenu de leurs cours d'informatique. Le nombre total d'enseignants en informatique en Belgique francophone n'étant pas connu, les résultats de cette enquête doivent être considérés avec précaution.

Parmi les 36 enseignants ayant complété l'enquête, seuls 21 peuvent être pris en compte : un enseignant en D1 et 20 en D2 et/ou D3, dans les formes d'enseignement G et/ou TT. Confrontés aux résultats de l'enquête auprès des étudiants, les réponses fournies par l'échantillon, en ce qui concerne l'enseignement G, vont permettre de se faire une idée de l'informatique qui y est enseignée.

Dans un premier temps, le profil sociodémographique des enseignants en informatique peut être dressé. La majorité de ceux-ci sont des hommes (65,2%) et ont plus de 41 ans (73,9%). Le nombre moyen d'années d'expérience est de 17,3 (écart-type = 10,3) et le nombre moyen d'années d'expérience en tant qu'enseignant en informatique, de 15,1 (écart-type = 8,85). En termes de formation initiale, une grande diversité est observée : 13 enseignants (sur 21, 60,8%) ont suivi un cursus orienté "sciences exactes" (ingénierie (5), mathématiques (2), sciences (4) ou informatique (2)), 7 enseignants ont suivi un cursus orienté "sciences humaines et sociales" (économie (4), comptabilité (2) et études commerciales (1)). La majorité des participants enseignent dans le réseau LS (17, 80,9%) et les autres, dans le réseau FWB.

Sur les 4 enseignants du réseau FWB, aucun n'enseigne au niveau D1, un enseigne au niveau D2 et 4 au niveau D3. L'enseignant du niveau D2 utilise le programme officiel de l'enseignement G (FWB-1) (cfr Table 2.3) qui aborde les thèmes TXT, TAB et PAO. Dans le niveau D3, 3 enseignants travaillent dans l'enseignement G : un n'utilise pas de programme officiel et les deux autres utilisent l'un ou les deux programmes officiels fournis (FWB-1 et FWB-3) (cfr Table 2.3). De façon générale, la plupart des thèmes (14/19) repris par le BAI du réseau FWB (cfr Table 2.6) sont enseignés par plus de 50% des enseignants (cfr Table 2.11).

TABLE 2.11 – Informatique enseignée en D3 de l'enseignement G (n = 3) *vs* BAI, dans le réseau FWB

		BAI
HW	**66.6**	
SE	**66.6**	
TXT	**66.6**	
TAB	**66.6**	
PAO	**66.6**	
TIM	**66.6**	
TSV		
BDD	100	
NAV	**66.6**	
MDR	**66.6**	

		BAI
INFO	**66,6**	
HTML	**66,6**	
XCSS		
EMAIL	**66,6**	
NET	33,3	
CODE		
MATH		
ALGO		
PROG	33,3	
CN		

Sur les 17 enseignants du réseau LS, un enseigne au niveau D1, 11 enseignent au niveau D2 et 14 au niveau D3. Malgré qu'il n'existe pas de programme d'enseignement officiel au D1, l'enseignant concerné aborde 4 thèmes avec ses classes : HW, TXT, INFO et EMAIL. Au niveau D2, 7 enseignants (sur 11) travaillent dans l'enseignement G. Parce qu'il n'existe pas de programme officiel pour cette forme d'enseignement, 6 d'entre eux utilisent le programme officiel de l'enseignement TT. Le dernier n'utilise aucun programme. Seuls 2 thèmes sont enseignés par plus de 50% des enseignants de D2 : TXT et TAB (cfr Table 2.12).

Au niveau D3, 8 enseignants (sur 14) travaillent dans l'enseignement G. Parce qu'il n'y pas de programme d'enseignement officiel, 5 enseignants utilisent le programme de l'enseignement TT. Les autres ne se reposent sur rien. Les thèmes majoritairement enseignés sont liés aux outils de bureautique (cfr Table 2.13).

TABLE 2.12 – Informatique enseignée en D2 de l'enseignement G (n = 7) *vs* BAI (G et TT), dans le réseau LS

		BAI (G)	BAI (TT)
HW			
SE	28,4		
TXT	**71,4**		
TAB	**71,4**		
PAO	42,6		
TIM	28,4		
TSV			
BDD	28,4		
NAV	28,4		
MDR	28,4		

		BAI (G)	BAI (TT)
INFO	42,6		
HTML	28,4		
XCSS			
EMAIL	14,2		
NET	14,2		
CODE	14,2		
MATH	14,2		
ALGO	14,2		
PROG	14,2		
CN	14,2		

TABLE 2.13 – Informatique enseignée en D3 de l'enseignement G (n = 8) *vs* BAI (G et TT), dans le réseau LS

		BAI (G)	BAI (TT)
HW	25		
SE	37,5		
TXT	**75**		
TAB	**87,5**		
PAO	**62,5**		
TIM	37,5		
TSV			
BDD	25		
NAV	12,5		
MDR	25		

		BAI (G)	BAI (TT)
INFO	25		
HTML	**50**		
XCSS			
EMAIL	37,5		
NET	25		
CODE	12,5		
MATH			
ALGO	12,5		
PROG	25		
CN			

CE QU'IL FAUT EN RETENIR

En 2016, les résultats obtenus via une enquête menée auprès de 21 enseignants en informatique semblent plus positifs que ceux obtenus trois ans auparavant auprès des primo-arivants :

- Les outils bureautiques sont les thèmes majoritairement enseignés dans les réseaux de la Fédération Wallonie bruxelles et libre subventionné, dans l'enseignement général. À noter qu'aucune donnée n'a été collectée pour le réseau officiel subventionné.
- Dans le réseau de la Fédération Wallonie-Bruxelles, les enseignants prennent en charge une bonne partie des thèmes annoncés dans les programmes d'enseignement officiels.
- Dans le réseau libre subventionné, bien qu'aucun programme d'enseignement n'existe, des cours d'informatique sont dispensés à tous les niveaux. Cela confirme les résultats obtenus auprès des primo-arrivants.

— Par ailleurs, certains enseignants déclarent utiliser les programmes de l'enseignement technique de transition, sans pour autant prendre en charge l'ensemble des thèmes.
— Les thèmes les plus avancés sont soit non enseignés, soit peu enseignés. Ainsi, que ce soit dans le réseau la Fédération Wallonie-Bruxelles ou le réseau libre subventionné, les jeunes ont peu de chance d'acquérir des compétences en algorithmique et en programmation durant leur cursus obligatoire.

2.7 L'avenir de l'enseignement de l'informatique en Belgique francophone

ARTICLE PUBLIÉ INSPIRANT CETTE SECTION

- Henry, J., & Smal, A. (2018). "Et si demain je devais enseigner l'informatique ?" Le cas des enseignants de Belgique francophone. In Parriaux, J, Pellet, J.P., Baron, G.L., Bruillard, E., & Komis, V. *De 0 à 1 ou l'heure de l'informatique à l'école*. Lausanne, Suisse, Peter Lang Verlag, 129-150.

Cette recherche doctorale a été menée en parallèle à une réforme du tronc commun de l'enseignement obligatoire en Belgique francophone, à savoir touchant les élèves de 5 à 15 ans. Cette réforme vise notamment à garantir une littératie numérique pour tous en intégrant des cours de "numérique". Un référentiel a donc été construit, inspiré par le référentiel européen DigComp : le référentiel de Formation Manuelle, Technique, Technologique et Numérique (FMTTN). Saisissant l'opportunité d'introduire l'informatique à l'école, les établissements d'enseignement supérieur de la FWB ont étaboré leur propre référentiel. Si ce dernier n'aura probablement jamais l'occasion d'être mis en place, il a permis de mesurer la réalité du terrain en ce qui concerne les compétences des enseignants qui seraient amenés à dispenser les cours de "numérique" du référentiel FMTTN.

2.7.1 Le référentiel de Formation Manuelle, Technique, Technologique et Numérique

En 2015, le **Pacte pour un Enseignement d'Excellence** [17] est élaboré. Son objectif est de "déployer au sein du monde de l'enseignement une démarche transversale de qualité en vue, notamment, de renforcer la qualité de l'offre d'enseignement pour chaque apprenant - principalement en améliorant la formation initiale et continuée (...) des acteurs de l'enseignement (...) ; ensuite, en adaptant nos savoirs,

17. Le Pacte pour un Enseignement d'Excellence [rfie.ares-ac.be/boite-a-outils/referentiels-duTC], consulté le 15/04/2021

compétences et pratiques pédagogiques aux besoins de la société du 21e siècle, en intégrant la révolution numérique (...)".

Les changements planifiés par le Pacte ont engendré "la mise en œuvre d'une formation polytechnique et pluridisciplinaire s'incarnant en particulier, au sein d'une plus grande variété de domaines d'apprentissage, dans le développement d'une formation technologique, manuelle et numérique visant notamment à assurer une littératie numérique à chaque élève". Le référentiel de Formation Manuelle, Technique, Technologique et Numérique [18] fait partie des référentiels qui remplaceront les socles de compétences du tronc commun (cours obligatoires), à savoir de la première primaire (P1) à la troisième secondaire (S3). Il est composé de deux volets liés (cfr Figure 2.5) : le volet "Formation manuelle, technique et technologique" et le volet "Numérique".

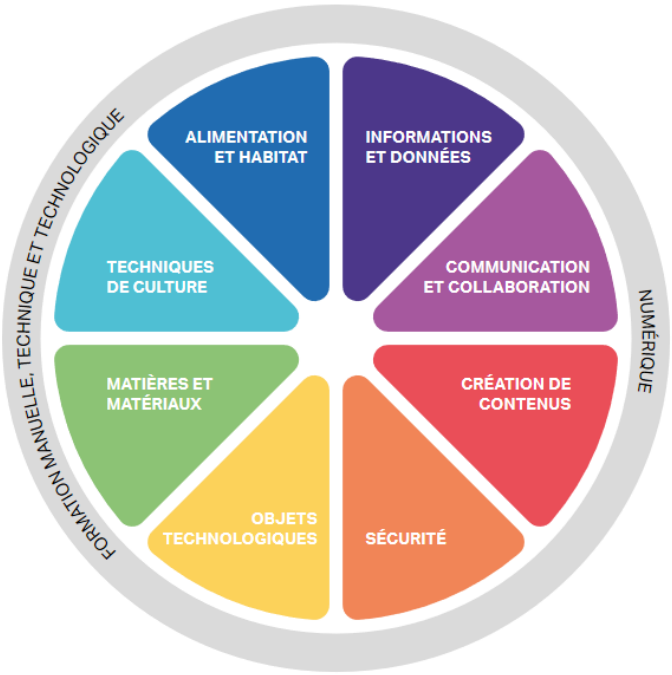


FIGURE 2.4 – Les volets du référentiel FMTTN

Le volet numérique se base à la fois sur DigComp 2.0 et sur les compétences en éducation aux médias proposées par le Conseil Supérieur de l'Éducation aux Médias [19]. Parmi les quatre champs composant ce volet, certains contenus visent plus particulièrement les dimensions informationnelle, technique et sociale propres à l'éducation aux médias numériques. C'est principalement le cas des champs "com-

18. Référentiel de Formation Manuelle, Technique, Technologique et Numérique, version provisoire consultée le 19/01/2022 [rfie.ares-ac.be/boite-a-outils/referentiels-duTC]

19. Les compétences en éducation aux médias - Cadre général [csem.be/sites/default/files/2021-01/cadre-competences-education-aux-medias-portefeuille-activites-pedagogiques-2016.pdf], consulté le 19/01/2022

munication et collaboration" et "sécurité", mais aussi, dans une moindre mesure, le champ "informations et données".

	P1	P2	P3	P4	P5	P6	S1	S2	S3
Formation manuelle, technique et technologique									
ALIMENTATION		✘		✘		✘			✘
HABITAT		✘			✘			✘	
TECHNIQUES DE CULTURE			✘		✘				✘
MATIÈRES ET MATÉRIAUX	✘	✘		✘		✘	✘	✘	
OBJETS TECHNOLOGIQUES				✘		✘	✘	✘	✘
Numérique									
INFORMATIONS ET DONNÉES			✘	✘			✘	✘	
COMMUNICATION ET COLLABORATION					✘		✘		✘
CRÉATION DE CONTENUS			✘	✘	✘	✘	✘	✘	✘
SÉCURITÉ						✘	✘		

FIGURE 2.5 – Répartition des contenus d'apprentissage par champ thématique et par année d'études - *P = primaire, S = secondaire*

En réalisant, comme pour tous les programmes d'enseignement en informatique organisés en Belgique francophone, un inventaire des contenus (cfr Table 2.14), il apparait que certains thèmes ne sont pas abordés. Par contre, d'autres sont à ajouter à la liste : les médias sociaux, les application de collaboration et de partage, l'outil de communication qu'est la messagerie instantanée. En outre, le thème CN est bien plus riche que dans les programmes d'enseignement existant dans les trois réseaux : identité numérique, vie privée, protection des données, droits de propriété, droits à l'image, cyberharcèlement, *fake news*, netiquettes, etc.

TABLE 2.14 – Inventaire des thèmes enseignés dans le référentiel FMTTN

	FMTTN
HW	
SE	
TXT	
TAB	
PAO	
TIM	
TSV	
BDD	
NAV	
MDR	

	FMTTN
INFO	
HTML	
XCSS	
EMAIL	
NET	
CODE	
MATH	
ALGO	
PROG	
CN	

2.7.2 Le référentiel de Sciences Informatiques pour le Secondaire Inférieur

Dans le cadre de Digital Wallonia, stratégie numérique de la Wallonie, et du Pacte d'Excellence, les sections informatiques des universités et hautes écoles francophones ont été mandatées par la Région Wallonne, en accord avec la FWB, pour élaborer un projet cohérent de cursus pour le premier degré du secondaire inférieur et mettre en place une offre de formation et d'accompagnement des (futurs) enseignants/formateurs d'enseignants. Un consortium a donc été créé, dénommé "Sciences Informatiques pour le Secondaire Inférieur" ({SI}²) [20].

Au moment de créer ce consortium, le contenu du référentiel FMTTN n'était pas connu. Il était juste annoncé la réintroduction d'un cours de "numérique". L'hypothèse a alors été faite, au vu de la composition du consortium, que l'informatique y tiendrait un rôle important. Un référentiel de compétences a été rédigé, largement inspiré par les expériences menées ailleurs dans le monde : le programme Numérique et Sciences Informatiques [21] en France, *Computing At School* [22] au Royaume-Uni, ou encore *CS for All* [23] aux États-Unis. Les compétences à développer chez les apprenants y sont organisées en cinq thématiques : représentation des données, algorithmique, programmation, matériel, réseau et sécurité (cfr Annexe A).

Pour identifier les besoins en termes de formation chez les enseignants (et futurs enseignants), une étude a été menée en 2016-2017. Il s'agissait de déterminer qui, parmi eux, étaient en l'état capables d'assurer un enseignement des sciences informatiques (tel que prévu par le référentiel {SI}²), et d'identifier, si ça n'était pas le cas, les thématiques de formation continue à privilégier. Dans le cadre de cette recherche doctorale, les résultats sont à discuter à la lumière des thématiques abordées dans le cadre du référentiel FMTTN.

Collecte de données auprès des enseignants

La récolte de données est passée par enquête composée de trois volets. Le premier volet permet de dresser le portait des participants : leur sexe, leur âge, s'ils enseignent ou sont en cours de formation, leur (futur) titre didactique (instituteur/institutrice, agrégation de l'enseignement inférieur/supérieur, certificat d'aptitude pédagogique aproprié à l'enseignement supérieur, etc.), le titre de leur formation initiale, la discipline enseignée et le niveau où elle est organisée (le cas échéant). Dans un second volet, les enseignants sont invités à auto-évaluer, au moyen d'une échelle de Likert, leurs compétences à enseigner chacun des items composant le référentiel {SI}². Enfin, le troisième volet demande aux enseignants de se déclarer leur intérêt à enseigner ce référentiel.

20. sicarre.be [https://sicarre.be/], consulté le 19/01/2022
21. Programme de numérique et sciences informatiques de terminale générale [cache.media.eduscol.education.fr/file/SPE8_MENJ_25_7_2019/93/3/spe247_annexe_1158933.pdf], consulté le 25/01/2022
22. Teaching Computing Curriculum [teachcomputing.org/curriculum], consulté le 25/01/2022
23. CS for All [csforall.org], consulté le 25/01/2022

Cette enquête a été diffusée auprès de 355 directions d'établissements organisant un enseignement secondaire en Belgique francophone, auprès du corps enseignant des Hautes Écoles, via les listes de diffusion du consortium {SI}² et du projet HETICE [24], ainsi qu'aux sections scientifiques organisant l'AESS à l'Université de Namur. Il a été demandé aux destinataires du courriel accompagnant l'enquête de diffuser celle-ci auprès d'un maximum de (futurs) enseignants/futurs enseignants concernés par son sujet.

293 réponses ont été récoltées, venant d'enseignants de tous niveaux : fondamental (30), secondaire inférieur (99), secondaire supérieur (104) et supérieur non universitaire (69).

Si le référentiel FMTTN cible l'ensemble du tronc commun, les compétences numériques ne seront dispensées qu'à partir de la P3 (cfr Figure 2.5). En outre, une lecture approfondie des savoir-faire et compétences listées montrent des attendus plus typés "informatique" dans le secondaire. Les résultats discutés ci-après sont donc centrés sur les 99 participants qui enseignent (ou enseigneront) au niveau secondaire inférieur.

Parmi ceux-ci, 73,7% sont en cours de formation pour acquérir un titre pédagogique. Le taux de participation moins élevé des enseignants en place peut laisser penser qu'ils se sont sentis moins concernés par l'enquête. Toutefois, il peut également s'agir d'un manque de communication entre les directions ayant réceptionné le courriel et leur corps enseignant, là où il est facile pour un professeur en Haute École/Université de diffuser l'enquête auprès de futurs enseignants. Il est intéressant d'obtenir les réponses d'enseignants "en devenir" qui constituent sans doute le public le plus susceptible [25] de se retrouver du cours de "numérique".

Enfin, les participants été catégorisés sur base de leur formation initiale : informatique, mathématiques, formation à caractère scientifique et autres. Cette dernière catégorie reprend tous les diplômes qui, de prime abord, sont moins susceptibles de proposer des sciences informatiques (et donc d'avoir développé chez les enseignants les compétences associées) dans leur programme. Parmi les participants, 5 sont issus du domaine informatique, 34 ont une formation de base en mathématiques, 30 sont plutôt scientifiques et 30 ont un diplôme majoritairement issus des sciences dites humaines.

Profil informatique des enseignants

Concernant les items issus du référentiel {SI²}, il est intéressant de constater qu'à part les enseignants ayant une formation initiale en informatique (5), les autres témoignent des lacunes pour un nombre plus ou moins important d'items. Une première analyse globale des résultats montre, en effet, que, pour chacune des thématiques du référentiel, les informaticiens sont majoritaires à répondre "tout-à-fait" à la question "j'estime être capable d'enseigner..." (cfr Figures 2.6- 2.10). Ils sont généralement suivis, mais de loin, par les mathématiciens. Excepté pour les théma-

24. Hetice [hetice.ulg.ac.be], consulté le 25/01/2022

25. Les jeunes enseignants se voient souvent attribuer les cours à faible charge horaire et n'ayant pas de lien avec une discipline en particulier.

tiques "Algorithmique" et "Matériel", les mathématiciens répondent majoritairement "pas du tout". En ce qui concerne les algorithmes, il n'est pas surprenant de voir cette matière maîtrisée par un certain nombre de mathématiciens, celle-ci faisant souvent partie de leur cursus de formation initiale. L'aspect matériel, quant à lui, présente une répartition plus homogène des enseignants (mis à part les informaticiens).

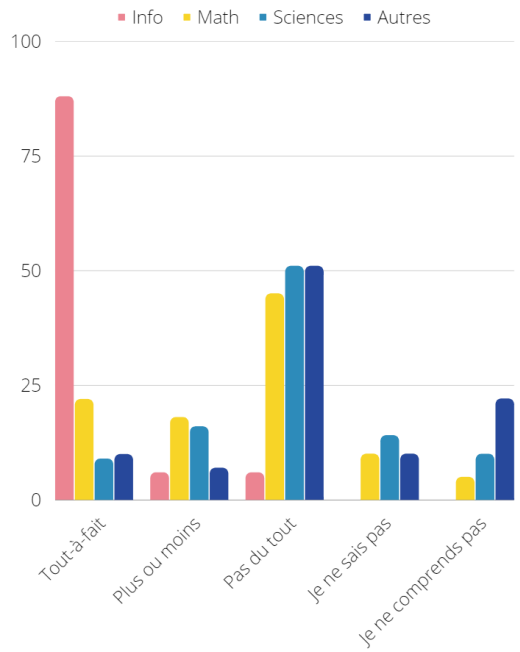


FIGURE 2.6 – Résultats (en %) pour la thématique "Représentation des données"

Ces résultats sont à nuancer de par le fait qu'il s'agit d'une auto-évaluation basée sur la seule compréhension d'un énoncé. Le participant pouvait d'ailleurs stipuler son incompréhension le cas échéant.

En conclusion, outre les informaticiens (dont il est logique qu'ils possèdent les compétences nécessaires), il semble que ce soit les mathématiciens qui soient, en l'état actuel, les plus à même d'enseigner les sciences informatiques. Les résultats des scientifiques (chimie, physique, biologie, etc.) s'avèrent plus négatifs que prévu, dans le sens où il n'existe quasiment pas de différence entre eux et les enseignants possédant une formation initiale orientée sciences humaines et sociales (et supposés à priori moins à l'aise avec l'informatique).

Pour une analyse plus détaillée permettant de déterminer les items les plus problématiques, aucune distinction entre les catégories d'enseignants n'est faite (cfr Figures 2.6-2.10). Il apparaît ainsi que, mise à part la thématique "Matériel", les enseignants se sentent peu capables d'enseigner ce qui constitue le référentiel {SI²}. En effet, ils répondent majoritairement (entre 40 et 50%) "pas du tout" lorsqu'il s'agit de s'exprimer quant à leurs capacités d'enseigner la représentation des données, l'algorithmique, la programmation et le couple réseau-sécurité.

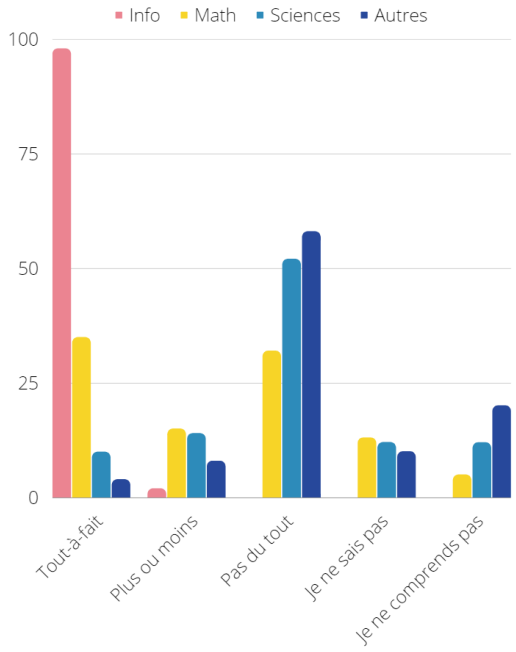


FIGURE 2.7 – Résultats (en %) pour la thématique "Algorithmique"

Pour chaque thématique et pour l'ensemble des items la composant, des tendances globales ont été calculées (cfr Figure 2.11). Celles-ci font apparaitre des profils similaires en ce qui concerne les thématiques "Représentation des données", "Algorithmique" et "Programmation". Les items des thématiques "Matériel", "Réseau et Sécurité" sont visiblement plus connus des enseignants.

À un item (R9 - cfr Annexe A) près, c'est plus d'un enseignant sur trois qui ne sent pas capables d'enseigner la thématique de "Représentation des données" (cfr Figure 2.6). Pour sept items sur treize, plus d'un enseignant sur deux avouent leurs lacunes. Pour les items R11 à R13, moins de 10% se sentent "tout-à-fait" capables, environ 60% ne s'en sentent pas capables et 15% ne comprennent tout simplement pas de quoi il s'agit. Les items présentant la plus grosse différence entre les enseignants qui se disent capables et ceux qui pensent ne pas l'être sont principalement axés sur le traitement des images (R7, R8 et R10). C'est également le cas pour les items ayant à coeur de susciter la discussion entre l'enseignant et les apprenants ("discuter des avantages et des inconvénients…", "expliquer l'intérêt…") notamment à propos des standards d'encodage de caractères (R6) et des techniques de cryptographie (R11 à R13).

Les thématiques d'"Algorithmique" et de "Programmation" présentent des profils fortement similaires, y compris lorsqu'on y regarde plus en détails (cfr Figures 2.7 et 2.8), à savoir des items non maîtrisés pour plus de 40% des enseignants (exception faite du A1). Il existe cependant une différence notable quant au enseignants qui se sentent capables d'enseigner ces matières. Concernant l'algorithmique,

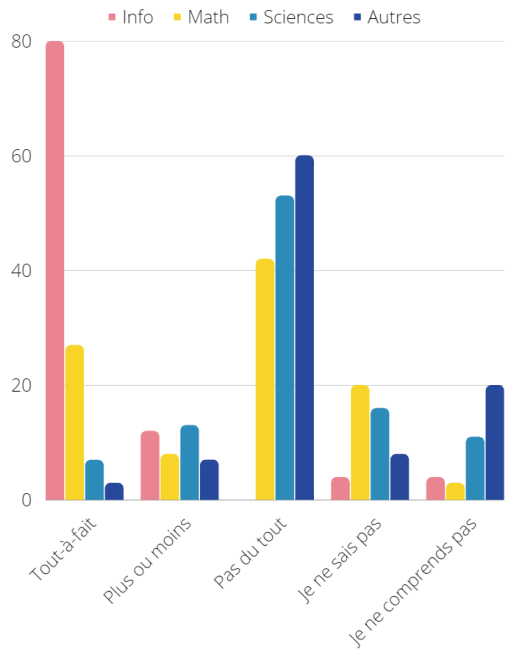


FIGURE 2.8 – Résultats (en %) pour la thématique "Programmation"

un enseignant sur cinq s'estime capable d'enseigner onze items sur les quinze. Au niveau de la "Programmation", ça n'est le cas que pour un seul item sur les dix proposés.

Comme dit, précédemment, la thématique "Matériel" semble être la mieux maîtrisée par les enseignants. Les items "basiques" (M1 à M3) sont les plus connus (cfr Figure 2.9). Les deux autres posent problème à plus de 40% des enseignants. Dans l'ensemble, les items composant cette thématique sont compris par la quasi totalité des enseignants, ce qui pourrait s'expliquer par une présence plus importante des concepts liés à cette thématique dans le quotidien de monsieur tout-le-monde.

Enfin, les réactions par rapport à la thématique "Réseau et Sécurité" sont plus mitigées : trois items étant maîtrisés par un tiers des enseignants (cfr Figure 2.10), tandis que sept autres posent problème à plus de 40% d'entre eux. Ici aussi, les concepts semblent plus familiers aux enseignants.

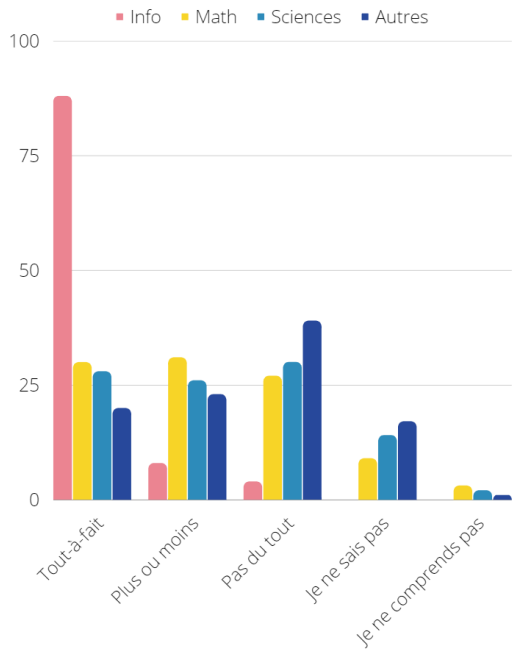


FIGURE 2.9 – Résultats (en %) pour la thématique "Matériel"

L'enquête passée auprès de 99 enseignants du secondaire inférieur laisse apparaître un besoin de formation chez ceux-ci vis-à-vis du contenu du référentiel {SI}[2]. Parmi eux, 38 se disent absolument intéressés à enseigner les sciences informatiques "à la rentrée prochaine" (2018), 27 répondent "peut-être" et 32 ne le souhaitent pas. Deux enseignants n'émettent aucun avis. Lorsqu'il leur est demandé s'ils trouveraient cela intéressant pour leurs apprenants, 58 enseignants en sont persuadés, 30 se posent la question et sept trouvent que ça n'a pas d'intérêt. Quatre restent sans avis. Enfin, 45 se disent prêts à suivre une formation pour pouvoir enseigner cette discipline, 36 sont mitigés et 17 ne le souhaitent pas. Un seul enseignant n'a pas d'avis.

Dès lors, pas loin d'un enseignant sur deux seraient prêts à se former afin de pouvoir intégrer de l'informatique dans le cursus de leurs apprenants. Plus d'un sur deux trouvent cela intéressant pour l'apprenant lui-même. Près de un sur trois seraient déjà partants pour l'enseigner dès la rentrée suivante. Ces résultats s'avèrent particulièrement engageants.

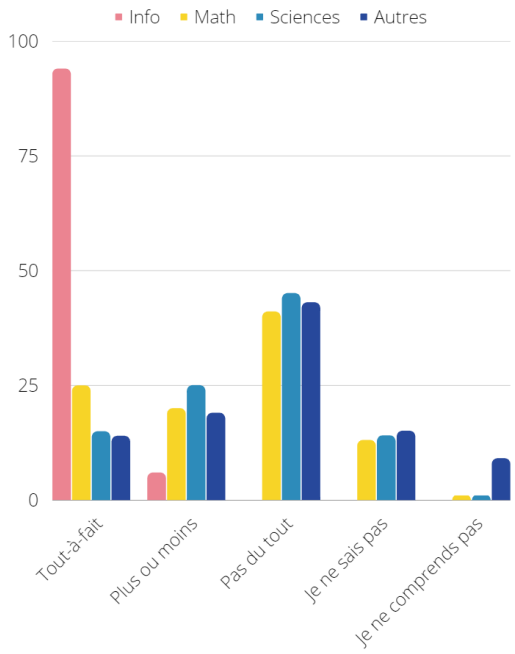


FIGURE 2.10 – Résultats (en %) pour la thématique "Réseau et sécurité"

CE QU'IL FAUT EN RETENIR
La réforme du Pacte d'Excellence prévoit l'introduction d'un cours de "numérique" dans le tronc commun de l'enseignement obligatoire (de 5 à 15 ans). Si le contenu apparait peu ambitieux en ce qui concerne l'informatique, ce référentiel a l'avantage de proposer une éducation au numérique dès le plus jeune âge (troisième primaire). Tout apprenant aura la possibilité, en théorie, d'acquérir un bagage en informatique plus conséquent que la majorité des apprenants suivant l'enseignement pré-Pacte d'Excellence : notamment en ce qui concerne l'algorithmique et la programmation. Mais garantir l'acquisition de ce bagage par les jeunes passera par la formation des enseignants. En effet, la majorité des (futurs) enseignants du secondaire inférieur se disent incapables d'enseigner les sous-domaines de l'informatique que sont la représentation des données, l'algorithmique, la programmation, le matériel, le réseau et la sécurité. Par conséquent, ils risquent d'être en difficulté face aux sujets liés à ces sous-domaines dans le référentiel de Formation Manuelle, Technique, Technologique et Numérique.

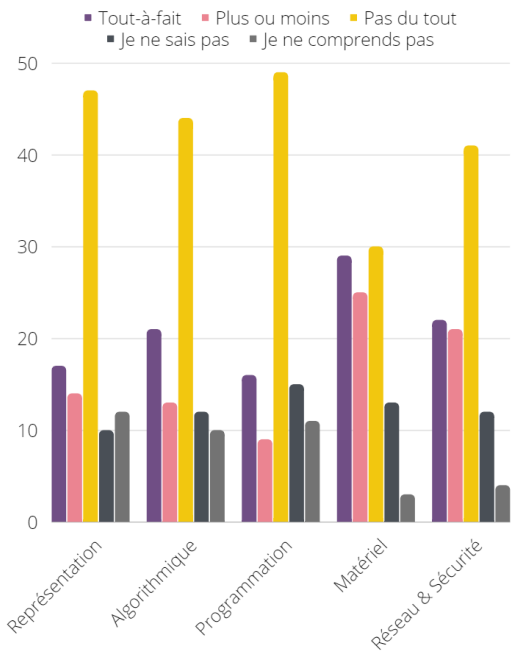
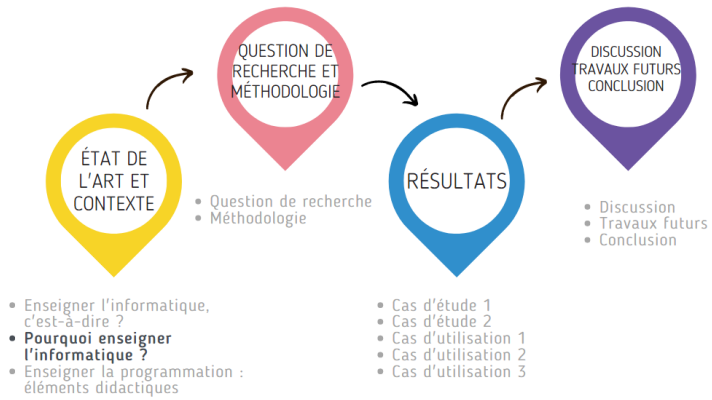


FIGURE 2.11 – Profils globaux (en %) pour les différentes thématiques

CHAPITRE 3

POURQUOI ENSEIGNER L'INFORMATIQUE ?



ARTICLE PUBLIÉ INSPIRANT CE CHAPITRE

- Henry, J., Hernalesteen, A., Dumas, B., & Collard, A. S. (2018). Que signifie éduquer au numérique ? Pour une approche interdisciplinaire. In Parriaux, G., Pellet, J., Baron, G., Bruillard, E., Komis, V. *De 0 à 1 ou l'heure de l'informatique à l'école.* Actes du colloque Didapro 7 - DidaSTIC. Lausanne, Suisse, Peter Lang Verlag.

La question de la (ré)introduction de l'informatique dans les écoles se pose depuis des années. Outre le fait de définir ce qui devrait être enseigné (cfr Chapitre

2), il est important de répondre à la question du pourquoi l'enseigner, les deux questionnements étant liés et s'enrichissant mutuellement.

Pour répondre à cette question, une réflexion est menée autour de trois axes. Le premier axe concerne l'intégration dans les programmes scolaires d'une éducation à la pensée informatique (cfr Section 3.1). Le deuxième axe recouvre les activités d'initiation à la technologie et à l'informatique en vue d'attirer dans ces filières de formation (cfr Section 3.2). Le dernier axe s'inscrit dans une approche d'éducation citoyenne et d'éveil au numérique en vue de comprendre le fonctionnement et les enjeux du numérique dans notre société (cfr Section 3.3).

3.1 Penser comme un informaticien

Dans le cadre d'une éducation aux sciences, les compétences cognitives que sont la démarche scientifique (plus rarement appelée pensée scientifique), la pensée critique et le raisonnement sont considérées comme cruciales [Hasanah and Shimizu, 2020]. Lorsqu'il s'agit d'éducation à l'informatique, et principalement chez les jeunes, la pensée informatique ou pensée computationnelle (*computational thinking*) est évoquée : "*Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability*" [Wing, 2006, p. 33].

Il est généralement considéré que Wing a lancé le mouvement autour de la pensée informatique. Depuis, de nombreuses définitions ont été formulées [Cansu and Cansu, 2019], toutes mettant en avant des compétences cognitives non spécifiques à l'informatique (cfr Table 3.1).

En outre, ce qui compose la pensée informatique est également une source de divergence entre les chercheurs. Les composants les plus souvent mentionnés sont l'abstraction, la décomposition des problèmes, la pensée algorithmique, l'automatisation et la généralisation [Cansu and Cansu, 2019]. Une fois encore, il apparait que ceux-ci ne sont pas spécifiques à l'informatique, mais bien à la résolution de problèmes.

Si un enseignement de la pensée informatique n'équivaut pas à un enseignement de l'informatique, il est bien souvent associé à l'enseignement d'un de ses sous-domaines : la programmation. Déjà à l'époque de Seymour Papert et de ses prédécesseurs, l'hypothèse était faite que la pratique de la programmation aidait à l'acquisition de compétences en matière de résolution de problèmes [Papert and Harel, 1991]. Depuis, de nombreuses études ont confirmé ou infirmé cette hypothèse [Menon et al., 2020, Arfé et al., 2020, Cansu and Cansu, 2019, Popat and Starkey, 2019, Resnick et al., 2009, Romero et al., 2017, Moreno-León et al., 2016, Kalelioğlu, 2015]. De façon logique, il est désormais considéré que c'est l'association pensée informatique-programmation et non la programmation seule qui a une incidence sur l'augmentation des capacités de résolution de problèmes. C'est également le cas lorsque sont associés pensée informatique et mathématiques, pensée informatique et sciences naturelles, pensée informatique et sciences sociales, ou encore pensée informatique et art [Cansu and Cansu, 2019]. Ce qui ne signifie en rien que la pensée informatique soit la meilleure façon de résoudre un problème. Ainsi, Papert a tou-

TABLE 3.1 – Différentes définitions de la pensée informatique [Cansu and Cansu, 2019]

Définition	Source
"*(...) the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent*"	[Wing, 2011]
"*(...) the thought processes used to formulate a problem and express its solution or solutions in terms a computer can apply effectively*"	[Wing, 2014]
"*The mental process for abstraction of problems and the creation of automatable solutions*"	[Yadav et al., 2014]
"*(...) the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes*"	[Yadav et al., 2014]
"*Known in the 1950s and 1960s as algorithmic thinking, it means a mental orientation to formulating problems as conversions of some input to an output and looking for algorithms to perform the conversions. The term has been expanded to include thinking with many levels of abstractions, use of mathematics to develop algorithms, and examining how well a solution scales across different sizes of problems*"	[Denning, 2009]
"*(...) is to teach them how to think like an economist, a physicist, an artist, and to understand how to use computation to solve their problems, to create, and to discover new questions that can fruitfully be explored*"	[Hemmendinger, 2010]

jours pris une position d'ouverture vers d'autres types de pensée [Turkle and Papert, 1990], comme bien d'autres chercheurs [Tedre and Denning, 2016, Hemmendinger, 2010], telles que la pensée (démarche) scientifique, la pensée mathématique, la pensée logique, la pensée critique ou encore le *design thinking* [Tedre and Denning, 2016].

La pensée informatique a du sens même sans ordinateur, ce qui explique que de nombreuses activités réalisées pour développer cette pensée le soient de façon débranchée. Programmer y est considéré uniquement pour rendre exécutable la solution déterminée par les apprenants et mettre en pratique un langage de programmation. Ce rôle secondaire ne reflète pas vraiment l'importance qui est donnée à l'activité de programmation, ou plutôt de *coding*, ces dernières années dans le monde de l'éducation. Mais si la programmation n'est pas nécessaire au développement de la pensée informatique, qu'est-ce qui la rend si essentielle à la discipline informatique ? Chaque apprenant en informatique doit-il acquérir un minimum de compétences dans ce sous-domaine juste parce que c'est une des pratiques standard de la discipline ? C'est en tout cas ce que sous-entend Guzdial lorsqu'il souligne l'importance de la programmation dans l'alphabétisation informatique (*computational literacy*) [Guzdial, 2015]. Rushkoff et Purvis considèrent même cet apprentissage comme aussi important que celui de la lecture et de l'écriture (cfr Chapitre 4) [Rushkoff and Purvis, 2011].

CE QU'IL FAUT EN RETENIR
La pensée informatique est parfois considérée comme une composante essentielle dans une éducation à l'informatique. Pourtant, il s'avère qu'elle est spécifique à la résolution de problèmes et non à l'informatique. De plus, si son association avec la programmation semble jouer sur l'augmentation des capacités de résolution de problèmes, elle ne constitue pas pour autant la seule alternative pour atteindre cet objectif.

3.2 Recruter dans les filières informatiques

ARTICLES PUBLIÉS INSPIRANT LA SECTION

- Henry, J., Lombart, C., & Dumas, B. (2021). Changer la représentation de l'informatique chez les jeunes : recommandations. *"Apprendre la Pensée Informatique de la Maternelle à l'Université", Environnements Informatiques pour l'Apprentissage Humain (EIAH)*, pages 13-24. Fribourg, Suisse.
- Lombart, C., Smal, A., & Henry, J. (2020, September). Tips and Tricks for Changing the Way Young People Conceive Computer Science. *Proceedings of International Conference on Informatics in Schools : Situation, Evolution, and Perspectives (ISSEP)*, pages 79-93. Springer, Cham.
- Henry, J., & Dumas, B. (2018, April). Perceptions of computer science among children after a hands-on activity : a pilot study. *Proceedings of 2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 1811-1817. IEEE

MÉMOIRES ENCADRÉS

- Hallaert, E. (2021). *Conception, développement et évaluation d'un outil de sensibilisation à la cybersécurité.* [Unpublished master's thesis]. Université de Namur. (Co-promotrice)
- Théate, N. (2020). *Comprendre l'ordinateur à l'aide d'un système informatique tangible : le micro :bit, au cours d'une séquence d'activité à destination de jeunes de 12 à 15 ans.* [Unpublished master's thesis]. Université de Namur. (Co-promotrice)
- Olivier, B. (2018). *Enseigner l'intelligence artificielle pour les jeunes de 12 à 14 ans afin d'en changer leur image et leur représentation au travers d'activités de courte durée.* [Unpublished master's thesis]. Université de Namur. (Co-promotrice)

Il s'agit de considérer les enjeux liés au recrutement d'étudiants dans les filières de formation en informatique et à ses implications au niveau du développement des secteurs technologiques. La digitalisation de la société requiert des experts capables de porter les innovations technologiques. La carence continue de tels experts est par ailleurs régulièrement soulignée [Albessart et al., 2017]. Selon le Conseil Wallon de la Politique Scientifique, le nombre de jeunes s'orientant vers les études scientifiques et techniques, et particulièrement vers l'informatique, en FWB est insuffisant pour couvrir les besoins des entreprises [1] Ce manque d'attrait est un constat massivement établi dans la recherche [Pantic et al., 2018, Hansen et al.,

1. Attractivité des études et métiers scientifiques et techniques, rapport consulté le 01/02/2022 [cesw.be/uploads/publications/CPS_Rapport_janvier2014.pdf]

2017, Hansen et al., 2016, Grover et al., 2014, Hewner, 2013, Hewner and Guzdial, 2008, Martin, 2004]. Ces études suggèrent sans équivoque que les jeunes entre 12 et 18 ans ignorent totalement ce qu'est l'informatique et entretiennent généralement des stéréotypes négatifs et/ou une vision de celle-ci centrée sur les ordinateurs.

3.2.1 Les représentations de l'informatique

Si de nombreux facteurs sont susceptibles d'influencer le manque d'intérêt des jeunes envers l'informatique et ses filières d'étude, leur méconnaissance ("*little public understanding of the broader dimensions of the computing field*") et leurs représentations erronées (par exemple, "*fear of becoming isolated in jobs perceived to involve little human contact*") du domaine ont probablement un impact majeur [Cassel et al., 2007]. Greening se pose la question : "*Is it the case that many students who enroll for a first computer science course do so with some very limiting misconceptions of what the discipline entails ?*". Il en est convaincu : "*students might also not be enrolling in computer science courses due to misconceptions*" [Greening, 1998]. Cette conviction que les jeunes choisissent de ne pas se spécialiser en informatique parce qu'ils ont une représentation incorrecte ou inadéquate de la discipline (ou même aucune représentation) est étayée par plusieurs études [Ruslanov and Yolevich, 2011, Biggers et al., 2008, Carter, 2006, Brinda et al., 2009]. Pour Brinda, les jeunes développent des croyances inadéquates en informatique parce qu'ils n'ont souvent que des expériences avec les TIC [Brinda et al., 2009]. Pour Carter, le principal responsable est le manque d'éducation. En ce sens, il recommande de contextualiser l'enseignement : "*educating students on how computing is really used in the real world - for special effects in the movies, to improve the quality of life for people with missing limbs, and for allowing communication for people with speech impediments*" [Carter, 2006]. En effet, une façon de remédier à cette situation est de travailler sur les représentations. C'est pourquoi l'enseignement de l'informatique doit viser à sensibiliser les jeunes à ce domaine dans son ensemble.

Greening définit "*a basis for evolving coursework in such a way that it increases the likelihood that students become excited about their learning*" [Greening, 1998]. Pour Yardi et Bruckmann, "*there is an opportunity to increase interest in computing among teenagers by bridging the gap between their conceptions of programming and the actual opportunities that are offered in computing disciplines*". Ils proposent, par ailleurs, un programme d'enseignement "*to prepare and motivate teenagers for careers in today's expanding, Internet-based, global economy*". Ils décrivent l'informatique comme "*an innovative, creative and challenging field with authentic, real-world applications*" et pensent qu'il existe une possibilité d'augmenter l'intérêt des jeunes envers l'informatique en reliant leurs représentations aux opportunités offertes par le domaine [Yardi and Bruckman, 2007].

Plus récemment, Grover et al. présentent les résultats d'une "*curricular intervention that aims to show computer science to [12-14 year-old] students in a new light–in real world contexts and as a creative and problem-solving discipline; as something bigger and broader than the computer-centric view that many students are known to harbor*" [Grover et al., 2014]. Deux ans plus tard, ils recommandent aux enseignants

"*to shift the initial focus of introductory courses to the deeper ideas of computing, computation and computability*" [Grover et al., 2016].

Selon la perspective constructiviste de l'apprentissage [Ben-Ari, 2001], les suggestions faites par ces experts vont toutes dans le même sens : il faut tenir compte, dans son enseignement, des représentations existantes des élèves en matière d'informatique. Mais cela semble plus facile à dire qu'à faire. En effet, selon Hewner et Guzdial, des cours d'introduction à l'informatique conçus pour être attrayants et pertinents par rapport aux intérêts des jeunes, mais axés uniquement sur la pratique de la programmation, n'ont pas d'effets significatifs sur l'attitude de ces jeunes vis-à-vis de l'informatique [Hewner and Guzdial, 2008]. Taub est légèrement plus optimiste : "*computer science unplugged activities did start a process of changing the students' views, but that this process was partial*" [Taub et al., 2009]. Parmi les obstacles à surmonter, Taub cite les difficultés que connaissent les étudiants à identifier des "*relationships between computer science unplugged activities and central ideas in computer science*" [Taub et al., 2009]. En outre, en plus d'une contextualisation de l'enseignement, il semble que la période à laquelle s'organise cet enseignement ait également une importance. En effet, Hewner et Guzdial montrent qu'une éducation à l'informatique organisée postérieurement à l'enseignement obligatoire ne présente pas un impact important sur l'attitude des jeunes envers ce domaine [Hewner and Guzdial, 2008]. Il est donc essentiel d'enseigner l'informatique dès le plus jeune âge.

3.2.2 Les représentations de l'ordinateur

Les jeunes ont majoritairement une vision de l'informatique centrée sur l'ordinateur. Ils grandissent entourés d'ordinateurs, parfois sans s'en rendre compte, et interagissent rapidement avec eux, se construisant ainsi leur propre représentation du fonctionnement et des capacités d'un ordinateur.

Grover et al. rapportent que certains jeunes expliquent leur intérêt (ou désintérêt) pour l'informatique par leur intérêt (ou désintérêt) pour l'ordinateur. Et leurs représentations d'un ordinateur sont limitées. Ainsi, pour les jeunes, appeler un objet "ordinateur" alors qu'il ne s'agit pas d'un ordinateur de bureau ou d'un ordinateur portable pose problème. Cette vision restreinte se répercute sur leurs représentations de l'informatique [Grover et al., 2016].

En reprenant différentes études traitant des représentations que possèdent les jeunes de l'ordinateur, Rücker et Pinkwart identifient cinq représentations distinctes [Rücker and Pinkwart, 2016].

— L'ordinateur humanisé
  "*The computer is often anthropomorphized and seen as some kind of living entity that is better understood in terms of psychology rather than technology*". Les jeunes attribuent à l'ordinateur "some form of mind or brain as well as various mental states like motivations, intentions or even emotions".
— L'ordinateur base de donnée
  L'ordinateur ne calcule pas. Il sait tout (stockage illimité des données) et retient tout par cœur (récupération rapide des données).

— L'ordinateur mécanique
  En raison de la façon dont il est construit, l'ordinateur est considéré comme une horloge complexe, où les données et les processus sont des entités physiques.
— L'ordinateur dispositif électronique
  L'ordinateur est un ensemble complexe de composants électroniques : "*what exactly is wired to what initially remains a complete mystery*".
— L'ordinateur programmable
  "Behaviour and capabilities are determined by humans and can be changed by humans".

Selon Rücker et Pinkwart, toutes ces représentations ne semblent pas être persistantes dans le temps : les capacités intrinsèques sont plus tenaces alors que les représentations liées au matériel sont logiquement plus influencées par les développements technologiques. En outre, plusieurs représentations peuvent émerger simultanément dans un contexte et/ou une situation donnés [Rücker and Pinkwart, 2016].

3.2.3 Les représentations des métiers de l'informatique

La technique du "dessine moi..." est souvent utilisée pour déterminer les attitudes et les croyances concernant un sujet quelconque. Lorsqu'il s'agit de dessiner des informaticiens, les enfants les voient comme des "*white males in various degrees of geekiness*" : ils portent des lunettes, ont de l'acné et des cheveux en désordre, sont devant un écran d'ordinateur H24, présentent un excès de poids, sont accro à la malbouffe et portent un t-shirt illustrant un code informatique, entre autres [Martin, 2004]. Ces résultats sont similaires à ceux mesurés par Hansen et al. auprès de jeunes de 9 à 10 ans. Soumis à un programe d'enseignement en informatique, ces mêmes jeunes voient leurs représentations modifiées. Ainsi, il apparait qu'après le programme les filles sont plus nombreuses à dessiner des femmes informaticiennes. En outre, les actions représentées sont plus spécifiques à l'informatique et non à la technologie en général (dactylographie, impression, etc.) [Hansen et al., 2017, Hansen et al., 2016].

3.2.4 Recommandations pour modifier la façon dont les jeunes se représentent l'informatique

De 2017 à 2020, dans le cadre du projet School-IT [2], des ateliers d'éducation à l'informatique ont été créés, permettant de développer des compétences techniques en privilégiant la variété dans les domaines couverts (communication, réseau, interaction humain-machine, intelligence artificielle, etc.), dans les processus mis en évidence (analyse, codage, test, etc.), et dans les équipements utilisés (activités débranchées, micro :bit, Makeblock, thymio, Bee-Bot, etc.). Ces ateliers ont pour objectif premier d'accroître l'intérêt des jeunes pour l'informatique en leur faisant prendre conscience de la richesse et de la diversité qui caractérisent ce domaine.

2. school-it [school-it.info.unamur.be], consulté le 25/01/2022

Pour mesurer un quelconque changement sur les représentations des jeunes suite à ces ateliers, une étude a été menée en situation réelle, sur une période de deux ans (entre septembre 2017 et juin 2019).

Cinq établissements scolaires ont organisé des ateliers dans le cadre du cours d'éducation par la technologie. Au total, onze classes ont participé, soit 232 élèves âgés de 12 à 15 ans. Les enseignants responsables du cours, tous scientifiques de formation initiale, ont pris part à l'étude.

Les élèves ont eu entre quatre et huit périodes de cours consacrées aux ateliers. Tous les élèves n'ont pas eu les mêmes ateliers, en fonction du temps disponible dans chaque école et des affinités de leur enseignant pour la matière abordée. Deux ateliers, organisés sur trois périodes, ont toutefois été imposés : un atelier introduisant la numérisation de l'information et un atelier expliquant le fonctionnement d'un ordinateur par la découverte des entrées et sorties d'un appareil tangible (ordinateur portable, micro :bit, thymio, tablette ou smartphone). Il restait alors aux enseignants à compléter leur programme par des ateliers au choix parmi ceux disponibles sur le site School-IT : concepts de base en programmation, intelligence artificielle, cybersécurité, entre autres. Sans concertation, l'ensemble des enseignants a opté pour les ateliers abordant la programmation.

En 2017-2018, les ateliers choisis ont été donnés par une membre de l'équipe School-It, informaticienne de formation. Les enseignants y ont assisté pour être en mesure de les dispenser eux-mêmes lors de la deuxième année d'étude, en 2018-2019. Deux échantillons sont donc considérés : 134 élèves (échantillon 1) ont reçu un enseignement par une experte en informatique et 98 élèves (échantillon 2) ont suivi les ateliers avec leur enseignant.

Un questionnaire a été administré aux élèves avant et après l'ensemble des ateliers dispensés. Une période pouvant aller jusqu'à quatre mois s'est écoulée entre les deux passations. Trois questions ouvertes ont été posées :

— Qu'est-ce qu'un ordinateur ? Qu'est-ce qu'un ordinateur peut faire ? Qu'est-ce qu'un ordinateur ne peut pas faire ?
— Qu'est-ce que l'informatique ?
— Quels sont, selon toi, les aspects positifs et négatifs des métiers de l'informatique ?

Les réponses à ces questions ont été codées indépendamment par deux chercheurs de façon à les regrouper dans des catégories. Pour tenir compte au mieux du vocabulaire utilisé par les élèves, certaines catégories ont été doublées bien qu'elles expriment un même thème : par exemple, les catégories "Machine" et "Boîte" sont restées distinctes, de même que les catégories "Écran/Souris/Clavier" et "Entrées/-Sorties". Pour chaque question, un comptage des catégories a été effectué. Une comparaison a ensuite été faite entre les résultats obtenus durant la première année d'étude (ateliers donnés par l'experte) et ceux de la deuxième année. Les résultats sont présentés sous forme de tableaux et seuls les points posant questions sont discutés.

L'ordinateur, ses capacités et ses limites

Dans le pré-test, à la question "Qu'est-ce qu'un ordinateur?" (cfr Table 3.2), l'échantillon 1 parle principalement d'une machine avec un écran, un clavier et une souris, ayant une mémoire et étant capable d'agir seule. Ils parlent aussi de l'utilisation qu'ils en ont couramment, à savoir la recherche d'information. Dans le post-test, les élèves décrivent un ordinateur composé d'entrées et de sorties, et possédant une mémoire et un processeur.

L'échantillon 2, quant à lui, décrit l'ordinateur d'un point du vue matériel (écran, souris, clavier) avant et après les ateliers. Si peu d'élèves parlent des composants que sont la mémoire et le processeur dans le pré-test, ce nombre diminue encore après les ateliers. Par contre, le nombre d'élèves décrivant l'ordinateur à partir de ses utilisations (communication et recherche) augmente. Enfin, 12,8% des élèves parlent de l'intelligence artificielle après la série d'ateliers alors que ceux-ci n'évoquaient pas ce sous-domaine de l'informatique.

TABLE 3.2 – Catégorisation et fréquence (en %) des réponses à la question "Qu'est-ce qu'un ordinateur?"

	Échantillon 1			**Échantillon 2**		
	Pré-test	**Post-Test**	Δ	**Pré-test**	**Post-Test**	Δ
"Machine"	27,9	18,0	-9,9	21,0	27,7	+6,7
"Boite"	12,0	10,5	-1,5	2,0	13,8	+11,8
Écran/Souris/Clavier	25,3	6,0	-19,3	16,0	19,1	+3,1
Entrées/Sorties	0,0	36,1	+36,1	3,0	1,1	-1,9
Processeur	2,7	13,5	+10,8	3,0	1,1	-1,9
Mémoire	20,0	24,1	+4,1	3,0	2,1	-0,9
Code/Programme	4,0	2,3	-1,7	3,0	6,4	+3,4
Internet	9,3	1,5	-7,8	6,0	6,4	+0,4
Communication	6,7	0,0	- 6,7	0,0	1,1	+1,1
Recherche	27,9	5,3	-22,6	3,0	8,5	+5,5
Logiciels	6,7	6,0	-0,7	4,0	6,4	+2,4
Intelligence artificielle	5,3	6,0	+0,7	3,0	12,8	+9,8
"Obéit aux ordres"	6,7	3,0	-3,7	7,0	8,5	+1,5
"Fait des choses par lui-même"	23,9	3,0	-20,9	4,0	8,5	+4,5

Les élèves devaient également répondre à deux questions : "Qu'est-ce qu'un ordinateur peut faire?" (cfr Table 3.3) et "Qu'est-ce qu'un ordinateur ne peut pas faire?" (cfr Table 3.4)

Les résultats du pré-test montrent que les élèves des deux échantillons humanisent l'ordinateur en lui attribuant des actions qui sont des utilisations possibles : effectuer des recherches, naviguer sur Internet ou encore enregistrer des données. Après les ateliers, la possibilité de faire des recherches est encore mentionnée par un élève sur trois. Par contre, pour les élèves de l'échantillon 1, la fréquence des autres utilisations diminuent.

Une augmentation du nombre d'élèves pensant que l'ordinateur peut tout faire est observée au sein des deux échantillons : +7,5% pour l'échantillon 1 et +2,4% pour l'échantillon 2. Ces résultats sont à mettre en relation avec les résultats à la question "Qu'est ce qu'un ordinateur ne peut pas faire?". En effet, après les ateliers,

TABLE 3.3 – Catégorisation et fréquence (en %) des réponses à la question “Qu’est-ce qu’un ordinateur peut faire?”

	Échantillon 1			Échantillon 2		
	Pré-test	**Post-Test**	Δ	**Pré-test**	**Post-Test**	Δ
Effectuer des recherches	31,6	30,8	-0,8	36,0	35,5	-0,5
Naviguer sur Internet	13,5	5,3	-8,2	12,0	12,9	+0,9
Programmer	2,3	3,0	+0,7	7,0	22,6	+15,6
Enregistrer des données	13,5	4,5	-9,0	12,0	9,7	-2,3
Travailler avec un traitement de texte	3,8	3,0	-0,8	8,0	5,4	-2,6
Communiquer	5,3	5,3	/	7,0	4,3	-2,7
Jouer	8,3	1,5	-6,8	11,0	1,1	-9,9
Tout	12,8	20,3	+7,5	3,0	5,4	+2,4
Rien sans l’action d’un humain	0,8	6,0	+5,2	0,0	1,1	+1,1

TABLE 3.4 – Catégorisation et fréquence (en %) des réponses à la question “Qu’est-ce qu’un ordinateur ne peut pas faire?”

	Échantillon 1			Échantillon 2		
	Pré-test	**Post-Test**	Δ	**Pré-test**	**Post-Test**	Δ
Bouger	14,3	12,0	-2,3	22,0	29,0	+7,0
Parler	14,3	12,8	-1,5	19,0	18,3	-0,7
Faire des actions du quotidien	10,0	15,0	+5,0	31,0	33,3	/2,3
Penser comme un humain	7,5	5,3	-2,2	3,0	1,1	-1,9
Rien (il peut tout faire)	4,5	9,0	+4,5	0,0	7,5	+7,5

plus d’élèves répondent “rien” à cette question : +4,5% pour l’échantillon 1 et +7,5% pour l’échantillon 2.

Après les ateliers dispensés par l’informaticienne, 6% des élèves pensent que l’ordinateur ne peut rien faire sans l’action d’un humain. Alors que les élèves animés par l’enseignant évoquent en plus grand nombre la programmation (+15,6%).

Enfin, pour les deux échantillons, l’évocation des jeux vidéos est moindre après les ateliers.

L’informatique

Dans le pré-test, la définition de 50% des élèves de l’échantillon 1 à la question “Qu’est-ce que l’informatique?” contient le mot “Ordinateur” (cfr Table 3.5). Après les ateliers, ce nombre diminue légèrement (41,8%). Dans l’échantillon 2, quasiment autant d’élèves mentionnent le mot “Ordinateur” aux pré-test et post-test. L’association informatique-ordinateur se ressent également lorsque les élèves parlent des composants écran/souris/clavier dans le pré-test. Après les ateliers, ces composants ne sont quasiment plus évoqués et les élèves de l’échantillon 1 citent davantage les entrées/sorties (+7,5%). Dans le même ordre d’idée, “Technologie”, “Machine électronique” et “Smartphone” arrivent respectivement en deuxième, troisième et quatrième positions, échantillons 1 et 2 confondus.

Le code et la programmation sont mentionnés par un plus grand taux d’élèves dans l’échantillon 2. Après les ateliers, le nombre de mentions augmente : +5,9% pour l’échantillon 1 et +16,2% pour l’échantillon 2.

TABLE 3.5 – Catégorisation et fréquence (en %) des réponses à la question "Qu'est-ce que l'informatique ?"

	Échantillon 1			Échantillon 2		
	Pré-test	**Post-Test**	Δ	**Pré-test**	**Post-Test**	Δ
Ordinateur	50,0	41,8	-8,2	39,4	38,8	-0,6
Technologie	21,6	9,7	-11,9	13,1	8,4	-4,7
Écran/Souris/Clavier	6,7	0,0	-6,7	8,1	2,1	-6,0
Entrées/Sorties	0,0	7,5	+7,5	0,0	0,0	/
Smartphone	14,2	4,5	-9,7	6,1	3,2	-2,9
Télévision	2,2	0,7	-1,5	1,0	3,2	+2,2
Machine éléctronique	10,4	8,2	-2,2	13,5	10,5	-3,0
Robot	0,7	4,5	+3,8	7,1	18,9	+11,8
Code/Programme	7,5	13,4	+5,9	10,1	26,3	+16,2
Réseaux	2,2	1,5	-0,7	1,0	1,1	+0,1
Internet	7,5	2,2	-5,3	2,0	4,2	+2,2
Communication	1,5	0,0	-1,5	3,0	0,0	-3,0
Données	10,4	0,0	-10,4	5,1	3,2	-1,9
Logiciels	8,2	1,5	-6,7	1,0	9,5	+8,5
Jeux vidéos	5,2	0,7	-4,5	0,0	4,2	+4,2
Intelligence artificielle	2,2	1,5	-0,7	3,0	8,4	+5,4

L'association informatique-Internet est plus présente au sein de l'échantillon 1 dans le pré-test et diminue de 5,3% après les ateliers. Par contre, cette association augmente pour l'échantillon 2 (+2,2%).

Dans le post-test, les élèves de l'échantillon 2 évoquent plus les robots que ceux de l'échantillon 1 : +11,8% contre +3,8%. De même que l'intelligence artificielle est mentionnée par plus d'élèves après les ateliers lorsque ceux-ci sont donnés par l'enseignant (+5,4%).

Si 15,5% des élèves parlent de données lors du pré-test, ce n'est plus le cas que de 3,2% d'entre eux au post-test.

Enfin, en ce qui concerne les logiciels, ceux-ci sont mentionnés par 8,2% des élèves de l'échantillon 1 avant les ateliers, contre 1% des élèves de l'échantillon 2. Après les ateliers, seulement 1,5% des élèves de l'échantillon 1 les citent, contre 9,5% des élèves de l'échantillon 2. Ce même comportement est observé concernant les jeux vidéos.

Les métiers de l'informatique

Les élèves ont exprimé les aspects positifs (cfr Table 3.6) et négatifs (cfr Table 3.7) qu'ils se représentaient dans les métiers de l'informatique.

Les aspects positifs mentionnés avant les ateliers sont, dans l'ordre et pour les deux échantillons confondus, l'utilité, les possibilités d'apprentissage continu qu'offre le domaine, le côté ludique et le fait d'utiliser un ordinateur. Après les ateliers, les deux premiers aspects sont maintenus, voire même renforcé pour le premier. L'aspect "Métier d'avenir" termine le podium.

TABLE 3.6 – Catégorisation et fréquence (en %) des réponses liées aux aspects positifs des métiers de l'informatique

	Échantillon 1			Échantillon 2		
	Pré-test	**Post-Test**	Δ	**Pré-test**	**Post-Test**	Δ
Job stable	0,8	3,0	+2,2	5,0	3,0	-2,0
Aider la population	3,8	6,0	+2,2	1,0	1,0	/
Utile pour la société	15,8	24,1	+8,3	22,0	35,0	+13,0
Utiliser l'ordinateur	10,5	6,8	-3,7	4,0	6,0	+2,0
Apprentissage continu	11,3	8,3	-3,0	13,0	11,0	+2,0
Ludique	6,0	3,0	-3,0	9,0	4,0	-5,0
Métier d'avenir	8,2	8,9	+0,7	4,0	6,0	+2,0
Créativité	4,5	5,3	+0,8	2,0	1,0	-1,0

TABLE 3.7 – Catégorisation et fréquence (en %) des réponses liées aux aspects négatifs des métiers de l'informatique

	Échantillon 1			Échantillon 2		
	Pré-test	**Post-Test**	Δ	**Pré-test**	**Post-Test**	Δ
Mauvais pour les yeux	31,9	16,5	-15,4	10,0	25,0	+15,0
Mauvais pour la santé car addictions	14,6	7,5	-7,1	6,0	6,0	/
Trop statique	37,2	21,1	-16,1	16,0	21,0	+5,0
Solitaire	10,6	7,5	-3,1	2,0	2,0	/
Pas attirant	5,3	6,8	+1,5	15,0	5,0	-10,0
Difficile	16,0	6,8	-9,2	11,0	8,0	-3,0
Effrayant	1,3	0,0	-1,3	3,0	4,0	+1,0

Si les élèves de l'échantillon 1 associent moins les métiers de l'informatique à l'utilisation d'un ordinateur après les ateliers, ce n'est pas le cas des élèves de l'échantillon 2 (+3,2%).

L'aspect ludique est délaissé par les deux échantillons et la créativité n'augmente que faiblement au sein de l'échantillon 1.

Concernant les aspects négatifs, les fréquences de l'échantillon 1 diminuent pour pratiquement toutes les catégories après les ateliers. Par contre, les élèves de l'échantillon 2 mentionnent plus les méfaits pour la vue et le coté statique du métier.

Discussion et recommandations

Outre l'objectif éducatif visé par les ateliers School-IT qui est de faire découvrir un domaine, des concepts, des techniques et des usages, l'enseignant doit se fixer un deuxième objectif : faire évoluer les représentations qu'ont les jeunes de l'informatique. C'est sur ce deuxième objectif que porte la discussion des résultats.

Un premier constat à faire est la confusion toujours présente chez les jeunes entre l'informatique, les métiers de l'informatique et l'ordinateur sous sa forme la plus classique (avec écran, clavier et souris). Bien que distinctes, les questions posées souffrent de cette confusion. Cependant, il reste possible de tirer des leçons des réponses collectées.

En général, l'experte informaticienne a pris davantage soin de s'éloigner, dans son discours, de l'ordinateur sous sa forme classique pour en faire comprendre le

fonctionnement interne, incluant le fonctionnement des composants (mémoire, processeur) en vue d'en expliquer les forces et les limites. L'experte a également été attentive à transmettre l'image d'une informatique "as a creative and engaging discipline with an impact in almost all other domains", comme l'ont recommandé Grover et al [Grover et al., 2014]. Compte tenu des ateliers imposés (initiation à la numérisation de l'information et au fonctionnement de l'ordinateur), les enseignants devaient également pouvoir transmettre ces idées. La différence mesurée entre les deux échantillons pourrait s'expliquer par leur manque de formation, par leur manque de confiance en eux lorsqu'il s'agit de donner des cours d'informatique, ou par le fait que les enseignants ont probablement moins insisté sur cette image jugée cruciale par l'experte.

Les résultats, au regard des ateliers organisés, laissent penser que, comme les jeunes, les enseignants semblent s'inspirer des médias pour développer leurs connaissances dans le domaine de l'informatique. Cela expliquerait la présence, dans les réponses de leurs élèves, de thèmes tels que robot et intelligence artificielle, notions non abordées dans les ateliers dispensés mais qui font régulièrement la Une des médias. L'évocation des logiciels et des jeux vidéo, plus souvent cités par les élèves de l'échantillon 2, montre également que les enseignants s'appuient sur leurs connaissances et leurs utilisations fréquentes.

L'augmentation des références à la programmation est logique considérant le contenu des ateliers. Cependant, elle est beaucoup plus prononcée dans l'échantillon 2. Une fois de plus, cela peut s'expliquer par l'attention portée par l'experte à montrer la variété qui existe dans les métiers de l'informatique.

Enfin, les évolutions concernant les métiers mettent une fois de plus en évidence la précaution prise par l'experte pour valoriser son travail. L'intervention des enseignants semble avoir un impact relativement faible, voire plutôt négatif.

En conclusion, il semble difficile de changer les représentations des jeunes si certaines conditions ne sont pas remplies. Plusieurs recommandations peuvent être faites, nourries par les résultats obtenus, mais également par les discussions informelles menées avec les enseignants durant ces deux années d'étude. Elles portent à la fois sur la conception d'ateliers pour enseigner l'informatique et sur les enseignants eux-mêmes.

- **Être clair sur le message à transmettre : ne pas enseigner l'informatique pour n'enseigner que l'informatique.** Si l'experte est consciente de ce qu'elle doit apporter, le combat est différent pour les enseignants qui doivent évoluer dans un domaine qu'ils maîtrisent mal.
- **Découvrir les différents sous-domaines de l'informatique.** Il est important d'enseigner différents sous-domaines de l'informatique (intelligence artificielle, cybersécurité, interface humain-machine, etc.) pour éviter de renforcer les stéréotypes et pour changer la façon dont les jeunes perçoivent l'informatique.
- **Prendre suffisamment de temps.** Il est difficile de changer des représentations profondément ancrées, surtout sur un temps court. En cela, une initiation ponctuelle à l'informatique reste insuffisante.
- **Se détacher de l'ordinateur sous sa forme la plus classique (écran, souris et**

clavier). Comme les jeunes parlent principalement de l'ordinateur lorsqu'ils se réfèrent à l'informatique, l'idéal est de s'en éloigner le plus possible pour développer leur vision et leur montrer d'autres aspects et dispositifs (micro :bit, makeblock, thymio, etc.). Une activité débranchée peut être tout aussi efficace pour comprendre la matière et changer les représentations.

— **Renforcer la formation des enseignants.** S'il est nécessaire de jouer sur les thèmes abordés (intelligence artificielle, cybersécurité, interface humain-machine, etc.), cela oblige les enseignants à maîtriser davantage que les compétences de plus en plus demandées dans l'enseignement que sont l'algorithmique et la programmation. Maîtriser la matière, c'est maîtriser les concepts de base, maîtriser le vocabulaire associé, avoir confiance en soi, pouvoir se détacher de la matière elle-même, mais aussi des médias. Prendre du recul est essentiel pour encourager les jeunes à adopter une attitude critique à l'égard de l'informatique (cfr Section 3.3).

— **Faire attention à ce qui est dit.** Les enseignants doivent être informés des stéréotypes existants afin de les déconstruire et non de les alimenter. Cela leur permettrait également de porter un regard critique sur les médias et les métaphores couramment utilisées (cfr Section 3.3).

— **Devenir autant que possible un modèle ou promouvoir des modèles inspirants.** L'experte devient un modèle pour les jeunes qui ont alors devant eux une personne qui devrait travailler quotidiennement devant un ordinateur (selon leurs représentations), mais qui dit faire beaucoup plus que cela. S'il n'est pas possible pour les enseignants d'être eux-mêmes des modèles, ils peuvent faire découvrir à leurs élèves des modèles inspirants. Il faut cependant veiller à montrer des modèles accessibles et non de rares cas de réussite [Spieler et al., 2019].

CE QU'IL FAUT EN RETENIR

Les représentations erronées de l'informatique et des métiers associés sont un facteur (parmi d'autres) expliquant le désintérêt des jeunes pour ce domaine, mais également l'inégalité qui existe entre les femmes et les hommes en informatique, ainsi que les stéréotypes de genre [Drot-Delange, 2011, Baron et al., 2010, Barker and Aspray, 2006]. Enseigner l'informatique peut avoir comme objectif de faire changer ces représentations auprès des jeunes et potentiellement d'intéresser ceux-ci aux filières d'études associées. Il faut cependant rester attentif à ne pas proposer une vision "restrictive" de l'informatique. Il s'agit de montrer, à travers le contenu de cet enseignement et sa contextualisation, toute la diversité du domaine et ce afin de ne pas produire l'effet inverse de celui attendu.

3.3 Devenir citoyen à l'ère du numérique

Pour rappel, le Conseil de l'Europe désigne la citoyenneté numérique comme "le maniement efficace et positif des technologies numériques (créer, travailler, partager, établir des relations sociales, rechercher, jouer, communiquer et apprendre), la participation active et responsable (valeurs, aptitudes, attitudes, connaissance) aux communautés (locales, nationales, mondiales) à tous les niveaux (politique, économique, social, culturel et interculturel), l'engagement dans un double processus d'apprentissage tout au long de la vie (dans des structures formelles, informelles et non formelles) et la défense continue de la dignité humaine" (cfr Section 2.2). L'éducation à la citoyenneté numérique (ECN), quant à elle, désigne "l'autonomisation des enfants par le biais de l'éducation ou l'acquisition de compétences pour apprendre et participer activement à la société numérique". À son niveau le plus simple, l'ECN cherche à "s'assurer que ceux qui ne sont pas des natifs du numérique ou qui n'ont pas la possibilité de devenir des citoyens numériques ne soient pas marginalisés dans les sociétés futures". Si l'éducation aux médias numériques (EAMN) constitue un pan important de l'ECN, l'informatique, en dehors de l'usage des technologies, n'y est pas toujours envisagée. Pourtant, la question de l'ECN ramène notamment à la culture informatique que Duchateau définit comme un "ensemble, aussi stable et adapté que possible, de savoirs et de savoir-faire qui permettent d'être à l'aise face à l'ordinateur et aux outils informatiques, de comprendre et de juger ce que permet l'informatique et ce qui est hors de sa portée... (Des) connaissances et (des) pratiques (qui) vont (...) colorer (notre) vision (...) du réel et de nous-mêmes pour s'intégrer à ce qui fait notre culture globale" [Duchâteau, 1992]. Dans leur travail de synthèse sur les compétences du 21e siècle dans une société transformée par les TIC, Voogt et Roblin introduisent, eux aussi, la nécessité de comprendre ces technologies, leurs principes et leurs stratégies pour développer des solutions et réaliser des objectifs [Voogt and Roblin, 2012]. Les travaux de Vuorikari et al. vont dans le même sens en ajoutant le traitement de données et la programmation dans les compétences de création de contenu : "*to plan and develop a sequence of understandable instructions for a computing system to solve a given problem or perform a specific task*" [Vuorikari et al., 2016, p.11].

Le Cadre de référence de la compétence numérique (CRCN) [3] fait figure d'exemple d'intégration d'une dimension informatique dans l'ECN. Élaboré dans le contexte du Plan d'action numérique en éducation et en enseignement supérieur (2018-2023) [4], le CRCN vise à développer la compétence numérique de chaque citoyen à partir du préscolaire, à savoir qu'il soit autonome et critique dans son utilisation du numérique. Chaque citoyen doit pouvoir "faire face aux innovations technologiques qui se concrétiseront dans les années à venir, notamment les avancées en matière d'intelligence artificielle". Il doit être capable de "poser un regard critique sur ces innovations et sera pleinement capable de se les approprier et d'y recourir s'il juge

3. Cadre de référence de la compétence numérique, consulté le 03/02/2022. [education.gouv.qc.ca/fileadmin/site_web/documents/ministere/Cadre-reference-competence-num.pdf]

4. Plan d'action numérique en éducation et en enseignement supérieur, consulté le 03/02/2022 [education.gouv.qc.ca/fileadmin/site_web/documents/ministere/PAN_Plan_action_VF.pdf]

qu'elles peuvent lui être utiles". Pour y parvenir, le CRCN offre une compréhension globale de la compétence numérique, décrite comme unique, et définie comme "un ensemble d'aptitudes relatives à une utilisation confiante, critique et créative du numérique pour atteindre des objectifs liés à l'apprentissage, au travail, aux loisirs, à l'inclusion dans la société ou à la participation à celle-ci". Cette compétence numérique est déclinée selon 12 dimensions indispensables pour apprendre et évoluer au 21e siècle (cfr Figure 3.1).

— Agir en citoyen éthique (dimension 1)
— Développer et mobiliser ses habiletés technologiques (dimension 2)
— Exploiter le potentiel numérique pour l'apprentissage (dimension 3)
— Développer et mobiliser sa culture informationnelle (dimension 4)
— Collaborer à l'aide du numérique (dimension 5)
— Communiquer à l'aide du numérique (dimension 6)
— Produire du contenu avec lé numérique (dimension 7)
— Mettre à profit le numérique en tant que vecteur d'inclusion et pour répondre à des besoins diversifiés (dimension 8)
— Adopter une perspective de développement personnel et professionnel avec le numérique dans une posture d'autonomisation (dimension 9)
— Résoudre une variété de problèmes avec le numérique (dimension 10)
— Développer sa pensée critique à l'égard du numérique (dimension 11)
— Innover et faire preuve de créativité à l'égard du numérique (dimension 12).

En plus d'être un exemple à suivre pour l'ECN (et l'éducation au numérique, de façon globale), le CRCN répond déjà aux attentes de l'Europe vis-à-vis de la version 2.2 de DigComp (cfr Section 2.3). Ainsi, sont mentionnés le développement d' "une compréhension globale à l'égard de l'intelligence artificielle et de ses impacts sur l'éducation, la société, la culture ou la politique" (p.14), le développement de la "pensée informatique, notamment par le développement de sa compréhension et de ses habiletés à l'égard de la programmation informatique" (p.14) et l'exploration du "fonctionnement mécanique, électronique ou informatique d'appareils du quotidien" (p.14). De plus, il est suggéré aux enseignants de "discute(r) avec les (élèves), et favorise(r) les échanges entre eux autour de grandes questions contemporaines, par exemple en matière d'intelligence artificielle" (p.23). Enfin, la culture numérique est développée par les enseignants en traitant "un fait d'actualité en classe, en amenant les (élèves) à analyser la crédibilité des médias et, au besoin, à déconstruire certaines rumeurs" (p.16) (désinformation).

Finalement, l'approche de l'ECN vise les compétences permettant d'utiliser, de comprendre et d'évaluer les technologies [Voogt and Roblin, 2012, Vuorikari et al., 2016]. Avec le développement de technologies complexes, dont le fonctionnement devient de plus en plus opaque pour le grand public, la compréhension des aspects techniques est un enjeu fondamental. Cette dimension technique est prise en charge, au niveau de l'éducation, par l'enseignement de l'informatique qui soutient l'acquisition de compétences suffisamment avancées pour comprendre les concepts fondamentaux et identifier les logiques de ces technologies [De la Higuera, 2015].

Comme le souligne De la Higuera, l'enseignement de l'informatique permet d'accompagner le changement de paradigme dans la manière de penser introduit

FIGURE 3.1 – Les 12 dimensions du Cadre de référence de la compétence numérique

par les technologies numériques. Son objectif est d'assurer une compréhension des technologies numériques passées, actuelles et futures [De la Higuera, 2015]. Si cet enseignement, au niveau des écoles, se traduit souvent par une initiation à la pensée informatique, aux algorithmes, au codage ou à la programmation, il devrait être étendu à des sujets considérés comme avancés par les experts mais auxquels le grand public est régulièrement confronté, comme la cybersécurité ou l'intelligence artificielle.

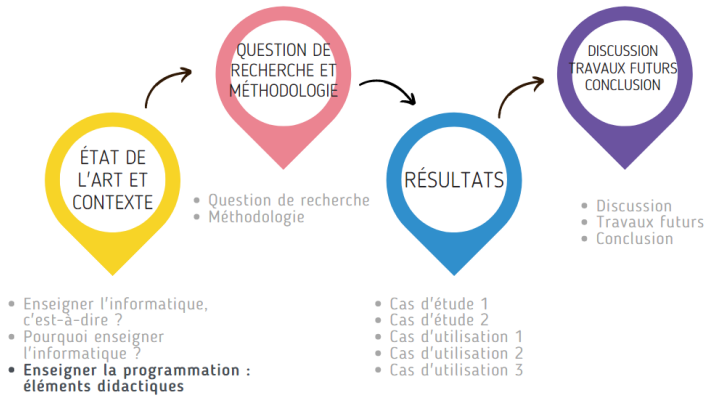
Cependant, enseigner les aspects techniques ne suffit pas pour amener les usagers à questionner la place des technologies dans la société et la manière dont elles s'inscrivent dans des pratiques qu'elles façonnent en même temps. Faire appel aux cadres de l'EAMN permet d'envisager une éducation aux technologies qui soit critique par rapport aux problématiques éthiques et sociétales, et réflexive par rapport aux pratiques de chacun [Saariketo, 2014a, Saariketo, 2014b]. L'EAMN vise en effet à développer les compétences nécessaires pour être critique, créatif, autonome et socialisé dans l'environnement médiatique contemporain, en prenant en compte la dimension technique des médias numériques mais également leurs dimensions

informationnelle (sémiotique) et sociale (pragmatique) [Fastrez, 2011]. Il s'agit donc non seulement de soutenir une compréhension des technologies au niveau de leur fonctionnement informatique réel et actuel, mais aussi d'en développer une compréhension qui s'inscrit dans un contexte d'usage, mettant en jeu des intérêts et des intentions humaines, prenant place au sein de relations sociales et s'inscrivant dans la compréhension que nous pouvons avoir d'une situation. En d'autres termes, cela consiste à ouvrir la "boîte noire" pour saisir les failles et les stratégies mises au point techniquement, mais aussi à la "déplier" dans un contexte médiatique et social pour souligner la manière dont sont envisagées les technologies en tant que constructions sociales. Cette approche intégrée, croisant les perspectives de l'enseignement de l'informatique et de l'EAMN dans un objectif d'éducation citoyenne critique au numérique [Henry et al., 2020], a permis le développement de deux dispositifs d'éducation à l'intelligence artificielle [Henry et al., 2021] et à la cybersécurité [Henry et al., 2022].

CE QU'IL FAUT EN RETENIR
Il est intéressant d'envisager l'enseignement de l'informatique dans une perspective plus large : posséder les clés pour comprendre le fonctionnement des technologies numériques est essentiel pour rendre chaque enfant autonome, au sens de l'éducation à la citoyenneté numérique. Cependant, c'est insuffisant car il faut également les amener à questionner la place de ces technologies dans la société. C'est le rôle de l'éducation aux médias numériques. Dès lors, il faut considérer une approche croisant les perspectives de l'enseignement de l'informatique, incluant des technologies complexes auxquelles les citoyens sont régulièrement confrontés, et de l'éducation aux médias numériques.

CHAPITRE 4

ENSEIGNER LA PROGRAMMATION : ÉLÉMENTS DIDACTIQUES



Lorsqu'il s'agit d'enseigner l'informatique, un sous-domaine est plus souvent mis en place que les autres : la programmation. Mais qu'est-ce qui rend la programmation si essentielle ? Chaque apprenant doit-il acquérir un minimum de compétences dans ce sous-domaine juste parce que c'est une des pratiques standard en informatique ? Il ne s'agit pas déterminer si la programmation est fondamentale ou non, mais parce qu'elle est considérée comme telle, il est nécessaire de se pencher sur des aspects didactiques aidant à son enseignement.

Jusqu'à il y a plus ou moins 20 ans, l'apprentissage de la programmation était décrit comme difficile. Un large éventail de problèmes et de déficits était attribué aux novices (cfr Section 4.1) et les cours de programmation présentaient souvent

les taux d'abandon les plus élevés [Robins et al., 2003]. Pourtant, rien ne permettait alors d'étayer ces conclusions. Par exemple, si certains auteurs associaient la difficulté aux taux d'échecs élevés (plus de 30%) [Kinnunen and Malmi, 2006, Guzdial and Soloway, 2002], ils ne pouvaient prouver que ces taux étaient inhabituels dans le cas d'un cours introductif [Bennedsen and Caspersen, 2019, Watson and Li, 2014, Bennedsen and Caspersen, 2007]. Depuis, même si des études plus récentes suggèrent que la situation n'est pas aussi désastreuse qu'annoncée [Luxton-Reilly et al., 2018a], comprendre comment s'apprend la programmation, pour aider à la fois les apprenants et les enseignants, reste le défi majeur de la recherche dans le domaine de l'enseignement de l'informatique [Guzdial, 2015].

4.1 Les novices en programmation

Si le manque de connaissances (générales et spécifiques) et de compréhension reste un obstacle logique à la réussite en programmation [Kohn, 2017, Cetin, 2015, Özmen and Altun, 2014, Piteira and Costa, 2013, Piteira and Costa, 2012, Derus and Ali, 2012, Tan et al., 2009, Dehnadi, 2009, Seppälä et al., 2006, Lahtinen et al., 2005, Ginat, 2004, Milne and Rowe, 2002, Du Boulay, 1986, Kessler and Anderson, 1986, Bayman and Mayer, 1983], l'apprenant lui-même est à considérer. En effet, la relation entre caractéristiques individuelles et succès de l'apprentissage de la programmation n'est plus à prouver [Gilberto et al., 2015, Renumol et al., 2010, Mancy and Reid, 2004]. De nombreuses recherches ont tenté d'identifier les caractéristiques d'un apprenant qui réussit un cours d'introduction à la programmation. Ces caractéristiques, appelées facteurs de réussite, trouvent leur origine à la fois dans la sociologie et dans la psychologie cognitive [White and Sivitanides, 2002]. Les facteurs les plus fréquemment mentionnés sont l'expérience antérieure en programmation [Lambert, 2015, Watson and Li, 2014, Wiedenbeck, 2005, Rountree et al., 2002, Wilson and Shrock, 2001, Hagan and Markham, 2000], les capacités en mathématiques [Lambert, 2015, Watson and Li, 2014, Bergin and Reilly, 2005, Oman, 1986, Leeper and Silver, 1982] et l'habileté spatiale [Cooper et al., 2015, Lau and Yuen, 2011, Jones and Burnett, 2008]. Cependant, l'éventail des facteurs potentiellement pertinents est bien plus large. Sharma et al. en répertorient 38, parmi lesquels l'enthousiasme et la motivation, les styles d'apprentissage, le sexe, mais aussi les stratégies et les modèles mentaux [Sharma and Shen, 2018].

Selon Winslow, il faut environ 10 ans pour faire d'un novice un programmeur expert [Winslow, 1996]. Entre les deux niveaux extrêmes existe un continuum [Dreyfus et al., 2000] :

— Le **novice** apprend des faits, des caractéristiques et des règles pour déterminer les actions à mener basées sur ces faits et caractéristiques. Tout ce qu'un novice fait est sans contexte.
— Le **débutant avancé** commence à reconnaître et à gérer des situations non couvertes par des faits, des caractéristiques et des règles donnés (sensibles au contexte) sans vraiment comprendre ce qu'il fait.
— Après avoir considéré l'ensemble de la situation, un programmeur **compétent** choisit consciemment un plan organisé pour atteindre l'objectif.

— Un programmeur qui **maîtrise** n'a plus besoin de raisonner consciemment à travers toutes les étapes pour déterminer un plan.
— Un **expert** sait généralement ce qu'il faut faire sur la base d'une compréhension mûre et pratiquée.

Plus récemment, la théorie néo-piagétienne a été appliquée à l'apprentissage de la programmation [Lister, 2016, Teague and Lister, 2014, Lister, 2011]. Trois stades de développement ont été définis pour le novice, du moins mature au plus mature.

— **Sensorimoteur** - Le novice a un modèle incorrect de l'exécution ("traçage") du programme. Lister évoque le stade de **pré-traçage** [Lister, 2016].
— **Pré opérationnel** - Le novice peut exécuter manuellement ("tracer") plusieurs lignes de code de manière fiable. Il fait souvent des suppositions inductives sur ce que fait un morceau de code, en effectuant une ou plusieurs traces, et en examinant la relation entre les entrées et les sorties. Lister parle du stade du **traçage** [Lister, 2016].
— **Opérationnel concret** - Le novice raisonne sur le code de manière déductive, en lisant le code lui-même, plutôt qu'en utilisant l'approche inductive pré-opérationnelle. C'est le premier stade où le novice commence à raisonner sur les abstractions du code. Par exemple, c'est seulement à ce troisième stade qu'il peut utiliser des diagrammes pour raisonner de manière productive sur le code. C'est le stade **post-traçage**, ou encore le stade du **traçage abstrait** [Teague et al., 2015].

Avant la perspective néo-piagétienne, il était considéré que tous les novices pensaient, par nature, comme le **novice opérationnel concret**. Pourtant, il est maintenant admis que tous les novices ne passeront pas tous rapidement et facilement au stade opérationnel concret. Il faut donc les aider à progresser entre les stades, tout en gardant en tête que la progression à travers les stades n'est pas linéaire. Parfois, à l'apprentissage d'un nouveau concept-clé, les novices peuvent temporairement régresser à un stade inférieur du raisonnement piagétien. Selon Lister, cette régression n'est pas une mauvaise chose [Lister, 2016]. Il soutient l'idée qu'un enseignant devrait chercher à ce que ses apprenants développent les stades supérieurs du raisonnement piagétien avec le moins de concepts-clé possibles, et seulement alors introduire des nouveaux concepts-clé. À noter qu'un novice ne peut pas être catégorisé comme étant à un stade unique de développement à un moment donné : selon le modèle *Overlapping Waves*, les stades de raisonnement coexistent [Boom, 2015].

Dans la littérature, la distinction est faite entre un novice *efficient* et un novice *inefficient*. Par définition, le premier apprend à programmer sans effort ou assistance excessifs, alors que le second n'apprend pas ou apprend au prix d'efforts démesurés et d'une attention particulière. Ici encore, il ne s'agit pas de deux catégories distinctes, mais d'un continuum entre deux extrêmes. Chaque apprenant devrait atteindre l' *efficience*. Déjà en 2003, Robins et al. suggèrent, pour assurer la réussite des apprenants, de ne plus se concentrer uniquement sur la connaissance de la programmation, mais de s'attacher à "créer" des novices *efficient*. Selon eux, les différences les plus significatives entre les novices *efficient* et les novices *inefficient* sont à chercher au niveau des stratégies utilisées pour accéder aux connaissances et

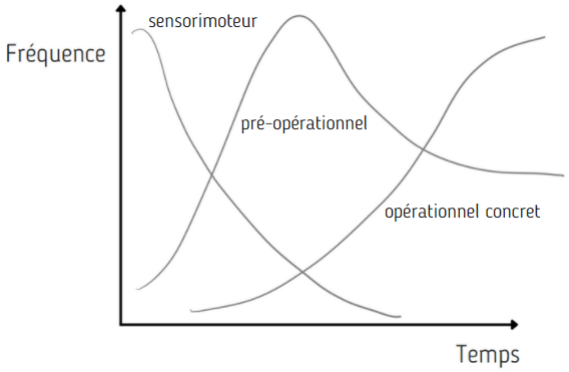


FIGURE 4.1 – Le modèle *Overlapping Waves* et les stades de raisonnement

pour les appliquer à la compréhension et à la création de programmes [Robins et al., 2003]. Ce constat a déjà été fait dix ans plus tôt par Davies qui affirme qu'un novice *inefficient* souffre en réalité d'un déficit de stratégies. Selon lui, les connaissances étant faibles ou inexistantes chez les novices, ce sont les stratégies préexistantes qui distinguent les *efficients* des *inefficients* [Davies, 1993]. Il s'agirait donc d'identifier les stratégies employées par les novices *efficient*, de déterminer comment celles-ci sont liées à leurs connaissances et à leurs modèles mentaux, et de les transmettre, à travers des cours bien pensés, aux novices *inefficient*.

La capacité de résolution de problèmes, couplée à des aspects de connaissance et de stratégie, est donc un indicateur important du niveau d'expertise d'une personne. Ainsi, les caractéristiques suivantes sont citées :

— **Le novice ne possède pas un modèle mental adéquat** du domaine étudié [**?**, Dehnadi, 2009, Caspersen et al., 2007, Ma, 2007, Kessler and Anderson, 1986].
— Le novice est limité à des connaissances superficielles et organisées de manière trop sommaires [Dreyfus et al., 2000].
— **Le novice a des connaissances fragiles** qu'il ne parvient pas à utiliser [Perkins and Martin, 1986]. Ces connaissances peuvent être **manquantes** (oubliées), **inertes** (apprises mais non utilisées), ou encore **mal placées** (apprises mais mal utilisées).
— Le novice néglige les stratégies (qui peuvent être fragiles) et planifie peu [Perkins and Martin, 1986].
— Le novice utilise des stratégies générales de résolution de problèmes, c'est-à-dire copier une solution similaire ou travailler à rebours à partir de l'objectif pour déterminer la solution, plutôt que des stratégies adaptées au problème particulier [Dreyfus et al., 2000].
— Le novice utilise une approche ligne par ligne, ascendante (*bottom-up*), pour résoudre un problème [Anderson, 2005]. Lorsqu'il s'agit de tester (ce qu'il ne fait pas spontanément), il a tendance à tenter de petites corrections "locales" plutôt que de reformuler les programmes [Linn and Dalbey, 1985].

CE QU'IL FAUT EN RETENIR
La notion de novice en programmation est plus compliquée qu'elle n'y paraît. Il ne s'agit pas seulement de considérer comme novice un apprenant n'ayant jamais eu de contact avec la programmation. Est novice tout apprenant qui ne possède pas un modèle mental adéquat et présente des déficits au niveau de ses connaissances et stratégies. Un novice, dans son développement, passerait par trois stades sans que cette progression soit linéaire et surtout, en considérant que plusieurs stades peuvent coexister à un moment donné. Ces constats devraient idéalement avoir un impact sur l'enseignement dispensé dans le cadre d'un cours d'introduction à la programmation. Il s'agirait d'identifier les stratégies employées par les novices *efficient*, de déterminer comment celles-ci sont liées à leurs connaissances et à leurs modèles mentaux, et de les transmettre aux novices *inefficient*.

4.2 Les modèles mentaux et conceptuels

Selon Johnson-Laird, les modèles mentaux sont des modèles incomplets, dynamiques et fonctionnels de situations du monde qui, grâce à leur manipulation mentale, permettent de comprendre et d'expliquer des phénomènes, mais également d'agir en anticipant les comportements [Johnson-Laird, 1983].

Dans un contexte éducatif, les modèles mentaux sont construits par les apprenants à partir de leur perception du monde et/ou à partir de leur compréhension du discours de l'enseignant [Greca and Moreira, 2000]. C'est ce que Johnson-Laird définit comme un modèle conceptuel, à savoir une structure artificielle inventée par quelqu'un pour faciliter l'apprentissage [Johnson-Laird, 1983]. Le modèle conceptuel est tout simplement ce qui est enseigné aux apprenants, contrairement au modèle mental d'un phénomène qui forme la connaissance des apprenants [Greca and Moreira, 2000]. Dans le domaine des interactions humain-machine, Norman distingue les deux termes en considérant [Staggers and Norcio, 1993] :

— Le **système-cible**, à savoir le système qu'un utilisateur doit prendre en main ;
— Le **modèle conceptuel du système-cible**, à savoir le modèle précis, cohérent et complet construit par le développeur du système et qui doit être transmis à l'utilisateur ;
— L'**image du système**, à savoir la représentation que s'en fait l'utilisateur ;
— Le **modèle mental**, incomplet, dynamique et fonctionnel, que l'utilisateur se construit en interagissant avec le système-cible et/ou son image ;
— La **perception** qu'a le **développeur** du modèle mental de l'utilisateur.

Le parallèle peut être fait avec une situation d'enseignement/apprentissage [Norman, 2014] (cfr Figure 4.2) :

— Le **concept-cible**, à savoir le concept à enseigner ;
— Le **modèle conceptuel du concept-cible**, à savoir le modèle précis, cohérent et complet construit par l'enseignant et qui doit être transmis à l'apprenant (à

noter que ce modèle provient du modèle mental de l'enseignant, qui lui-même repose sur l'image du concept-cible que possède l'enseignant) ;
— L'**image du concept-cible**, à savoir la représentation que s'en fait soit l'apprenant, soit l'enseignant ;
— Le **modèle mental**, incomplet, dynamique et fonctionnel, que l'apprenant ou l'enseignant se construit en interagissant avec le concept-cible et/ou son image (à notre qu'en ce qui concerne l'apprenant, le modèle mental se construit également sur base du modèle conceptuel transmis par l'enseignant) ;
— La **perception** qu'a l'**enseignant** du modèle mental de l'apprenant (et qui va nourrir son modèle conceptuel).

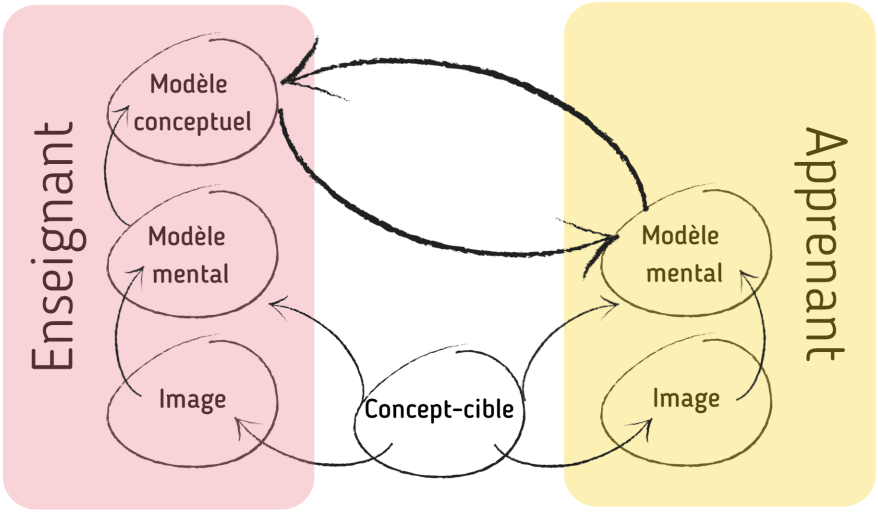


FIGURE 4.2 – Interactions entre modèles conceptuel et mentaux dans un contexte d'apprentissage

Il est attendu des apprenants qu'ils développent les bons modèles mentaux en se basant sur les modèles conceptuels présentés par l'enseignant. Selon Greca et Moreira, ce n'est pas la bonne manière d'enseigner car les apprenants ne possèdent pas toutes les connaissances nécessaires pour comprendre que le modèle conceptuel qui leur est enseigné est une représentation simplifiée d'un phénomène, et pas nécessairement le phénomène réel lui-même. En conséquence, les modèles mentaux construits par les apprenants ne seront pas nécessairement cohérents avec ce que les enseignants essaient de transmettre [Greca and Moreira, 2000]. Cette situation se mesure d'autant plus dans le cadre d'un cours d'introduction à la programmation. En effet, les apprenants disposent rarement des modèles mentaux efficaces dont ils ont besoin pour développer une compréhension appropriée des concepts enseignés : *"a (beginner) computer science student has no effective (mental) model of a computer. (...) Since computer science deals with artifacts—programming languages and software, the creator of the artifact employed a very detailed model and the learner must construct a similar, though not necessarily identical, model"* [Ben-Ari, 2001]. De plus,

tous les apprenants ne sont pas forcément capables de construire ces modèles mentaux simplement en écoutant le discours de l'enseignant ou en lisant les supports pédagogiques fournis. Le risque de construire des modèles mentaux inappropriés, basés sur une mauvaise utilisation de leurs connaissances antérieures et de leurs modèles mentaux intuitifs, est grand. Ben-Ari est formel : les modèles mentaux doivent être enseignés de manière explicite [Ben-Ari, 2001]. Dès lors, l'identification des modèles mentaux construits par les apprenants concernant les concepts de base en programmation est essentielle pour préparer des supports pédagogiques adaptés : "*if typical incorrect models are understood, then instructors and designers can create materials that minimize the chances of triggering errors*" [Gentner and Stevens, 2014]. Dans la littérature, les modèles mentaux des novices ont notamment été identifiés en ce qui concerne l'affectation de variable [Dehnadi, 2009], l'affectation par valeur et l'affectation par référence [Rørnes et al., 2019, Ma, 2007], la récursion [Scholtz and Sanders, 2010], ou encore les concepts de base en programmation orientée-objet que sont les classes, les objets, les appels de méthodes (et leurs effets sur les objets) [Sajaniemi et al., 2008].

Identifier les modèles mentaux des apprenants peut se faire en utilisant des questions à choix multiples [Haladyna, 2004]. Un exemple pertinent dans le cadre de cette recherche doctorale est l'outil développé par Dehnadi et visant à identifier les modèles mentaux des apprenants en ce qui concerne l'affectation de variables [Dehnadi, 2009, Dehnadi et al., 2006]. Cet outil prédisait, selon lui, la réussite (ou non) du cours d'introduction à la programmation. Bien que cette affirmation ait été discutée et rétractée [Bornat, 2014, Caspersen et al., 2007], cet outil a mis en évidence que la plupart des apprenants qui réussissent le cours sont ceux qui possèdent des modèles mentaux cohérents [**?**, Dehnadi, 2009, Ma et al., 2008, Ma, 2007]. De la même manière, selon Ahadi et al. [Ahadi et al., 2014], les apprenants qui appliquent un modèle cohérent d'exécution de programme seraient plus susceptibles d'apprendre la programmation, même lorsque leur modèle est erroné. La cohérence est donc un critère important pour la viabilité des modèles mentaux : un modèle mental n'est pas viable si son détenteur ne peut pas l'utiliser de manière cohérente. Déterminer la cohérence d'un modèle mental se fait via l'étude des réponses à une série de questions portant sur un seul et même concept. Les apprenants cohérents utilisent le même modèle mental pour la plupart des questions (quel que soit le modèle). Les apprenants incohérents, eux, en utilisent plusieurs [Gentner and Stevens, 2014].

CE QU'IL FAUT EN RETENIR
Les apprenants construisent leur modèle mental, incomplet, dynamique et fonctionnel, à partir du modèle conceptuel transmis notamment par les enseignants. Dans le cadre d'un cours d'introduction à la programmation, les apprenants ne possèdent généralement pas les connaissances nécessaires pour comprendre que le modèle conceptuel qui leur est enseigné est une représentation simplifiée de la programmation. En conséquence, les modèles mentaux construits par les apprenants ne seront pas nécessairement cohérents avec ce que les enseignants essaient de transmettre. Dès lors, l'identification de ces modèles mentaux s'avère essentielle pour préparer des supports d'enseignement adaptés et efficaces.

4.3 Les représentations erronées

Selon la théorie du constructivisme, la connaissance est construite par l'apprenant, de manière récursive sur base des connaissances qu'il possède déjà [Ben-Ari, 2001], mais aussi en s'appuyant sur les manuels et les discours de l'enseignant. Cette connaissance construite est une version propre à l'apprenant qui peut être différente de la connaissance scientifique standard, en ce sens qu'elle peut être constituée de représentations erronées.

Dans le contexte de l'enseignement de la programmation, Sorva définit les représentations erronées comme "*understandings that are deficient or inadequate for many practical programming contexts*", incluant entre autres une compréhension insuffisante des concepts-clé [Sorva et al., 2012]. Cette vision est partagée par Qian et Lehman qui décrivent les représentations erronées comme des erreurs de compréhension conceptuelle, à savoir une mauvaise compréhension des constructions de programmation telles que les variables et les instructions d'affectation, les expressions conditionnelles ou les boucles [Qian and Lehman, 2017].

Parce qu'il est attendu des enseignants qu'ils trouvent de meilleures façons d'enseigner la programmation, les représentations erronées, à savoir insuffisantes ou inadéquates, constituent une des plus grandes pistes de solution à considérer dans le domaine [Veerasamy et al., 2016, Bringula et al., 2012, Kaczmarczyk et al., 2010, Özdener, 2008]. Une façon de réduire ou de prévenir les représentations erronées est de confronter directement l'apprenant à une expérience qui provoque un déséquilibre [Maier, 2004]. Souvent, les apprenants ne se rendent pas compte que leurs représentations existantes sont en conflit avec les représentations enseignées [Hewson and Hewson, 1984]. Il est donc important que l'enseignant soit capable d'aider les apprenants à prendre conscience de ce problème, avant de les aider à construire des représentations cohérentes. Toutefois, il n'est pas facile pour les apprenants d'abandonner leurs représentations existantes, parfois erronées, et d'en adopter de nouvelles [Orey, 2010]. Remettre en question les représentations existantes de l'apprenant l'encourage à détecter les problèmes de compréhension et le motive à construire des compréhensions appropriées [Scott et al., 1997]. Sirkiä et Sorva

parlent de conflit cognitif : "*a dissonance between the learner's existing understanding and what they observe*" [Sirkiä and Sorva, 2012, p. 20]. En général, une stratégie d'enseignement par conflits cognitifs comporte trois étapes :

— Identifier les connaissances antérieures de l'apprenant et ses représentations existantes ;
— Mettre l'apprenant (et ses représentations) au défi avec des informations contradictoires ;
— Évaluer le changement conceptuel entre les représentations antérieures de l'apprenant et ses représentations actuelles.

Identifier les représentations erronées va permettre aux enseignants d'évaluer la compréhension qu'ont les apprenants des concepts fondamentaux de la programmation. De là, ils pourront fournir une intervention appropriée en cas de difficulté [Taylor and Kowalski, 2014]. Cette identification se fait idéalement via des interviews avec les apprenants (cfr Section 7.2) [Evans et al., 2003, Treagust, 1988]. L'intervention d'enseignement proposée doit alors provoquer un conflit cognitif. Parmi les stratégies suggérées pour y arriver figurent l'enseignement par les pairs, les outils de visualisation, le traçage de code et les questionnaires [Veerasamy et al., 2016]. Enfin, il est recommandé d'explorer l'évolution des représentations erronées qui peut être mise en relation avec les stratégies d'enseignement et les supports pédagogiques utilisés dans les cours, mais aussi en relation avec les différences individuelles [Qian and Lehman, 2017]. Parmi les facteurs généralement liés aux représentations erronées figurent les modèles mentaux incomplets ou inexacts [Qian and Lehman, 2017]. Malgré ce lien évident entre modèles mentaux et représentations erronées, les travaux scientifiques existants qui se concentrent sur les uns ne font généralement que peu référence aux autres.

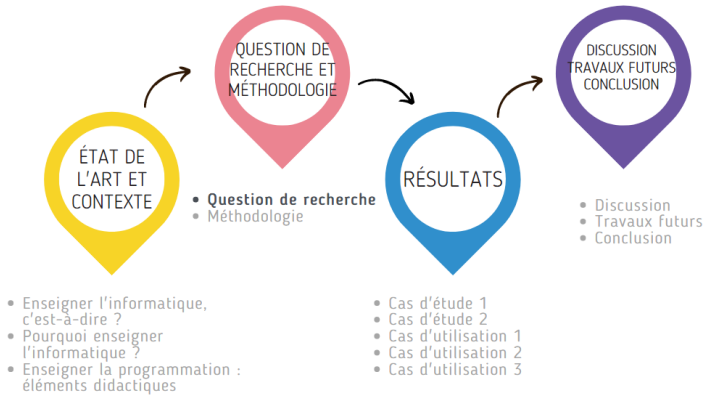
CE QU'IL FAUT EN RETENIR
La connaissance est construite par l'apprenant, de manière récursive sur base de ses connaissances antérieures et des discours de l'enseignant, entre autres. De ce fait, la connaissance construite est une version propre à l'apprenant qui peut différer de la connaissance scientifique standard et contenir des représentations erronées. Identifier ces mauvaises compréhensions qui existent chez les apprenants va non seulement permettre aux enseignants d'évaluer la compréhension qu'ont les apprenants, mais va également fournir un matériel de choix pour élaborer des ressources pédagogiques plus appropriées et efficaces.

Deuxième partie

Question de recherche et méthodologie

CHAPITRE 5

QUESTION DE RECHERCHE



La problématique au cœur de cette thèse est l'enseignement de l'informatique. Si celui-ci a un intérêt certain pour la société, ce n'est pas une évidence pour tous. Il est courant d'entendre dire, à propos des *digital natives*, qu'il n'est pas nécessaire de leur enseigner l'informatique et que bien souvent ils sont meilleurs dans cette discipline que leurs propres enseignants. Cette idée reçue a cependant été ébranlée durant la période imposée d'enseignement à distance faisant suite à la pandémie de Covid-19. En effet, ces "*native speakers of the digital language of computers, video games and the Internet*" [Prensky, 2001] se sont révélés bien moins experts qu'attendu, plus aptes à utiliser l'informatique qu'à en comprendre les concepts-clé.

En Belgique, si ce n'est dans le cadre d'une formation professionnalisante (Haute-École ou Université, entre autres), l'enseignement de l'informatique n'est plus envisagé en tant que tel dans l'enseignement obligatoire. Avec la mise en place du

Pacte d'Excellence, une éducation au numérique est cependant envisagée, avec une composante informatique essentiellement centrée sur la programmation et l'algorithmique.

5.1 Une recherche en didactique de l'informatique

Cette recherche doctorale se veut **une recherche en didactique de l'informatique**. L'informatique est une science et "toute science doit assumer, comme sa condition première, de se vouloir science d'un objet, existant d'une existence indépendante du regard qui le transformera en objet de connaissance" [Chevallard, 1981, p. 1]. Le passage du savoir savant au savoir enseigné est appelé la transposition didactique. L'étude de "l'obligatoire distance qui les sépare" [Chevallard, 1981, p. 3] et qui permet de rendre compte des transformations subies est la didactique.

Il ne s'agit pas seulement d'analyser les relations de distance entre les savoirs de référence et les savoirs à enseigner (**niveau 1**), mais également entre les savoirs à enseigner et les savoirs effectivement enseignés (**niveau 2**) et entre les savoirs enseignés et les savoirs appris (**niveau 3**) [Garcia-Debanc, 2008]. Sur base de ces trois niveaux de distance, Halté définit trois types de recherche en didactique [Halté, 1992] :

— "(...) une réflexion sur les objets d'enseignement. Elle s'intéresse à leur nature cognitive (...), à leur statut épistémologique (...), à la méthodologie de leur construction (...), à leur organisation en curricula, à leur histoire institutionnelle (...). La dominante de cette tendance est **épistémologique**" (**type 1**).

— "(...) des recherches sur les conditions d'appropriation des savoirs. Elle s'interroge alors moins sur les concepts et les notions en eux-mêmes que sur leur construction dans l'apprentissage, les prérequis qu'ils supposent, les représentations ordinaires qu'en ont les apprenants, les différentes sortes d'obstacles à l'apprentissage qu'ils peuvent susciter (...). La dominante est **psychologique**" (**type 2**).

— "(...) des recherches sur l'intervention didactique. Systémique, la didactique alors articule les points précédents aux tâches de l'enseignant, à l'organisation des situations d'enseignement, à la construction de cycles ou de séquences didactiques, à l'adaptation au type de public, bref, à l'approche de la classe et de son fonctionnement propre. La dominante est **praxéologique**" (**type 3**).

Parce qu'elle vise à évaluer la compréhension des apprenants à partir des représentations erronées, cette thèse s'inscrit dans **les niveau 3 et type 2**.

Dans les pays francophones, la didactique de l'informatique met bien souvent à l'honneur la programmation, et ce depuis sa naissance dans la deuxième moitié des années 70 [Duchateau, 2002, Baron and Bruillard, 2001]. La programmation est alors incontournable parce que omniprésente, à l'instar de ce qui se passe aujourd'hui, dans les programmes scolaires [Rogalski, 1989, Romainville, 1988, Rogalski and Samurçay, 1986, Samurçay and Rouchier, 1985]. Cette mise sur un piédestale a inspiré l'objet de cette recherche : l'**enseignement de la programmation** .

5.2 Une recherche en psychologie de la programmation

Dans la littérature, les préoccupations didactiques liées à l'enseignement de la programmation peuvent être diverses. Les plus courantes sont soit issues du courant *psychology of programming* (PP) [Blackwell et al., 2019], soit touchent les environnements spécifiques adaptés à l'apprentissage de la programmation.

Les recherches en PP portent sur les caractéristiques de l'apprentissage et de la maîtrise de l'activité de programmation. Elles se basent notamment sur les difficultés rencontrées par les novices. La PP rassemble une communauté de recherche interdisciplinaire, dont font partie l'enseignement de l'informatique (*CS education*) et l'interaction humain-machine (*Human-Computer Interaction*, HCI) [Blackwell et al., 2019]. Cette thèse se situe à l'intersection de ces trois communautés (cfr Figure 5.1) en développant outils d'évaluation et approche pour mesurer la compréhension des apprenants, sur une période d'enseignement donnée, en ce qui concerne notamment les concepts abordés dans un cours de programmation.

FIGURE 5.1 – Situation de cette thèse de doctorat par rapport aux communautés de recherche de la psychologie de la programmation (PP), de l'enseignement de l'informatique (*CS education*) et de l'interaction humain-machine (HCI)

5.3 Une recherche autour des représentations erronées

Les outils envisagés dans le cadre de cette recherche doctorale sont des *concept inventories* (cfr Chapter 5). Ces outils, fondés sur les représentations erronées (cfr Section 4.3) des concepts-clé en programmation, permettent notamment d'identifier les modèles mentaux (cfr Section 4.2) utilisés par les apprenants. L'approche élaborée, pour sa part, tient compte de l'organisation du cours qui constitue le contexte de l'étude : activités d'enseignement/apprentissage, plan du cours, calendrier, entre autres. Les outils sont administrés plusieurs fois pour un même concept afin de capturer une évolution dans l'usage des modèles mentaux, et par là un gain de compréhension.

Deux cas d'étude (un cours d'initiation à la programmation et un cours de programmation orientée-objet) et trois cas d'utilisation (les deux cours de program-

mation cités, plus un cours de modélisation des systèmes embarqués) (cfr Section 6.5) vont apporter des éléments de réponse à la question de recherche suivante :

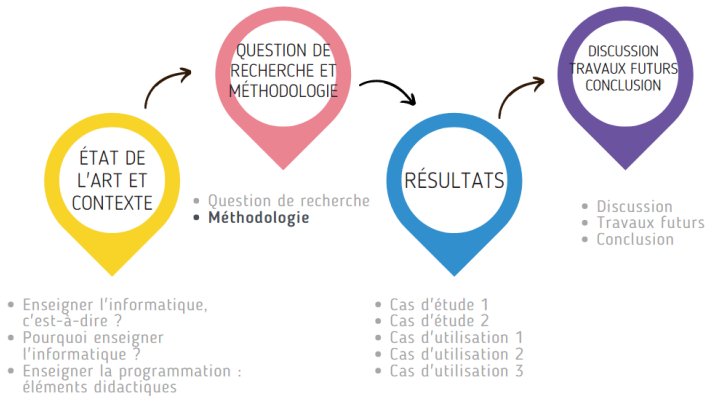
Quel(s) rôle(s) les *concept inventories* peuvent-ils jouer dans l'enseignement de l'informatique ?

Il s'agit, entre autres, de développer (et de tester) des *concepts inventories* permettant de :

- **Identifier les représentations erronées** présentes chez les apprenants **et/ou les modèles mentaux** qu'ils utilisent ;
- **Quantifier les représentations erronées et/ou modèles mentaux**, à savoir mesurer leur fréquence au sein d'un groupe d'apprenants ;
- **Mesurer** l'évolution des représentations erronées et/ou modèles mentaux, à savoir **le gain de compréhension** des apprenants sur une période donnée.

CHAPITRE 6

MÉTHODOLOGIE



Ce chapitre se concentre sur la méthodologie mise en place pour développer les outils d'évaluation et l'approche permettant de :

— Identifier les représentations erronées présentes chez les apprenants en ce qui concerne des concepts-clé en programmation et/ou les modèles mentaux qu'ils utilisent pour résoudre des problèmes mettant en oeuvre ces mêmes concepts-clé ;

— Quantifier les représentations erronées et/ou modèles mentaux, à savoir mesurer leur fréquence au sein d'un groupe d'apprenants ;

— Mesurer l'évolution des représentations erronées et/ou modèles mentaux, à savoir le gain de compréhension des apprenants.

Au coeur de cette recherche se trouve un outil d'évaluation couramment utilisé et basé sur les représentations erronées des apprenants : le *concept inventory*. Cette

thèse porte son intérêt à la fois sur le processus de développement de cet outil (cfr Section 6.3), mais également sur son utilisation (et donc, son administration - cfr Section 6.4) dans différents contextes (cfr Section 6.5).

6.1 L'évaluation formative

En 2015, l'*Assessment Task Force* de la *Computer Science Teachers Association* mène une étude sur l'évaluation de l'enseignement de l'informatique auprès d'enseignants du secondaire [Yadav et al., 2015]. Les résultats mettent en évidence les pratiques d'évaluation que les enseignants utilisent pour mesurer la compréhension des concepts informatiques par leurs élèves, ainsi que les difficultés auxquelles ils sont confrontés lors de ces évaluations. Les enseignants font état d'un certain nombre de défis qui, selon eux, sont propres à l'évaluation d'un enseignement de l'informatique, notamment la difficulté à trouver des évaluations formatives et sommatives valides et fiables.

Cette thèse considère exclusivement l'évaluation formative, c'est-à-dire les données sur les apprentissages collectées durant les activités d'enseignement/apprentissage [Andrade et al., 2019]. Par définition, l'évaluation formative vise à :

— Identifier les forces et les faiblesses de l'apprenant ;
— Aider l'apprenant à réfléchir à ses propres processus d'apprentissage et à guider ses démarches en vue de progresser ;
— Accroître l'autonomie et la prise de responsabilité de l'apprenant face à ses apprentissages ;
— Orienter la planification de l'enseignement.

En ce sens, l'évaluation formative consiste en deux choses : une collaboration apprenants-enseignant, ce dernier devant juger de la situation de chaque apprenant face aux apprentissages ciblés, et une intervention visant à soutenir les apprentissages.

6.2 Les *concept inventories*

Un nombre croissant de recherches font état d'outils mesurant l'apprentissage dans le contexte d'un cours afin de fournir une évaluation formative de l'enseignement (*Formative ASsessment of Instruction*, FASI). Dans le domaine de l'enseignement de l'informatique, les outils FASI sont souvent des *concept inventories* (CIs) [Taylor et al., 2014] : mathématiques discrètes [Almstrum et al., 2006], arbres de recherche binaires [Karpierz and Wolfman, 2014], logique numérique [Herman et al., 2010], tables de hachage [Karpierz and Wolfman, 2014], systèmes d'exploitation [Webb and Taylor, 2014, Andrus and Nieh, 2012], algorithmes et structures de données [Danielsiek et al., 2012], la récursion [Hamouda et al., 2017], affectations [Dehnadi, 2009] et autres concepts fondamentaux de la programmation [Caceffo et al., 2018, Luxton-Reilly et al., 2018b, Kaczmarczyk et al., 2010, Tew and Guzdial, 2010, Ragonis and Ben-Ari, 2005], entre autres.

Un CI est un questionnaire à choix multiple, inspiré de résultats issus de la recherche, qui vise à mesurer les connaissances d'un apprenant sur un concept (ou

un ensemble de concepts) tout en saisissant les conceptions et les représentations erronées qu'il possède de ce concept [Wittie et al., 2017]. Chaque question autour d'un concept spécifique comporte une réponse correcte et un certain nombre de réponses incorrectes, appelées distracteurs, inspirées des représentations erronées (cfr Section 4.3) que possèdent les apprenants de ce concept. En outre, les résultats obtenus lors de l'administration d'un CI permettent de vérifier l'efficacité d'une méthode d'enseignement particulière [Bailey et al., 2012] et de guider les futures interventions [Taylor et al., 2014].

Si la pédagogie utilisée est inefficace, il y aura peu ou pas de gain d'apprentissage. Cela ne veut pas dire que les apprenants n'auront pas appris quelque chose, mais il se peut que ce ne soit pas ce qui était prévu. Concernant l'évaluation, un apprenant qui réussit une évaluation sommative (par exemple, un examen) aura appris quelque chose, mais la réussite d'une telle évaluation ne suffit pas à garantir la qualité de cet apprentissage. Les CIs le peuvent en ce sens qu'ils sont destinés à évaluer le gain d'apprentissage, et particulièrement le gain de compréhension [Sands et al., 2018]. Là où l'évaluation sommative est effectuée une seule fois, les CIs sont généralement administrés deux fois au cours de la séquence d'enseignement permettant de mesurer une évolution de la compréhension par comparaison (pré-test *vs* post-test). Toutefois, pour évaluer un gain de compréhension, les CIs doivent remplir un certain nombre d'exigences : ils doivent être valides (mesurer ce qu'ils doivent mesurer), standardisés (permettre des comparaisons significatives) et longitudinaux (permettre d'apparier les résultats collectés à des moments différents) [McGrath et al., 2015].

6.3 Le processus de développement d'un *concept inventory*

Si de nombreuses approches de conception et de validation d'outils FASI, et particulièrement de CIs, ont été rapportées [Wittie et al., 2017, Farghally et al., 2017, Caceffo et al., 2016, Webb and Taylor, 2014, Tew and Guzdial, 2010, Almstrum et al., 2006], celles-ci ne sont pas toujours complètes et rigoureuses [Adams and Wieman, 2011]. Par exemple, lorsqu'il s'agit de mesurer les représentations erronées des apprenants sur un sujet particulier, il apparait important de mener des interviews avec les apprenants [Evans et al., 2003, Treagust, 1988]. Pourtant, peu de processus de développement et de validation utilisent des interviews [Adams and Wieman, 2011]. Pour cause, les développeurs d'outils FASI sont avant tout des experts en contenu, pas forcément familiers avec la conception d'évaluation.

Dans le domaine de l'enseignement de l'informatique, la plupart des CIs existants ne sont pas validés et restent au stade de projet pilote. Pour pallier ce manque de rigueur, certains développeurs s'inspirent des processus de développement issus d'autres disciplines, notamment le bien connu *Force Concept Inventory* en physique [Evans et al., 2003]. Ainsi, Goldman et al. proposent un processus en quatre étapes [Goldman et al., 2010].

1) **Définir la portée du CI** en sollicitant l'opinion des experts dès le début du processus de développement. Cela garantit que le CI évalue ce qu'il prétend évaluer.
2) **Identifier les représentations erronées** en s'entretenant avec les seuls capables de fournir des informations fiables : les apprenants. Si les experts peuvent identifier les sujets que les apprenants ont du mal à comprendre, ils ne savent pas toujours quelles sont les représentations erronées les plus répandues.
3) **Élaborer des questions** à choix multiples dont les réponses incorrectes correspondent aux représentations erronées des apprenants en utilisant les données de l'étape 2.
4) **Valider les questions** par des essais, des interviews de suivi et les commentaires des experts. En outre, la fiabilité du CI doit être mesurée par des méthodes statistiques.

D'autres s'inspirent des normes pour les tests éducatifs et psychologiques [Association et al., 1999]. Tew et Guzdial mettent en place un processus en sept étapes pour développer et valider un outil d'évaluation sommative consacré à des concepts fondamentaux dans le cadre d'un cours d'introduction à la programmation [Tew and Guzdial, 2010].

1) **Établir l'objectif et la définition du CI**, à savoir ce qui doit être mesuré et ce que les scores signifient.
2) **Faire réviser le CI par des experts** pour fournir des preuves de la validité de son contenu, s'assurer que tous les concepts visés sont adéquatement représentés et que des concepts étrangers ne sont pas inclus.
3) **Créer une banque de questions**.
4) **Rendre le CI indépendant d'un langage de programmation particulier**, pour s'assurer que les apprenants sont capables de transférer de manière appropriée leur compréhension.
5) **Tester le CI dans le cadre d'une phase pilote**, pour apporter les révisions nécessaires avant la sélection des questions candidates finales.
6) **Établir la validité du CI** pour s'assurer qu'il mesure ce qui devait l'être.
7) **Établir la fiabilité du CI** en vérifiant la cohérence des résultats lors d'administrations répétées.

Enfin, Adams et Wieman établissent un processus en six étapes, dont les dernières sont itératives [Adams and Wieman, 2011].

1) **Établir les sujets de l'évaluation** en discutant avec des enseignants expérimentés ou des experts.
2) **Identifier**, par observations et interviews, **comment la pensée des apprenants à propos de ces sujets s'écarte de la pensée des experts**.
3) **Créer des questions ouvertes** et les administrer aux apprenants pour approfondir les problèmes soulevés à l'étape précédente.
4) **Créer des questions fermées** qui mesurent la pensée des apprenants.
5) **Valider les questions** via des interviews avec des apprenants et des experts.
6) Administrer les questions à une plus large population et **effectuer des tests statistiques** sur les résultats.

Un constat peut être fait : développer un CI de façon rigoureuse prend du temps. Cela explique en partie la déclaration des enseignants quant au peu d'évaluations formatives et sommatives valides et fiables à leur disposition.

Comparativement, ces trois processus sont grandement similaires. De leurs points communs est issu le **processus de développement en cinq étapes génériques et itératives** (cfr Figure 6.1) utilisé dans le cadre de cette thèse, auxquelles sont proposées des alternatives.

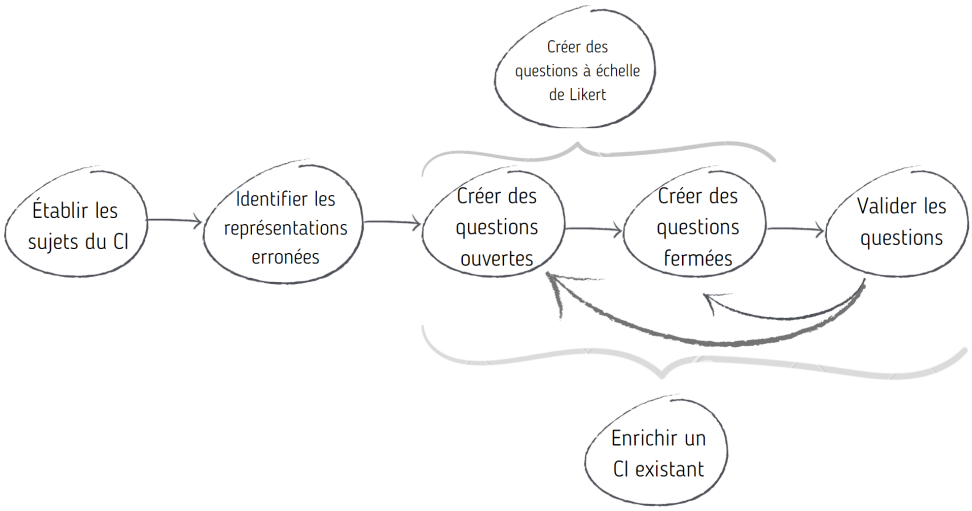


FIGURE 6.1 – Processus de développement d'un CI utilisé dans le cadre de cette thèse : ajout de deux alternatives

1) **Établir les sujets du CI** grâce à des observations, des discussions avec les apprenants, avec des enseignants expérimentés ou encore des experts, mais également à travers la lecture de la littérature scientifique.
2) **Identifier les représentations erronées** chez les apprenants suivant une approche similaire : observations réalisées en contexte, échanges avec des apprenants, des enseignants et experts et lecture de la littérature scientifique.
3) **Créer des questions ouvertes** pour récolter des données supplémentaires quant aux représentations erronées.
4) **Créer des questions fermées**, notamment leurs distracteurs, à partir des réponses obtenues via les questions ouvertes, entre autres. La clé du développement d'un CI efficace réside dans la sélection de bonnes questions avec des distracteurs appropriés [Dick-Perez et al., 2016].
5) **Valider les questions** via administration à une large population. Il s'agit de vérifier que le CI mesure bien ce qu'il doit mesurer, que les apprenants interprètent les questions de manière cohérente, mais aussi que, pour chaque question, les distracteurs (réponses incorrectes) sont le résultat d'une représentation erronée chez l'apprenant.

Plutôt que de créer des questions ouvertes et fermées, **deux alternatives sont envisagées et testées dans le cadre de cette recherche doctorale**. La première alternative consiste à **enrichir un CI existant avec des questions valides, telles que des doublons traduits dans un langage de programmation différent ou utilisant des métaphores** (pour en mesurer l'impact sur la compréhension des apprenants). La deuxième alternative revient à **développer un CI "avec échelle de Likert"** plutôt qu'un CI à questions fermées **de façon à obtenir une vision globale, mesurer une tendance et utiliser, sur les résultats obtenus, des tests statistiques**.

6.4 L'administration d'un *concept inventory*

Dans des contextes autres que l'enseignement de l'informatique, un CI est administré avant (pré-test) et après (post-test) une séquence d'enseignement. Les résultats des pré-tests et post-tests de l'ensemble des apprenants sont ensuite comparés afin d'évaluer l'efficacité des méthodes d'enseignement mises en place [Bailey et al., 2012]. Les résultats de cette comparaison peuvent alors être utilisés pour adapter les méthodes d'enseignement et répondre ainsi aux lacunes de compréhension révélées par les CIs [Sands et al., 2018].

Cependant, il a été avancé que les novices en informatique n'avaient pas d'idée préconçue du domaine et que, par conséquent, l'administration d'un CI comme pré-test pouvait ne pas être utile : "*It is likely that a majority of student misconceptions are a result of instruction in computer science rather than based upon a set of common, naive understandings [that students] bring to the topic from their experience in the world*" [Tew and Guzdial, 2010]. Pourtant, les sujets relatifs à l'informatique feront, à l'avenir, de plus en plus partie du programme scolaire obligatoire (cfr Section 2.7), ce qui génèrera des préconceptions. En outre, celles-ci peuvent exister hors d'un contexte d'enseignement de l'informatique, inspirées par les connaissances antérieures en mathématiques, notamment. Dès lors, dans le cadre de cette thèse, l'administration d'un CI est considérée suivant une approche pré-test-post-test [Shuttleworth, 2009].

6.5 Une recherche exploratoire

Cette recherche doctorale se veut exploratoire, mettant en jeu des CIs de formes différentes (questions à choix multiple ou énoncés avec échelle de Likert), visant des objectifs différents, dans des contextes relativement différents (cfr Table 6.1).

Cinq CIs ont été développés : deux CIs sur le concept de variable, un sur le concept d'instruction conditionnelle, un sur le concept de fonction et un sur le concept de *finite state machine* (FSM).

Deux de ces CIs ont été élaborés sur base du processus en cinq étapes décrit ci-avant (cfr Section 6.3) : celui sur le concept d'instruction conditionnelle et celui sur le concept de fonction. Les autres CIs sont issus des alternatives proposées (cfr Figure 6.1). La première alternative a abouti à un CI sur le concept de variable, version enrichie du CI de Dehnadi [Dehnadi, 2009]. La deuxième a conduit à développer

TABLE 6.1 – Synthèse des résultats décrits dans cette thèse

Usages	**CIs instr. cond. et fonction**	**CI variable choix multiple**	**CI variable échelle Likert**	**CI FSM**
Identifier les représentations et/ou modèles mentaux	cas d'étude 1	cas d'étude 1	cas d'étude 2	cas d'util. 3
Quantifier les représentations erronées et/ou modèles mentaux			cas d'util. 2	cas d'util. 3
Mesurer le gain de compréhension		cas d'util. 1		cas d'util. 3

deux CIs "à échelle de Likert" : un sur le concept de variable et l'autre, sur le concept de FSM.

Les cinq CIs ainsi obtenus ont fait l'objet deux cas d'étude et trois cas d'utilisation (cfr Figure 6.2) :

— un cours d'introduction à la programmation en première année de bachelier (**cas d'étude 1** - cfr Chapitre 7 - et **cas d'utilisation 1** - cfr Chapitre 9) ;
— un cours de programmation orientée-objet en deuxième année de bachelier (**cas d'étude 2** - cfr Chapitre 8 - et **cas d'utilisation 2** - cfr Chapitre 10) ;
— un cours de modélisation des systèmes embarqués en master (**cas d'utilisation 3** - cfr Chapitre 11).

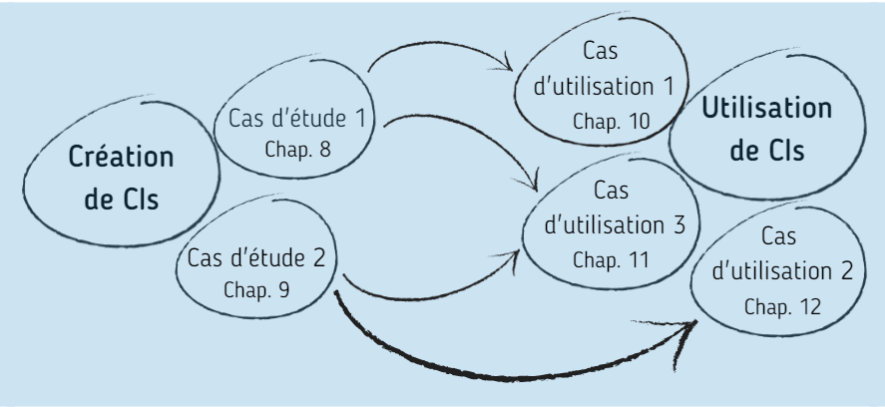


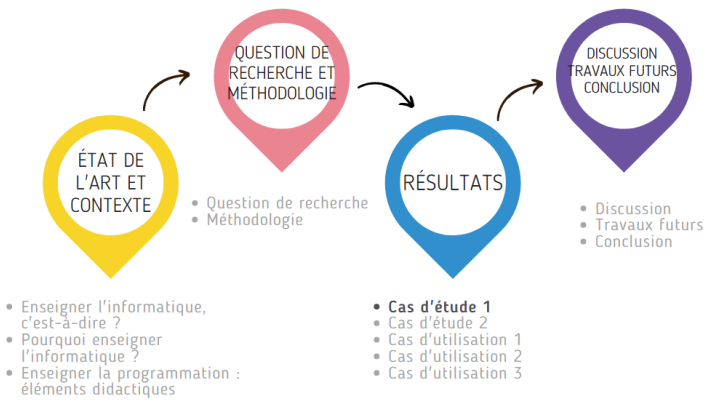
FIGURE 6.2 – Vue globale des cas d'étude et d'utilisation décrits dans le cadre de cette thèse

Troisième partie

Résultats

CHAPITRE 7

CRÉER DES CONCEPT INVENTORIES POUR IDENTIFIER REPRÉSENTATIONS ERRONÉES ET MODÈLES MENTAUX (CAS D'ÉTUDE 1)



ARTICLE PUBLIÉ INSPIRANT CE CHAPITRE

- Henry, J., & Dumas, B. (2020, October). Approach to Develop a Concept Inventory Informing Teachers of Novice Programmers' Mental Models. *Proceedings of 2020 Frontiers in Education Conference (FIE)*. IEEE.

Le premier cas d'étude est un cours d'introduction à la programmation (INFOB131) organisé en première année de bachelier à l'Université de Namur et suivi

par des étudiants issus de deux filières : informatique (INFO) et ingénierie de gestion en management de l'information (INGMI).

Ce cas d'étude illustre le processus de développement de CIs (cfr Section 6.3) consacrés aux concepts de base en programmation que sont la variable, l'instruction conditionnelle et la fonction.

7.1 Contexte

7.1.1 Le cours d'introduction à la programmation (INFOB131)

Il s'agit d'un cours imposé dans le programme des étudiants INFO et INGMI. Il s'étale sur 13 semaines, au premier semestre de l'année académique. Chaque semaine, un concept-clé en programmation est abordé en quatre heures de cours théoriques (deux heures le vendredi et deux heures le lundi de la semaine suivante) et trois séances de pratique (comprises entre le lundi et le vendredi suivant) (cfr Figure 7.1).

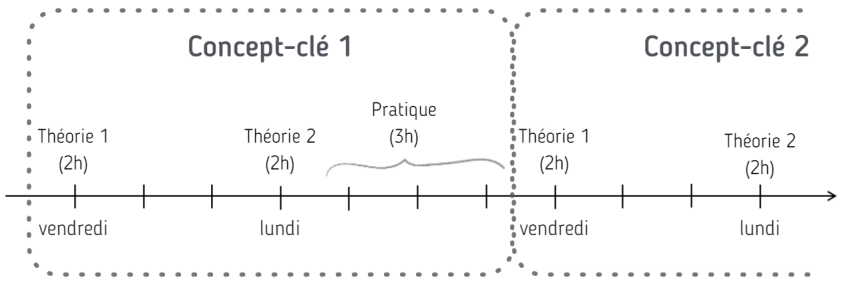


FIGURE 7.1 – Organisation de l'apprentissage d'un concept-clé au sein d'INFOB131

Durant les cours théoriques, l'enseignant s'appuie sur une présentation sous forme de diaporama et un environnement de développement intégré enrichi, Jupyter Notebook [1]. Les séances de pratique sont structurées en deux parties : une heure de pratique sur papier et deux heures de pratique sur machine. Au sein d'INFOB131, la mise en pratique se fait dans le langage de programmation Python.

Sur le semestre, trois semaines sont consacrées à des projets de plus grande envergure mettant en jeu, dans un cas concret, plusieurs concepts-clé.

Il n'est pas obligatoire d'assister aux cours théoriques ni aux séances de pratique, mais bien de participer aux projets.

7.1.2 Le cours préparatoire

Avant le début de l'année académique (et le début du cours INFOB131), les étudiants ont la possibilité de participer à un cours préparatoire facultatif qui propose

1. [ipython.org/notebook.html]

une introduction rapide à l'algorithmique et aux concepts de base de la programmation (variable, instruction conditionnelle et boucle).

Les étudiants sont d'abord initiés aux algorithmes par des activités débranchées. Ensuite, après une brève explication théorique, chaque concept de programmation est mis en pratique à travers des mini-projets de difficulté croissante. Tous les projets sont programmés via l'interface micro :bit [2], à l'aide d'un langage de programmation en blocs. Les étudiants travaillent de manière autonome, sous la supervision d'un formateur.

7.2 Le processus de développement des *concept inventories*

La méthodologie de développement des CIs utilisée dans ce cas d'étude est celle décrite en Section 6.3 et illustrée en Figure 6.1. Le développement s'est déroulé sur quatre ou cinq années académiques.

Au cours de la première année, une approche ethnographique a été appliquée pour **établir les sujets des CIs** (étape 1) et **identifier les représentations erronées** (étapes 2) des étudiants suivant le cours INFOB131. Cette approche a d'abord été mise en place au sein même du cours (semestre 1), puis a été répétée, avec les mêmes étudiants, dans le cadre d'un projet de programmation à grande échelle (+/- 10 000 lignes de code - semestre 2).

Pour établir les sujets des CIs, des discussions ont eu lieu avec l'équipe enseignante chaque semaine durant le cours INFOB131. Les réponses fournies aux examens ont été analysées. Des discussions informelles ont également eu lieu avec des experts du domaine et des enseignants en place dans d'autres contextes.

Pour identifier les représentations erronées, les étudiants ont été observés, individuellement et en groupe, pendant qu'ils programmaient. Des discussions informelles ont eu lieu régulièrement avec eux. En outre, la lecture de la littérature sur les représentations erronées dans le domaine a permis d'enrichir les observations.

Sur base des données ainsi collectées, il a été décidé de développer des CIs qui se concentreraient sur les quatre premiers concepts abordés dans le cours INFOB131, à savoir la **variable**, l'**instruction conditionnelle** et la **fonction** et la **boucle**.

En outre, l'enseignement de ces concepts s'étalent sur une période durant laquelle les étudiants peuvent encore abandonner leur cursus et se réorienter. Á travers les mesures effectuées via les CIs, il est notamment question d'éviter au maximum ces réorientations en agissant au plus vite auprès des étudiants.

Le processus de **création des questions** (étapes 3 et 4) varie en fonction du concept de programmation considéré, s'appuyant soit sur des travaux existants, soit sur les observations faites au cours de l'étude ethnographique.

L'identification des modèles mentaux étant cruciale dans ce cas d'étude, un CI publié mesurant les modèles mentaux utilisés par les étudiants pour résoudre des affectations de variables sert de base au CI dédié aux variables [Dehnadi, 2009]. En outre, les questions qu'il contient sont également une source d'inspiration pour la création de questions au sein des autres CIs.

2. [makecode.microbit.org/]

Par ailleurs, les observations ethnographiques ont mis en évidence les principales difficultés rencontrées par les étudiants vis-à-vis des concepts considérés dans le cas d'étude. Sur base de ces résultats, des questions ouvertes ont été créées : des énoncés mettant en jeu respectivement des instructions conditionnelles, des fonctions et des boucles. À travers les réponses à ces questions, il s'agit de définir les modèles mentaux utilisés par les étudiants pour résoudre les énoncés.

Pour chaque CI développé (un par concept-clé considéré), trois ou quatre itérations ont été réalisées, de la deuxième à la quatrième ou cinquième année, avec à chaque fois un groupe différent d'étudiants composés des deux populations décrites (INFO et INGMI). Chaque CI a donc été administré trois fois par an. Les données ont été recueillies à des moments précis tout au long du semestre durant lequel est organisé le cours INFOB131 (Figure 7.9), en s'inspirant de la conception pré-test post-test [Shuttleworth, 2009]. Pour chacun des concepts-clé, le CI a été administré avant le cours théorique correspondant, entre le cours théorique et les séances de cours pratique, et après celles-ci. Les CIs ont été présentés aux étudiants en tant qu'évaluation formative [Fisher and Frey, 2014]. Il n'a pas été précisé aux étudiants que les questionnaires étaient les mêmes pour les trois administrations. Si certains étudiants répondaient de façon erronées après le troisième questionnaire sur le même sujet, ils étaient contactés individuellement et se voyaient proposer une aide.

La **validation des questions** (étape 5) repose sur l'analyse des données collectées lors des années suivantes (année 2 à année 5).

Dans les sections qui suivent, seuls les trois CIs sur lesquels un travail a été fait concernant l'identification des modèles mentaux sont présentés, à savoir un CI sur la variable, un CI sur l'instruction conditionnelle et un CI sur la fonction. Bien qu'un CI ait été développé pour le concept de boucle, aucune des données collectées sur trois années n'a pu être traitée dans le cadre de cette thèse.

7.3 Un *concept inventory* sur le concept de variable

7.3.1 Le développement

Pour le CI sur le concept de variable, le choix a été fait de se baser sur un travail existant ayant fait l'objet de plusieurs articles scientifiques [Ahadi et al., 2014, Bornat, 2014, Caspersen et al., 2007, Dehnadi et al., 2006] et permettant d'identifier des modèles mentaux utilisés par les étudiants lors de problèmes d'affectaction de variables : le questionnaire de Dehnadi [3] [Dehnadi, 2009].

Ce questionnaire est composé de 12 questions proposant trois types de problèmes distincts à résoudre, à savoir des affectations simples (3 questions) (cfr Figure 7.2), doubles (3) (cfr Figure 7.3) et triples (6) (cfr Figure 7.4). Les questions sont des questions à choix multiples : les répondants doivent choisir les valeurs correctes contenues dans des variables définies parmi un ensemble de réponses proposées (cfr Figure 7.5).

3. Questionnaire [eis.mdx.ac.uk/staffpages/r_bornat/tests/distribution.html], consulté le 25/01/2022

```
int a = 10;
int b = 20;

a = b;
```

FIGURE 7.2 – Affectation simple

```
int a = 10;
int b = 20;

b = a;
a = b;
```

FIGURE 7.3 – Affectation double

```
int a = 5;
int b = 3;
int c = 7;

b = a;
c = b;
a = c;
```

FIGURE 7.4 – Affectation triple

1. Read the following statements and tick the box next to the correct answer in the next column.

```
int  a = 10;
int  b = 20;

a = b;
```

The new values of a and b are:

- ☐ a = 10 b = 10
- ☐ a = 30 b = 20
- ☐ a = 0 b = 10
- ☐ a = 20 b = 20
- ☐ a = 0 b = 30
- ☐ a = 10 b = 20
- ☐ a = 20 b = 10
- ☐ a = 20 b = 0
- ☐ a = 10 b = 30
- ☐ a = 30 b = 0

Any other values for a and b:

a = b =
a = b =
a = b =

Use this column for your rough notes please

FIGURE 7.5 – Exemple de question proposée dans le questionnaire “papier” de Dehnadi

Les problèmes sont rédigés en Java. Cette différence de langage de programmation entre le questionnaire de Dehnadi et le cours INFOB131 a été volontairement conservée. Les étudiants sont sensés apprendre les concepts de programmation de manière théorique. Le langage utilisé dans le cours INFOB131, Python, n’est qu’une façon de les mettre en œuvre. Il est attendu des étudiants qu’ils soient capables de transférer leurs connaissances dans un autre langage, notamment Java. Les différences de syntaxe entre une affectation en Python et une affectation en Java sont, de plus, mineures : le typage à la déclaration des variables et les points-virgules. Par ailleurs, le typage des variables est abordé, de façon théorique, dans le cours INFOB131. Finalement, aucun étudiant n’a soulevé ces différences lors de l’administration du CI.

Dehnadi a identifié onze modèles mentaux différents (cfr Tableau 7.1) utilisés par les étudiants pour résoudre les problèmes proposés dans son questionnaire. Chaque question à choix multiple comprend une réponse correcte unique et des distracteurs (Figure 7.5) qui sont soigneusement construits pour illustrer tous les modèles mentaux identifiés par Dehnadi. Des choix ouverts (“autres”) sont disponibles si

l'étudiant souhaite proposer une solution personnelle.

TABLE 7.1 – Les modèles mentaux liés au concept de variable et identifiés par Dehnadi.
La numérotation ne correspond pas à celle proposée par Dehnadi dans sa publication originale.

M1	The value of the variable on the right is assigned to the variable on the left; the variable on the right is initialized to 0
M2	The value of the variable on the right is assigned to the variable on the left; the variable on the right retains its original value
M3	The value of the variable on the left is assigned to the variable on the right; the variable on the left is initialized to 0
M4	The value of the variable on the left is assigned to the variable on the right; the variable on the left retains its original value
M5	The value of the variable on the right is added to the value of the variable on the left; the variable on the right is initialized to 0
M6	The value of the variable on the right is added to the value of the variable on the left; the variable on the right retains its original value
M7	The value of the variable on the left is added to the value of the variable on the right; the variable on the left is initialized to 0
M8	The value of the variable on the left is added to the value of the variable on the right; the variable on the left retains its original value
M9	The values assigned to the two variables are exchanged
M10	Variables retain their original values
M11	Mathematical equality is applied

En outre, les questions de Dehnadi peuvent être mises en relation avec les principales représentations erronées des variables identifiées par Sorva [Sorva et al., 2012] (cfr Tableau 7.2, Figure 7.6).

TABLE 7.2 – Les représentations erronées liées au concept de variable et décrites par Sorva

MIS8	Magical parallelism : Several lines of a (simple non-concurrent) program can be simultaneously active or known
MIS9a	A variable can hold multiple values at a time
MIS9b	A variable “remembers” old values
MIS11	Primitive assignment works in opposite direction
MIS12	Primitive assignment works both directions (swaps)

1.
int a = 10;
int b = 20;
a = b;
	a = 20, b = 0	M1	MIS11

	a = 20, b = 20	M2 or M11	MIS9a
	a = 0, b = 10	M3	MIS11
	a = 10, b = 10	M4 or M11	MIS9a, MIS11
	a = 30, b = 0	M5	MIS9b
	a = 30, b = 20	M6	MIS9b
	a = 0, b = 30	M7	MIS9b, MIS11
	a = 10, b = 30	M8	MIS9b, MIS11
	a = 20, b = 10	M9	MIS12
	a = 10, b = 20	M10	MIS9b

FIGURE 7.6 – Question 1 du questionnaire de Dehnadi - Chaque réponse a été associée à un modèle mental (défini par Dehnadi) et à une (ou plusieurs) représentation(s) erronée(s) (identifiée(s) par Sorva)

Durant la deuxième année de l'étude, le questionnaire de Dehnadi a été utilisé tel quel.

Au cours de la troisième année, **la version originale du questionnaire de Dehnadi a été enrichie**. Les valeurs de certaines questions ont été modifiées pour éviter toute confusion entre celles-ci. De plus, trois questions ont été ajoutées, portant le nombre total de questions à 15. Ces questions sont des versions dupliquées, formatées dans un langage de programmation par blocs, de questions existantes (cfr Figure 7.7) : une affectation simple, une double et une triple. L'objectif visé par cet ajout était d'identifier la dépendance des réponses au langage de programmation. Elles ont été placées dans le questionnaire de façon à ne pas suivre directement leur homologue en langage Java.

La quatrième année a vu l'ajout d'une seizième question : une version dupliquée enrichie d'une représentation métaphorique. L'idée était de mesurer l'impact d'une telle représentation, souvent utilisée durant les cours d'introduction à la programmation, pour diminuer le caractère abstrait des variables.

Enfin, le CI à **16 questions** a été administré une cinquième année.

Au terme de ces cinq années, **aucun modèle mental n'a été ajouté à la liste définie par Dehnadi** (Tableau 7.1)

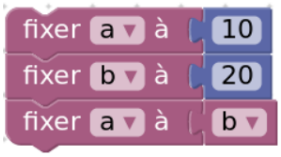


FIGURE 7.7 – Duplicata de la question 1 de Dehnadi traduite en langage de programmation par blocs

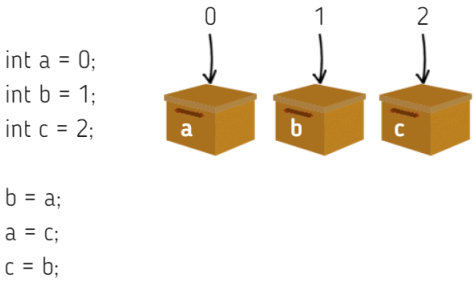


FIGURE 7.8 – Duplicata de la question 11 de Dehnadi enrichie de la métaphore de la "boite"

7.3.2 L'administration

Étant donné l'organisation du cours INFOB131 (cfr Figure 7.1) et la méthodologie envisagée dans le cadre de cette thèse (cfr Section 6.3), le CI a été administré selon

une approche pré-test post-test rendant possible la mesure d'une évolution au sein des données collectées [Shuttleworth, 2009] (cfr Figure 7.9). Une administration intermédiaire (mid-test) a été envisagée pour mesurer l'impact des différentes modalités offertes par le cours INFOB131, à savoir les cours théoriques et les séances de pratique. Le CI traitant des variables a donc été complété trois fois par an, durant quatre années, par respectivement 107, 112, 104 et 108 étudiants, soit un total de 431 étudiants.

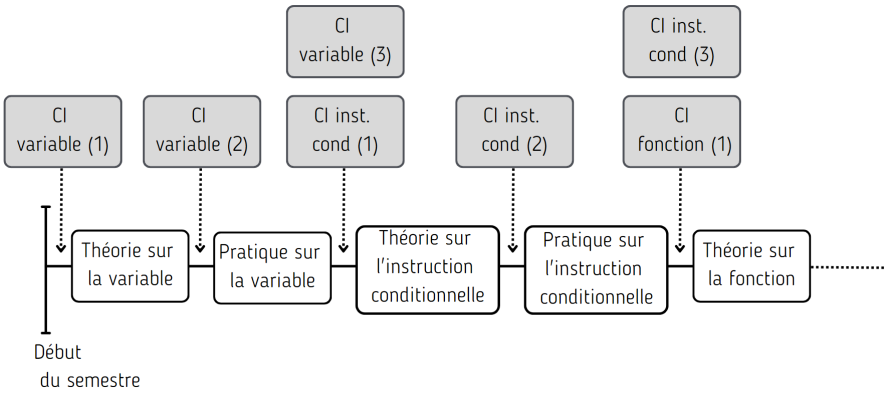


FIGURE 7.9 – Organisation de l'administration des CIs en fonction du cours INFOB131

7.3.3 Le *concept inventory*

Questions	Réponses	Modèles mentaux	Rep. erronées
Affectations simples			

1.
int a = 10;
int b = 20;
a = b;
	a = 20, b = 0	M1	MIS11

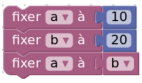
	a = 20, b = 20	M2 or M11	MIS9a
	a = 0, b = 10	M3	MIS11
	a = 10, b = 10	M4 or M11	MIS9a, MIS11
	a = 30, b = 0	M5	MIS9b
	a = 30, b = 20	M6	MIS9b
	a = 0, b = 30	M7	MIS9b, MIS11
	a = 10, b = 30	M8	MIS9b, MIS11
	a = 20, b = 10	M9	MIS12
	a = 10, b = 20	M10	MIS9b

Questions	Réponses	Modèles mentaux	Rep. erronées

2.
int a = 40;
int b = 30;
b = a;
	a = 0, b = 40	M1	MIS11

	a = 40, b = 40	M2 or M11	MIS9a
	a = 30, b = 0	M3	MIS11
	a = 30, b = 30	M4 or M11	MIS9a, MIS11
	a = 0, b = 70	M5	MIS9b
	a = 40, b = 70	M6	MIS9b
	a = 70, b = 0	M7	MIS9b, MIS11
	a = 70, b = 30	M8	MIS9b, MIS11
	a = 30, b = 40	M9	MIS12
	a = 40, b = 30	M10	MIS9b

3.
int big = 10;
int small = 20;
big = small;
	big = 20, small = 0	M1	MIS11

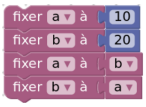
	big = 20, small = 20	M2 or M11	MIS9a
	big = 0, small = 10	M3	MIS11
	big = 10, small = 10	M4 or M11	MIS9a, MIS11
	big = 30, small = 0	M5	MIS9b
	big = 30, small = 20	M6	MIS9b
	big = 0, small = 30	M7	MIS9b, MIS11
	big = 10, small = 30	M8	MIS9b, MIS11
	big = 20, small = 10	M9	MIS12
	big = 10, small = 20	M10	MIS9b
3'. 	a = 20, b = 0	M1	MIS11
	a = 20, b = 20	M2 or M11	MIS9a
	a = 0, b = 10	M3	MIS11
	a = 10, b = 10	M4 or M11	MIS9a, MIS11
	a = 30, b = 0	M5	MIS9b
	a = 30, b = 20	M6	MIS9b
	a = 0, b = 30	M7	MIS9b, MIS11
	a = 10, b = 30	M8	MIS9b, MIS11
	a = 20, b = 10	M9	MIS12
	a = 10, b = 20	M10	MIS9b
Affectations doubles			

4.
int a = 10;
int b = 20;
a = b;
b = a;
	a = 0, b = 20	M1/S	MIS11

	a = 20, b = 20	M2/S or M11/SA	MIS8, MIS9a
	a = 10, b = 0	M3/S	MIS11
	a = 10, b = 10	M4/S or M11/SA	MIS8, MIS9a, MIS11
	a = 0, b = 30	M5/S	MIS9b
	a = 30, b = 50	M6/S	MIS9b
	a = 30, b = 0	M7/S	MIS9b, MIS11
	a = 40, b = 30	M8/S	MIS9b, MIS11
	a = 10, b = 20	M9/S or M10	MIS9b, MIS12
	a = 20, b = 10	M2/SA or M9/NS	MIS9b, MIS12
	a = 30, b = 30	M5/SA or M6/SA or M7/SA or M8/SA	MIS8, MIS9b, MIS11

Questions	Réponses	Modèles mentaux	Rep. erronées

5.
int a = 40;
int b = 30;
b = a;
a = b;
	a = 40, b = 0	M1/S	MIS11

	a = 40, b = 40	M2/S or M11/SA	MIS8, MIS9a
	a = 0, b = 30	M3/S	MIS11
	a = 30, b = 30	M4/S or M11/SA	MIS8, MIS9a, MIS11
	a = 70, b = 0	M5/S	MIS9b
	a = 110, b = 70	M6/S	MIS9b
	a = 0, b = 70	M7/S	MIS9b, MIS11
	a = 70, b = 100	M8/S	MIS9b, MIS11
	a = 40, b = 30	M9/S or M10	MIS9b, MIS12
	a = 30, b = 40	M2/SA or M9/NS	MIS9b, MIS12
	a = 70, b = 70	M5/SA or M6/SA or M7/SA or M8/SA	MIS8, MIS9b, MIS11
5'. 	a = 0, b = 20	M1/S	MIS11
	a = 20, b = 20	M2/S or M11/Sa	MIS9a
	a = 10, b = 0	M3/S	MIS11
	a = 10, b = 10	M4/S or M11/SA	MIS9a, MIS11
	a = 0, b = 30	M5/S	MIS9b
	a = 30, b = 50	M6/S	MIS9b
	a = 30, b = 0	M7/S	MIS9b, MIS11
	a = 40, b = 30	M8/S	MIS9b, MIS11
	a = 10, b = 20	M9/S or M10	MIS9b, MIS12
	a = 20, b = 10	M2/SA or M9/NS	MIS9b, MIS12
	a = 30, b = 30	M5/SA or M6/SA or M7/SA or M8/SA	MIS8, MIS9b, MIS11

6.
int a = 10;
int b = 20;
int c = 30;
a = b;
b = c;
	a = 20, b = 30, c = 0	M1/S	MIS11

	a = 20, b = 30, c = 30	M2/S or M1/SA or M2/SA	MIS8
	a = 0, b = 0, c = 10	M3/S	MIS11
	a = 10, b = 10, c = 10	M4/S	MIS11
	a = 30, b = 30, c = 0	M5/S	MIS9b
	a = 30, b = 50, c = 30	M6/S or M5/SA or M6/SA	MIS8, MIS9b
	a = 0, b = 0, c = 60	M7/S	MIS9b, MIS11
	a = 10, b = 30, c = 60	M8/S	MIS9b, MIS11
	a = 20, b = 30, c = 10	M9/S	MIS12
	a = 10, b = 20, c = 30	M10	MIS9b
	a = 10, b = 10, c = 20	M3/SA or M4/SA	MIS8, MIS11
	a = 10, b = 50, c = 30	M7/SA or M8/SA	MIS8, MIS9b, MIS11

Questions	Réponses	Modèles mentaux	Rep. erronées
Affectation triples			

7.
int a = 5;
int b = 3;
int c = 7;
a = c;
b = a;
c = b;
	a = 0, b = 0, c = 7	M1/S	MIS11

	a = 7, b = 7, c = 7	M2/S	/
	a = 3, b = 5, c = 0	M3/S	MIS11
	a = 3, b = 5, c = 5	M4/S	MIS11
	a = 0, b = 0, c = 15	M5/S	MIS9b
	a = 12, b = 15, c = 22	M6/S	MIS9b
	a = 3, b = 12, c = 0	M7/S	MIS9b, MIS11
	a = 8, b = 15, c = 12	M8/S	MIS9b, MIS11
	a = 3, b = 5, c = 7	M9/S	MIS12
	a = 5, b = 3, c = 7	M10	MIS9b
	a = 7, b = 5, c = 3	M1/SA or M2/SA	MIS8, MIS11
	a = 3, b = 7, c = 5	M3/SA or M4/SA	MIS8, MIS11
	a = 12, b = 8, c = 10	M5/SA or M6/SA	MIS8, MIS9b
	a = 8, b = 10, c = 12	M7/SA or M8/SA	MIS8, MIS9b, MIS11

8.
int a = 6;
int b = 9;
int c = 5;
c = b;
b = a;
a = c;
	a = 9, b = 6, c = 0	M1/S	MIS11

	a = 9, b = 6, c = 9	M2/S	/
	a = 0, b = 0, c = 5	M3/S	MIS11
	a = 5, b = 5, c = 5	M4/S	MIS11
	a = 14, b = 6, c = 0	M5/S	MIS9b
	a = 20, b = 15, c = 14	M6/S	MIS9b
	a = 0, b = 0, c = 20	M7/S	MIS9b, MIS11
	a = 20, b = 14, c = 25	M8/S	MIS9b, MIS11
	a = 9, b = 6, c = 5	M9/S	MIS12
	a = 6, b = 9, c = 5	M10	MIS9b
	a = 5, b = 6, c = 9	M1/SA or M2/SA	MIS8, MIS11
	a = 9, b = 5, c = 6	M3/SA or M4/SA	MIS8, MIS11
	a = 11, b = 15, c = 14	M5/SA or M6/SA	MIS8, MIS9b
	a = 15, b = 14, c = 11	M7/SA or M8/SA	MIS8, MIS9b, MIS11

9.
int a = 14;
int b = 20;
int c = 3;
c = b;
a = c;
b = a;
	a = 0, b = 20, c = 0	M1/S	MIS11

	a = 20, b = 20, c = 20	M2/S	/
	a = 3, b = 0, c = 14	M3/S	MIS11
	a = 3, b = 3, c = 14	M4/S	MIS11
	a = 0, b = 37, c = 0	M5/S	MIS9b
	a = 37, b = 57, c = 23	M6/S	MIS9b
	a = 23, b = 0, c = 14	M7/S	MIS9b, MIS11
	a = 37, b = 23, c = 17	M8/S	MIS9b, MIS11
	a = 3, b = 20, c = 14	M9/S	MIS12
	a = 14, b = 20, c = 3	M10	MIS9b
	a = 3, b = 14, c = 20	M1/SA or M2/SA	MIS8, MIS11
	a = 20, b = 3, c = 14	M3/SA or M4/SA	MIS8, MIS11
	a = 17, b = 34, c = 23	M5/SA or M6/SA	MIS8, MIS9b
	a = 34, b = 23, c = 17	M7/SA or M8/SA	MIS8, MIS9b, MIS11

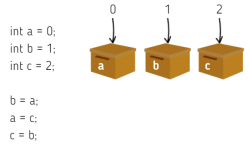
Questions	Réponses	Modèles mentaux	Rep. erronées

10.
int a = 3;
int b = 10;
int c = 2;
b = a;
c = b;
a = c;
	a = 3, b = 0, c = 0	M1/S	MIS11

	a = 3, b = 3, c = 3	M2/S	/
	a = 0, b = 2, c = 10	M3/S	MIS11
	a = 10, b = 2, c = 10	M4/S	MIS11
	a = 15, b = 0, c = 0	M5/S	MIS9b
	a = 18, b = 13, c = 15	M6/S	MIS9b
	a = 0, b = 2, c = 13	M7/S	MIS9b, MIS11
	a = 13, b = 12, c = 15	M8/S	MIS9b, MIS11
	a = 3, b = 2, c = 10	M9/S	MIS12
	a = 3, b = 10, c = 2	M10	MIS9b
	a = 2, b = 3, c = 10	M1/SA or M2/SA	MIS8, MIS11
	a = 10, b = 2, c = 3	M3/SA or M4/SA	MIS8, MIS11
	a = 5, b = 13, c = 12	M5/SA or M6/SA	MIS8, MIS9b
	a = 13, b = 12, c = 5	M7/SA or M8/SA	MIS8, MIS9b, MIS11

11.
int a = 0;
int b = 1;
int c = 2;
b = a;
a = c;
c = b;
	a = 2, b = 0, c = 0	M1/S or M2/S	MIS11

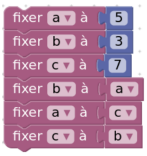
	a = 0, b = 1, c = 0	M3/S	MIS11
	a = 1, b = 1, c = 1	M4/S	MIS11
	a = 2, b = 0, c = 1	M5/S or M1/SA or M2/SA	MIS8, MIS9b, MIS11
	a = 2, b = 1, c = 3	M6/S or M5/SA or M6/SA	MIS8, MIS9b
	a = 0, b = 3, c = 0	M7/S	MIS9b, MIS11
	a = 1, b = 4, c = 3	M8/S	MIS9b, MIS11
	a = 2, b = 1, c = 0	M9/S	MIS12
	a = 0, b = 1, c = 2	M10	MIS9b
	a = 10, b = 2, c = 3	M3/SA or M4/SA	MIS8, MIS11
	a = 13, b = 12, c = 5	M7/SA or M8/SA	MIS8, MIS9b, MIS11

11'.
 
int a = 0;
int b = 1;
int c = 2;
b = a;
a = c;
c = b;
	a = 2, b = 0, c = 0	M1/S or M2/S	MIS11

	a = 0, b = 1, c = 0	M3/S	MIS11
	a = 1, b = 1, c = 1	M4/S	MIS11
	a = 2, b = 0, c = 1	M5/S or M1/SA or M2/SA	MIS8, MIS9b
	a = 2, b = 1, c = 3	M6/S or M5/SA or M6/SA	MIS8, MIS9b
	a = 0, b = 3, c = 0	M7/S	MIS9b, MIS11
	a = 1, b = 4, c = 3	M8/S	MIS9b, MIS11
	a = 2, b = 1, c = 0	M9/S	MIS12
	a = 0, b = 1, c = 2	M10	MIS9b
	a = 10, b = 2, c = 3	M3/SA or M4/SA	MIS8, MIS11
	a = 13, b = 12, c = 5	M7/SA or M8/SA	MIS8, MIS9b, MIS11

Questions	Réponses	Modèles mentaux	Rep. erronées

12.
int a = 2;
int b = 4;
int c = 8;
a = c;
c = b;
b = a;
	a = 0, b = 8, c = 4	M1/S	MIS11

	a = 8, b = 8, c = 4	M2/S	/
	a = 2, b = 0, c = 0	M3/S	MIS11
	a = 2, b = 2, c = 2	M4/S	MIS11
	a = 0, b = 10, c = 4	M5/S	MIS9b
	a = 10, b = 14, c = 12	M6/S	MIS9b
	a = 14, b = 0, c = 0	M7/S	MIS9b, MIS11
	a = 16, b = 14, c = 10	M8/S	MIS9b, MIS11
	a = 2, b = 8, c = 4	M9/S	MIS12
	a = 2, b = 4, c = 8	M10	MIS9b
	a = 8, b = 2, c = 4	M1/SA or M2/SA	MIS8, MIS11
	a = 4, b = 8, c = 2	M3/SA or M4/SA	MIS8, MIS11
	a = 10, b = 6, c = 12	M5/SA or M6/SA	MIS8, MIS9b
	a = 6, b = 12, c = 10	M7/SA or M8/SA	MIS8, MIS9b, MIS11
12'. 	a = 2, b = 0, c = 1	M1/S or M2/S	MIS11
	a = 0, b = 1, c = 0	M3/S	MIS11
	a = 1, b = 1, c = 1	M4/S	MIS11
	a = 2, b = 0, c = 1	M5/S or M1/SA or M2/SA	MIS8, MIS9b
	a = 2, b = 1, c = 3	M6/S or M5/SA or M6/SA	MIS8, MIS9b
	a = 0, b = 3, c = 0	M7/S	MIS9b, MIS11
	a = 1, b = 4, c = 3	M8/S	MIS9b, MIS11
	a = 2, b = 1, c = 0	M9/S	MIS12
	a = 0, b = 1, c = 2	M10	MIS9b
	a = 10, b = 2, c = 3	M3/SA or M4/SA	MIS8, MIS11
	a = 13, b = 12, c = 5	M7/SA or M8/SA	MIS8, MIS9b, MIS11

7.3.4 Résultats préliminaires

Le CI sur le concept de variable en programmation a été utilisé dans le cadre d'une étude visant à mesurer le gain d'apprentissage des étudiants durant leur suivi d'un cours d'introduction à la programmation. Les résultats de cette étude font l'objet du Chapitre 9.

7.4 Un *concept inventory* sur le concept d'instruction conditionnelle

7.4.1 Le développement

D'après les résultats de l'approche ethnographique menée durant la première année de l'étude, l'instruction conditionnelle semble être le concept de programmation le plus facilement compréhensible pour les étudiants. Dès lors, le CI élaboré et administré durant la deuxième année comptait seulement trois questions, comparables aux problèmes proposés dans le CI de Dehnadi : un énoncé présentant une instruction conditionnelle "if" avec une condition vraie et une clause "then" (cfr Figure 7.10), un énoncé présentant une instruction conditionnelle "if" avec une condition fausse et des clauses "then" et "else" (cfr Figure 7.11), un énoncé présentant deux instructions conditionnelles "if" imbriquées, chacun avec une condition fausse (cfr Figure 7.12). Les différences de syntaxe, en ce qui concerne les instructions conditonnelles, entre le langage Python et le langage Java sont mineures. Les problèmes sont donc proposés en Java, comme dans le CI de Dehnadi. Lors de la première administration, les questions étaient des questions ouvertes. Sur base des résultats obtenus et des discussions informelles menées avec les étudiants concernant le CI, **onze modèles mentaux ont été identifiés** (cfr Tableau 7.4). En outre, il est apparu un problème syntaxique qui n'a pas été considéré dans la définition des modèles mentaux, à savoir que certains étudiants ont assimilé l'opérateur d'égalité ("==") à l'opérateur d'affectation ("="). Ces étudiants peuvent toutefois être considérés dans les résultats : mis de côté ce problème syntaxique, ils utilisent les modèles mentaux M1, M4 ou M6 (cfr Table 7.4).

```
int a = 5;
int b = 3;
if (a > b) {
   a = a + b;
}
b = a;
```

FIGURE 7.10 – "if(true)-then"

```
int a = 5;
int b = 3;
if (a == b) {
   a = a + b;
   b = 2;
}
else {
   a = a - b;
   b = 3;
}
b = a;
```

FIGURE 7.11 – "if(false)-then-else"

```
int a = 5;
int b = 3;
if (a < b) {
   a = a + b;
   b = 2;
}
else {
   if (a == b) {
      a = a - b;
      b = 3;
   }
   else {
      b = a + b ;
      a = 7;
   }
}
b = a;
```

FIGURE 7.12 – Nested "if(false)-then-else"

En troisième année, le CI s'est enrichi de trois questions : un énoncé présentant une instruction conditionnelle "if" avec une condition fausse et une clause "then",

ainsi que deux énoncés traduits en langage de programmation par blocs ("if-then" et "if-then-else"). Le CI est alors composé de six questions à choix multiple. Les distracteurs sont soigneusement construits pour aborder tous les modèles mentaux identifiés (cfr Table 7.4). Il reste possible pour les étudiants de proposer des solutions personnelles ("autres").

TABLE 7.4 – Les modèles mentaux liés au concept d'instruction conditionnelle

M1	Code in *if statement* is not executed, nor "then" clause (even if the condition is true) nor "else" clause (if it exists). Code before and after *if statement* is executed.
M2	Code before, after and in *if statement* ("then" **OR** "else" - if it exists - clause according the condition) are executed.
M3	Code executed in *if statement* has an effect only on the variables that are changed in it. Code after *if statement* is executed but is not affected by the changes made in *if statement*.
M4	Code before and in *if statement* ("then" **OR** "else" - if it exists - clause according the condition) are executed. Code after *if statement* is not executed.
M5	Only code before *if statement* is correctly executed. Code in *if statement* is not executed, nor "then" clause (even if the condition is true) nor "else" clause (if it exists). Code after *if statement* is not executed.
M6	Code before, after, and in *if statement* ("then" **AND** "else" - if it exists - clauses regardless of condition) are executed. "All lines of the code, without exception, are executed".
M7	Code before and in *if statement* ("then" **AND** "else" - if it exists - clauses regardless of condition) are executed. Code after *if statement* is not executed.
M8	***If statement* with an with an "else" clause** - Code before is executed. In *if statement*, only the (first) "then" clause is executed, regardless of condition and the code stops directly after.
M9	***If statement* with an with an "else" clause** - Code before is executed. In *if statement*, only the (first) "then" clause is executed, regardless of condition. Code after *if statement* is executed.
M10	**Nested *if statements* with an with an "else" clause** - Code before is executed. In *if statement*, the first condition is correctly considered (false) but in the "else" clause, only the (first) "then" clause is executed, regardless of condition and the code stops directly after. (Similar to M8)
M11	**Nested *if statements*** - Code before is executed. In *if statement*, the first condition is correctly considered (false) but in the "else" clause, only the (first) "then" clause is executed, regardless of condition. Code after *if statement* is executed. (Similar to M9)

Si certaines questions peuvent être liées aux représentations erronées identifiées par Sorva [Sorva et al., 2012] (cfr Tableau 7.5), il apparaît que, d'une part, certaines représentations erronées restent difficiles à mesurer par le biais d'une question et, d'autre part, certaines réponses fournies par les élèves révèlent des représentations erronées non listées par Sorva. **Trois nouvelles représentations erronées** sont ainsi énoncées (cfr Table 7.6).

TABLE 7.5 – Représentations erronées liées au concept d'instruction conditionnelle et décrites par Sorva

MIS23	Difficulties in understanding the sequentiality of statements
MIS24	Code after if statement is not executed if the then clause is
MIS25	If statement gets executed as soon as its condition becomes true
MIS26	A false condition ends program if no else branch
MIS27	Both then and else branches are executed
MIS28	The then branch is always executed
MIS29	Using else is optional (the next statement is always the else branch)
MIS36	All statements of a program get executed at least once

TABLE 7.6 – Nouvelles représentations erronées liées au concept d'instruction conditionnelle

new-MIS01	"What happens in the if statement stays in the if statement"
new-MIS02	The code stops at the exit of the if statement
new-MIS03	Neither then or else branches are executed

Durant la quatrième année de l'étude, deux problèmes ont été dupliqués et proposés sous forme de logigramme (cfr Figure 7.13). Ces questions avaient pour objectif de fournir une aide visuelle pour diminuer le caractère abstrait de l'énoncé. Le CI est alors composé de huit questions. Ce CI a été administré une cinquième année.

7.4.2 L'administration

Comme pour le CI traitant de la variable, ce CI a été administré selon une approche pré-test post-test (cfr Figure 7.9). Il a été complété trois fois par an, durant quatre années, par respectivement 112, 101, 108 et 47 étudiants, soit un total de 368 étudiants.

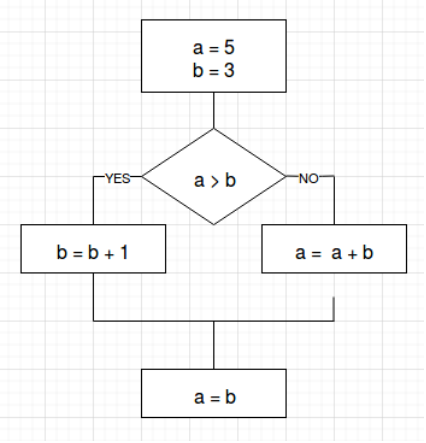


FIGURE 7.13 – Logigramme d'un énoncé présentant une instruction conditionnelle

7.4.3 Le *concept inventory*

Questions	Réponses	Modèles mentaux	Rep. erronées

1.
int a = 5;
int b = 3;
if (a > b) {
a = a + b;
}
b = a;
	a = 5, b = 5	M1	MIS23

	a = 8, b = 8	M2 or M6 or M9	/ or MIS27, MIS28 or MIS28
	a = 8, b = 5	M3	new-MIS01
	a = 8, b = 3	M4 or M7 or M8	new-MIS02 (MIS24, MIS26) or MIS27, MIS28, new-MIS02 or MIS28, new-MIS02
	a = 5, b = 3	M5	new-MIS02, new-MIS03

2.
int a = 5;
int b = 3;
if (a == b) {
a = a + b;
b = 2;
}
else {
a = a - b;
b = 3;
}
b = a;
	a = 5, b = 5	M1	MIS23

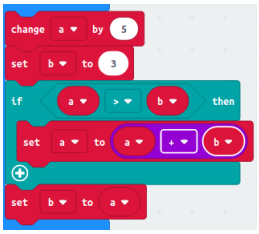
	a = 2, b = 2	M2	/
	a = 2, b = 5	M3	new-MIS01
	a = 2, b = 3	M4	new-MIS02 (MIS24, MIS26)
	a = 5, b = 3	M5	new-MIS02, new-MIS03
	a = 6, b = 6	M6	MIS27, MIS28
	a = 6, b = 3	M7	MIS27, MIS28, new-MIS02
	a = 8, b = 2	M8	MIS28, new-MIS02
	a = 8, b = 8	M9	MIS28

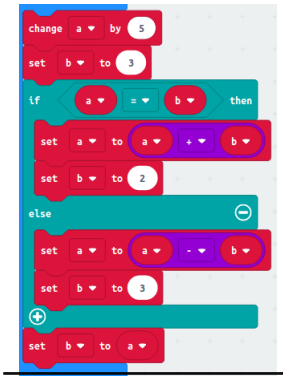
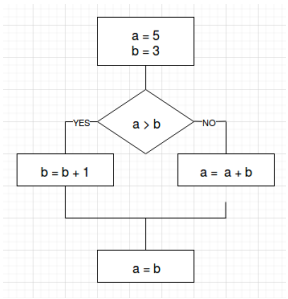
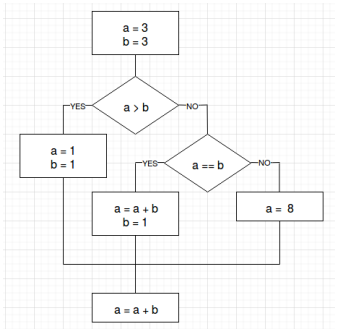
Questions	Réponses	Modèles mentaux	Rep. erronées

3.
int a = 5;
int b = 3;
if (a < b) {
a = a + b;
b = 2;
}
else {
if (a == b) {
a = a - b;
b = 3;
}
else {
b = a + b;
a = 7;
}
}
b = a;
	a = 5, b = 5	M1	MIS23

	a = 7, b = 7	M2 or M6	/ or MIS27, MIS28
	a = 7, b = 5	M3	new-MIS01
	a = 7, b = 8	M4	new-MIS02 (MIS24, MIS26)
	a = 5, b = 3	M5	new-MIS02, new-MIS03
	a = 7, b = 9	M7	MIS27, MIS28, new-MIS02
	a = 8, b = 2	M8	MIS28, new-MIS02
	a = 8, b = 8	M9	MIS28, new-MIS04
	a = 2, b = 3	M10	MIS28, new-MIS02, new-MIS04
	a = 2, b = 2	M11	MIS28

4.
int a = 4;
int b = 6;
if (a > b) {
a = a + b;
}
b = a;
	a = 4, b = 4	M1 or M2 or M3	MIS23 or / or new-MIS01

	a = 4, b = 6	M4 or M5	new-MIS02 (MIS24, MIS26) or new-MIS02, new-MIS03
	a = 10, b = 10	M6 or M9	MIS27, MIS28 or MIS28
	a = 10, b = 6	M7 or M8	MIS27, MIS28, new-MIS02 or MIS28, new-MIS02
5. 	a = 5, b = 5	M1	MIS23
	a = 8, b = 8	M2 or M6 or M9	/ or MIS27, MIS28 or MIS28
	a = 8, b = 5	M3	new-MIS01
	a = 8, b = 3	M4 or M7 or M8	new-MIS02 (MIS24, MIS26) or MIS27, MIS28, new-MIS02 or MIS28, new-MIS02
	a = 5, b = 3	M5	new-MIS02, new-MIS03

Questions	Réponses	Modèles mentaux	Rep. erronées
6. 	a = 5, b = 5	M1	MIS23
	a = 2, b = 2	M2	/
	a = 2, b = 5	M3	new-MIS01
	a = 2, b = 3	M4	new-MIS02 (MIS24, MIS26)
	a = 5, b = 3	M5	new-MIS02, new-MIS03
	a = 6, b = 6	M6	MIS27, MIS28
	a = 6, b = 3	M7	MIS27, MIS28, new-MIS02
	a = 8, b = 2	M8	MIS28, new-MIS02
	a = 8, b = 8	M9	MIS28
7. 	a = 3, b = 3	M1	MIS23 or / or new-MIS02 (MIS24, MIS26) or MIS27, MIS28
	a = 4, b = 4	M2 or M3	/ or new-MIS01
	a = 5, b = 4	M4 or M6	new-MIS02 (MIS24, MIS26) or MIS27,MIS28
	a = 5, b = 3	M5	new-MIS02, new-MIS03
	a = 8 or 9, b = 4	M7	MIS27,MIS28, new-MIS02
8. 	a = 6, b = 3	M1	MIS23
	a = 7, b = 1	M2 or M3 or M11	/ or new-MIS01 or IS28, new-MIS04
	a = 6, b = 1	M4 or M10	new-MIS02 (MIS24, MIS26) or IS28, new-MIS02
	a = 3, b = 3	M5	new-MIS02, new-MIS03
	a = 10, b = 1	M6	MIS27, MIS28
	a = 9, b = 1	M7	MIS27, MIS28, new-MIS02
	a = 1, b = 1	M8	MIS28, new-MIS02, new-MIS04
	a = 2, b = 1	M9	MIS28

7.4.4 Résultats et discussion

Au terme de cinq années de développement du CI sur l'instruction conditionnelle et au vu des réponses obtenues aux questions ouvertes, il apparait que les questions composant le CI à ce stade de sont insuffisantes pour déterminer la totalité des modèles mentaux utilisés par les programmeurs novices.

Des exercices plus complexes devraient être ajoutés : des problèmes impliquant des *if statements* imbriqués, présentant plusieurs *if statements* consécutifs ou encore, des conditions non numériques. En effet, les résultats obtenus avec le CI sur le concept de variable montrent que certains étudiants peuvent être influencés par le nom des variables. Concernant la question 3 de ce CI, certains étudiants expriment leur doute : "*big*... je ne sais pas si le terme à un impact sur la réponse...", "si un objet est large (*big*), il ne peut pas être étroit (*small*)". Dès lors, il serait intéressant de complexifier les conditions de certains problèmes, la compréhension de ces dernières étant cruciale dans la compréhension du concept d'instruction conditionnelle.

En outre, certaines questions composant actuellement le CI échouent à distinguer efficacement les modèles mentaux. Elles ont cependant été conservées sur l'ensemble de l'étude pour garantir une collecte de données cohérente.

Malgré tout, les résultats obtenus peuvent être considérés comme satifaisants. Le concept d'instruction conditionnelle est considéré comme un des plus simples en programmation. Ce constat souvent fait par les enseignants se confirme par les chiffres. Ainsi, sur l'étude complète, le plus bas taux d'étudiants utilisant le modèle mental correct (M2 - cfr Table 7.4) en pré-CI, toutes questions confondues, est de 68,3% (question 2, année 4, n = 104). Ces résultats sont évidemment à relativiser compte tenu du fait que le concept d'intruction conditionnelle a été abordé lors du cours préparatoire sans qu'aient été mesurées les représentation erronées existant chez les participants au préalable.

Fait intéressant, un CI composé d'une sélection de questions (1, 2, 3, 5, 6, 7 et 8) a été administré à un tout autre public durant l'année académique 2020-2021 : des étudiants en première année de bachelier de la filière ingénieur de gestion (INGE) suivant un cours d'introduction à la programmation propre à leur Faculté. À période équivalente (au niveau de l'apprentissage) à l'administration du pré-CI chez les INFO et INGMI, les INGE obtiennent un taux similaire (68,8%, n = 16) pour une seule question : la question "visuelle" 7. Par contre, aucune bonne réponse n'a été fournie pour la question "visuelle" 8 (*if statements* imbriqués). Les questions rédigées en langage de programmation par blocs ne semblent pas aider plus les étudiants : respectivement 18,8% et 25% de réponses correctes pour les questions 5 et 6. La question 2, plus bas taux de compréhension observé chez les INFO et INGMI, n'est considérée correcte que par 18,8% des étudiants INGE.

En bref, même si le concept d'instruction conditionnelle semble plus facilement acquis par les étudiants (sur base notamment de leur compréhension du concept de variable), il n'en reste pas moins pertinent d'étudier les modèles mentaux utilisés par les étudiants et les représentations erronées qui les induisent. Notamment, ces cinq années d'étude ont montré que les représentations erronées listées par Sorva (cfr Tableau 7.5) n'étaient pas exhaustives et qu'il restait du travail à ce niveau.

7.5 Un *concept inventory* sur le concept de fonction

7.5.1 Le développement

L'approche ethnographique menée en première année d'étude a mis en évidence une situation bien plus complexe pour le concept de fonction que pour les autres concepts abordés jusqu'ici (la variable et l'instruction conditionnelle). Les étudiants ont beaucoup de difficulté à exprimer leurs représentations (erronées) de la fonction. Il a donc été décidé, dans une première approche, de se concentrer uniquement sur certaines des représentations erronées décrites par Sorva (cfr Table 7.8). Dans sa première version (deuxième année de l'étude), le CI traitant du concept de fonction n'est composé que de quatre problèmes à résoudre : deux énoncés où des fonctions sont définies mais jamais appelées (cfr Figure 7.14) ; un énoncé où des fonctions sont définies et où l'une d'entre elles est appelée sans que la valeur retournée ne soit stockée (cfr Figure 7.15) ; un énoncé où des fonctions sont définies et appelées avec stockage des valeurs retournées (cfr Figure 7.16). Les questions étaient des questions ouvertes. En raison des différences de syntaxe entre une déclaration de fonction dans le langage Python et dans le langage Java, les problèmes sont proposés en Python. Le changement de langage aurait pu ajouter des problèmes davantage liés à la syntaxe qu'à la compréhension du concept.

TABLE 7.8 – Sélection de représentations erronées liées au concept de fonction et décrites par Sorva

MIS37	Subprogram code is executed according to the order in which the subprograms are defined
MIS38	A return values does not need to be stored (even if one needs it later)
MIS39	A method can be invoked only once
MIS40	Numbers or numeric constants are the only appropriate actual parameters corresponding to integer formal parameters
MIS42	Difficulties understanding the invocation of a method from another method
MIS43	Confusion over where return values go
MIS46	Parameter passing requires different variable names in call and signature
MIS47	Subprograms can (routinely) use the variables of calling subprograms
MIS50	A function (always) changes its input variable to become the output
MIS53	Upon return, the value of a variable changes to match a previously given parameter

Au cours de la troisième année, trois questions ouvertes supplémentaires sont venues enrichir le CI. Ces questions portent sur les représentations erronées liées aux paramètres des fonctions : un énoncé présentant deux appels pour une même

```
a = 3
def f(a) :
        return a + 1
def g(a) :
        return a * 10
def h(a) :
        a = a * 2
        return f(a) + g(a)
b = a
```

FIGURE 7.14 – Pas d'appel

```
a = 3
def f( a ) :
        return a + 1
def g( a ) :
        return a * 10
def h( a ) :
        a = a * 2
        return f(a) + g(a)
g(a)
b = a
```

FIGURE 7.15 – Appels, valeur retournée non stockée

```
a = 3
def f( a ) :
        return a + 1
def g( a ) :
        return a * 10
def h( a ) :
        a = a * 2
        return f(a) + g(a)
a = h(a)
b= a
```

FIGURE 7.16 – Appels, valeur retournée stockée

fonction avec des paramètres différents (cfr Figure 7.17), un énoncé présentant deux appels dont un avec un paramètre numérique (cfr Figure 7.18), et un énoncé présentant des appels de fonction avec deux paramètres (cfr Figure 7.19).

Étant donné la grande variété des réponses collectées durant les deuxième et troisième années, il semblait encore prématuré de construire les distracteurs des questions à choix multiples. À ce stade de l'étude, des modèles mentaux avaient pourtant été définis pour les quatre premières questions. Mais des réponses d'étudiants restaient inexplicables. La décision a alors été prise de conserver des questions ouvertes lors de la quatrième année d'étude.

En outre, si certaines réponses aux quatre premières questions peuvent être liées aux représentations erronées identifiées par Sorva [Sorva et al., 2012] (cfr Table 7.8), il apparaît que, d'une part, certaines représentations erronées restent difficiles à mesurer par le biais d'une question et, d'autre part, certaines réponses fournies par les élèves révèlent des représentations erronées non listées par Sorva.

À l'heure d'écrire cette thèse, **onze modèles mentaux** (cfr Tables 7.9 et 7.10) et **quatre nouvelle représentations erronées** (cfr Table 7.11) ont été énoncés. Il reste à définir les modèles mentaux et les représentations erronées pour les trois dernières questions, ainsi que les représentations erronées liées aux modèles mentaux M3 et M4. Le travail qu'il reste à fournir en ce sens, et qui pourrait faire l'objet d'une recherche en elle-même tant le concept de fonction semble poser problème aux

étudiants, est mentionné dans la description du CI (cfr Section 7.5.3) par la mention "TBD" (*to be done*).

```
a = 3
def f(a):
        return 3*a
b = f(a)
c = f(b)
```

FIGURE 7.17 – Une même fonction, des paramètres différents

```
a = 1
def f(x):
        a = 56 + x
        return a
a = f(1)
b = f(a)
```

FIGURE 7.18 – Un paramètre numérique

```
a = 3
def f(x,y):
        return x + y
b = f(a,3)
c = f(b,0)
```

FIGURE 7.19 – Deux paramètres

7.5.2 L'administration

Comme pour les autres CI, l'administration s'est déroulée selon une approche pré-test post-test (cfr Figure 7.9). Le CI traitant du concept de fonction a été complété trois fois par an, durant trois années, par respectivement 111, 94 et 57 étudiants, soit un total de 262 étudiants.

TABLE 7.9 – Les modèles mentaux liés au concept de fonction (1/2)

M1'	**No called *function*** - Code of only one *function* (from those defined) is executed and the obtained value is returned and used even if it is not stored. Return instruction is read as an assignment. Codes before and after the definitions of *functions* are executed.
M2'	**No called *function*** - Code of all defined *functions* (in the order of the given definitions) is executed and the obtained value is returned and used even if it is not stored. Return instruction is read as an assignment. Codes before and after the definitions of *functions* are executed.
M3'	**No called *function*** - Code of all defined *functions* is executed in parallel (not sequentially). It is the variable value declared just before the *functions* that is considered in each of them. The obtained values are added and the result is returned and used even if it is not stored. Codes before and after the definitions of *functions* are executed.
M4'	**No called *function*** - Code of all defined *functions* is executed sequentially. Each *function* uses the variable value calculated in the previous *function*. The obtained values are added and the result is returned and used even if it is not stored. Codes before and after the definitions of *functions* are executed.
M5'	**No called *function*** - *Function* code is partially executed : Only the return instructions are executed. The obtained value is returned even if it is not stored. Codes before and after the definitions of *functions* are executed.
M6'	**Called *function*, but returned value not stored** - Codes before, called *functions* codes and code after the call are executed. The obtained value is returned and used even if it is not stored.

TABLE 7.10 – Les modèles mentaux liés au concept de fonction (2/2)

M1	*Function* code is perhaps executed but no value is returned. "What happens in the function stays in the *function*". Code before and after the call is executed.
M2	Code before the call, called *functions* codes and code after the call are correctly executed.
M3	Code before the call, called *functions* codes and code after the call are executed, but the value of the variable passed in parameter remains constant despite the *function* codes.
M4	*Function* code is partially executed : Only the return instruction is executed. Code before and after the call is executed.
M5	*Function* code is partially executed : Everything but the return instruction is executed and the obtained value is returned. Code before and after the call is executed.

TABLE 7.11 – Nouvelles représentations erronées liées au concept de fonction

new-MIS04	"What happens in the function stays in the *function*"
new-MIS05	A return instruction is equivalent to an assignment
new-MIS06	*Function* codes are executed in parallel, no matter if they are called or not.
new-MIS07	If a variable in the main code (external variable of the *function*) has the same name as the internal variable of the *function* and its parameter, no matter what value is passed as a parameter, the *function* considers the last value of this external variable, in the order of writing (and not of execution) of the main code

7.5.3 Le *concept inventory*

Questions	Réponses	Modèles mentaux	Rep. erronées

1.
a = 3
def f(a) :
  return a + 1
def g(a) :
  return a * 10
b = a
	b = 3	M1 or M2	new-MIS04 or /

	b = 4	M1'	MIS38, new-MIS05
	b = 30	M1'	MIS38, new-MIS05
	b = 40	M2'	MIS37, MIS38, new-MIS05
	b = 34	M3'	new-MIS06, new-MIS07
	b = 44	M4'	MIS37, new-MIS07

2.
a = 3
def f(a) :
  return a + 1
def g(a) :
  return a * 10
def h(a) :
  a = a * 2
  return f(a) + g(a)
b = a
	b = 3	M1 or M2	new-MIS04 or /

	b = 4	M1'	MIS38, new-MIS05
	b = 30	M1'	MIS38, new-MIS05
	b = 67	M1'	MIS38, new-MIS05
	b = 40	M1'	MIS38, new-MIS05
	b = 34	M1' + M5'	MIS38, new-MIS05
	b = 881	M4'	new-MIS06, new-MIS07
	b = 44	M4' + M5'	MIS37, MIS38, new-MIS07

3.
a = 3
def f(a) :
  return a + 1
def g(a) :
  return a * 10
def h(a) :
  a = a * 2
  return f(a) + g(a)
g(a)
b = a
	b = 3	M1 or M2	new-MIS04 or /

	b = 4	M1'	MIS38, new-MIS05
	b = 40	M1'	MIS38, new-MIS05
	b = 67	M1'	MIS38, new-MIS05
	b = 40	M1'	MIS38, new-MIS05
	b = 34	M1' + M5'	MIS38, new-MIS05
	b = 881	M3'	new-MIS06, new-MIS07
	b = 44	M4' + M5'	MIS37, MIS38, new-MIS07
	b = 30	M6'	MIS38

4.
a = 3
def f(a) :
  return a + 1
def g(a) :
  return a * 10
def h(a) :
  a = a * 2
  return f(a) + g(a)
a = h(a)
b = a
	b = 3	M1	new-MIS04

	b = 67	M2	/
	b = 40	M3 or M1'	TBD or MIS38, new-MIS05
	b = 34	M4	TBD
	b = 6	M5	MIS42
	b = 881	M3'	new-MIS06, new-MIS07
	b = 30	M6'	MIS38
	b = 44	M4' + M5'	MIS37, MIS38, new-MIS07

Questions	Réponses	Modèles mentaux	Rep. erronées
5. a = 3 def f(a) : return 3 * a b = f(a) c = f(b)	a = 3, b = 3, c = 3	TBD	TBD
	a = 3, b = 9, c = 9	TBD	TBD
	a = 3, b = 9, c = 27	TBD	TBD
	a = 3, b = 9, c = 21	TBD	TBD
	a = 9, b = 9, c = 9	TBD	TBD
	a = 3, b = 9, c = 81	TBD	TBD
	a = 3, b = 9, c = 0	TBD	TBD
	a = 3, b = 9, c = 18	TBD	TBD

6.
a = 1
def f(x) :
  a = 56 + x
  return a
a = f(1)
b = f(a)
	a = 1, b = 1	TBD	TBD

	a = 57, b = 113	TBD	TBD
	a = 57, b = 57	TBD	TBD
	a = 57, b = 58	TBD	TBD
	a = 55, b = 55	TBD	TBD
	a = 57, b = 0	TBD	TBD
	a = 113, b = 113	TBD	TBD
	a = 57, b = 1	TBD	TBD
	a = 1 , b = 0	TBD	TBD
	a = 57 , b = 28	TBD	TBD
7. a = 3 def f(x,y) : return x + y b = f(a,3) c = f(b,0)	a = 3, b = 3, c = 0	TBD	TBD
	a = 3, b = 6, c = 6	TBD	TBD
	a = 3, b = 6, c = 3	TBD	TBD
	a = 3, b = 4, c = 0	TBD	TBD
	a = 3, b = 9, c = 0	TBD	TBD

7.5.4 Résultats et discussion

Avant tout chose, il convient de rappeler que le concept de fonction n'est pas abordé dans le cours préparatoire.

Au terme de quatre années de développement et au vu des réponses obtenues, il apparait que le nombre d'itérations et que les questions composant le CI sont largement insuffisants pour déterminer la totalité des modèles mentaux utilisés par les novices en ce qui concerne le concept de fonction. Une tentative a cependant été faite, mais la liste des modèles mentaux obtenus est incomplète et les descriptions de ceux-ci sont à améliorer (cfr Tables 7.9 et 7.10). Le travail d'identification doit également être poursuivi en ce qui concerne les représentations erronées liées au concept de fonction (cfr Tables 7.8 et 7.11), idéalement à travers des interviews avec des étudiants. Une fois les modèles mentaux identifiés (et avec eux, possiblement des nouvelles représentations erronées), il sera plus facile de créer des questions supplémentaires permettant de tester ces modèles.

En outre, le choix avait été délibérément fait de rester dans la syntaxe du langage de programmation Python, plus haut niveau et jugé plus abordable en ce qui concerne les fonctions, que le langage de programmation Java. Toutefois, beaucoup d'étudiants ont été malgré tout en difficulté face à la syntaxe d'un appel de fonction et à la notion de retour d'une fonction. En ce sens, l'ajout de questions aidant à mieux visualiser ces deux notions cruciales pour la compréhension du concept de fonction est plus que nécessaire.

7.6 Conclusion intermédiaire

Un processus de développement de CIs est au coeur de ce premier cas d'étude. Ce processus est illustré par l'élaboration de trois CIs dans le cadre d'un cours d'initiation à la programmation. Chaque CI porte sur un concept de base abordé dans le cadre de ce cours, à savoir la variable, l'instruction conditionnelle et la fonction.

Basé sur les représentatione erronées qu'ont les apprenants d'un concept spécifique, un CI doit permettre d'identifier les modèles mentaux que ceux-ci utilisent pour résoudre des problèmes mettant en jeu le concept considéré. Pour s'assurer de l'efficacité de ces CIs, des données ont été collectées durant quatre ou cinq ans, auprès de différents groupes d'étudiants (un groupe par an).

Au cours de la première année, une approche ethnographique a été appliquée pour définir les sujets problématiques et identifier les représentations erronées des étudiants (groupe 1) inscrits au cours. De la deuxième à la quatrième ou ciquième année, chaque année et pendant un semestre, les CIs ont été administrés aux étudiants (du groupe 2 au groupe 4 ou 5) selon un calendrier spécifique. Les CIs développés ont donc été itérés trois ou quatre fois.

Bien que les résultats n'aient pas été traités dans les détails pour l'ensemble des CIs, les résultats préliminaires montrent qu'ils sont efficaces pour identifier les représentations erronées des apprenants et les modèles mentaux qu'ils utilisent.

LES CONTRIBUTIONS À SOULIGNER
Dans le cadre d'un cours d'introduction à la programmation en première année de bachelier (cas d'étude 1), **deux outils d'évaluation ont été créés et un troisième, enrichi** :

- un *concept inventory* sur le concept de variable, version enrichie du *concept inventory* de Dehnadi
- un *concept inventory* sur le concept d'instruction conditionnelle
- un *concept inventory* sur le concept de fonction

À partir des données collectées lors de leurs administrations, des **représentations erronées et modèles mentaux** liés aux concepts-clé étudiés **ont pu être identifiés** :

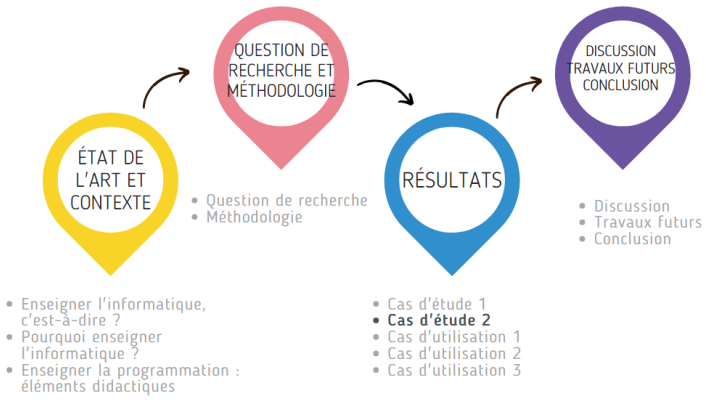
- trois représentations erronées et onze modèles mentaux, grâce au *concept inventory* sur le concept d'instruction conditionnelle
- quatre représentations erronées et onze modèles mentaux, grâce au *concept inventory* sur le concept de fonction

7.7 Menaces à la validité

Les menaces à la validité de ce chapitre sont discutées dans le cadre du cas d'utilisation 1 (cfr Section 9.4, Chapitre 9).

CHAPITRE 8

CRÉER UN CONCEPT INVENTORY À ÉCHELLE DE LIKERT (CAS D'ÉTUDE 2)



ARTICLE PUBLIÉ INSPIRANT CE CHAPITRE

- Henry, J., Dumas, B., Heymans, P. & Leclercq, T. (2020, October). Object-Oriented Programming : Diagnosis Understanding by Identifying and Describing Novice Perceptions. *Proceedings of 2020 Frontiers in Education Conference (FIE)*. IEEE.

Le deuxième cas d'étude s'est déroulé dans le cadre du cours de "Conception et programmation orientées objet" (INFOB234) organisé en deuxième année de ba-

chelier en informatique, à l'Université de Namur. Ce cours est suivi par les étudiants issus des filières INFO et INGMI.

Ce cas d'étude propose une approche alternative pour identifier et quantifier les représentations erronées auprès d'un grand nombre d'étudiants, à savoir remplacer les questionnaires ouverts et fermés dans le processus de développement d'un CI (étapes 3 et 4 - cfr Section 6.3) par un questionnaire à échelle de Likert.

Les interviews et les questions ouvertes apportent des précisions et permettent d'expliquer le pourquoi des représentations erronées. Cependant, ils ne se prêtent pas facilement à l'analyse. Les échelles de Likert permettent d'opérationnaliser facilement les perceptions. Elles peuvent donc être considérées comme de bons outils pour identifier les principales représentations erronées. De plus, l'utilisation des échelles de Likert facilite la collecte et l'analyse des données, mais aussi la complétion des CI par les étudiants. Dès lors, l'hypothèse est posée que cette alternative permettrait de collecter des mesures statistiquement exploitables pour, notamment, mettre en évidence les différences qui peuvent exister entre les étudiants INFO et INGMI (cfr Chapitre 10).

8.1 Le contexte : le cours de programmation orientée-objet (INFOB234)

Le cours INFOB234 vise à introduire les concepts fondamentaux de la programmation orientée-objet (POO), illustrés et mis en œuvre à l'aide du langage de programmation Java. Il est imposé dans les programmes des étudiants INFO et INGMI et se compose de 30 heures de cours théorique et de 30 heures de cours pratique (sur machine).

En aucun cas le paradigme orienté-objet ne constitue une particularité de ce cas d'étude. Il s'agissait tout simplement de suivre la progression logique d'apprentissage des novices en programmation. En effet, pour intégrer le cours INFOB234, les étudiants doivent avoir suivi avec succès un cours d'introduction à la programmation équivalent au cours INFOB131 (cfr Chapitre 7). Par conséquent, l'hypothèse de travail est posée que chaque étudiant inscrit au cours INFOB234 possède une connaissance des concepts-clé en programmation et est capable de mettre en œuvre ces concepts dans un programme informatique.

8.2 Le processus de développement du *concept inventory*

Parce que le concept de variable est l'un des premiers à être abordé dans le cadre et est considéré comme difficile pour les étudiants, il est celui sur lequel repose les résultats préliminaires issus de ce cas d'étude, mené durant trois ans auprès des étudiants du cours INFOB234 (étape 1 du processus de développement d'un CI - cfr Section 6.3). En outre, ce concept est, à ce stade, considéré connu par ces étudiants car logiquement abordé dans le cadre d'un cours d'introduction à la programmation.

Durant la première année, une approche phénoménographique (**collecte de données qualitatives**) a été utilisée pour découvrir comment les étudiants percevaient

le concept de variable [Richardson, 1999]. Des interviews semi-directives ont été conduites [Ketele and Roegiers, 2009]. Cette première phase de l'étude correspond à l'étape 2 de la procédure de développement d'un CI. Au cours des deuxième et troisième années, un CI "à échelle de Likert" a été créé sur base des données collectées durant la première année et a été administré aux étudiants selon une approche pré-test post-test [Shuttleworth, 2009] afin de mesurer, entre autres, l'évolution de leurs représentations (cfr Chapitre 11) (**collecte de données quantitatives**).

8.2.1 Collecte de données qualitatives

Pendant un semestre, de février 2019 à mai 2019, des interviews semi-directives ont été menées avec un groupe d'étudiants inscrits au cours INFOB234. Chaque étudiant a été interviewé trois fois : avant le début du cours (pré-interview), au milieu du semestre (mid-interview) et à la fin du cours, avant l'examen final (post-interview). Pour déterminer l'échantillon des interviews, il a été décidé de suivre les lignes directrices décrites dans la littérature de recherche qualitative. Il est recommandé de réaliser des interviews jusqu'à ce que la saturation soit atteinte [Guest et al., 2006]. Dans un projet nécessitant la transcription de 60 interviews, la saturation a été constatée après 12 transcriptions [Guest et al., 2006]. Enfin, Nielsen et Molich montrent que très peu de nouveaux résultats émergent au-delà de dix utilisateurs [Nielsen and Molich, 1990]. Sur base de ces constats, 13 interviews ont été menées avec des étudiants volontaires : 8 étudiants INFO et 5 étudiants INGMI.

Le guide d'interview était unique, inspiré à la fois des représentations erronées identifiées chez les novices par Sorva [Sorva et al., 2012] et du contenu du cours INFOB234. Il visait plusieurs concepts : la variable, l'objet, les primitives et les types. En ce qui concerne la variable, les étudiants étaient invités à définir le concept (ce qui entraînait souvent des questions de précision quant à son stockage et sa manipulation), à énumérer ses types possibles, à la représenter schématiquement, et enfin à commenter et à corriger une représentation donnée de la pile et du tas. Du papier et un crayon étaient mis à disposition des étudiants s'ils souhaitaient étayer leur discours par des annotations ou un schéma.

Les interviews ont été entièrement enregistrées et transcrites. Les verbatim obtenus ont été codés, par deux chercheurs, en deux étapes : codage ouvert et codage axial [Lejeune, 2019]. Tout désaccord a fait l'objet d'une discussion entre les chercheurs, jusqu'à ce que soit trouvé un consensus. Au final, six codes génériques ont été définis : VAR_ModèleMental, VAR_Théorie, VAR_TypeDeDonnée, VAR_TypeDeVariable et VAR_Utilisation.

Suite à ce codage et à partir de l'analyse des verbatims obtenus pour chaque code, des représentations erronées ont été définies, faisant écho aux acquis d'apprentissage attendus pour le cours INFOB134.

8.2.2 Collecte de données quantitatives

Ces représentations erronées ont inspiré des énoncés délibérément rédigés pour exprimer l'évolution des représentations mesurée chez les étudiants au cours des

Pour faire des manipulation par après.	VAR_Utilisation		
Ben en programmation des boucles c'est plus facile en utilisant une variable ou tout ca.	VAR_Utilisation		
La donnée on l'utiliserait pour faire plein de choses comme je l'ai déjà expliqué.	VAR_Utilisation		
Donc quand on mettra dans notre programme le X vaudra la valeur 7 et on pourra s'en servir.	VAR_Utilisation		
pour être utilisé dans le langage de programmation ou dans les calculs.	VAR_Utilisation		
qui est utilisé dans le programme et qui peut changer de valeur au fil de l'exécution du programme	VAR_Utilisation		
C'est un espace de stockage d'une donnée afin d'en faire un calcul comment expliquer....Je ne saurai pas	VAR_Utilisation	VAR_ModèleMental	
Afin de faire des additions des multiplications des trucs comme ça.	VAR_Utilisation		
Une variable, c'est quoi et bien tu prends ce que tu veux mettre avant et tu appelles ta variable variable	VAR_Utilisation	VAR_TypeDeVariable	VAR_ModèleMental
String, float, integer, caractère... réfléchi... booléen...réfléchi... surement des autres mais je ne sais plus trop.	VAR_TypeDeVariable		
Entier, Integer, Float, flottant, double, short et long. Il y a booléen. Char euh les... String et tout. Les chaines	VAR_TypeDeVariable		
il existe des variables de type primitif il y a plusieur type. type primitif et type objet.	VAR_TypeDeVariable		
type primitif (liste les types) int boolean short long double etc... byte aussi	VAR_TypeDeVariable		
Et en objet il y a en a beaucoup et on peut créer des types d'objets, enfin le programmeur peut créer des	VAR_TypeDeVariable		
Types primitives; int bytes boolean,short, long double j'ai dit long? Il en manque un.	VAR_TypeDeVariable		
Il y en a bien 8?	VAR_TypeDeVariable		
Ah... c'est une question piège?	VAR_TypeDeVariable		
Booléen entier réel string caractère, voilà. J'en ai peut-être oublié...	VAR_TypeDeVariable		
Réfléchi, les concepts de tableaux, constantes ...	VAR_TypeDeVariable		
Des listes...	VAR_TypeDeVariable		
Type objet et type primitif.	VAR_TypeDeVariable		
int, char, bytes, float, boolean euh.... short long...il y en a encore un... il y en a huit au total... je cale toujours	VAR_TypeDeVariable		
On a String, Integer, Caractère, Entier, Float, Booléen, je pense avoir tout dit.	VAR_TypeDeVariable	VAR_Théorie	
Je ne sais plus, je dirai euh..... réfléchi... les types ce qu'on peut mettre avant le égale après le égale, on	VAR_TypeDeVariable	VAR_Utilisation	
Alors après est-ce que l'ordinateur, je suppose que du coup l'ordinateur ne l'interprète pas de la même	VAR_TypeDeVariable		
... je ne vois pas...	VAR_TypeDeVariable	VAR_Théorie	
Par exemple, une chaîne de caractère, un chiffre un entier un réel. Je ne sais pas comment expliquer, on va	VAR_TypeDeDonnée	DON_Exemple	
Euh oui, je pense...je ne sais plus du tout. Et euh, à l'intérieur on met une donnée de variable enfin de type	VAR_TypeDeDonnée	VAR_Théorie	VAR_Utilisation
Une variable c'est... je ne sais pas vraiment la théorie ce qu'on dit.	VAR_Théorie		
On stocke une donnée.	VAR_Théorie		
Une données que l'on met à l'intérieur.	VAR_Théorie		
Euh... ensuite cette variable on lui donne un nom. On va l'appeler X et on y stocke un 7.	VAR_Théorie		

FIGURE 8.1 – Extrait du codage axial

pré-, mid- et post-interviews. D'autres énoncés ont été ajoutées, inspirés par la définition donnée aux étudiants dans le cours INFOB131 et par les représentations erronées listées dans la littérature [Kaczmarczyk et al., 2010, Sorva, 2008, Sorva, 2007,Eckerdal and Thuné, 2005]. Dès lors, le CI développé pour le concept de variable (mais non encore validé) propose des énoncés différents en fonction du moment où il est administré aux étudiants : le pré-CI comporte 12 énoncés, le mid-CI en comporte 14 et le post-CI en comporte 15.

Les étudiants sont amenés à se positionner par rapport à chacun de ces énoncés grâce à une échelle de Likert avec quatre options de réponse : "Entièrement vrai", "Vrai, mais à enrichir", "Partiellement vrai, contient des erreurs" et "Entièrement faux". Ce positionnement vise à donner une vue globale de la compréhension des étudiants. Une cinquième option, "Je ne sais pas", est également disponible (cfr Figure 8.2).

Au cours des deuxième et troisième années de l'étude, le CI a été administré à 171 étudiants (83 INFO et 88 INGMI), trois fois par an, suivant la même logique que les interviews : avant le début du cours (pré-CI), au milieu du semestre (mid-CI), et à la fin du cours, avant l'examen final (post-CI) (cfr Chapitre 10).

8.3 Un *concept inventory* "à échelle de Likert" sur le concept de variable

Les données présentées dans cette section sont issues de l'encodage de 39 interviews (3 x 13) et concernent le concept de variable dans la POO. Six représentations erronées ont été identifiées à partir des transcriptions et exprimées dans les termes utilisés par les étudiants au cours des interviews (cfr Table 8.1). Elles constituent la base des énoncés du CI.

FIGURE 8.2 – Extrait du CI “à échelle de Likert” sur le concept de variable

TABLE 8.1 – Représentations erronées liées au concept de variable - *Chacune est associée à un label qui la “résume”*

mis-a : Memory Value	*A variable allows storing a data value in the memory of a computer.*
mis-b : Math Point of View (PoV)	*A variable in programming is similar/assimilated to a mathematical variable.*
mis-c : Program Segment	*A variable is a portion of code in a program.*
mis-d : Manipulable Memory Value	*A variable allows storing a data value in the memory of a computer (mis-a) in order to use it (read, write).*
mis-e : Memory Place	*A variable is in the stack.*
mis-f : Memory Organisation	*A variable is in the stack (mis-e) and may have a reference to the heap.*

8.3.1 Les représentations erronées issues des pré-interviews

Avant le cours, les étudiants semblent se raccrocher à leurs connaissances antérieures : leur perception de la structure et du fonctionnement d’un ordinateur, ou encore des notions de mathématiques ou de programmation (cfr Table 8.1). En ce qui concerne la représentation erronée **mis-a** (**Memory Value**), les étudiants discutent de l’emplacement/du stockage d’une variable dans un espace mémoire : *“Nous stockons une donnée dans la ram [...] la mémoire vive du PC”*, *“C’est une réfé-*

rence à quelque chose dans la mémoire [...]". Quant à la représentation erronée **mis-b** (**Math PoV**), une comparaison est faite avec le concept de variable en mathématiques, *"pour les équations [...]"*, ou un apprentissage antérieur en programmation, *"la même variable que celle vue l'année dernière [...] dans le cours d'introduction à la programmation [...] c'est une valeur de données, cela peut être une chaîne de caractères"*. Au sujet de la représentation erronée **mis-c** (**Program Segment**), les étudiants restent sur l'idée globale qu'ils se font d'un cours de programmation : *"une variable, c'est du code"*.

8.3.2 Les représentations erronées issues des mid-interviews

Pendant l'apprentissage, la représentation erronée **mis-c** (**Program Segment**) est toujours présente : *"une variable est quelque chose qui prend de la place dans le programme, dans la mémoire"*. La représentation erronée **mis-a** (**Memory Value**) évolue et la **mis-b** (**Math PoV**) n'est plus évoquée par les étudiants, au profit des représentations **mis-d** (**Manipulable Memory Value**) et **mis-e** (**Memory Place**) (cfr Table 8.1). La catégorie **mis-d** (**Manipulable Memory Value**) est une version évoluée de la catégorie **mis-a** (**Memory Value**) où il est spécifié que le stockage est effectué dans un certain but : *"C'est un espace de stockage d'une valeur de données pour faire un calcul"*. La catégorie **mis-e** (**Memory Place**) est la réutilisation directe (mais incomplète) du modèle conceptuel transmis par l'enseignant : *"C'est donc soit une instance d'un objet, soit un type primitif et cela correspond à une zone mémoire [...] Les types primitifs sont dans la pile [...]"*.

8.3.3 Les représentations erronées issues des post-interviews

À la fin de la période donnée d'apprentissage, la catégorie **mis-c** (**Program Segment**) disparaît tandis que la catégorie **mis-e** (**Memory Place**) évolue. La catégorie **mis-d** (**Manipulable Memory Value**) est toujours présente : *"Une variable est le stockage de données qui peuvent être utilisées pour faire des calculs ou simplement stockées pour être utilisées par la suite"*. Une nouvelle catégorie apparaît (cfr Table 8.1) : la catégorie **mis-f** (**Memory Organisation**), évolution de la catégorie **mis-e** (**Memory Place**) où l'étudiant s'approprie le modèle conceptuel de l'enseignant qui devient ainsi son modèle mental : *"Une variable contient une valeur et est identifiée par son nom. Elle peut être soit une variable de type primitif, soit une variable de type données. Si c'est une variable de type données, elle a une référence [...] Si c'est une variable de type primitif, elle sera dans la pile, sinon dans le tas"*.

8.3.4 Les énoncés du *concept inventory*

Comme stipulé plus haut, les énoncés proposés sont différents en fonction du moment où le CI est administré aux étudiants : le pré-CI comporte 12 énoncés, le mid-CI en comporte 14 et le post-CI en comporte 15. Ces énoncés sont issus des représentations erronées identifiées lors des interviews, de la littérature ou des notions logiquement abordées lors d'un cours d'introduction à la programmation (cfr Table 8.2).

TABLE 8.2 – Énoncés (numérotés) composant le CI "à échelle de Likert" sur le concept de variable - *Chacun est associé à un label qui le "résume"*

	Énoncé	Source
Storage	1. A variable allows storing a data value in the memory of a computer.	mis-a
Math Assimilation	2. A variable in programming can be assimilated to a mathematical variable.	mis-b
Reference	3. A variable is a pointer.	Literature
Code Segment	4. A variable is a portion of code in a computer program.	mis-c
Unicity	5. A variable contains a unique value.	Introductory course
Modifiability	6. A variable can be modified.	Introductory course
Code Segment Reference	7. A variable is a portion of code that can be referenced in a program.	Based on mis-c, mis-d
Memory Address Reference	8. A variable is a memory address that is named and referenced in a program.	Based on mis-d
Function Component	9. A variable is the component of a function, allowing to vary the function.	mis-b
Storage Manipulation	10. A variable is used to manipulate a storage area.	Based on mis-d
Syntaxic Name	11. A variable associates a name with a value.	Introductory course
History	12. A variable contains a history of values that can be used by a program.	Literature
Updatable Storage	13. A variable allows storing a data value in the memory of a computer in order to use it (read, write).	mis-d
Stack Model	14. A variable is in the stack.	mis-e
Stack & Heap Model	15. A variable is in the stack and may have a reference to the heap.	mis-f

8.4 Conclusion intermédiaire

Ce cas d'étude vise à proposer une alternative dans le processus de développement d'un CI, à savoir remplacer les questionnaires ouverts et fermés (étapes 3 et 4 - cfr Section 6.3) par un questionnaire à échelle de Likert en vue de collecter des mesures statistiquement exploitables.

Une étude qualitative a permis d'identifier six représentations erronées que possèdent les étudiants sur le concept de variable (étape 2 du processus de développement d'un CI). Il est apparu que ces représentations erronées évoluaient au cours de l'apprentissage (s'étalant sur un quadrimestre). Un CI "à échelle de

Likert" a été créé, à partir des six représentations erronées et de représentations issues de la littérature, entre autres. Pour respecter l'évolution des représentations mesurées auprès des étudiants, ce CI est évolutif et se compose, selon le moment d'administration, de 12 à 15 énoncés. Étant donné sa forme, ce CI ne vise pas, à ce stade, à déterminer les modèles mentaux utilisés pour résoudre des problèmes en programmation, mais bien à identifier et quantifier les représentations erronées au sein d'un groupe d'étudiants.

Le passage à l'étape suivante du processus de développement, à savoir la validation des énoncés et l'administration à grande échelle (étape 5 - cfr Section 6.3) en vue d'effectuer des tests statistiques, est l'une des principales perspectives de cette étude. En ce sens, le CI a été administré, sur deux ans, à 171 étudiants (Cas d'utilisation 2 - cfr Chapitre 10).

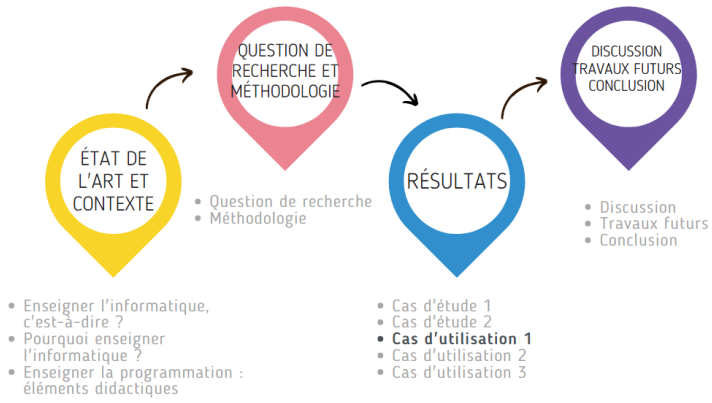
LES CONTRIBUTIONS À SOULIGNER
Dans le cadre d'un cours de programmation orientée-objet en deuxième année de bachelier (cas d'étude 2), un outil a été créé pour **quantifier les représentations erronées** des étudiants concernant la variable : **un *concept inventory* "à échelle de Likert"**.
Ce *concept inventory* a été élaboré selon un **processus original** mis au point dans le cadre de cette recherche doctorale. Ainsi, les étapes de création de questions ouvertes et fermées du processus "classique" de développement d'un *concept inventory* ont été remplacées par la création de questions à échelle de Likert.

8.5 Menaces à la validité

Les menaces à la validité de ce chapitre sont discutées dans le cadre du cas d'utilisation 2 (cfr Section 10.5, Chapitre 10).

CHAPITRE 9

UN CONCEPT INVENTORY POUR MESURER UN GAIN DE COMPRÉHENSION CHEZ LES APPRENANTS (CAS D'UTILISATION 1)



ARTICLES PUBLIÉS INSPIRANT CE CHAPITRE

- Henry, J., & Dumas, B. (2020). Developing an assessment to profile students based on their understanding of the variable programming concept. *Proceedings of the 2020 Conference on Innovation and Technology in Computer Science Education (ITICSE)*, pages 33-39. ACM.

- Henry, J., & Dumas, B. (2019, October). Towards the identification of profiles based on the understanding of programming concepts : the case of the variable. *Proceedings 2019 Frontiers in Education Conference (FIE)*, IEEE, 1-8.

La mesure de la compréhension via l'identification des modèles mentaux au moyen d'un CI a généralement lieu à un moment précis du processus d'apprentissage. Ce cas d'utilisation vise à valider l'utilisation d'un CI dans une approche permettant de mesurer l'évolution de cette compréhension sur une période donnée.

Une étude centrée sur le concept de variable a été menée sur deux ans, avec deux groupes d'étudiants suivant le même cours d'introduction à la programmation (INFOB131 - cfr Section 7.1). Le postulat est qu'il existe des profils de compréhension chez les étudiants et que ces profils sont notamment liés à leur cohérence en termes de modèles mentaux utilisés. Au total, le gain de compréhension de 219 étudiants a été mesuré selon une approche spécifique visant à identifier ces profils de compréhension.

La fiabilité de l'approche est testée en déterminant (à l'aide d'un test de signification statistique) si la proportion d'étudiants correspondant aux profils identifiés reste stable au fil des années.

H0 (hypothèse nulle) : Pour chaque profil de compréhension identifié, la fréquence des étudiants est équivalente d'une année académique à l'autre.

Si cette hypothèse est vérifiée, la mesure de la cohérence d'un étudiant dans un contexte d'auto-évaluation pourrait l'aider à prendre conscience de ses faiblesses potentielles. Il ne s'agit pas de prédire la réussite des étudiants, mais de leur fournir un outil pour réguler leur apprentissage dans un contexte universitaire où l'autonomie est fortement encouragée. En outre, le CI pourrait également aider les enseignants à mieux connaître leur public et à adapter leur enseignement en conséquence.

9.1 Méthodologie

L'approche mise en place est inspirée de la conception pré-test post-test [Shuttleworth, 2009]. Elle vise à évaluer l'efficacité d'un CI, ici celui liée au concept de variable en programmation (cfr Section 7.3), dans l'identification de profils de compréhension chez des programmeurs novices, ici les étudiants suivant le cours d'introduction à la programmation INFOB131 (cfr Section 7.1).

Les profils sont définis par le passage de l'utilisation de modèles mentaux incorrects à l'utilisation du modèle mental correct (unique). Les mesures ont été réalisées quatre fois, à des moments précis dans l'apprentissage du concept de variable déterminés par de l'organisation du cours INFOB131 et du cours préparatoire (cfr Section 7.1) : CI(0) avant le cours préparatoire, CI(1) avant les quatre heures de cours théorique, CI(2) avant les trois heures de pratique et enfin, CI(3) à la fin de la séquence d'apprentissage dédiée à la variable (cfr Figure 9.1). La période entre CI(0)

et CI(1) est d'un mois, de quatre jours maximum entre CI(1) et CI(2), et de trois jours maximum entre CI(2) et CI(3).

Certaines réponses caractérisent plusieurs modèles mentaux. Si un choix est porté quant à ces modèles mentaux, ce dernier est fait en s'appuyant sur les explications données par l'étudiant lui-même, quand elles existent. Si aucune explication n'est fournie mais qu'il est possible d'identifier le modèle mental utilisé en s'appuyant sur les réponses aux autres questions et sur la cohérence de l'étudiant sur l'ensemble du test, un choix est posé par le chercheur. Sinon, tous les modèles mentaux possibles sont pris en compte.

Cette étude s'est étendue sur deux ans (2017-2018, année 1 et 2018-2019, année 2), auprès de deux groupes d'étudiants. Le CI a été présenté à ces étudiants comme une évaluation formative [Fisher and Frey, 2014].

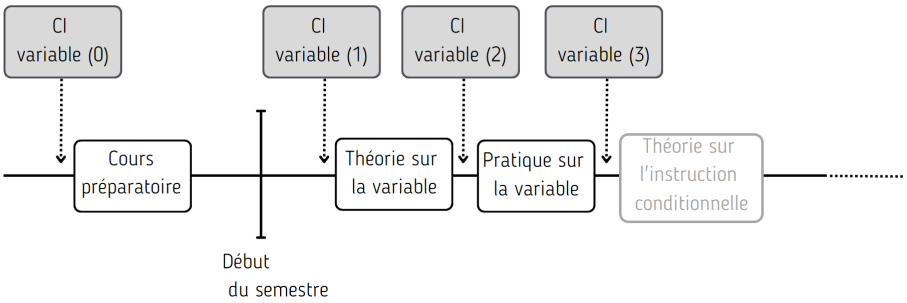


FIGURE 9.1 – Organisation de l'administration des CIs sur la variable

9.1.1 Participants

L'échantillon total est composé de 219 étudiants (cfr Table 9.1). Il présente une diversité intéressante dans le contexte de l'étude du comportement des novices en programmation : 131 (59,8%) étudiants sont inscrits en filière INFO et 88 sont inscrits en filière INGMI. Sur ces 219 étudiants, 82 (37,4%) ont participé au cours préparatoire en programmation (cfr Section 7.1) : 52 étudiants INFO et 30 étudiants INGMI.

Les étudiants qui échouent au cours de la première année de l'étude et qui doublent n'ont été pris en compte qu'une seule fois (lors de leur première tentative).

9.1.2 Collecte de données

Cette étude se concentre sur le concept de variable en programmation et sur les modèles mentaux associés. La version originale à 12 questions de Dehnadi [Dehnadi, 2009] a été utilisée en année 1 et la version enrichie à 15 questions, en année 2 (cfr Section 7.3). En raison de la durée limitée du cours préparatoire, le CI(0) consiste en une version comprenant 9 questions.

TABLE 9.1 – Composition de l'échantillon d'étudiants, caractérisé par leur filière et exprimé en nombre d'étudiants - Nombre total/Nombre d'étudiantes - *Les proportions entre parenthèses sont calculées indépendamment pour chaque année d'étude*

	INFO	INGMI	Total
Cours INFOB131 (année 1)	65/13 (60.7%)	42/9 (38.8%)	107
Cours préparatoire (année 1)	18/5 (16.8%)	10/1 (9.3%)	28
Cours INFOB131 (année 2)	66/7 (58.9%)	46/13 (41.1%)	112
Cours préparatoire (année 2)	34/5 (32.1%)	20/1 (17.0%)	54

Avant leur utilisation et en guise de validation, les deux CIs ont été analysés selon les sept étapes définies par Tew [Tew and Guzdial, 2010] :

— *Define Conceptual Content*
— *Expert Review of Test Specification*
— *Build Test Bank of Questions*
— *Verify Language Independence*
— *Pilot Questions*
— *Establish Validity*
— *Establish Reliability*

Les étapes 3 et 5 ne sont pas prises en compte. Les deux CIs utilisés sont basés sur le CI de Dehnadi. et, à ce titre, ont déjà fait l'objet de publications scientifiques. Les étapes 1, 2 et 4 sont présentées ci-après. Les étapes 6 et 7 sont, quant à elles, parties intégrantes de l'objet de l'étude au centre de ce cas d'utilisation.

Étape 1 : *Conceptual Content*

La première étape consiste à établir l'objectif et la définition du CI : ce qui doit être mesuré et ce que les scores obtenus signifient. Dehnadi a conçu un CI visant à identifier les modèles mentaux utilisés par les étudiants pour résoudre des problèmes d'affectation de variable [Dehnadi, 2009]. Dehnadi a identifié onze modèles mentaux différents. Outre la réponse correcte, chaque réponse possible à une question est un distracteur inspiré d'un modèle mental spécifique. Chaque question propose donc les onze modèles mentaux dans ses choix (cfr Figure 7.5 dans la Section 7.3), ainsi que le dernier choix "autre".

Un seul modèle mental est correct et rapporte un point lorsqu'il est sélectionné par le répondant. Le score global est, au maximum, égal au nombre de questions, à savoir 12 pour le CI original de Dehnadi et 15 pour la version enrichie.

Le CI de Dehnadi est pondéré : 25% des questions sont consacrées à des problèmes d'affectation simple, 25% à des problèmes d'affectation double et 50% à des problèmes d'affectation triple, plus difficiles. Dans le CI enrichi, cette pondération est légèrement différente dû à l'ajout de trois questions (une dans chaque catégorie de problèmes) : 26,7%, 26,7% et 46,6% respectivement.

Étape 2 : *CI Specification*

Le CI de Dehnadi a fait l'objet de plusieurs articles scientifiques [Ahadi et al., 2014, Bornat, 2014, Dehnadi, 2009, Dehnadi et al., 2006, Caspersen et al., 2007], suggérant une certaine confiance dans sa spécification.

Les deux CI utilisés ne concernent que le concept de variable en programmation. Tous les modèles mentaux décrits par Dehnadi (cfr Table 7.1 en Section 7.3) sont représentés dans chaque question (cfr Figure 7.5 dans la Section 7.3), ainsi que les principales représentations erronées liées au concept de variable identifiées par Sorva [Sorva et al., 2012] (cfr Table 7.2 dans la Section 7.3).

Étape 4 : *Language Independence*

Au moment de l'administration des CIs(0) et CIs(1), une majorité d'étudiants n'ont jamais programmé. Ils sont généralement perturbés par les questions composant les CIs, n'en comprenant bien souvent pas le sens. En année 2, le CI(1) administré comprenait trois questions dupliquées et traduites en langage de programmation par blocs (cfr Figure 7.7 dans la Section 7.3). Celles-ci ont été ajoutées au CI original de Dehnadi pour repérer les difficultés syntaxiques rencontrées par certains étudiants à la lecture des questions écrites en Java et tenter de les contourner. La comparaison des résultats obtenus pour les paires de questions Java/blocs aurait pu mettre en évidence une éventuelle dépendance linguistique. En effet, les étudiants qui n'identifient pas le bon modèle mental dans les trois paires de questions Java/blocs pourraient rencontrer des difficultés avec les langages utilisés et donc, potentiellement, ne pas être en mesure de transférer leurs connaissances de Python à Java. Cette dépendance n'a cependant pas été observée durant l'étude. Certains étudiants ont, par ailleurs, rapporté avoir compris le CI à la lecture de ces questions en blocs. "*Cette question m'a permis de comprendre la logique. On modifie la valeur de a par la valeur de b*" (étudiant 1 concernant la question 3' du CI(1) sur le concept de variable). "*Je donne à la variable a la valeur de 10 et à la variable b, 20*" (étudiant 2 concernant la question 1 du CI(1) sur le concept de variable). "*La valeur des variables est fixée pour a à 10 et b à 20. (a = b) Je fixe la valeur de a avec la valeur de b. (b = a) Je fixe la valeur de b à la nouvelle valeur de a*" (étudiant 2 concernant la question 4 du CI(1) sur le concept de variable, démontrant l'influence de la question 3' sur sa compréhension).

En outre, les mesures collectées via les CIs(2) et CIs(3) montrent que la majorité des étudiants deviennent capables, après le cours théorique et/ou les séances de pratique, de transférer leurs connaissances du langage de programmation Python (abordé durant le cours INFOB131) aux langages de programmation utilisés dans les CIs (Java et blocs).

9.2 Résultats et discussion

9.2.1 Profils de compréhension des étudiants

Au cours de l'année 1, sept profils (profils A à G - cfr Table 9.2) ont été identifiés parmi les 107 participants à l'étude. Ces profils ont été déterminés à partir des scores obtenus par les étudiants aux différents CIs : un point par question pour le modèle mental correct et zéro point pour les autres. À partir de ces scores, une classification manuelle a été réalisée, tenant compte des gains de compréhension entre les CIs.

En année 2, ces profils ont été confirmés auprès de 112 étudiants. En outre, un nouveau profil a été identifié (profil H) et une distinction a été faite au niveau du profil A entre les étudiants ayant (profil A') ou non (profil A) des connaissances en programmation avant l'administration du CI(0).

Une description générique est proposée pour chaque profil, comprenant ses principales caractéristiques (cfr Table 9.2).

Les données collectées sont discutées en deux sous-sections. D'abord les résultats obtenus en première année, à savoir la distribution des étudiants entre les sept profils initiaux, sont commentés vis-à-vis de la filière des étudiants, mais également de leur genre et de leur cohérence en ce qui concerne l'utilisation des modèles mentaux [Dehnadi, 2009]. Ces résultats permettent de confirmer le postulat de départ, à savoir qu'il existe des profils de compréhension chez les étudiants et que ces profils sont notamment liés à leur cohérence en termes de modèles mentaux utilisés.

Ensuite, la fiabilité de l'approche permettant d'identifier ces profils de compréhension, et finalement permettant de mesurer le gain de compréhension des étudiants sur une période donnée, est testée en déterminant (à l'aide d'un test de signification statistique) si la proportion d'étudiants correspondant aux profils identifiés reste stable au fil des deux années (année 1 et 2).

TABLE 9.2 – Profils de compréhension

Profils	Description
The student who already masters (**Profil A'**)	This student obtains a (near) maximum score for all the completed CIs. He/she had programming courses at school or learned on his/her own. Average number of mental models used : 1.7
The student who intuitively masters (**Profil A**)	This student obtains a (near) maximum score for all the completed CIs. He/she never had programming courses at school, and he/she didn't learn on his/her own.
The student who masters thanks to the summer school class (**Profil B**)	This student attended the summer school and failed the CI(0). He/she obtains a (near) maximum score for the next CIs. Average number of mental models used : 3
The student who masters thanks to the theory class (**Profil C**)	This student did not participate in the summer school class. CI(1) is failed with a high error rate and the next CIs are passed with a high score. Average number of mental models used : 4.3
The student who the summer school class did not help BUT who masters thanks to the theory class (**Profil D**)	Unlike C-profiles students, this student attended the summer school class. CI(0) and CI(1) are failed with a high error rate. CI(2) is passed with a high score. Average number of mental models used : 6.9
The student who masters thanks to the practical work session (**Profil E**)	This student did not attend the summer school class. Only CI(3) is a success. The other CIs were failed with a high error rate. Average number of mental models used : 6.2
The student who needs time (**Profil F**)	The scores of this student on the various CIs are gradually increasing. The error rate decreases by more than 50% between CI(1) and CI(2). Average number of mental models used : 5.6
The student who never masters (**Profil G**)	This student has not passed any of the completed CIs. Average number of mental models used : 6.5
The student who doubts after the theory class (**Profil H**)	This student passes CI(0) and/or CI(1) brilliantly, has a high error rate on CI(2), but passes CI(3) with an almost maximum score.

9.2.2 Distribution des étudiants entre les profils A à G (année 1)

TABLE 9.3 – Distribution des étudiants (et étudiantes) entre les sept profils identifiés en année 1 de l'étude - **mm pour modèles mentaux*

	Participant(e)s à l'étude		Participant(e)s au cours prépa		Nbre moyen
Profils	INFO	INGMI	INFO	INGMI	de mm*
Profil A	29 (0)	6 (0)	9 (0)	2 (0)	1.7
Profil B	3 (1)	1 (1)	3 (1)	1 (1)	3.0
Profil C	13 (1)	12 (1)	0 (0)	0 (0)	4.3
Profil D	4 (2)	8 (5)	3 (1)	2 (0)	6.9
Profil E	6 (1)	2 (0)	0 (0)	0 (0)	6.2
Profil F	7 (7)	3 (1)	3 (3)	1 (0)	5.6
Profil G	2 (1)	10 (1)	0 (0)	4 (0)	6.5
Total	65 (13)	42 (9)	18 (5)	10 (1)	/

Près de la moitié des étudiants inscrits dans la filière INFO appartiennent au **profil A**. Ce n'est pas surprenant étant donné que la moitié de ces étudiants ont déclaré avoir déjà travaillé avec un langage de programmation, la plupart du temps de manière autonome, avant d'entrer à l'université. Il n'en va pas de même pour les étudiants INGMI qui n'ont souvent aucune idée de ce que peut être le cours d'introduction à la programmation. Aucune femme de l'échantillon ne correspond au **profil A**. Étant donné le petit nombre de femmes dans cette étude, il est cependant difficile de tirer une quelconque conclusion. Toutefois, il est frappant de constater qu'aucune femme, quelle que soit la filière dans laquelle elle s'est inscrite, n'a déclaré s'être essayée seule à un langage de programmation avant d'entrer à l'université.

Moins de 5% des étudiants semblent avoir profité du cours préparatoire (**profil B**). Parmi les données recueillies au cours de cette première année d'étude, il n'y a aucun moyen d'identifier si leurs progrès dans la compréhension du concept de variable sont dus au contenu du cours préparatoire ou à la curiosité suscitée par ce cours. Près de la moitié des étudiants du **profil B** sont de sexe féminin.

Les étudiants ayant le **profil C** semblent comprendre le concept de variable après le cours théorique, sans avoir besoin de le pratiquer. Ce profil est le plus représenté dans le groupe des étudiants INGMI, avec près d'un tiers des étudiants. En revanche, seulement 20% des étudiants INFO correspondent au **profil C**. Ce résultat est cohérent avec le profil généralement plus scolaire des étudiants INGMI. Seules deux femmes présentent ce profil.

Le **profil D** est lié au profil C, à la différence que les étudiants qui y correspondent ont suivi le cours préparatoire. Pour le même résultat, à savoir une compréhension acquise après le cours théorique, les étudiants ayant le **profil D** ont eu une occasion supplémentaire d'être confrontés au concept de variable. Près de 60% des étudiants ayant ce profil sont des femmes. C'est le profil le plus fréquent pour les femmes dans la filière INGMI (plus d'une femme sur deux). Il est difficile de mesurer l'influence exacte des cours préparatoires sur ces étudiants. Si cette influence est considérée inexistante (ou presque), ce profil pourrait être regroupé avec le profil C et représenter 26,1% des étudiants INFO (17) et 47,6% des étudiants INGMI (20). Avec neuf femmes

concernées (7+2), le cours théorique serait déterminant pour la compréhension de 40,9% des femmes.

Le **profil E** montre que 7,5% (moins d'un étudiant sur dix) a besoin de mettre en oeuvre, dans les séances de pratique, le concept de variable pour le comprendre. Ceci peut être interprété de plusieurs manières : soit la compréhension du concept de variable est essentiellement le résultat d'un apprentissage théorique, soit l'évaluation de cette compréhension par un CI ne mesure finalement qu'une compréhension liée à l'apprentissage théorique, soit les étudiants des autres profils s'exercent seuls avant les séances de pratique.

Les étudiants ayant un **profil F** montrent une évolution progressive de leur compréhension au fil des tests. Ces étudiants semblent avoir besoin de temps et de différents outils d'apprentissage (cours théorique et séances de pratique) pour asseoir leur compréhension. Les scores montrent une augmentation à chaque CI administré. Ce profil est plus fréquent chez les femmes INFO.

Le profil G est le deuxième profil le plus représenté parmi les étudiants INGMI. Près d'un étudiant INGMI sur quatre termine la séquence complète d'enseignement sur la variable sans avoir compris le concept. Parmi ces étudiants, seul un sur quatre avoue ne pas avoir participé aux séances de pratique, malgré les difficultés rencontrées avec le concept de variable.

En ce qui concerne le nombre moyen de modèles mentaux utilisés pour résoudre les problèmes composant les CIs, les profils qui montrent une compréhension plus rapidement acquise sont plus cohérents (selon la définition de Dehnadi [Dehnadi, 2009], c'est-à-dire qu'ils utilisent une moins grande diversité de modèles mentaux pour un même CI. Les étudiants de **profil B**, même s'ils ne réussissent pas le CI(0), n'utilisent en moyenne que trois modèles mentaux. Ces étudiants semblent donc très cohérents dans leur approche pour résoudre les problèmes proposés. Les étudiants de **profil C** utilisent un peu plus de quatre modèles mentaux. Les étudiants de **profil D** ont un nombre moyen de modèles mentaux utilisés plus élevé, en raison notamment des mauvais scores obtenus pour deux des quatre CIs passés. D'après ces résultats, on peut supposer que les étudiants de **profil C** auraient tiré parti du cours préparatoire s'ils y avaient participé, car leur cohérence est similaire à celle mesurée pour le **profil B**. Le **profil D** est plus comparable aux **profils E, F et G**. Ces derniers sont caractérisés par une plus grande incohérence dans les modèles mentaux utilisés. La moyenne de 5,6 obtenue par les étudiants de **profil F** pourrait être liée à l'évolution progressive de leur compréhension. En effet, contrairement aux autres profils, les étudiants de **profil F** semblent construire leur compréhension de manière graduelle, ce qui implique une diminution du nombre de modèles utilisés entre les différents CIs administrés. Les autres semblent avoir besoin de temps pour déconstruire leurs représentations erronées antérieures, sans que cette déconstruction ne soit nécessairement assurée (**profil G**). Comme il s'agit de moyennes sur plusieurs CIs, ces résultats sur les modèles mentaux ne sont pas informatifs. Il serait nécessaire d'étudier comment le nombre de modèles mentaux utilisés évolue dans le temps pour les étudiants de chaque profil.

9.2.3 Distribution des étudiants entre les profils A' à H et validation (années 1 et 2)

TABLE 9.4 – Distribution (en nombre d'étudiants) et fréquence (en %) des étudiants entre les profils de compréhension identifiés

	Participants à l'étude		Participants au cours préparatoire	
Profils	Année 1	Année 2	Année 1	Année 2
Profil A'	17 (15,9%)	23 (20,5%)	9	17
Profil A	18 (16,8%)	13 (11,6%)	2	2
Profil B	4 (3,7%)	13 (11,6%)	4	13
Profil C	25 (23,4%)	29 (25,9%)	0	0
Profil D	12 (11,2%)	15 (13,4%)	5	14
Profil E	8 (7,5%)	2 (1,8%)	0	1
Profil F	10 (9,3%)	6 (5,3%)	4	2
Profil G	12 (11,2%)	7 (6,2%)	4	4
Profil H	0	4	0	1
Total	107	112	28	54

L'enseignement de l'informatique, et de la programmation en particulier, ne fait pas officiellement partie du cadre des programmes scolaires belges francophones (cfr Section 2.5). Cependant, certaines écoles semblent organiser cet enseignement. Ainsi, en moyenne 18% des étudiants (15,8% pour l'année 1 et 20,5% pour l'année 2) ont eu (selon leurs dires) la possibilité de suivre une initiation à la programmation au cours de leur enseignement secondaire (cfr Table 9.5).

Excepté les étudiants de **profils A, B et H**, tous les autres **profils** (**C à G**) ont échoué au CI(1). Les quelques étudiants qui déclarent avoir suivi des cours de programmation ou appris certaines notions par eux-mêmes ne semblent donc pas avoir acquis une compréhension suffisante du concept de variable (cfr Table 9.5).

Pour déterminer si les distributions sont stables d'une année à l'autre, les fréquences mesurées au cours des deux années d'étude (cfr Table 9.4) sont comparées. Les étudiants de **profil A'** et ceux de **profil H** (profil non observé en année 1) ne sont pas considérés dans le test statistique de comparaison.

Le test du khi carré a été effectué indépendamment de la filière des étudiants. En effet, s'il est tenu compte de cette caractéristique individuelle, le nombre d'étudiants dans certains profils n'atteint la valeur attendue de 5.

A ce stade de l'étude, H0 n'est pas rejetée. L'échantillon observé ne montre pas de différence significative dans les distributions de fréquence des profils de compréhension entre les deux années de l'étude ($\chi^2 = 11.79, df = 6, p-value = 0.0668, alpha = 0.05$).

Ainsi, la fréquence des étudiants pour chaque profil pourrait être équivalente d'une année académique à l'autre. Cependant, plus de données sont nécessaires pour confirmer cette hypothèse.

TABLE 9.5 – Nombre d'étudiants ayant des connaissances préalables en programmation dans chaque profil

	Ayant eu des cours		Apprentissage autonome	
Profils	Année 1	Année 2	Année 1	Année 2
Profil A'	8	10	9	13
Profil A	0	0	0	0
Profil B	1	2	0	0
Profil C	0	1	1	0
Profil D	0	1	0	0
Profil E	1	0	0	1
Profil F	0	1	0	2
Profil G	0	0	1	0
Profil H	0	1	0	0

9.2.4 Retours d'enseignants

Les résultats décrits ci-avant ont été présentés à cinq enseignants diplômés en informatique (doctorat et/ou master) : l'enseignant en charge du cours d'introduction INFOB131 (Enseignant 1), les deux assistants en charge des séances pratiques (Enseignant 2 et Enseignant 3), un enseignant en charge d'un cours d'introduction à la programmation organisé à l'Université de Namur, mais dans un autre contexte (Enseignant 4) et un expert international de l'enseignement de la programmation (Enseignant 5). Tous ont été interviewés sur l'apport perçu des résultats présentées en termes de matériel didactique et de pratiques pédagogiques.

À l'exception de l'expert (Enseignant 5) qui était déjà sensibilisé aux modèles mentaux, les enseignants ont pris conscience des difficultés que peuvent rencontrer leurs étudiants à la lecture des résultats. Si tous se sont étonnés du nombre de modèles mentaux différents utilisés par les étudiants, seul l'Enseignant 4 a souligné leur incohérence.

L'Enseignant 1 souhaite décrire ces modèles mentaux aux étudiants dans le cadre du cours et leur permettre de prendre eux-mêmes conscience de ce qui peut se passer dans leur tête. L'Enseignant 4 n'est pas convaincu de l'intérêt de montrer les différents modèles mentaux, de peur que les étudiants en adoptent certains qu'ils n'avaient pas auparavant. L'Enseignant 4 propose plutôt des rencontres individualisées en fonction des besoins personnels des étudiants. Selon lui, les modèles mentaux peuvent constituer la base d'un dialogue entre l'enseignant et l'étudiant.

Tous se sont intéressés au profilage des étudiants. Les profils ont plusieurs intérêts selon eux. Les Enseignants 1 et 3 espèrent pouvoir identifier plus rapidement les étudiants en difficulté et ainsi pouvoir leur accorder plus de temps. Pour l'Enseignant 2, grâce aux profils, les étudiants peuvent se rendre compte qu'ils ne sont pas seuls et ainsi être moins découragés. Selon les Enseignants 4 et 5 , les profils devraient permettre de mettre en évidence les faiblesses des étudiants et non de souligner quand ils comprennent. Dans le même ordre d'idée, l'Enseignant 1 souhaite connaître les modèles mentaux les plus utilisés dans chaque profil. Pour l'Enseignant 4, la notion de profilage est sensible, mais reste en accord avec le travail de l'enseignant. Selon

lui, s'il existe une relation entre tous les concepts de programmation, l'évaluation permettrait de prédire à l'avance les faiblesses des étudiants par rapport à d'autres concepts.

Les Enseignants 1, 2 et 3 soulignent l'intérêt d'une évaluation "concept par concept", permettant d'identifier les concepts les plus problématiques et le moment où les étudiants décrochent. L'Enseignant 4 souligne l'intérêt d'étendre cette évaluation à d'autres concepts de base de la programmation.

Les Enseignants 1 et 4 veulent utiliser les résultats de l'évaluation pour savoir où mettre l'effort dans leur cours. L'Enseignant 1 est particulièrement intéressé à comprendre pourquoi les étudiants n'adoptent pas le modèle mental présenté pendant le cours.

Si l'effet "révélateur" de l'évaluation n'est plus à démontrer, son intérêt pour les enseignants reste à préciser en termes d'action directe. Faut-il informer les étudiants des résultats de l'évaluation ou non ? Faut-il leur présenter les différents modèles mentaux existants ? Comment les résultats pourraient-ils être pris en compte dans un cours ? Les données obtenues à travers ces cinq entretiens montrent qu'il y a matière à discussion.

9.3 Conclusion intermédiaire

Dans ce cas d'utilisation, un CI est utilisé pour mesurer le gain de compréhension chez des apprenants, sur une période donnée. Ce gain de compréhension est considéré à travers l'évolution des modèles mentaux utilisés par les apprenants pour résoudre des problèmes mettant en jeu le concept de variable en programmation. Durant deux ans, des données ont été collectées auprès de 219 étudiants suivant le cours d'introduction à la programmation (INFOB131 - 7.1) à quatre moments spécifiques dans leur apprentissage : CI(0) avant le cours préparatoire, CI(1) avant les quatre heures de cours théorique, CI(2) avant les trois heures de pratique et enfin, CI(3) à la fin de la séquence d'apprentissage dédiée à la variable.

Sur base des données collectées, neufs profils de compréhension ont été identifiés. Sept d'entre eux ont été analysés en détail à la lumière de critères tels que la filière (INFO ou INGMI), le genre, la fréquentation des cours de programmation et le nombre moyen de modèles mentaux utilisés. Comme l'avait démontré Dehnadi [Dehnadi, 2009], les étudiants qui montrent une compréhension plus rapidement acquise sont plus cohérents, c'est-à-dire qu'ils utilisent moins de modèles mentaux différents au sein d'un même CI.

En outre, la fiabilité de l'approche d'identification des profils de compréhension a été évaluée à l'aide d'un test de signification statistique. Il s'agissait de déterminer si les fréquences d'étudiants correspondant aux profils identifiés restaient stables au fil des années. Pour sept des neuf profils, les fréquences ne sont pas significativement différentes d'une année académique à l'autre. L'approche semble donc être fiable, mais doit être confirmée avec des données supplémentaires.

Finalement, l'intérêt d'une telle approche a été brièvement discuté avec cinq enseignants.

LES CONTRIBUTIONS À SOULIGNER
Une **approche pour mesurer un gain de compréhension** chez les étudiants au moyen d'un *concept inventory* a été mise en place au sein d'un cours d'introduction à la programmation en première année de bachelier (cas d'utilisation 1). Il s'agit de mesurer l'évolution des modèles mentaux utilisés pour résoudre des problèmes mettant en jeu le concept de variable.
En outre, cette approche a permis de **profiler les étudiants** : neufs profils de compréhension ont été identifiés et définis.

9.4 Menaces à la validité

Cette étude ne comporte pas de groupe de contrôle. La comparaison se fait entre deux groupes suivant le même cours (avec le même enseignant), à un an d'intervalle. Cependant, des conditions cohérentes ont été maintenues pour les deux groupes. Le temps entre les différents tests (sauf le test zéro) était très court. De plus, les résultats n'ont pas été discutés avec l'enseignant avant la fin de l'étude.

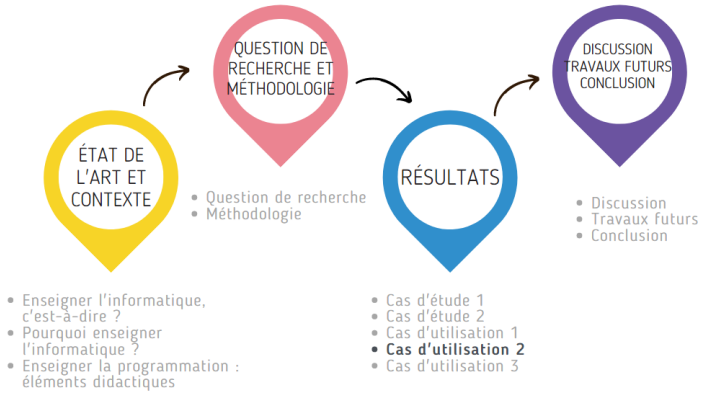
Il n'a pas été précisé aux étudiants que les tests étaient les mêmes pour les quatre mesures. L'intuition était que les étudiants ne se souviendraient pas des questions du test et n'en tireraient pas d'enseignements spécifiques. S'il n'y a aucun moyen de juger si le CI(0) a réellement influencé les résultats, cette intuition a plutôt été confirmée par des discussions informelles avec certains étudiants qui ont répondu non à la question : "avez-vous cherché comment résoudre les problèmes après avoir rempli le questionnaire ?".

La collecte de données a été présentée aux étudiants comme une évaluation formative [Fisher and Frey, 2014]. Il a également été précisé que l'enseignant en charge du cours n'aurait pas accès aux résultats avant la fin de l'étude. Il aurait alors accès à des données anonymes. De cette manière, le biais de désirabilité sociale devait être limité. Par contre, tout a été fait pour que les étudiants perçoivent les CIs comme une opportunité pour eux de s'auto-évaluer et de recevoir une aide personnalisée.

Enfin, toutes les informations relatives à l'évaluation ont été formulées clairement à des fins de réplication. Les formulaires d'évaluation et les instructions pour le codage et le décodage ont été conservés.

CHAPITRE 10

UN CONCEPT INVENTORY À ÉCHELLE DE LIKERT POUR COMPARER DEUX POPULATIONS D'APPRENANTS (CAS D'UTILISATION 2)



Ce cas d'utilisation vise un double objectif : (1) valider une approche alternative pour confirmer et quantifier des représentations erronées auprès d'un grand nombre d'apprenants en utilisant un CI à "échelle de Likert" (cfr Chapitre 8) et (2), grâce à cette nouvelle approche, avoir la possibilité de collecter des mesures statistiquement exploitables pour mettre en évidence les différences qui peuvent exister entre des populations d'apprenants.

En effet, le cours INFOB234 (cfr Section 8.1), à l'instar du cours INFOB131 (cfr Section 7.1), est suivi par des étudiants issus de filières différentes : INFO et INGMI.

Dans les faits, il est interpellant d'observer chez ces deux populations des différences de compréhension, perçues par l'enseignant à la fois dans leurs interactions durant le cours (questions, remarques, etc.) et constatées dans les évaluations sommatives. Ces deux populations jouissent pourtant des mêmes ressources pour atteindre les objectifs attendus du cours. Dès lors, parvenir à mettre le doigt sur ce qui fonde ces différences permettrait à l'enseignant de mettre en place des actions visant à offrir à chacun les mêmes chances d'apprentissage.

10.1 Méthodologie

Dans le cadre du cours INFOB234 (cfr Section 8.1) et pendant deux années académiques (années 2 et 3 du cas d'étude 2 - cfr Chapitre 8), un CI "à échelle de Likert" traitant du concept de variable (cfr Section 8.3) a été administré trois fois par an à deux groupes d'étudiants : avant le début du cours (pré-CI, 12 énoncés), au milieu du semestre (mid-CI, 14 énoncés), et à la fin du cours, avant l'examen final (post-CI,15 énoncés).

Au total, 171 étudiants (83 étudiants INFO et 88 étudiants INGMI) ont été amenés à se positionner par rapport à chacun des énoncés du CI (cfr Table 8.2) grâce à une échelle de Likert avec quatre options de réponse ("*Entirely true*/Entièrement vraie", "*True, but to be expanded*/Vraie, mais à enrichir", "*Partially true, it contains errors*/Partiellement vraie, contient des erreurs" et "*Entirely false*/Entièrement fausse") et une option "*I don't know*/Je ne sais pas".

À ce stade, seules des interprétations et des comparaisons basées sur des statistiques descriptives ont pu être faites.

10.2 Résultats

10.2.1 Pré-CI

171 étudiants ont rempli le pré-CI (cfr Figures 10.1 et 10.2).

L'énoncé **Storage** est considéré comme entièrement vrai ou à développer par une grande majorité d'étudiants. Cela confirme les résultats obtenus lors des pré-interviews quant à la représentation erronée **mis-a** (**Memory Value**) (cfr Section 8.3).

Les résultats concernant les énoncés relatifs aux mathématiques (**Math Assimilation** et **Function Component**) montrent qu'il existe encore une confusion entre variable en programmation et variable mathématique pour plus de 2/3 des étudiants. Ceci confirme les résultats obtenus lors des pré-interviews concernant la représentation erronée **mis-b** (**Math PoV**).

Les résultats obtenus lors des pré-interviews au sujet de la représentation erronée **mis-c** (**Program Segment**) sont confirmés : une confusion existe vis-à-vis de l'énoncé **Code Segment**. Cette confusion est également constatée au niveau de l'énoncé **Code Segment Reference**.

Près de 50% des étudiants INGMI ne se positionnent pas par rapport à l'énoncé **Storage Manipulation**. Ce résultat montre que la compréhension des étudiants

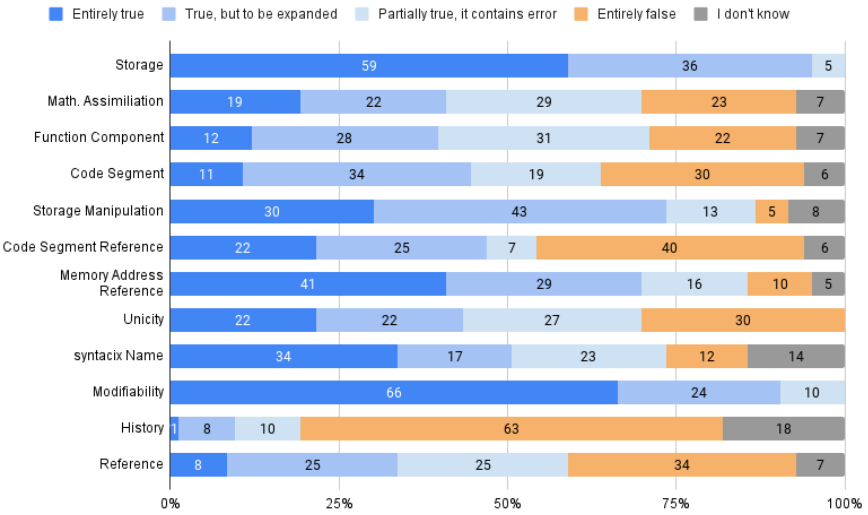


FIGURE 10.1 – Pré-CI : distribution des étudiants INFO (n = 83) par énoncé

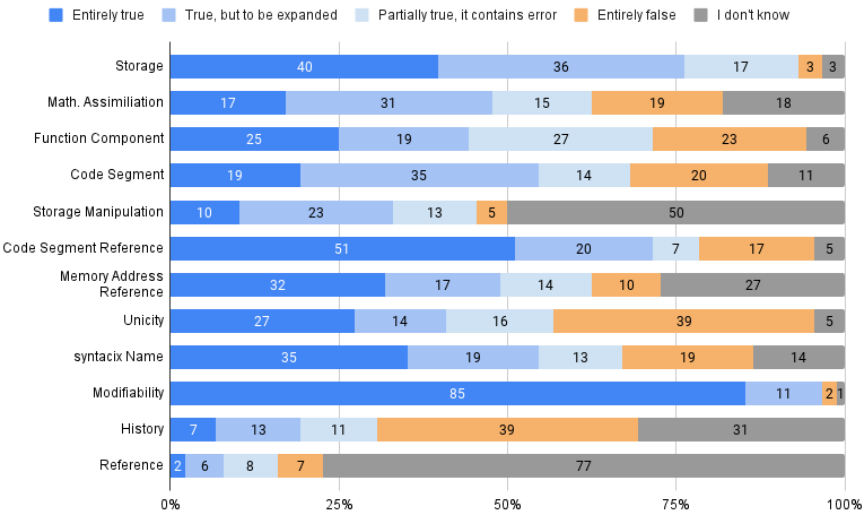


FIGURE 10.2 – Pré-CI : distribution des étudiants INGMI (n = 88) par énoncé

INFO est plus “concrète”, liée à leur connaissance du fonctionnement d’un ordinateur. Les étudiants INGMI ne suivent aucun cours leur permettant d’acquérir cette connaissance. Ce constat est confirmé par le positionnement de ces étudiants par rapport à l’énoncé **Memory Address Reference**.

Les énoncés **Unicity** et **Syntaxic Name**, liés aux connaissances théoriques du cours d’introduction à la programmation, semblent difficiles à interpréter pour les étudiants. En revanche, les étudiants semblent être assez confiants quant à l’énoncé

Modifiability.

Moins de 40% des étudiants INGMI considèrent que l'énoncé **History** est faux, contre plus de 60% des étudiants INFO.

Si près de 75% des étudiants INGMI ne savent pas comment se positionner par rapport à l'énoncé **Reference**, près de 60% des élèves INFO estiment que cet énoncé est vrai, ou partiellement vrai. Ce résultat témoigne de la plus grande compréhension des étudiants INFO. Pourtant, sur papier, les étudiants comptent tous le même nombre d'heures d'apprentissage encadré.

En bref, les résultats obtenus montrent une différence entre les deux publics (étudiants INFO et étudiants INGMI) dès le début du cours INFOB234. Prendre conscience de cette différence peut aider à y être attentif en situation d'enseignement et à agir pour garantir la meilleure compréhension possible pour tout étudiant, quelle que soit sa filière.

10.2.2 Mid-CI

77 étudiants ont rempli le mid-CI (cfr Figures 10.3 et 10.4). Seuls les énoncés pour lesquels des changements de perception sont observés par rapport au pré-CI sont discutés ci-dessous.

Si la confusion entre variable en programmation et variable mathématique existe toujours, plus d'un tiers des étudiants INGMI estiment que l'énoncé **Function Component** est faux.

La confusion concernant la représentation erronée **mis-c** (**Program Segment**) peut encore être mesurée. Si plus d'un tiers des étudiants INGMI sont d'accord pour dire que l'énoncé **Code Segment** est faux, moins de 15% disent la même chose pour l'énoncé **Code Segment Reference**.

Un peu moins d'un tiers des étudiants INGMI ne se positionnent pas par rapport à l'énoncé **Storage Manipulation**. Cette évolution entre le pré-CI et le mi-CI se confirme avec l'énoncé **Memory Address Reference**.

Les étudiants semblent douter de l'énoncé **Modifiability**. Une grande majorité des étudiants (INFO et INGMI) considèrent que l'énoncé **History** est faux. Près d'un quart des étudiants INGMI ne savent toujours pas comment se positionner par rapport à l'énoncé **Reference** et près de 30% pensent même que cet énoncé est faux. La proportion d'étudiants INFO qui pensent que cet énoncé est vrai, ou partiellement, a, quant à elle, augmenté (plus de 85%).

En ce qui concerne les énoncés **Updatable Storage** et **Stack Model**, les profils de réponse sont similaires entre les étudiants INFO et INGMI. Cela confirme les résultats obtenus lors des mid-interviews concernant les représentations erronées **mis-d** (**Manipulable Memory Value**) et **mis-e** (**Memory Place**).

Les résultats soulignent qu'il existe encore, au milieu du quadrimestre, une différence marquée entre les deux publics (INFO et INGMI).

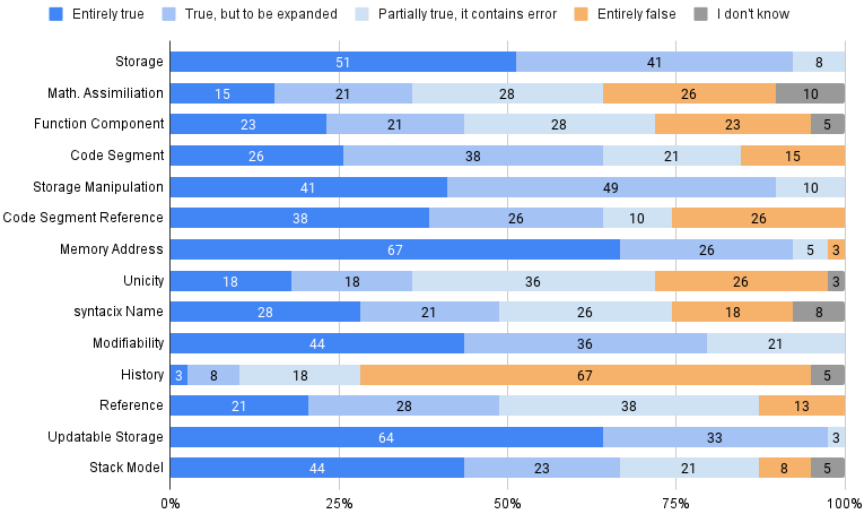


FIGURE 10.3 – Mid-CI : distribution des étudiants INFO (n = 39) par énoncé

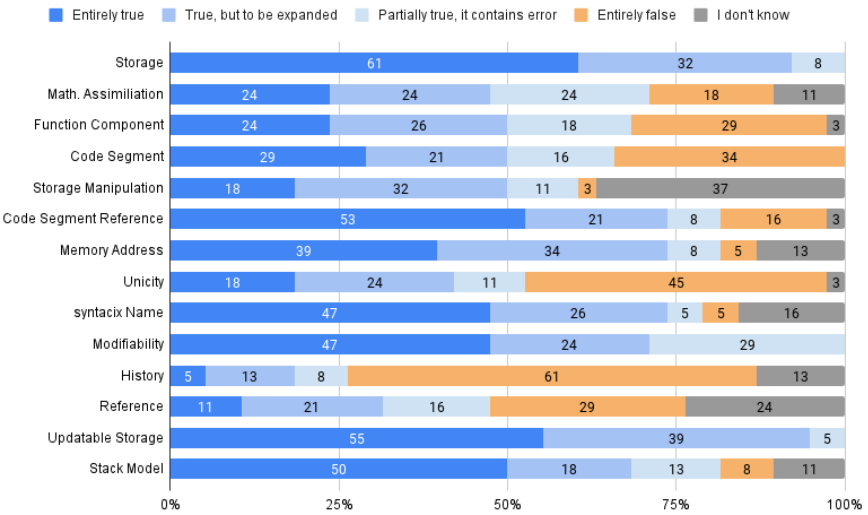


FIGURE 10.4 – Mid-CI : distribution des étudiants INGMI (n = 38) par énoncé

10.2.3 Post-CI

26 étudiants ont rempli le post-CI (cfr Figures 10.5 et 10.6). Ce faible taux de participation s'explique par le fait qu'aucune mesure n'a pu être collectée la deuxième année en raison de la crise sanitaire COVID-19. Les réponses obtenues sont donc à nuancer : elles font apparaître des tendances qui demandent à être confirmées par des mesures supplémentaires.

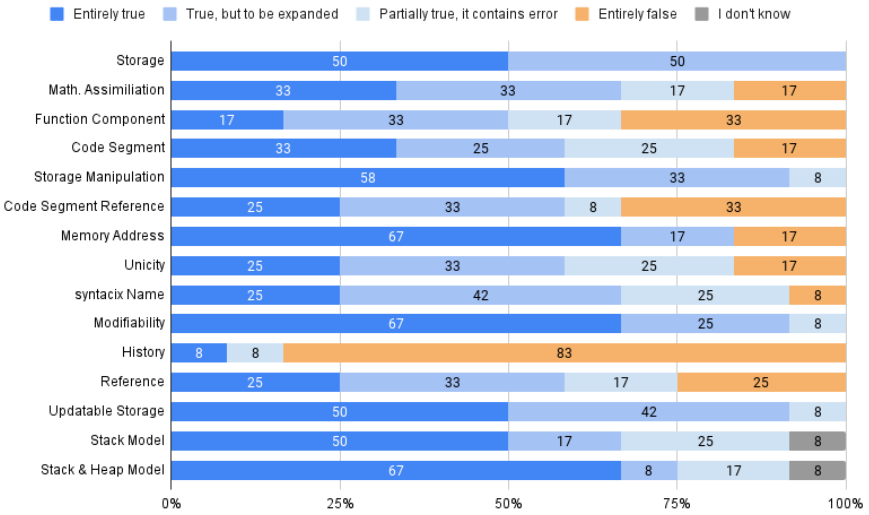


FIGURE 10.5 – Post-CI : distribution des étudiants INFO (n = 12) par énoncé

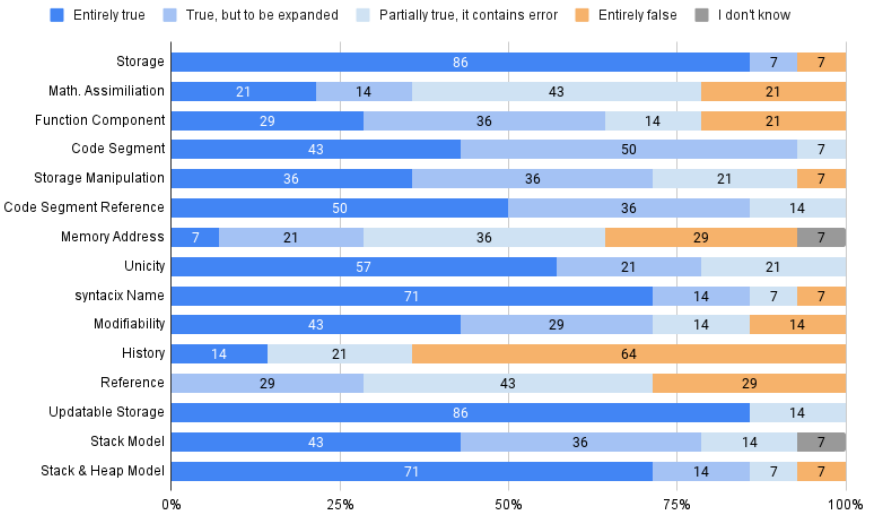


FIGURE 10.6 – Post-CI : distribution des étudiants INGMI (n = 14) par énoncé

Seuls les énoncés pour lesquels des changements de perception sont observés par rapport au mid-CI sont abordés ci-dessous.

Tout d'abord, la quasi-totalité des étudiants sont d'accord sur tous les énoncés. Seuls quatre "je ne sais pas" sont comptés sur le nombre total de réponses récoltées.

Considérant les énoncés **Math Assimilation** et **Function Component**, il semble que la confusion entre variable en programmation et variable mathématique (**mis-b - Math PoV**) existe toujours, tout comme la confusion au sujet de la représentation er-

ronée **mis-c** (**Program Segment**), surtout pour les étudiants INFO. Tous les étudiants INGMI sont d'accord sur le fait que les énoncés **Code Segment** et **Code Segment Reference** sont vrais.

Concernant l'énoncé **Storage Manipulation**, tous les étudiants INGMI prennent position pour affirmer que cet énoncé est vrai. En revanche, leur positionnement concernant l'énoncé **Memory Address Reference** est beaucoup plus distribué que dans le mid-CI.

Les énoncés **Unicity** et **Syntaxic Name** sont considérés comme vrais par presque tous les étudiants INGMI. Ces mêmes étudiants répondaient majoritairement faux à l'énoncé **Unicity** aux deux administrations précédentes. En revanche, les réponses des étudiants INFO restent distribuées.

Tous les étudiants INGMI savent comment se positionner par rapport à l'énoncé **Reference**, mais aucun ne soutient qu'il est entièrement vrai.

Concernant l'énoncé **Stack & Heap Model**, les profils de réponse sont similaires entre les étudiants INFO et INGMI. Cela confirme les résultats obtenus lors des post-interviews concernant la représentation erronée **mis-f** (**Memory Organisation**).

En bref, si la différence entre les deux publics (INFO et INGMI) se résorbe, elle est encore observée sur certains énoncés.

10.3 Discussion

En considérant les réponses obtenues sur l'ensemble des administrations, des constats peuvent être faits, touchant les deux ou l'un des deux publics (INFO ou INGMI).

Un premier constat général est que les étudiants ont abordé le cours INFOB234 en ayant, au moins théoriquement, la perception que la variable permet de stocker une valeur dans la mémoire de l'ordinateur (**mis-a, Memory Value**). Un deuxième est la confusion qui existe encore, après deux ans de cours (le cours d'introduction à la programmation ET le cours INFOB234) entre le terme "variable" en mathématiques et en programmation (**mis-b, Math PoV**).

Les énoncés (cfr Table 8.2) inspirés des représentations erronées identifiées lors des interviews (cfr Table 8.1) semblent poser quelques problèmes de positionnement aux étudiants. Il semble même exister une différence de compréhension des énoncés entre les deux publics : les étudiants INFO paraissent parfois plus confus à propos de ces énoncés. Concernant les énoncés issus du cours d'introduction à la programmation, les étudiants INGMI semblent plus confiants dans leurs réponses. Leurs représentations acquises pendant ce cours ne sont en aucun cas remises en question, tandis que les étudiants INFO semblent, ici aussi, plus confus.

Les différentes administrations du CI mettent en évidence certains constats utiles concernant la compréhension de la variable par les étudiants. Cependant, il est plus facile de tirer des conclusions lorsque les réponses ne sont pas distribuées parmi les solutions proposées. Dans ce cas, il devient difficile d'affirmer si le problème de l'étudiant provient d'une mauvaise compréhension de la variable ou si c'est l'énoncé lui-même qui pose problème. L'étape de validation (étape 5 du processus de

développement - cfr Section 6.3) des énoncés du CI doit être particulièrement bien organisée pour s'assurer que les étudiants interprètent correctement ces derniers.

10.4 Conclusion intermédiaire

Un CI à "échelle de Likert" conçu pour quantifier les représentations erronées auprès d'un grand nombre d'étudiants est au cœur de cette étude. L'échelle de Likert constitue une alternative aux questions ouvertes et fermées utilisées dans les étapes 3 et 4 du processus de développement d'un CI (cfr Section 6.3). Une étude préliminaire de trois ans à méthodes mixtes constitue la première mise en œuvre de cette approche, axée sur le concept de variable. La première année, 13 étudiants suivant le cours INFOB234 ont été interviewés trois fois au cours d'un semestre : avant le cours, à mi-parcours du semestre, et après le cours. Six représentations erronées ont été identifiées à partir de ces interviews et utilisées pour établir les quinze énoncés du CI (cfr Chapter 8). Celui-ci a ensuite été administré à 171 étudiants, issus de deux populations distinctes, sur deux ans. Un questionnaire avec échelle de Likert permet d'utiliser des tests statistiques tels que les techniques d'analyse de la variance. Dès lors, l'idée était de pouvoir comparer, au niveau de la compréhension mesurée, ces deux populations pour garantir des actions d'enseignement offrant à chacune les mêmes chances d'apprentissages.

Les résultats obtenus montrent que l'approche alternative peut être utilisée pour quantifier les représentations erronées que possèdent les étudiants. Les données recueillies illustrent également la façon dont les représentations évoluent au cours des première et deuxième années d'un enseignement de la programmation, en particulier à la lumière des différences entre les étudiants en filière informatique et les autres. En outre, l'approche devrait permettre d'obtenir un grand nombre de réponses facilement traitables d'un point de vue statistique et donc permettre la comparaison des populations INFO et INGMI. Malheureusement, cela n'a pu être confirmé étant donné les difficultés à obtenir, dans le cadre de cette recherche doctorale, un nombre suffisant de réponses pour les questionnaires 2 et 3.

LES CONTRIBUTIONS À SOULIGNER
Un ***concept inventory* "à échelle de Likert"** a été utilisé pour **quantifier les représentations erronées** liées au concept de variable que possèdent les étudiants suivant un cours de programmation orientée-objet en deuxième année de bachelier (cas d'utilisation 2). En outre, les données collectées au moyen de ce *concept inventory* sont **statistiquement exploitables**, ce qui devrait permettre notamment de **comparer des populations d'apprenants**.

10.5 Menaces à la validité

Les menaces discutées ici concernent l'étude complète, menée sur trois ans (cfr Chapitre 8).

Les biais d'échantillonnage ont été limités. L'échantillon représente la population d'intérêt. En année 1, un appel a été lancé aux étudiants volontaires pour participer aux interviews. La sélection des candidats s'est alors faite en respectant les minorités (dans le cadre de cette étude, les filles) et du ratio étudiants INFO/INGMI. En ce qui concerne les années 2 et 3, les étudiants ont été motivés à répondre aux CIs, sans obligation. Aucune sélection n'a été faite (randomisation) et l'échantillon a été considéré pour les deux années. Par chance, le ratio INFO/INGMI est respecté en ce qui concerne les répondants aux trois administrations.

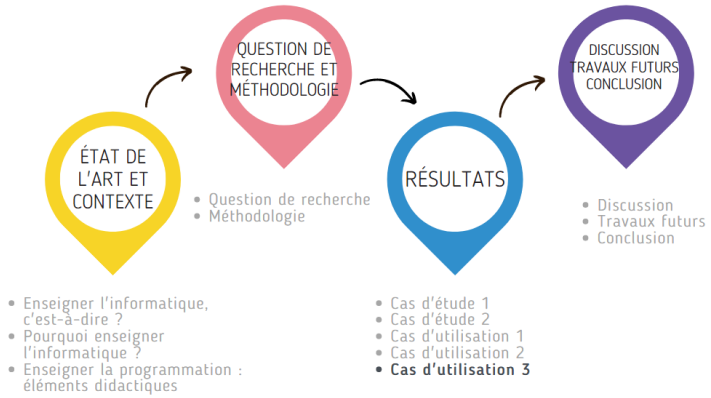
La diminution du nombre de répondants entre le pré-CI et le post-CI est un problème qui pourrait être atténué en ne prenant en compte que les étudiants ayant passé les trois CI. Toutefois, Cela réduirait considérablement le nombre d'observations possibles.

L'étude peut souffrir d'un biais de confusion. En effet, il n'a pas été jugé utile de créer un groupe de contrôle. Dès lors, l'hypothèse peut être émise que la confrontation des étudiants aux CIs a eu une influence sur l'attention qu'ils ont porté aux concepts interrogés. Toutefois, la randomisation limite ce biais en entraînant une répartition aléatoire des facteurs de confusion potentiels entre les groupes à comparer.

Enfin, les énoncés n'ont pas été validés (étape 5 du processus de développement d'un CI - cfr Section 6.3) avant la prise de mesures.

CHAPITRE 11

UN CONCEPT INVENTORY À ÉCHELLE DE LIKERT POUR MESURER UN GAIN DE COMPRÉHENSION CHEZ LES APPRENANTS (CAS D'UTILISATION 3)



ARTICLE SOUMIS INSPIRANT CE CHAPITRE

- Henry, J. et al. (2023). Student Misconceptions about Finite State Machines : Identify Them in Order to Create a Concept Inventory. **Submitted to ACM Transactions on Computing Education (TOCE)**

Ce troisième cas d'utilisation s'est déroulé dans le cadre du cours de "*Computa-*

tional Models for Embedded Systems" (CMES) organisé en deuxième année de master d'un programme "*Software Engineering*" et en première année de master d'un programme "*Distributed Systems in Internet*" à l'Université Babeș-Bolyai en Roumanie [1]. Il combine les intérêts des deux premiers : la mesure du gain de compréhension sur une période donnée ET l'utilisation d'un CI "à échelle de Likert" pour quantifier les représentations erronées (et possiblement comparer des populations) (cfr Chapitres 8 et 9). Ce cas d'utilisation concerne le concept de *finite state machine* (FSM).

Au moment d'écrire cette thèse, aucune représentation erronée sur les FSM n'apparait dans la littérature, du moins en ce qui concerne l'utilisation des FSM pour modéliser le comportement d'un système embarqué. Pourtant, les FSM constituent un véritable défi d'apprentissage pour les étudiants : ceux-ci les considèrent souvent comme non déterministes et créent facilement des états supplémentaires inutiles [Sanders et al., 2015], entre autres.

11.1 Le contexte : le cours CMES

Les données ont été collectées dans le cadre du cours CMES organisé dans deux programmes de master : "*Software Engineering*" et "*Distributed Systems in Internet*". Huit heures sont consacrées au concept de FSM : quatre heures de cours théorique (2 x 2h) et quatre heures de laboratoire. La théorie repose sur le livre de référence "*Introduction to Embedded Systems. A Cyber-Physical Systems Approach*" de Lee et Seshia [Lee and Seshia, 2017]. En ce qui concerne le laboratoire, les étudiants sont notamment évalués sur l'implémentation, par groupe de deux, d'une solution pour un système embarqué. Dans ce cadre, ils travaillent en autonomie et disposent de tutoriels. L'environnement de programmation graphique utilisé est LabVIEW [2]. Le travail est à rendre à une date précise considérée dans cette étude.

11.2 Le processus de développement du *concept inventory*

Bien que les étudiants soient les mieux placés pour fournir des informations fiables sur leurs représentations erronées, la durée limitée de cette étude n'a pas permis d'envisager l'étape 2 du processus de développement d'un CI (cfr Section 6.3). L'expérience et les observations accumulées en dix ans par l'enseignante en charge du cours CMES ont donc été utilisées pour définir quatre représentations erronées, à savoir dans le contexte du cours de CMES des définitions témoignant d'une compréhension incomplète (cfr Table 11.1).

— La représentation erronée **mis-a** est la définition la plus simple fournie par les étudiants lorsqu'ils sont interrogés (dans le cadre du cours) sur les FSM, c'est-à-dire un ensemble d'états et de transitions simples qui décrivent le comportement du système lorsqu'il change d'état, sans mentionner ni l'entrée déclenchée ni la sortie produite ;

1. À noter que dans ce cas d'utilisation, l'auteure n'est pas experte du contenu, mais considérée experte de l'approche mise en place.
2. [ni.com/fr-be/shop/labview.html]

— Les représentations erronées **mis-b** et **mis-c** ajoutent à la représentation **mis-a** respectivement l'entrée présente ou la sortie produite ;
— La représentation **mis-d** ajoute à la représentation **mis-c** des informations sur l'état actif.

TABLE 11.1 – Représentations erronées liées au concept de FSM

mis-a	*A FSM is a behavior model that consists of finite states ; transitions are describing the behavior by changing from one state to the another.*
mis-b	*A FSM is a mathematical model of computation that contains a finite set of states and transitions ; it can change from one state to another when inputs are present.*
mis-c	*A FSM models the behavior of a system by using states, transitions from one state to another with production of outputs.*
mis-d	*A FSM models the behavior of a system by using finite states, transitions from one state to another ; one single state can be active at once.*

Avant d'envisager des futures administrations du CI développé auprès d'un large public, ces représentations devront être étayées par des interviews auprès des étudiants.

À partir de ces quatre représentations erronées, l'enseignante a défini sept énoncés pour composer le CI traitant du concept de FSM (cfr Table 11.2).

Comme dans le cas d'utilisation 2 (cfr Chapitre 10), les étudiants sont amenés à se positionner par rapport à chacun de ces énoncés grâce à une échelle de Likert avec quatre options de réponse : "*Entirely true*/Entièrement vrai", "*True, to be expanded*/Vrai, mais à enrichir", "*Partially true, it contains errors*/Partiellement vrai, contient des erreurs" et "*Entirely false*/Entièrement faux". Ce positionnement vise à donner une vue globale de la compréhension des étudiants. Une cinquième option, "*I don't know*/Je ne sais pas", est également disponible.

11.3 Méthodologie

Dans le cadre du cours CMES, un CI traitant du concept de FSM a été administré à 22 étudiants : cinq étudiants de première année de master, quinze étudiants de deuxième année de master et deux étudiants Erasmus (niveau master).

Le CI a été administré cinq fois durant le semestre : une fois en début de semestre (CI(1), pré-CI), deux fois après chacun des deux cours théoriques (CI(2) et CI(3)), une fois après que les étudiants aient rendu leur travail de laboratoire (CI(4)) (à ce stade, l'ensemble des heures consacrées au concept de FSM ont été dispensées) et une fois à la fin du cours CMES (CI(5), post-CI) (cfr Figure 11.1). Cette organisation a été décidée en concertation avec l'enseignante en charge du cours CMES.

Un intervalle de deux semaines séparait les CI(1), CI(2) et CI(3). Entre les CI(3) et CI(4), cinq semaines se sont écoulées (incluant la remise du travail de laboratoire). Enfin, le CI(5) a été administré trois semaines plus tard.

TABLE 11.2 – Énoncés composant le CI sur le concept de FSM

	Énoncé	**Source**
sm-1	A state machine is a behavior model that consists of finite states; transitions are describing the behavior by changing from one state to the another.	mis-a
sm-2	A state machine is a mathematical model of computation that contains a finite set of states and transitions; it can change from one state to another when inputs are present.	mis-b
sm-3	A state machine is used to model behavior of a system by using states, transitions from one state to another with production of outputs.	mis-c
sm-4	A state machine is a mathematical model of computation that consists of a finite set of states and transitions; a transition from one state to the another is triggered when inputs are present.	mis-a and mis-b
sm-5	A state machine models the behavior of a system by using a finite set of states and transitions from one state to another when inputs are present and outputs are produced.	mis-b and mis-c
sm-6	A state machine is a mathematical model of computation that consists of finite states; transitions from one state to another is triggered when inputs are present; one single state can be active at once.	mis-b and mis-d
sm-7	A state machine models the behavior of a system by using finite states, transitions from one state to another and producing outputs; one single state can be active at once.	mis-c and mis-d

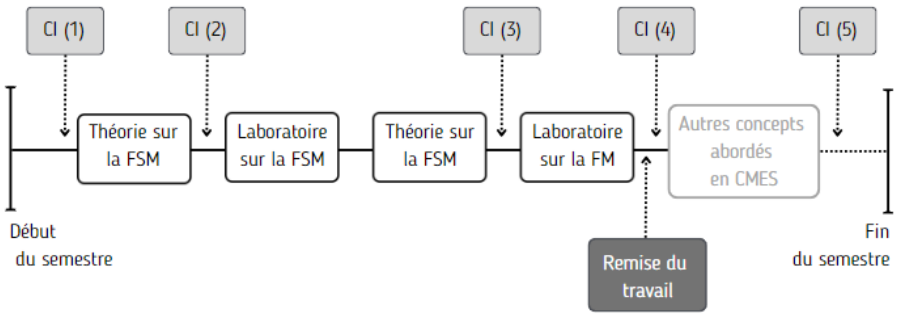


FIGURE 11.1 – Organisation de l'administration des CIs en fonction du cours CMES

11.4 Résultats

Durant un semestre, 22 étudiants se sont positionnés, à cinq reprises, vis-à-vis de sept énoncés composant un CI sur le concept de FSM (cfr Table 11.2). À travers cette étude, il s'agit de confirmer l'intérêt d'un CI "à échelle de Likert" pour quantifier les représentations erronées présentes chez les étudiants et mesurer le gain de compréhension de ceux-ci.

11.4.1 Première lecture

Le pré-CI (CI(1) - cfr Figure 11.2) confirme l'existence des représentations erronées identifiées par l'enseignante chez les étudiants. La moitié des étudiants considèrent que les énoncés **sm-4** et **sm-5** sont "Vrais, mais à enrichir". L'énoncé considéré comme "Entièrement vrai" par le plus grand nombre d'étudiants est l'énoncé **sm-6**, à savoir celui qui intègre les notions d'entrée et d'état unique.

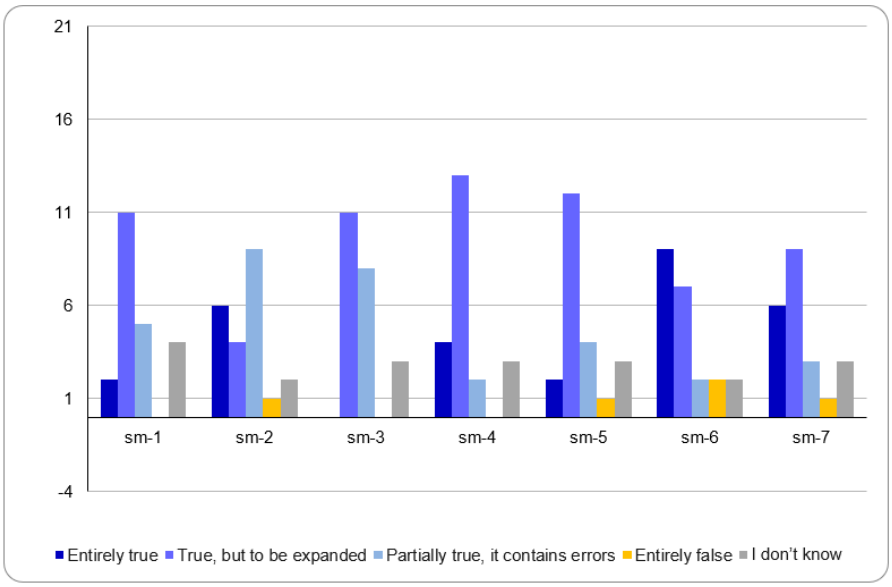


FIGURE 11.2 – CI(1) : distribution des étudiants par énoncé (n = 22)

Les résultats des CIs après les deux scéances de cours théorique (CI(2) et CI(3) - cfr Figures 11.3 et 11.4) montrent que pour la majorité des énoncés, les réponses "Vrai, mais à enrichir" et "Partiellement vrai, contient des erreurs" sont majoritaires. Entre les deux CIs, la proportion de "Vrai, mais à enrichir" augmente même. Pour l'enseignante, cela témoigne d'une certaine indécision chez les étudiants qui identifient une vérité dans chaque énoncé proposé. Seuls les énoncés **sm-6** et **sm-7** présentent une distribution légèrement différente. L'enseignante précise qu'au niveau du contenu, les diagrammes d'états (machines à état hiérarchiques) ne sont abordés que lors du deuxième cours théorique.

Après le dépôt du travail de laboratoire, les représentations des étudiants semblent se stabiliser. Davantage d'étudiants sont capables d'identifier pour chaque énoncé que certains éléments sont vrais et que d'autres sont encore manquants. Ainsi, plus de la moitié des étudiants ont répondu par "Vrai, mais à enrichir" (cfr Figure 11.5).

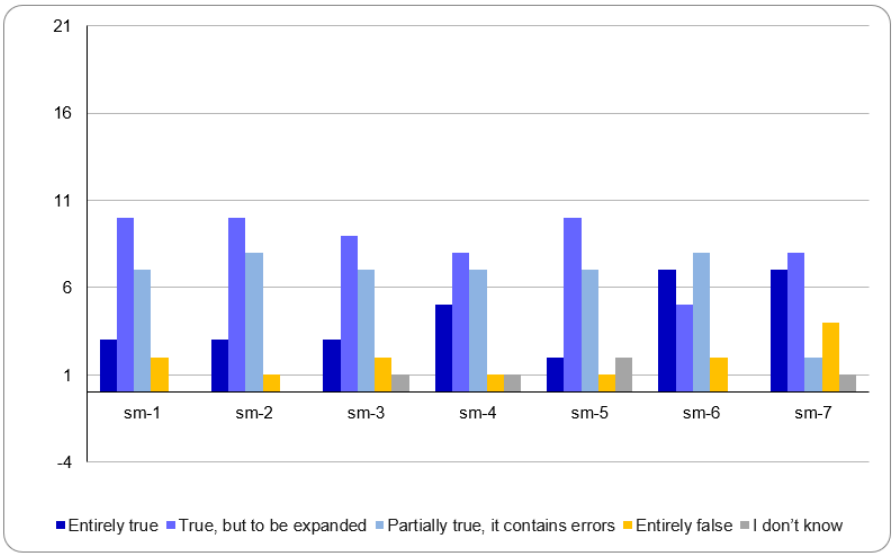


FIGURE 11.3 – CI(2) : distribution des étudiants par énoncé (n = 22)

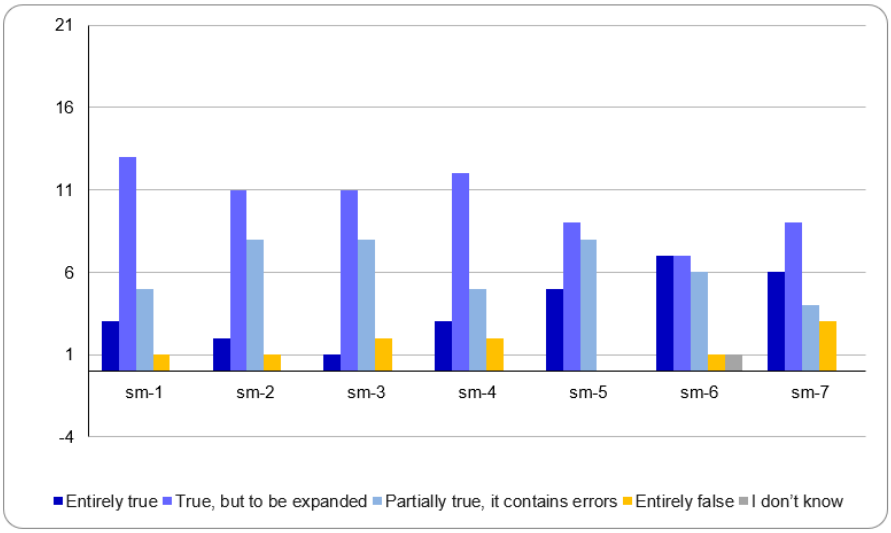


FIGURE 11.4 – CI(3) : distribution des étudiants par énoncé (n = 22)

Les résultats obtenus lors du post-CI (CI(5) - cfr Figure 11.6) ont révélé un changement de perspective majeur pour l'énoncé **sm-3** : la réponse "Partiellement vrai, contient des erreurs" est désormais majoritaire. Cet énoncé est celui qui contient l'élément de sortie de la FSM. L'enseignante pose l'hypothèse que les étudiants se sont concentrés sur la notion d'entrée d'une FSM, n'ayant pas totalement compris la notion de sortie.

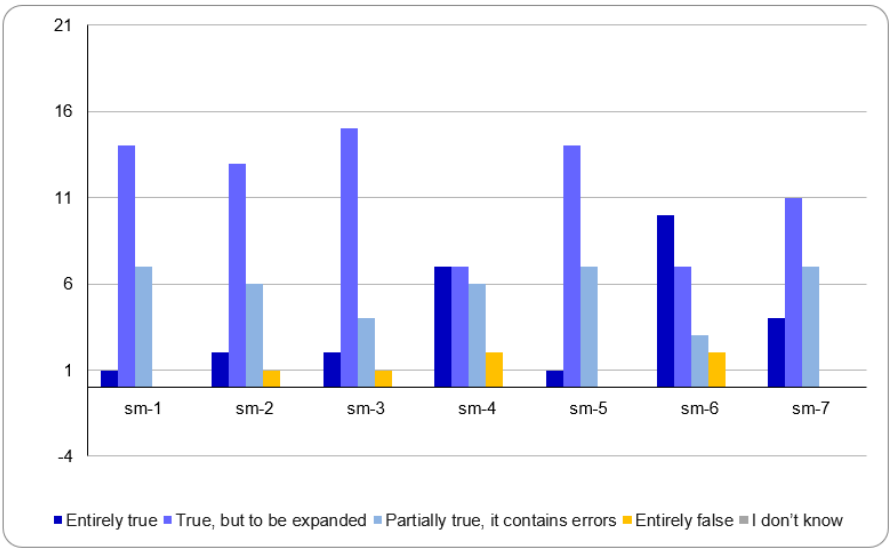


FIGURE 11.5 – CI(4) : distribution des étudiants par énoncé (n = 22)

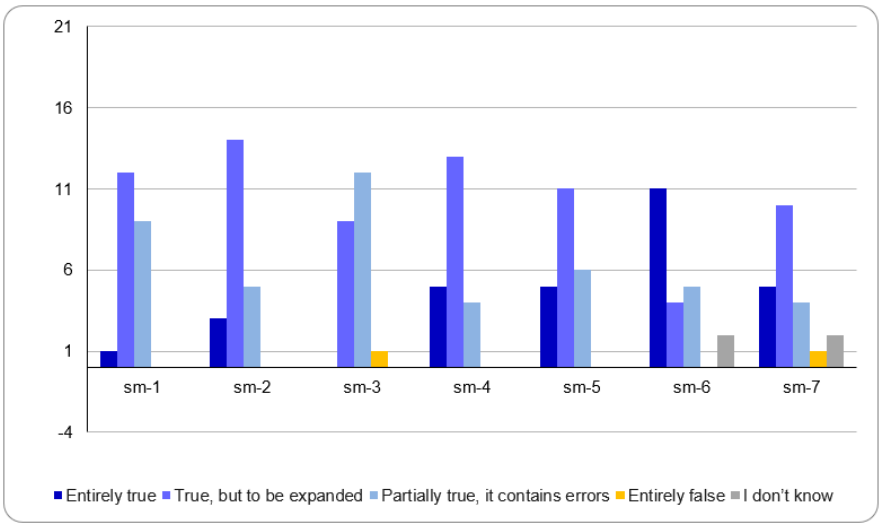


FIGURE 11.6 – CI(5) : distribution des étudiants par énoncé (n = 22)

11.4.2 Deuxième lecture

Dans cette sous-section, une deuxième visualisation est proposée pour enrichir la lecture des données collectées.

Comme mentionné précédemment, l'énoncé **sm-1** est la définition la plus simple fournie par les élèves, souvent au début de leur apprentissage. Les étudiants savent, grâce à des cours antérieurs ou à un apprentissage individuel, que cet énoncé est "Vrai, mais à enrichir" (cfr Figure 11.7). Cette réponse est restée majoritaire dans les cinq CIs et a même connu un pic après la remise du travail de laboratoire.

Cette tendance est également observée dans l'évolution de l'énoncé **sm-2** (cfr Figure 11.8) : la réponse "Vrai, mais à enrichir" augmente jusqu'au post-CI (CI(5)).

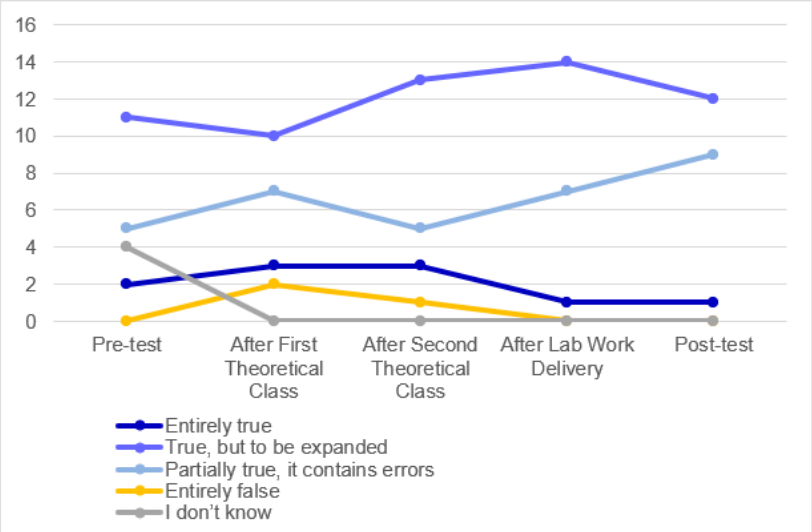


FIGURE 11.7 – Évolution de l'énoncé sm-1 (n = 22)

L'évolution de l'énoncé **sm-3** (cfr Figure 11.9) met en évidence le fait qu'à la fin de leur apprentissage, la moitié des étudiants se positionnent différemment par rapport aux réponses "Vrai, mais à enrichir" et "Partiellement vrai, contient des erreurs".

L'évolution de l'énoncé **sm-4** (cfr Figure 11.10) montre que les étudiants oscillent entre deux réponses : "Vrai, mais à enrichir" et "Partiellement vrai, contient des erreurs". Selon l'enseignante, cela pourrait s'expliquer par une mauvaise compréhension du concept de FSM en ce qui concerne l'élément d'entrée.

Une fois de plus, la réponse "Vrai, mais à enrichir" a été choisie par plus de la moitié des étudiants pour l'énoncé **sm-5** (cfr Figure 11.11), avec une forte augmentation après la remise du travail de laboratoire.

L'évolution de l'énoncé **sm-6** (cfr Figure 11.12) montre que le nombre de réponses "Entièrement vrai" augmente au fur et à mesure que les CIs sont administrés.

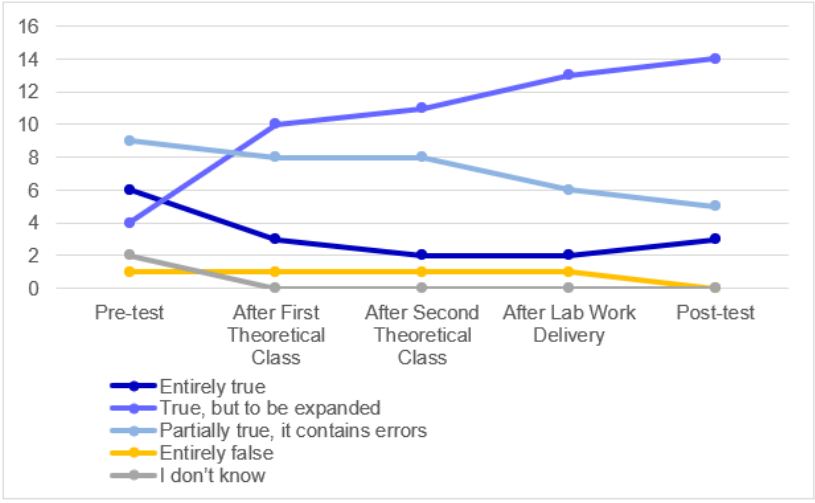


FIGURE 11.8 – Évolution de l'énoncé sm-2 (n = 22)

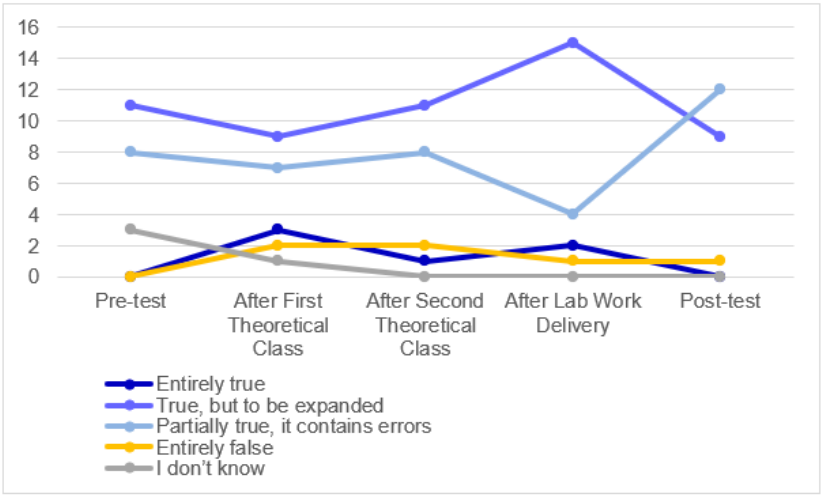


FIGURE 11.9 – Évolution de l'énoncé sm-3 (n = 22)

Enfin, le taux de réponses "Entièrement faux" à l'énoncé **sm-7** (cfr Figure 11.13) est le plus élevé, tous CIs confondus. Un pic est atteint après la première séance théorique.

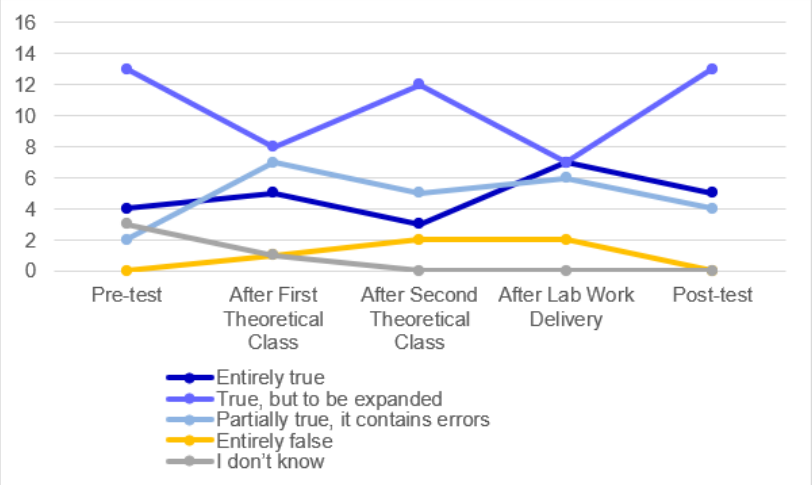


FIGURE 11.10 – Évolution de l'énoncé sm-4 (n = 22)

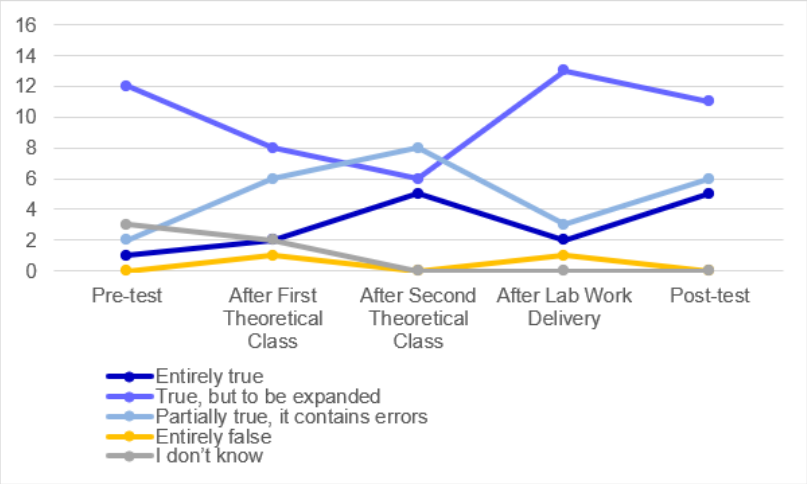


FIGURE 11.11 – Évolution de l'énoncé sm-5 (n = 22)

11.5 Discussion

Étant donné le petit nombre de participants à cette étude, les résultats sont à considérer avec une extrême prudence. Dans le cadre de cette thèse, il s'agit avant tout de voir si l'approche développée pour quantifier les représentations erronées et mesurer le gain de compréhension à partir d'un CI "à échelle de Likert" est adaptable à un tout autre contexte.

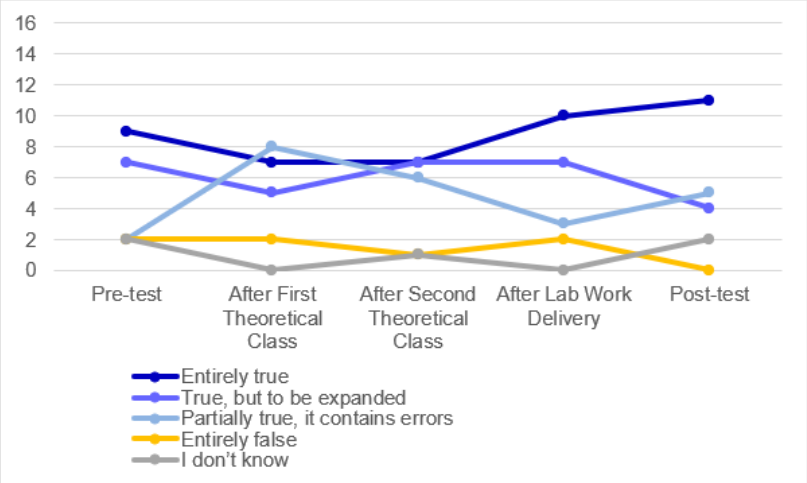


FIGURE 11.12 – Évolution de l'énoncé sm-6 (n = 22)

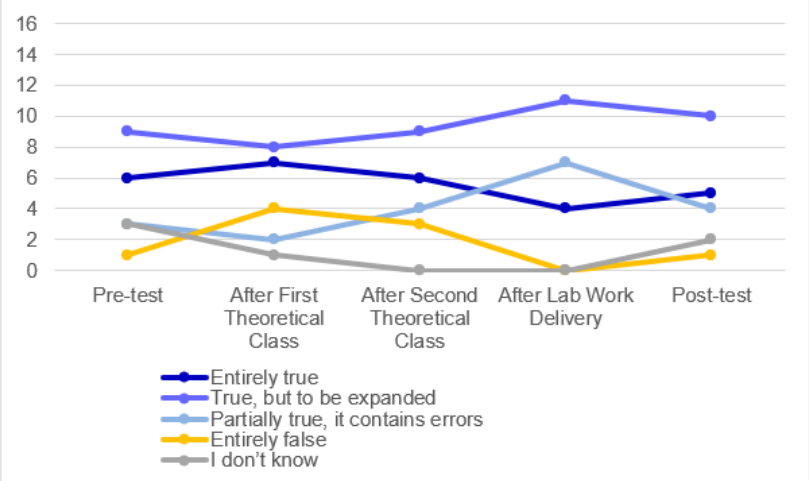


FIGURE 11.13 – Évolution de l'énoncé sm-7 (n = 22)

Si les résultats semblent montrer que l'approche alternative peut être utilisée pour quantifier les représentations erronées que possèdent les étudiants et mesurer leur gain de compréhension dans le contexte particulier du cours CMES, ces constats doivent être validés par un plus grand nombre de répondants. L'étude sera donc poursuivie en ce sens.

D'un point de vue méthodologique, l'administration des CIs telle qu'organisée dans cette étude permet moins de voir l'apport des cours théoriques sur la compréhension en comparaison aux séances pratiques (laboratoire) (ce qui était le cas dans le cadre du cours INFOB131 - cfr Chapitre 9). Il s'agirait dès lors d'évaluer l'intérêt

de cinq passations plutôt que trois, comme dans le cadre du cours INFOB234 (cfr Chapitre 10).

L'enseignante témoigne : "*First of all, being involved in this study was helpful for me to think back on all years that I taught this concept (FSM) how the students understand it, what difficulties they have in understanding it and putting it into practice, i.e. in concrete problems (problem statement to be designed as an FSM), and more importantly how I can improve my teaching (the way I explain, the problems that I give them to solve, the examples that I provide to them, maybe also providing them wrong examples and see if they see that is wrong). I also realized that students prefer more examples, immediately after I present them with the theoretical concept it is best to also have a real-world example. The results also show that their understanding improved after the lab work, so this was a confirmation for me that students better learn when doing themselves (Learning by doing, Experiential learning). They also mention that they learn better in groups so when a student does not understand right the concept, the other can explain it. As a teacher, I realized how important is the previous knowledge of the student of the concept (from previous courses). We first have to understand what the student remembers and how correctly he/she remembers, such that the elements that are added newly to the concept are built on top of a correct meaning for the students. Secondly, based on this experience, I think that concept inventories must be done for each discipline we teach.*"

11.6 Conclusion intermédiaire

Un CI "à échelle de Likert" a été conçu pour quantifier les représentations erronées que les étudiants possèdent du concept de FSM et mesurer leur gain de compréhension sur une période d'enseignement donnée. L'échelle de Likert constitue une alternative aux questions ouvertes et fermées utilisés dans les étapes 3 et 4 du processus de développement d'un CI (cfr Section 6.3). Cette étude prend place dans le cadre du cours CMES organisé à l'Université Babeș-Bolyai en Roumanie. À partir de quatre représentations erronées identifiées par l'enseignante en charge du cours sur base de ses dix années d'expérience, sept énoncés ont été définis et constituent un CI "à échelle de Likert". Ce dernier a été administré aux étudiants à cinq moments différents de leur processus d'apprentissage : avant le début du cours CMES, après le premier et le deuxième cours théoriques, après la remise du travail de laboratoire et à la fin du cours CMES.

Bien que le nombre de répondants soient insuffisants pour tirer des conclusions, certains constats ont déjà pu être faits par l'enseignante. Ainsi, les résultats du pré-CI confirment la présence de représentations erronées basées sur les connaissances acquises via des cours antérieurs au cours CMES ou un apprentissage autonome. Une évolution de ces représentations erronées est (logiquement) mesurée après les cours théoriques. En outre, une certaine stabilisation est constatée une fois le travail de laboratoire rendu, mettant en évidence, selon l'enseignante, l'impact positif de la pratique sur la compréhension du concept abstrait de FSM. Enfin, le post-CI laisse entrevoir l'existence, en fin de cours, de certaines représentations erronées.

À ce stade de l'étude, il s'agit de compléter la liste des représentations erronées via des interviews auprès des étudiants (étape 2 du processus de développement d'un CI). L'administration des CIs doit alors prendre place auprès d'un plus grand nombre d'étudiants pour confirmer l'intérêt d'une approche avec "échelle de Likert", à la fois pour quantifier statistiquement les représentations erronées des étudiants (voire même comparer les deux publics du cours CMES) et mesurer le gain de compréhension de ceux-ci sur une période donnée.

LES CONTRIBUTIONS À SOULIGNER
Un ***concept inventory* "à échelle de Likert"** sur le concept de *finite state machine* a été élaboré selon un **processus original** mis au point dans le cadre de cette recherche doctorale. Ainsi, les étapes de création de questions ouvertes et fermées du processus "classique" de développement d'un *concept inventory* ont été remplacées par la création de questions à échelle de Likert. Ce *concept inventory* a été utilisé pour **quantifier les représentations erronées** et **mesurer le gain de compréhension** des étudiants du cours de "*Computational Models for Embedded Systems*" à l'Université Babeș-Bolyai en Roumanie (cas d'utilisation 3).

11.7 Menaces à la validité

Les énoncés ont été définis, non pas à partir des étudiants, mais à partir d'une seule experte du domaine. Outre des interviews avec les étudiants (étape 5 du processus de développement d'un CI - cfr Section 6.3), une rencontre pourrait être organisée entre plusieurs enseignants du domaine afin qu'ils comparent et affinent leur compréhension mutuelle des difficultés rencontrées par leurs étudiants. De plus, les énoncés n'ont pas été validés (étape 5 du processus de développement d'un CI) avant la prise de mesures. Il faut s'assurer de la bonne compréhension des énoncés par les étudiants avant de poursuivre l'étude.

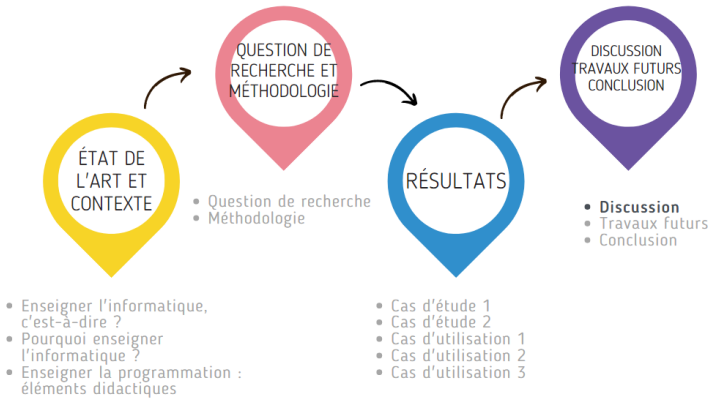
Les participants sont volontaires. Ils ont été bien moins nombreux que ce qu'avait espéré l'enseignante (+/- 20% de la population totale). Dès lors, la taille de l'échantillon est trop petite pour que soient utilisées des méthodes quantitatives. Aucune méthode qualitative appropriée n'avait été envisagée compte tenu de l'approche choisie pour cette étude.

Quatrième partie

Discussion, travaux futurs et conclusion

CHAPITRE 12

DISCUSSION



Cette recherche doctorale avait pour objectif d'apporter des éléments de réponse à la question :

Quel(s) rôle(s) les *concept inventories* peuvent-ils jouer dans l'enseignement de l'informatique ?

Pour ce faire, une étude exploratoire a été menée, mettant en jeu des *concept inventories* (CIs) de formes différentes (questions à choix multiple ou énoncés avec échelle de Likert), visant des objectifs différents, dans des contextes relativement différents (cfr Table 12.1) :

— un cours d'introduction à la programmation en première année de bachelier (cas d'étude 1 et cas d'utilisation 1)
— un cours de programmation orientée-objet en deuxième année de bachelier (cas d'étude 2 et cas d'utilisation 2)
— un cours de modélisation des systèmes embarqués en première et deuxième années de master (cas d'utilisation 3)

TABLE 12.1 – Synthèse des résultats décrits dans cette thèse

Usages	**CIs instr. cond. et fonction**	**CI variable choix multiple**	**CI variable échelle Likert**	**CI FSM**
Identifier les représentations erronées et/ou modèles mentaux	cas d'étude 1	cas d'étude 1	cas d'étude 2	cas d'util. 3
Quantifier les représentations erronées et/ou modèles mentaux			cas d'util. 2	cas d'util. 3
Mesurer le gain de compréhension		cas d'util. 1		cas d'util. 3

Parce que des discussions ont déjà été menées après chacune des études constituant cette recherche doctorale, seuls les différents rôles explorés seront considérés dans ce chapitre.

12.1 Les *concept inventories* dans l'enseignement : le point de vue des enseignants

De façon générale, le corps enseignant ayant eu l'opportunité de vivre, de près ou de loin, une étude mise en place dans le cadre de cette recherche doctorale a été intéressé par les résultats de l'administration d'un CI auprès de leurs étudiants. Cet intérêt s'explique notamment parce que cet outil permet de provoquer un conflit cognitif chez les apprenants [Veerasamy et al., 2016, Sirkiä and Sorva, 2012], de **vérifier l'efficacité d'une méthode d'enseignement particulière** [Bailey et al., 2012] et de **guider les futures interventions** [Taylor et al., 2014].

L'administration des CIs organisée en fonction du cours (avant les cours théoriques, après les cours théoriques et avant les séances pratiques, etc.) vise notamment à pouvoir fournir des résultats ciblant directement certaines actions d'enseignement. Les cas d'utilisation 1 et 3 présentent une administration qui permet de mesurer l'apport cumulé des actions de l'enseignant sur la période d'apprentissage du concept. Cela peut conduire l'enseignant à adapter ses actions et/ou le contenu de celles-ci. Ainsi, lors de la première administration du CI sur la variable dans le cadre du cours d'introduction à la programmation (cas d'utilisation 1), il a vite été repéré qu'un certain nombre d'étudiants n'avait pas compris qu'un pro-

gramme était séquentiel. De fait, l'enseignant a pris conscience qu'il ne l'évoquait pas explicitement durant les premiers cours théoriques, jugeant cela (trop) logique.

Être capable de **s'adapter à leurs étudiants et à la diversité de ceux-ci** est également un souhait exprimé par les enseignants. Notamment, les assistants en charge des séances pratiques visent à **identifier les étudiants en difficulté pour agir rapidement auprès d'eux**. En ce sens, les cinq CIs développés dans le cadre de cette thèse répondent à ce besoin. En outre, le profilage des étudiants permet de se rendre compte non seulement de la diversité des profils présents dans un auditoire, mais également de l'évolution (en lien avec l'organisation du cours) de ces profils. Autre exemple, le CI "à échelle de Likert" sur la variable a été développé sur demande du corps enseignant qui souhaitait comprendre les différences existant entre les étudiants issus de la filière informatique (INFO) et les autres (INGMI) (cas d'utilisation 2) pour les résoudre à travers les actions d'enseignement. Concernant la nécessité d'enseigner explicitement les représentations erronées et les modèles mentaux identifiés [Ben-Ari, 2001], les avis sont mitigés : si certains enseignants s'interrogent, d'autres sont convaincus.

Enfin, le cas d'utilisation 3 souligne l'**intérêt et** le **besoin, pour des enseignants en charge de cours plus avancés dans les filières informatiques, de disposer de ce type d'outil d'évaluation**. La littérature fait effectivement peu état d'exemples de CI sur des concepts plus avancés.

12.2 Les processus de développement

Comme mentionné dans la littérature, les résultats obtenus pour l'ensemble des CIs montrent qu'il est compliqué d'outrepasser l'étape 2 du processus de développement, à savoir **consulter les apprenants en amont de la création du CI**.

Il apparait nécessaire également de repasser, après l'administration du CI, par des **séances de *think aloud*** qui permettraient à la fois de **valider les questions (et leur bonne compréhension par les apprenants)**, mais aussi de **comprendre les réponses des apprenants** (étape 5). Sans cela, seul le regard/la perception du concepteur (et/ou de l'enseignant) est pris en compte. Ce point de vue s'illustre parfaitement dans le cas du CI sur le concept de fonction. Dans certains cas, pour certaines réponses fournies, il est extrêmement difficile d'arriver à définir le modèle mental utilisé par certains apprenants simplement en regardant la réponse qu'ils fournissent. Une confrontation des apprenants à leurs réponses pourrait remédier à ce problème. C'est aussi à ce prix qu'il sera possible d'**identifier des représentations erronées non perçues directement à la lecture des réponses incorrectes**.

L'**alternative "échelle de Likert"** (en lieu et place des étapes 3 et 4) semble **prometteuse pour développer des CIs**. Si les résultats trop peu nombreux obtenus dans les cas d'utilisation 2 et 3 ne permettent pas de tirer des vraies conclusions, ils laissent entrevoir le potentiel des CIs "à échelle de Likert" pour identifier des représentations erronées (cfr Section suivante).

12.3 Les *concept inventories* pour identifier les représentations et/ou modèles mentaux

Peu importe l'approche utilisée quant au processus de développement (et donc la forme du CI) et le contexte, **le CI reste un outil d'évaluation efficace dans l'identification des représentations erronées présentes chez les apprenants et/ou dans l'identification des modèles mentaux qu'ils utilisent pour résoudre des problèmes liés aux concepts informatiques**.

Dans le cas d'étude 1, des représentations erronées et des modèles mentaux non décrits dans la littérature ont été définis pour les concepts d'instruction conditionnelle et de fonction.

Les cas d'étude 2 et cas d'utilisation 3 proposent également des résultats originaux qui témoignent de l'**intérêt des CIs "à échelle de Likert" pour identifier les représentations erronées**. En outre, les CIs composés d'énoncés textuels seraient plus efficaces pour mesurer la compréhension des concepts et des relations entre eux. Un parallèle peut être fait avec la compréhension en mathématique [Bair, 2017]. Reconnaître un algorithme utilisé dans un exercice ou une méthode de résolution de problème, observer comment l'enseignant procède et le reproduire sont jugés plus faciles, en mathématiques, que de comprendre réellement les concepts. Dès lors, se positionner par rapport à un énoncé (et mieux encore devoir soi-même produire cet énoncé) permettrait de mieux mesurer la compréhension que de résoudre un exercice simple. En contrepartie, du fait de leurs énoncés textuels, les CIs "à échelle de Likert" sont **moins adaptés à l'identification des modèles mentaux**.

Enfin, le cas d'utilisation 3 laisse penser que **l'utilisation de cet outil, pour cet usage précis, est généralisable à bien d'autres contextes**. Pour confirmer ce résultat, deux études ont été mises en place au sein de l'Université Babeș-Bolyai (Roumanie) autour des concepts de fraction en mathématiques et de délégation en programmation orienté-objet.

12.4 Les *concept inventories* pour quantifier les représentations erronées et/ou modèles mentaux, voire comparer des populations

La **quantification des représentations erronées** semble **validée** dans le cas d'utilisation 2. Même si le nombre de répondants est insuffisant pour que soit réellement réalisée l'**analyse statistique**, il apparait assez clairement que celle-ci sera **possible**.

Si la **comparaison de populations** reste **possible** à travers tous les CIs développés, l'attention était portée principalement sur les CIs "à échelle de Likert", notamment parce qu'ils permettraient de mieux mesurer la compréhension. Cela n'a pas pu être confirmé par manque de données. Pourtant, les **résultats** obtenus pour le cas d'utilisation 2 s'avéraient très **prometteurs**.

12.5 Mesurer un gain de compréhension

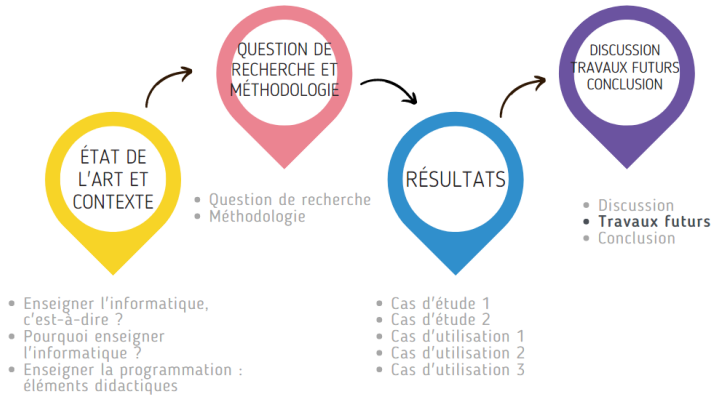
Dans un contexte où l'enseignant souhaite une garantie des apprentissages, les CIs sont un outil prisé pour **évaluer le gain de compréhension chez les apprenants** [Sands et al., 2018].

Pour ce faire, les CIs doivent remplir un certain nombre d'exigences : être valides (mesurer ce qu'ils doivent mesurer), standardisés (permettre des comparaisons significatives) et longitudinaux (permettre d'apparier les résultats collectés à des moments différents) [McGrath et al., 2015]. Ces étapes restent à mener pour les CIs développés dans le cadre de cette thèse. Toutefois, certains constats peuvent être faits.

Concernant le cas d'utilisation 1, le **profilage** fait bien état d'une évolution dans l'apprentissage. Cependant, dans sa forme actuelle (passage des modèles mentaux incorrects vers le modèle mental correct), il est difficile d'en tirer des conclusions quant à la compréhension des étudiants. Cela **met surtout en évidence les effets des actions d'enseignement sur l'apprenant**. Considérant ce qui a été dit concernant la compréhension et les énoncés textuels, **un CI "à échelle de Likert" serait potentiellement mieux adapté pour mesurer le gain de compréhension**. Le cas d'utilisation 3, malgré le petit nombre de participants, semble confirmer cette hypothèse.

CHAPITRE 13

TRAVAUX FUTURS



Cette thèse est le début d'un processus de recherche qui demande à être complété et qui permet de dégager des perspectives intéressantes pouvant faire l'objet de recherches ultérieures.

Ces perspectives vont être discutées par section, dans l'ordre d'apparition des études décrites dans ce document.

13.1 Les compétences informatiques des jeunes

Cartographier les compétences informatiques des jeunes en Belgique francophone permettrait d'analyser celles qui font défaut, et donc les besoins en acquisition. À l'heure d'écrire cette thèse, il existe un fossé entre les besoins réels de formation (qui doivent être définis) et ceux effectivement dispensés (que ce soit dans l'enseignement obligatoire ou en formation continue). Pour ce faire, les collectes de données réalisées en 2013 auprès des jeunes sortant de l'enseignement obligatoire et en 2016 auprès des enseignants en charge des cours d'informatique pourraient être reproduites à très grande échelle. Une adaptation devrait idéalement être apportée aux questionnaires pour inclure les attendus, en termes de compétences informatiques, au niveau européen.

13.2 Les compétences informatiques des enseignants

Cartographier les compétences informatiques des enseignants en Belgique francophone, notamment au regard de ce qui est attendu d'eux dans le référentiel FMTTN, permettrait d'identifier leurs besoins en termes de formation.

Bien que le référentiel du consortium {SI}² aille plus loin que ce qui est mentionné dans le référentiel FMTTN, il serait également intéressant d'analyser les données collectées non traitées dans le cadre de l'étude menée en 2016-2017, notamment en ce qui concerne les enseignants du niveau fondamental et les enseignants de Hautes Écoles. En effet, si les premiers sont directement concernés par la mise en place d'une éducation au numérique dans l'enseignement obligatoire, les seconds le sont également : ils pourraient être amenés très rapidement à inclure une préparation au cours FMTTN dans la formation initiale des enseignants. S'il est considéré que l'enseignant doit en savoir plus que l'élève, les enseignants devraient idéalement posséder certaines compétences énoncées dans le référentiel {SI}². Celles-ci pourront être identifiées sur base de la cartographie, entre autres. En outre, le questionnaire existant permetrait de relancer une collecte de données aisément.

13.3 Développer des *concept inventories* sur les concepts-clé en programmation

Il s'agirait avant tout de **continuer le processus de développement des CIs** : ajout de questions, validation du CI et administration à grande échelle. Ces nouvelles itérations permettraient de **compléter le travail d'identification des représentations erronées et modèles mentaux utilisés par les novices en programmation**.

Concernant les CIs avec questions à choix multiple, il convient de préciser que des données ont été collectées, sur quatre à cinq ans, pour les concepts de variable, instruction conditionnelle, fonction et boucle.

Aucune donnée n'a été traitée concernant le concept de boucle. Le CI sur la boucle a cependant suivi le même processus de développement que les autres, subissant trois itérations. Il est composé de sept questions, dont quatre exprimées dans le langage de programmation Java, deux en langage de programmation par

blocs et deux en pseudo-code. Le travail d'identification des modèles mentaux reste complètement à faire.

Concernant l'instruction conditionnelle, le CI obtenu au terme des cinq années de récolte de données devrait être enrichi de questions plus complexes : des problèmes supplémentaires impliquant des *if statements* imbriqués, des problèmes présentant plusieurs *if statements* consécutifs ou encore, des conditions non numériques. En effet, les questions proposées dans la version actuelle sont insuffisantes pour déterminer la totalité des modèles mentaux pouvant exister chez les novices. Certaines questions pourraient être remplacées, échouant à distinguer facilement les modèles mentaux. En outre, des nouvelles représentations erronées ont été identifiées. Il s'agirait de les confronter à la littérature scientifique et de les valider le cas échéant.

Au terme de quatre années de développement et au vu des réponses obtenues, il apparait que le nombre d'itérations et les questions proposées sont largement insuffisants pour déterminer la totalité des modèles mentaux pouvant exister chez les novices concernant le concept de fonction. La liste des modèles mentaux obtenus est incomplète et les descriptions de ceux-ci devraient être améliorées. Le travail d'identification devrait également être poursuivi en ce qui concerne les représentations erronées, idéalement à travers des interviews avec des étudiants. Une fois les modèles mentaux identifiés (et avec eux, possiblement des nouvelles représentations erronées), il sera plus facile de créer des questions supplémentaires perrmettant de tester ces modèles. Dans tous les cas, l'ajout de questions aidant à mieux visualiser les notions cruciales pour la compréhension du concept de fonction que sont l'appel et le retour est plus que nécessaire.

Outre les réponses aux questions composant les CIs, les étudiants ont également fourni des définitions personnelles pour certaines notions jugées essentielles dans la compréhension des concepts-clé. "Pour moi, une variable est…", "Pour moi, une instruction conditionnelle est…", "Pour moi, une valeur de retour d'une fonction est…", "Pour moi, un appel de fonction est…" sont des exemples de définitions demandées à chaque administration d'un CI, en fonction du concept visé. Les verbatims obtenus devraient être confrontés aux modèles mentaux et aux représentations erronées identifiés chez les étudiants. Cette confrontation pourrait aider à s'assurer que la compréhension mesurée chez les étudiants n'est pas seulement dûe à l'application d'un protocole de résolution de problème, mais bien à une compréhension profonde du concept. Le protocole *think-aloud* pourrait également être utilisé ici pour valider les résultats.

Le lien entre les représentations erronées et le discours tenu par l'enseignants devant les apprenants n'est plus à prouver. Il est donc plus que nécessaire de s'interroger sur l'utilisation des métaphores dans des contextes d'enseignement/apprentissage. Un travail est en cours concernant les métaphores utilisées pour enseigner la variable et les représentations erronées qu'elles génèrent chez les étudiants. Pendant trois années, il a été demandé à des étudiants de master de se filmer en train d'expliquer ce qu'est une variable (notamment à travers ses propriétés) à partir d'un objet du quotidien. D'autres recherche ont également été menées avec des expertes, notamment concernant les robots [Boraita et al., 2020].

Enfin, afin de **créer des outils d'aide à l'apprentissage de la programmation s'appuyant sur les représentations erronées**, des prototypes ont été développés dans le cadre de mémoires : un premier prototype visant à tangibiliser le CI sur la variable (cf Figure 13.1) et un deuxième visant à offrir une aide visuelle aux étudiants au sein même de l'environnement de programmation. Il serait intéressant de reprendre la conception de ces prototypes pour tester leur potentiel dans un contexte d'apprentissage réel.

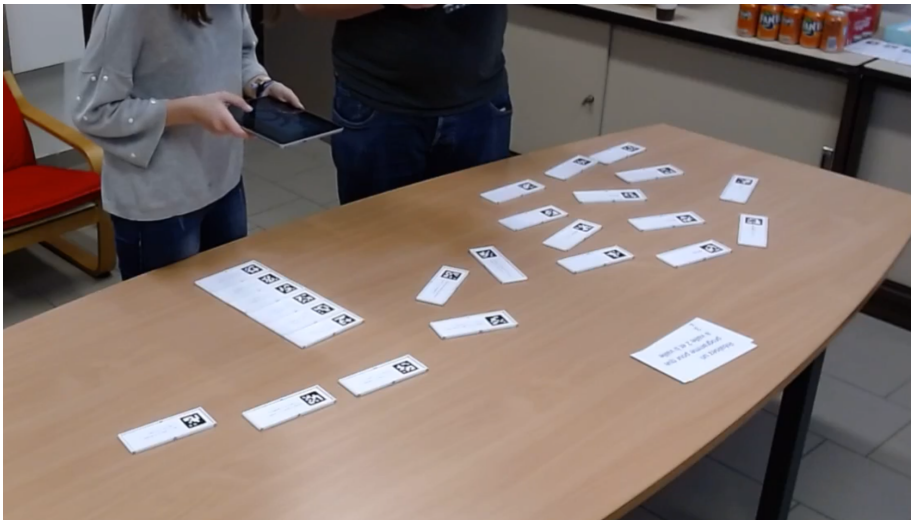


FIGURE 13.1 – Un prototype de CI tangible

ARTICLE PUBLIÉ

- Henry, J., Magis, T., Clarinval, A., Vanderose, B., & Dumas, B. (2020, August). A Tangible-Augmented Concept Inventory to Identify Novices' Misconceptions in Programming. *Proceedings of the 15th International Conference on Computer Science Education (ICCSE)*, pages 370-374. IEEE.

MÉMOIRES ENCADRÉS

- Magis, T. (2020). *Cartorithmique. Un Concept Inventory augmenté pour identifier les mauvaises représentations des novices.* [Unpublished master's thesis]. Université de Namur. (Co-promotrice)
- Moreels, M. (2022). *Animer l'exécution d'un programme : une solution pour un outil d'aide à l'apprentissage de la programmation plus didactique et adapté aux besoins des étudiants.* [Unpublished master's thesis]. Université de Namur. (Co-promotrice)

13.4 Développer des *concept inventories* “à échelle de Likert” sur les concepts-clé en programmation orientée-objet

Lors des interviews avec les 13 étudiants, des données ont été collectées pour les concepts d'objet, de primitive et de type. Ces interviews ont été totalement transcrites et le codage a déjà été effectué pour l'ensemble des concepts. Il reste possible, dès lors, d'effectuer pour les trois concepts non étudiés le même travail qui a été fait pour le concept de variable, à savoir **créer des CIs “à échelle de Likert” sur le concept d'objet, sur le concept de primitive et sur le concept de type**.

13.5 Utiliser un *concept inventory* pour mesurer le gain de compréhension chez les apprenants

De nombreuses perspectives peuvent être envisagées en ce qui concerne le cas d'utilisation 1. D'abord, les données collectées en années 4 et 5 pourraient être analysées pour **identifier les profils de compréhension et confirmer la fiabilité de l'approche**.

Ensuite, une toute autre approche pourrait être prise : **mettre le focus sur l'incompréhension plutôt que la compréhension**. Il s'agirait ici d'étudier les passages d'un modèles mental incorrect à un autre avec l'objectif de déterminer si ces passages suivent des *patterns* spécifiques. Une tentative de visualisation [1] de ces passages avait été tentée avec les données du cas d'utilisation 1 (cfr Figure **??**).

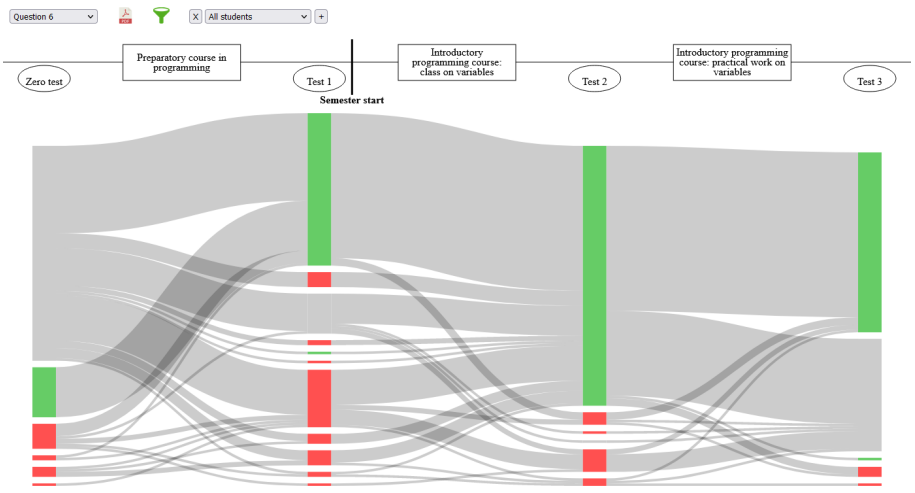


FIGURE 13.2 – Visualisation des passages d'un modèle mental à un autre entre chaque administration, par question issue du CI sur la variable

1. Prototype disponible ici : projects.info.unamur.be/programming_concepts/

Les analyses effectuées sur le concept de variable pourraient être étendues à l'ensemble des concepts pour lesquels des données ont été collectées (instruction conditionnelle, fonction et boucle).

Il serait alors intéressant de **comparer les profils de compréhension identifiés pour chacun des concepts** afin de déterminer s'il existe des similitudes en termes d'apprentissage entre les différents concepts, ou encore pour déterminer la co-existence de différents stades de raisonnement telle que décrite dans le modèle d'*Overlapping Waves* [Boom, 2015].

Confronter les résultats à des enseignants issus de contextes différents leur donnerait du sens et permettrait de **définir des actions concrètes possibles** en ce qui concerne l'enseignement, **voire de rédiger des recommandations** applicables par tout enseignant.

Enfin, la tentative de **généraliser l'approche** ayant été avortée suite à la crise sanitaire, il s'agirait de réenvisager des tests dans des contextes présentant des publics et des organisations d'enseignement très différents du cas d'utilisation 1.

13.6 Utiliser un *concept inventory* pour comparer des populations d'apprenants

Concernant le cas d'utilisation 2, **valider les énoncés du CI "à échelle de Likert"** serait l'étape suivante, avant d'**envisager une administration à très grande échelle**. En effet, les résultats obtenus laissent penser qu'il sera possible de comparer, sur base de tests statistiques, deux populations d'apprenants et d'identifier chez eux des différences en termes d'apprentissage et de compréhension. Cette recherche pourrait aider les enseignants à faire face à un public hétérogène composé d'étudiants inscrits en filière informatique et d'étudiants inscrits dans d'autres filières.

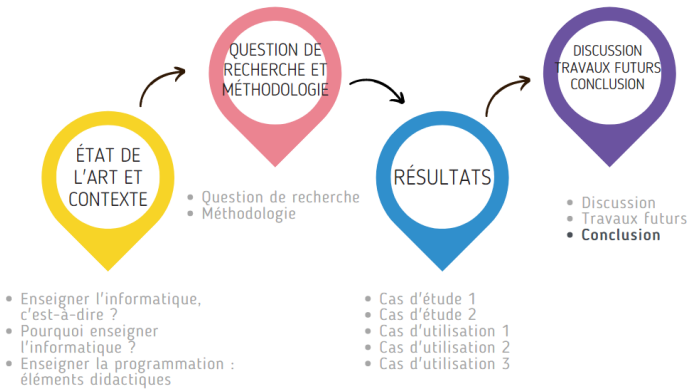
Ensuite, de nouveau, il s'agirait d'**effectuer ce même travail pour les trois concepts non étudiés** (objet, primitive et type).

Enfin, **confronter les résultats à des enseignants** issus de contextes différents leur donnerait du sens et permettrait de **définir des actions concrètes possibles** en ce qui concerne l'enseignement, **voire de rédiger des recommandations** applicables par tout enseignant.

En outre, des recherches sont actuellement en cours pour **généraliser l'approche alternative du CI "à échelle de Likert"**. Ces recherches prennent part dans des cours organisés à l'Université Babeș-Bolyai en Roumanie : un cours de mathématiques, autour du concept de fraction, un cours de programmation orientée-objet, autour du concept de délégation, et un cours de *testing* autour de concepts tels que le *white box testing* et le *black box testing*.

CHAPITRE 14

CONCLUSION



À l'heure d'écrire cette thèse, l'enseignement de l'informatique est quasi inexistant dans l'enseignement obligatoire en Belgique francophone. Pourtant, cette discipline est citée dans les discours européens comme un des éléments essentiels dans l'éducation au numérique de tout enfant. Si la situation est en passe d'être améliorée suite à une réforme de l'enseignement obligatoire, il apparait que l'accent sera mis principalement sur l'algorithmique et la programmation.

Pourtant, limiter l'enseignement de l'informatique a la programmation aurait des conséquences, notamment sur les représentations qu'ont les jeunes des métiers de l'informatique. D'autant que, dans la plupart des filières informatiques, les cours de programmation constituent à la fois le point d'entrée et l'obstacle principal pour les étudiants en première année : un obstacle jugé, par avance et par réputation, infranchissable ou sans intérêt par beaucoup d'entre eux. En se basant sur les re-

présentations qu'ils ont de la programmation et de ses concepts-clé, des potentiels candidats optent pour un autre choix de filière avant même d'avoir essayé ou sont poussés vers la sortie prématurément, ratant peut-être leur vocation. Pour pallier à ce problème, cette thèse vise à s'intéresser aux aspects didactiques aidant à l'enseignement de la programmation : les novices, les représentations erronées, leurs modèles mentaux et les outils d'évaluation que sont les *concept inventories* (CIs).

Les CIs sont conçus pour identifier rapidement les représentations erronées des apprenants et offrir ainsi aux enseignants du matériel pour améliorer leur enseignement. Il s'agit dés lors d'évaluer le potentiel de ces outils et de déterminer les **rôles qu'ils peuvent jouer dans l'enseignement de l'informatique, et dans l'enseignement de la programmation en particulier**. Pour déterminer ces rôles, un **travail exploratoire** a été **mené avec cinq CIs** : deux CIs sur le concept de variable, un sur le concept d'instruction conditionnelle, un sur le concept de fonction et un sur le concept de *finite state machine* (FSM).

Les CIs sur les concepts d'instruction et de fonction ont été développés en suivant un **processus en cinq étapes** : (1) Établir les sujets du CI, (2) Identifier les représentations erronées chez les apprenants, (3) Créer des questions ouvertes pour récolter des données supplémentaires quant aux représentations erronées, (4) Créer des questions fermées et leurs distracteurs et (5) Valider les questions par administration à une large population.

Pour les trois autres CIs, **deux alternatives** à ce processus ont été proposées, **visant à remplacer les étapes 3 et 4** :

— soit enrichir un CI existant ayant déjà été validé scientifiquement (et donc publié)
— soit créer des énoncés avec échelle de Likert et donc, développer des CIs "à échelle de Likert"

Les CIs développés et utilisés dans le cadre de cette recherche doctorale présentent donc des formes différentes : ils sont **composés soit de questions à choix multiple**, **soit d'énoncés avec échelle de Likert**. En outre, ils répondent à des besoins contextuels différents et visent des objectifs différents :

— **Identifier les représentations erronées présentes chez les apprenants et/ou les modèles mentaux qu'ils utilisent**
— **Quantifier les représentations erronées et/ou modèles mentaux, à savoir mesurer leur fréquence au sein d'un groupe d'apprenants**
— **Mesurer l'évolution des représentations erronées et/ou modèles mentaux, à savoir le gain de compréhension des apprenants sur une période donnée**

14.1 Le concept *inventory* sur le concept de variable

Le CI sur la variable a été pensé pour les besoins d'un cours d'initiation à la programmation (INFOB131) organisé en première année de bachelier à l'Université de Namur et suivi par des étudiants issus de deux filières : informatique (INFO) et ingénierie de gestion en management de l'information (INGMI). Basé sur le CI développé par Dehnadi [Dehnadi, 2009], ce CI a été enrichi, notamment par des questions en langage de programmation par blocs et des questions "visuelles" (métaphore de la boîte).

Le rôle du CI dans ce contexte a été de **mesurer le gain de compréhension des apprenants (et par là, d'identifier chez eux des profils de compréhension**). Durant deux ans, des données ont été collectées auprès de 219 étudiants à quatre moments spécifiques dans leur apprentissage, selon une approche de type pré-test post-test. Sur base des données collectées, **neufs profils de compréhension** ont été identifiés.

14.2 Les *concept inventories* sur les concepts d'instruction conditionnelle et de fonction

Comme le CI sur la variable, les CIs sur l'instruction conditionnelle et sur la fonction ont été développés dans le cadre du cours INFOB131. Le rôle de ces CI a été d'**identifier les représentations erronées des apprenants et les modèles mentaux qu'ils utilisent**.

Le **CI sur l'instruction conditionnelle** été administré selon une approche pré-test post-test. Il a été complété trois fois par an, durant quatre années, par un total de 368 étudiants. À partir des réponses obtenues, **trois nouvelles représentations erronées** ont été identifiées et **onze modèles mentaux** ont été définis.

Comme pour les autres CI, l'administration du **CI sur la fonction** s'est déroulée selon une approche pré-test post-test. Il a été complété trois fois par an, durant trois années, par un total de 262 étudiants. À partir des réponses obtenues, **quatre nouvelles représentations erronées** ont été identifiées et **onze modèles mentaux** ont été définis.

14.3 Le *concept inventory* "à échelle de Likert" sur le concept de variable

Le CI "à échelle de Likert" sur la variable a été développé dans le cadre du cours "Conception et programmation orientées objet" (INFOB234) organisé en deuxième année de bachelier en informatique, à l'Université de Namur. Ce cours est suivi par les étudiants issus des filières INFO et INGMI.

Le rôle de ce CI "à échelle de Likert" était double : (1) **quantifier des représentations erronées** auprès d'un grand nombre d'apprenants et (2) collecter des mesures statistiquement exploitables pour **mettre en évidence les différences qui peuvent exister entre des populations d'apprenants**. Si le premier rôle a été confirmé, le nombre de réponses n'a pas été suffisant pour permettre les analyses statistiques

nécessaires au deuxième rôle. Toutefois, les données recueillies ont souligné des tendances potentiellement intéressantes pour expliquer les différences existant entre les deux populations d'étudiants (INFO et INGMI).

14.4 Le *concept inventory* "à échelle de Likert" sur le concept de FSM

Le CI sur le concept de FSM répond aux besoins du cours "*Computational Models for Embedded Systems*" (CMES) organisé en deuxième année de master d'un programme "*Software Engineering*" et en première année de master d'un programme "*Distributed Systems in Internet*" à l'Université Babeș-Bolyai en Roumanie.

Ce CI "à échelle de Likert" jouait deux rôles : (1) mesurer le gain de compréhension des apprenants sur une période donnée et (2) quantifier leurs représentations erronées. Il a été administré cinq fois, durant un semestre, à 22 étudiants. Si le nombre de répondants ne permet pas de tirer des conclusions quant aux rôles du CI, ce cas d'utilisation a aidé à **généraliser le processus de développement d'un CI à "échelle de Likert" et l'approche d'administration de type pré-test post-test**.

Cinquième partie

Annexes

ANNEXE A

RÉFÉRENTIEL DE COMPÉTENCES EN SCIENCES INFORMATIQUES

Représentation des données

En ce qui concerne la représentation des données, l'élève doit être capable de :

- Comprendre la représentation interne des données par une machine
  - Expliquer pourquoi une machine représente les données sous forme binaire (0 et 1) **(R1)**
- Comprendre la représentation binaire des nombres
  - Passer de la représentation décimale d'un nombre naturel vers sa représentation binaire **(R2)**
  - Passer de la représentation binaire d'un nombre naturel vers sa représentation décimale **(R3)**
- Comprendre la représentation binaire des caractères
  - Utiliser un code pour représenter un caractère sous forme binaire (coder) **(R4)**
  - Utiliser un code pour retrouver un caractère à partir de son encodage binaire (décoder) **(R5)**
  - Discuter des avantages et inconvénients de l'utilisation d'un standard d'encodage des caractères (p.e. ASCII) **(R6)**
- Comprendre la représentation binaire des images
  - Numériser une image sous forme d'une grille de pixels **(R7)**
  - Reconstituer une image à partir d'une grille de pixels **(R8)**
  - Discuter du lien entre la qualité de l'image et sa résolution **(R9)**
  - Discuter des avantages et inconvénients d'un algorithme de compression d'image **(R10)**

Algorithmique

Du point de vue de l'algorithmique, l'élève doit être capable de :

- Lire et comprendre un algorithme simple permettant de résoudre un problème donné
  - Expliquer un algorithme simple **(A1)**
  - Simuler l'exécution d'un algorithme simple **(A2)**
  - Adapter un algorithme simple existant **(A3)**
- Concevoir un algorithme simple permettant de résoudre un problème donné
  - Écrire un algorithme comme une suite d'instructions **(A4)**
  - Identifier les instructions de base à disposition **(A5)**
  - Définir des instructions non ambiguës **(A6)**
  - Définir et utiliser des expressions de manière appropriée dans un algorithme **(A7)**
  - Définir et utiliser des variables de manière appropriée dans un algorithme **(A8)**
  - Utiliser des structures de contrôles (instructions conditionnelles ou boucles) de manière appropriée dans un algorithme **(A9)**
  - Combiner des instructions, des structures de contrôle, des variables pour écrire un algorithme **(A10)**
- Utiliser un algorithme simple pour résoudre un problème donné
  - Identifier les entrées/sorties d'un algorithme simple **(A11)**
  - Discuter des conditions d'utilisation d'un algorithme simple **(A12)**
  - Comparer l'efficacité de deux algorithmes **(A13)**
  - Identifier une suite d'instructions qui constituent un tout réutilisable **(A14)**
  - Réutiliser une suite d'instructions en définissant une fonction **(A15)**

Programmation

En programmation, l'élève doit être capable de :

- Comprendre qu'un langage de programmation est une façon de décrire un algorithme de sorte qu'il soit compréhensible par une machine
  - Identifier la correspondance entre les instructions de base d'un algorithme et celles du langage de programmation à utiliser **(P1)**
  - Identifier et utiliser les types de données adéquats **(P2)**
  - Traduire un algorithme simple dans un langage de programmation **(P3)**
  - Rédiger un programme simple dans un langage de programmation visuel **(P4)**
  - Exécuter sur machine un programme simple **(P5)**
  - Établir les cas de base à tester **(P6)**
  - Vérifier que l'exécution du programme produit les résultats attendus sur base d'un jeu de données de test **(P7)**
  - Argumenter de l'intérêt d'utiliser plusieurs jeux de données de test **(P8)**

Matériel

Concernant le matériel, l'élève doit être capable de :

- Comprendre le rôle des différents composants d'un ordinateur dans l'exécution d'un programme
  - Lister les composants de base d'un ordinateur (processeur, mémoire de travail, mémoire à long terme, périphériques d'entrée-sortie) **(M1)**
  - Expliquer la fonction de chaque composant de base d'un ordinateur **(M2)**
  - Discuter de la différence entre la mémoire à long terme et la mémoire de travail **(M3)**
  - Simuler l'exécution d'une séquence d'instructions pour illustrer le rôle des composants de base d'un ordinateur (par exemple, somme de 3 nombres) **(M4)**
  - Discuter de la manière dont une opération de haut niveau utilise les composants de base (par exemple, la visualisation d'une vidéo) **(M5)**

Réseau et sécurité

Pour la thématique réseau et sécurité, l'élève doit être capable de :

- Expliquer comment accéder à des données sur une machine distante
  - Décrire le rôle du navigateur comme programme qui affiche une page web distante (aller la chercher et interpréter le code) **(S1)**
  - Expliquer que le contenu et la mise en forme d'une page web reposent sur un langage de balises **(S2)**
  - Identifier les balises qui permettent d'insérer des liens hypertextes et des images dans une page web **(S3)**
  - Identifier les composants d'un lien internet (protocole, nom de domaine, nom du document) **(S4)**
  - Discuter de la relation entre le nom de domaine et l'adresse IP **(S5)**
  - Décrire le découpage des données en paquets et leur acheminement via des relais **(S6)**
  - Distinguer Web, Internet et réseau afin d'utiliser ces termes à bon escient **(S7)**

ANNEXE B

PUBLICATIONS

Liste de publications sélectionnées ayant fait l'objet d'une publication dans des actes de colloque, journaux ou revues à comité de lecture, ou dans des livres.

— Henry, J., Hernalesteen, A., & Collard, A.-S. (2022). Stop Hackers, un jeu de rôle pour éduquer les enfants à la cybersécurité de manière critique. *Actes du colloque DIDAPRO 9 - DIDASTIC*. Didapro 9, Le Mans, France.

— Collard, A.-S., Grandjean, N., Boraita, F., & Henry, J. (2022). Questionner les métaphores pour une éducation critique aux robots. *Cahiers du Gerse*, 51-76.

— Henry, J., & Boraita, F. (eds) (2021). *Éduquer au numérique : 12 clés pour comprendre l'informatique*. Politeia.

— Henry, J. (2021) L'information. In Henry, J., & Boraita, F. (eds) (2021). *Éduquer au numérique : 12 clés pour comprendre l'informatique*. Politeia, 17 - 34.

— Hermans, F., & Henry, J. (2021) La programmation. In Henry, J., & Boraita, F. (eds) (2021). *Éduquer au numérique : 12 clés pour comprendre l'informatique*. Politeia, 53 - 90.

— Boraita, F., Collard, A.-S., & Henry, J. (2021). De l'analyse de métaphores conceptuelles à la mise en place d'une éducation critique à la robotique. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, vol. 28.

— Henry, J., Hernalesteen, A., & Collard, A. S. (2021). Teaching Artificial Intelligence to K-12 Through a Role-Playing Game Questioning the Intelligence Concept. *KI-Künstliche Intelligenz*, 35(2) : 171-179.

— Henry, J., Lombart, C., & Dumas, B. (2021). Changer la représentation de l'informatique chez les jeunes : recommandations. *"Apprendre la Pensée Informatique de la Maternelle à l'Université", Environnements Informatiques pour l'Apprentissage Humain (EIAH)*, pages 13-24. Fribourg, Suisse.

— Henry, J., & Dumas, B. (2021). Réalité augmentée et interface tangible au ser-

vice des novices en programmation/Augmented Reality and Tangible Interaction to Help Programming Novices. *Actes de la 32e Conférence Francophone sur l'Interaction Homme-Machine*, pages 1-6.

— Henry, J., Praphamontripong, U., Serban, C., & Vescan, A. (2022). Report from the 3rd Int. Workshop on Education through Advanced Software Engineering and Artificial Intelligence (EASEAI'21). *ACM SIGSOFT Software Engineering Notes*, 47(1) : 22-24.

— Collard, A.-S., Henry, J., Hernalesteen, A., & Frénay, B. (2020). Chaire de recherche Educ0Num : Approche interdisciplinaire de l'éducation à l'intelligence artificielle. *Dossier IA Éducation*. vol. 108, Association française pour l'intelligence artificielle (AfIA).

— Henry, J., Magis, T., Clarinval, A., Vanderose, B., & Dumas, B. (2020). A Tangible-Augmented Concept Inventory to Identify Novices' Misconceptions in Programming. *Proceedings of the 15th International Conference on Computer Science Education (ICCSE)*, pages 370-374. IEEE.

— Lombart, C., Smal, A., & Henry, J. (2020). Tips and Tricks for Changing the Way Young People Conceive Computer Science. *Proceedings of the International Conference on Informatics in Schools : Situation, Evolution, and Perspectives (ISSEP)*, pages 79-93. Springer, Cham.

— Vanderose, B., Henry, J., Frénay, B., Devroey, X. (2021). Report from the 2nd Int. Workshop on Education through Advanced Software Engineering and Artificial Intelligence (EASEAI'20). *ACM SIGSOFT Software Engineering Notes*, 46(2) : 28-29.

— Henry, J., & Dumas, B. (2020). Approach to Develop a Concept Inventory Informing Teachers of Novice Programmers' Mental Models. *Proceedings of 2020 Frontiers in Education Conference (FIE)*, IEEE.

— Henry, J., Dumas, B., Heymans, P. & Leclercq, T. (2020). Object-Oriented Programming : Diagnosis Understanding by Identifying and Describing Novice Perceptions. *Proceedings of 2020 Frontiers in Education Conference (FIE)*, IEEE.

— Henry, J., Hernalesteen, A., & Collard, A.-S. (2020). Designing Digital Literacy Activities : An Interdisciplinary and Collaborative Approach. *Proceedings of 2020 IEEE Frontiers in Education Conference (FIE)*, IEEE.

— Henry, J., Boraita, F., & Collard, A.-S. (2020). Developing a Critical Robot Literacy for Young People from Conceptual Metaphors Analysis. *Proceedings of 2020 IEEE Frontiers in Education Conference (FIE)*, IEEE.

— Henry, J., & Dumas, B. (2020). Developing an assessment to profile students based on their understanding of the variable programming concept. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITICSE)*, pages 33-39. ACM.

— Vanderose, B., Frénay, B., Henry, J., & Devroey, X. (2020). Report from the 1st int. workshop on education through advanced software engineering and artificial intelligence (easeai'19). *ACM SIGSOFT Software Engineering Notes*, 45(1) : 25-27.

— Henry, J., & Dumas, B. (2019). Towards the identification of profiles based on the understanding of programming concepts : the case of the variable. In

Proceedings of 2019 Frontiers in Education Conference (FIE), IEEE.
— Henry, J., Boraita, F., & Dumas, B. (2019). Programmation tangible : vers une évaluation des concepts perçus par l'enfant à travers la manipulation d'un outil. *Des ressources numériques pour ressourcer les pratiques : Actes du 2e colloque scientifique LudoviaCH 2019*. Yverdon-Les-Bains, Suisse.
— Henry, J., Frénay, B., Dumas, B., Hernalesteen, A., & Collard, A.-S. (2019), AI in a Nutshell : Three Hands-on Activities for Teenagers. *EduAI - 28th International Joint Conference on Artificial Intelligence (IJCAI)*, Macao, China.
— Henry, J., & Dumas, B. (2018). Perceptions of computer science among children after a hands-on activity : a pilot study. *Proceedings of 2018 Global Engineering Education Conference (EDUCON)*, pages 1811-1817. IEEE.
— Henry, J., Hernalesteen, A., Dumas, B., & Collard, A. S. (2018). Que signifie éduquer au numérique? Pour une approche interdisciplinaire. In Parriaux, G., Pellet, J., Baron, G., Bruillard, E., Komis, V. (2018). *De 0 à 1 ou l'heure de l'informatique à l'école*. Lausanne, Suisse, Peter Lang Verlag.
— Vandeput, É., & Henry, J. (2018). Apprendre à programmer Comment les enseignants justifient-ils le choix d'un outil didactique? In Parriaux, G., Pellet, J., Baron, G., Bruillard, E., Komis, V. (2018). *De 0 à 1 ou l'heure de l'informatique à l'école*. Lausanne, Suisse, Peter Lang Verlag.
— Henry, J., & Smal, A. (2018, February). "Et si demain je devais enseigner l'informatique?" Le cas des enseignants de Belgique francophone. In Parriaux, G., Pellet, J., Baron, G., Bruillard, E., Komis, V. (2018). *De 0 à 1 ou l'heure de l'informatique à l'école*. Lausanne, Suisse, Peter Lang Verlag, pages 129-150.
— Henry, J., & Dumas, B. (2018). Apprentissage de la programmation par des novices : étude de l'adéquation entre concepts, outils d'apprentissage et besoins des utilisateurs. In *Une école numérique pour émanciper? Actes du colloque scientifique LudoviaCH 2018*. Yverdon-Les-Bains, Suisse.
— Théate, N., Smal, A., Frénay, B., Henry, J. (2018, March). Comprendre l'ordinateur à travers un système informatique tangible, le micro :bit. In *Une école numérique pour émanciper? Actes du colloque scientifique LudoviaCH 2018*. Yverdon-Les-Bains, Suisse.
— Olivier, B., Smal, A., Frénay, B., Henry, J. (2018). Intelligence Artificielle : Éduquer pour modifier la représentation qu'en ont les jeunes. *Une école numérique pour émanciper? Actes du colloque scientifique LudoviaCH 2018*. Yverdon-Les-Bains, Suisse.
— Henry, J., Nguyen, A., & Vandeput, É. (eds) (2017). *L'informatique et le numérique dans la classe : qui, quoi, comment?* Presses Universitaires de Namur (PUN), Namur.
— Henry, J. (2017). Formation à la maitrise des TIC pour les enseignants : un contenu inspiré de leurs pratiques. In Henry, J., Nguyen, A., & Vandeput, É. (eds) (2017). *L'informatique et le numérique dans la classe : qui, quoi, comment?* Presses Universitaires de Namur (PUN), Namur.
— Henry, J. (2016). De la "boite à applications" des enseignants au contenu d'une formation à la maîtrise des TIC. *Actes du colloque Didapro 6. Didapro 6*, Namur, Belgique.

— Henry, J., & Joris, N. (2016). Informatics at Secondary Schools in the French-Speaking Region of Belgium : Myth or Reality. *Proceedings of the International Conference on Informatics in Schools : Situation, Evolution and Perspectives (ISSEP)*, 13-24.
— Henry, J., & Joris, N. (2015). Le bagage TIC des étudiants en Belgique francophone. État des lieux. In Baron, G.-L., Bruillard, E. & Drot-Delange, B. (eds.). *Informatique en éducation : perspectives curriculaires et didactiques*. Presses Universitaires Blaise-Pascal.
— Joris, N., & Henry, J. (2014). L'enseignement de l'informatique en Belgique francophone : état des lieux. *1024 : Bulletin de la Société Informatique de France*, (2) : 107-116.
— Henry, J. & Joris, N. (2013). Maîtrise et usage des TIC : la situation des enseignants en Belgique francophone. *Actes du colloque Didapro 5 - Dida&STIC*, Clermont-Ferrand, France.

BIBLIOGRAPHIE

[Adams and Wieman, 2011] Adams, W. K. and Wieman, C. E. (2011). Development and validation of instruments to measure learning of expert-like thinking. **International Journal of Science Education**, 33(9) :1289–1312.

[Ahadi et al., 2014] Ahadi, A., Lister, R., and Teague, D. (2014). Falling behind early and staying behind when learning to program. In **PPIG**, volume 14.

[Albessart et al., 2017] Albessart, C., Calay, V., Guyot, J., Marfouk, A., and Verschueren, F. (2017). **La digitalisation de l'économie wallonne : une lecture prospective et stratégique**.

[Almstrum et al., 2006] Almstrum, V. L., Henderson, P. B., Harvey, V., Heeren, C., Marion, W., Riedesel, C., Soh, L.-K., and Tew, A. E. (2006). Concept inventories in computer science for the topic discrete mathematics. In **Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education**, pages 132–145.

[Anderson, 2005] Anderson, J. R. (2005). **Cognitive psychology and its implications**. Macmillan.

[Andrade et al., 2019] Andrade, H. L., Bennett, R. E., and Cizek, G. J. (2019). **Handbook of formative assessment in the disciplines**. Routledge.

[Andrus and Nieh, 2012] Andrus, J. and Nieh, J. (2012). Teaching operating systems using android. In **Proceedings of the 43rd ACM Technical Symposium on Computer Science Education**, pages 613–618.

[Arfé et al., 2020] Arfé, B., Vardanega, T., and Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. **Computers & Education**, 148 :103807.

[Association et al., 1999] Association, A. E. R., Association, A. P., on Measurement in Education, N. C., et al. (1999). **Standards for educational and psychological testing**. American Educational Research Association.

[Bailey et al., 2012] Bailey, J. M., Johnson, B., Prather, E. E., and Slater, T. F. (2012). Development and validation of the star properties concept inventory. **International Journal of Science Education**, 34(14) :2257–2286.

[Bair, 2017] Bair, J. (2017). **Quelques idées générales à propos de la compréhension en mathématiques**.

[Barker and Aspray, 2006] Barker, L. J. and Aspray, W. (2006). **The state of research on girls and IT**.

[Baron and Bruillard, 2001] Baron, G.-L. and Bruillard, É. (2001). Une didactique de l'informatique ? **Revue française de Pédagogie**, pages 163–172.

[Baron et al., 2010] Baron, G.-L., Drot-Delange, B., Khaneboubi, M., and Sedooka, A. (2010). Genre et informatique : compte rendu d'une enquête récente par questionnaire sur les opinions d'élèves de lycée. **Revue de l'EPI**.

[Bayman and Mayer, 1983] Bayman, P. and Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of basic programming statements. **Communications of the ACM**, 26(9) :677–679.

[Ben-Ari, 2001] Ben-Ari, M. (2001). Constructivism in computer science education. **Journal of Computers in Mathematics and Science Teaching**, 20(1) :45–73.

[Bennedsen and Caspersen, 2007] Bennedsen, J. and Caspersen, M. E. (2007). Failure rates in introductory programming. **ACM SIGCSE Bulletin**, 39(2) :32–36.

[Bennedsen and Caspersen, 2019] Bennedsen, J. and Caspersen, M. E. (2019). Failure rates in introductory programming : 12 years later. **ACM Inroads**, 10(2) :30–36.

[Bergin and Reilly, 2005] Bergin, S. and Reilly, R. (2005). Programming : factors that influence success. In **ACM SIGCSE Bulletin**, volume 37, pages 411–415. ACM.

[Biggers et al., 2008] Biggers, M., Brauer, A., and Yilmaz, T. (2008). Student perceptions of computer science : a retention study comparing graduating seniors with cs leavers. **ACM SIGCSE Bulletin**, 40(1) :402–406.

[Blackwell et al., 2019] Blackwell, A. F., Petre, M., and Church, L. (2019). Fifty years of the psychology of programming. **International Journal of Human-Computer Studies**, 131 :52–63.

[Boom, 2015] Boom, J. (2015). A new visualization and conceptualization of categorical longitudinal development : measurement invariance and change. **Frontiers in Psychology**, 6 :289.

[Boraita et al., 2020] Boraita, F., Henry, J., and Collard, A.-S. (2020). Developing a critical robot literacy for young people from conceptual metaphors analysis. In **2020 IEEE Frontiers in Education Conference (FIE)**, pages 1–7. IEEE.

[Bornat, 2014] Bornat, R. (2014). **Camels and humps : a retraction**. London, UK.

[Brinda et al., 2009] Brinda, T., Puhlmann, H., and Schulte, C. (2009). Bridging ict and cs : Educational standards for computer science in lower secondary education. **ACM SIGCSE Bulletin**, 41(3) :288–292.

[Bringula et al., 2012] Bringula, R. P., Manabat, G. M. A., Tolentino, M. A. A., and Torres, E. L. (2012). Predictors of errors of novice java programmers. **World Journal of Education**, 2(1) :3–15.

[Caceffo et al., 2018] Caceffo, R., Gama, G., Benatti, R., Aparecida, T., Caldas, T., and Azevedo, R. (2018). A concept inventory for cs1 introductory programming courses in c. **Technical Report 18-06, Institute of Computing**.

[Caceffo et al., 2016] Caceffo, R., Wolfman, S., Booth, K. S., and Azevedo, R. (2016). Developing a computer science concept inventory for introductory programming. In **Proceedings of the 47th ACM Technical Symposium on Computing Science Education**, pages 364–369.

[Cansu and Cansu, 2019] Cansu, S. K. and Cansu, F. K. (2019). An overview of computational thinking. **International Journal of Computer Science Education in Schools**, 3(1) :n1.

[Carter, 2006] Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. **ACM SIGCSE Bulletin**, 38(1) :27–31.

[Caspersen et al., 2007] Caspersen, M. E., Larsen, K. D., and Bennedsen, J. (2007). Mental models and programming aptitude. In **Proceedings of the 12th annual SIGCSE Conference on Innovation and Technology in Computer Science Education**, pages 206–210.

[Cassel et al., 2007] Cassel, L., McGettrick, A., Guzdial, M., and Roberts, E. (2007). The current crisis in computing : What are the real issues? **ACM SIGCSE Bulletin**, 39(1) :329–330.

[Cetin, 2015] Cetin, I. (2015). Students' understanding of loops and nested loops in computer programming : An apos theory perspective. **Canadian Journal of Science, Mathematics and Technology Education**, 15(2) :155–170.

[Chevallard, 1981] Chevallard, Y. (1981). Pourquoi la transposition didactique. **Communication au Séminaire de didactique et de pédagogie des mathématiques de l'IMAG, Université scientifique et médicale de Grenoble. Paru dans les Actes de l'année**, 1982 :167–194.

[Cooper et al., 2015] Cooper, S., Wang, K., Israni, M., and Sorby, S. (2015). Spatial skills training in introductory computing. In **Proceedings of the eleventh annual International Conference on International Computing Education Research**, pages 13–20. ACM.

[Danielsiek et al., 2012] Danielsiek, H., Paul, W., and Vahrenhold, J. (2012). Detecting and understanding students' misconceptions related to algorithms and data structures. In **Proceedings of the 43rd ACM Technical Symposium on Computer Science Education**, pages 21–26. ACM.

[Davies, 1993] Davies, S. P. (1993). Models and theories of programming strategy. **International journal of man-machine studies**, 39(2) :237–267.

[De la Higuera, 2015] De la Higuera, C. (2015). **À l'école, doit-on enseigner l'informatique ou le coding?**

[Dehnadi, 2009] Dehnadi, S. (2009). **A cognitive study of learning to program in introductory programming courses.** PhD thesis, Middlesex University.

[Dehnadi et al., 2006] Dehnadi, S., Bornat, R., et al. (2006). **The camel has two humps (working title)**. Middlesex University, UK.

[Denning, 2009] Denning, P. J. (2009). The profession of it beyond computational thinking. **Communications of the ACM**, 52(6) :28–30.

[Denning et al., 1989] Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., and Young, P. R. (1989). Computing as a discipline. **Computer**, 22(2) :63–70.

[Denning et al., 2017] Denning, P. J., Tedre, M., and Yongpradit, P. (2017). Misconceptions about computer science. **Communications of the ACM**, 60(3) :31–33.

[Derus and Ali, 2012] Derus, S. and Ali, A. M. (2012). Difficulties in learning programming : Views of students. In **1st International Conference on Current Issues in Education (ICCIE)**, pages 74–79.

[Dick-Perez et al., 2016] Dick-Perez, M., Luxford, C. J., Windus, T. L., and Holme, T. (2016). A quantum chemistry concept inventory for physical chemistry classes. **Journal of Chemical Education**, 93(4) :605–612.

[Dreyfus et al., 2000] Dreyfus, H., Dreyfus, S. E., and Athanasiou, T. (2000). **Mind over machine**. Simon and Schuster.

[Drot-Delange, 2011] Drot-Delange, B. (2011). Informatique et web : quelle place pour les filles ?. le cas d'étudiants d'une formation hybride. **Questions Vives. Recherches en éducation**, 8(15).

[Du Boulay, 1986] Du Boulay, B. (1986). Some difficulties of learning to program. **Journal of Educational Computing Research**, 2(1) :57–73.

[Duchâteau, 1992] Duchâteau, C. (1992). Peut-on définir une" culture informatique". **Journal de Réflexion sur l'informatique**.

[Duchateau, 2002] Duchateau, C. (2002). Mais qu'est la didactique de l'informatique devenue ? In **Les technologies en éducation : perspectives de recherche et questions vives : actes du symposium international francophone, Paris, maison des sciences de l'homme, 31 janvier-1er février 2002**, pages 33–42. INRP. Programme de numérisation pour l'enseignement et la recherche.

[Eckerdal and Thuné, 2005] Eckerdal, A. and Thuné, M. (2005). Novice java programmers' conceptions of" object" and" class", and variation theory. **ACM SIGCSE Bulletin**, 37(3) :89–93.

[Evans et al., 2003] Evans, D., Gray, G. L., Krause, S., Martin, J., Midkiff, C., Notaros, B. M., Pavelich, M., Rancour, D., Reed-Rhoads, T., Steif, P., et al. (2003). Progress on concept inventory assessment tools. In **33rd Annual Frontiers in Education (FIE)**, volume 1, pages T4G–1. IEEE.

[Farghally et al., 2017] Farghally, M. F., Koh, K. H., Ernst, J. V., and Shaffer, C. A. (2017). Towards a concept inventory for algorithm analysis topics. In **Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education**, pages 207–212.

[Fastrez, 2011] Fastrez, P. (2011). Quelles compétences le concept de littératie médiatique englobe-t-il? une proposition de définition matricielle. **Recherches en communication**, 33(33) :35–52.

[Fisher and Frey, 2014] Fisher, D. and Frey, N. (2014). **Checking for understanding : Formative assessment techniques for your classroom**. ASCD.

[Garcia-Debanc, 2008] Garcia-Debanc, C. (2008). De la configuration didactique au modèle disciplinaire en acte : trente ans de didactique du français avec jean-françois halté. **Pratiques. Linguistique, littérature, didactique**, (137-138) :39–56.

[Gentner and Stevens, 2014] Gentner, D. and Stevens, A. L. (2014). **Mental models**. Psychology Press.

[Gilberto et al., 2015] Gilberto, C.-R., Serrano-Rodríguez, M. T., Leyva-Figueredo, P. A., and Mendoza-Tauler, L. L. (2015). Methodology for characterization of cognitive activities when solving programming problems of an algorithmic nature. **Olympiads in Informatics**, page 27.

[Ginat, 2004] Ginat, D. (2004). On novice loop boundaries and range conceptions. **Computer Science Education**, 14(3) :165–181.

[Goldman et al., 2010] Goldman, K., Gross, P., Heeren, C., Herman, G. L., Kaczmarczyk, L., Loui, M. C., and Zilles, C. (2010). Setting the scope of concept inventories for introductory computing subjects. **ACM Transactions on Computing Education (TOCE)**, 10(2) :1–29.

[Greca and Moreira, 2000] Greca, I. M. and Moreira, M. A. (2000). Mental models, conceptual models, and modelling. **International Journal of Science Education**, 22(1) :1–11.

[Greening, 1998] Greening, T. (1998). Computer science : Through the eyes of potential students. In **Proceedings of the 3rd Australasian Conference on Computer Science Education**, pages 145–154.

[Grover et al., 2014] Grover, S., Pea, R., and Cooper, S. (2014). Remedying misperceptions of computer science among middle school students. In **Proceedings of the 45th ACM Technical Symposium on Computer Science Education**, pages 343–348.

[Grover et al., 2016] Grover, S., Rutstein, D., and Snow, E. (2016). " what is a computer" what do secondary school students think? In **Proceedings of the 47th acm technical symposium on computing science education**, pages 564–569.

[Guest et al., 2006] Guest, G., Bunce, A., and Johnson, L. (2006). How many interviews are enough? an experiment with data saturation and variability. **Field methods**, 18(1) :59–82.

[Guzdial, 2015] Guzdial, M. (2015). Learner-centered design of computing education : Research on computing for everyone. **Synthesis Lectures on Human-Centered Informatics**, 8(6) :1–165.

[Guzdial and Soloway, 2002] Guzdial, M. and Soloway, E. (2002). Teaching the nintendo generation to program. **Communications of the ACM**, 45(4) :17–21.

[Hagan and Markham, 2000] Hagan, D. and Markham, S. (2000). Does it help to have some programming experience before beginning a computing degree program ? In **ACM SIGCSE Bulletin**, volume 32, pages 25–28. ACM.

[Haladyna, 2004] Haladyna, T. M. (2004). **Developing and validating multiple-choice test items**. Routledge.

[Halté, 1992] Halté, J.-F. (1992). **La didactique du français**. Presses universitaires de France.

[Hamouda et al., 2017] Hamouda, S., Edwards, S. H., Elmongui, H. G., Ernst, J. V., and Shaffer, C. A. (2017). A basic recursion concept inventory. **Computer Science Education**, 27(2) :121–148.

[Hansen et al., 2016] Hansen, A. K., Dwyer, H., Harlow, D. B., and Franklin, D. (2016). What is a computer scientist ? developing the draw-a-computer-scientist-test for elementary school students. **AERA Online Paper Repository**.

[Hansen et al., 2017] Hansen, A. K., Dwyer, H. A., Iveland, A., Talesfore, M., Wright, L., Harlow, D. B., and Franklin, D. (2017). Assessing children's understanding of the work of computer scientists : The draw-a-computer-scientist test. In **Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education**, pages 279–284.

[Hartree, 2012] Hartree, D. R. (2012). **Calculating instruments and machines**. Cambridge University Press.

[Hasanah and Shimizu, 2020] Hasanah, U. and Shimizu, K. (2020). Crucial cognitive skills in science education : A systematic review. **Jurnal Penelitian dan Pembelajaran IPA**, 6(1) :36–72.

[Hemmendinger, 2010] Hemmendinger, D. (2010). A plea for modesty. **Acm Inroads**, 1(2) :4–7.

[Henry et al., 2020] Henry, J., Hernalesteen, A., and Collard, A.-S. (2020). Designing digital literacy activities : an interdisciplinary and collaborative approach. In **2020 IEEE Frontiers in Education Conference (FIE)**, pages 1–5. IEEE.

[Henry et al., 2021] Henry, J., Hernalesteen, A., and Collard, A.-S. (2021). Teaching artificial intelligence to k-12 through a role-playing game questioning the intelligence concept. **KI-Künstliche Intelligenz**, 35(2) :171–179.

[Henry et al., 2022] Henry, J., Hernalesteen, A., and Collard, A.-S. (2022). "stop hackers", un jeu de rôle pour éduquer les enfants à la cybersécurité de manière critique. In **L'informatique, objets d'enseignement et d'apprentissage. Quelles nouvelles perspectives pour la recherche ?**, pages 6–18.

[Henry and Joris, 2013] Henry, J. and Joris, N. (2013). Maîtrise et usage des tic : la situation des enseignants en belgique francophone. In **Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif, Clermont-Ferrand, France.**

[Henry and Smal, 2018] Henry, J. and Smal, A. (2018). "et si demain je devais enseigner l'informatique ?" le cas des enseignants de belgique francophone. **De 0 à 1 ou l'heure de l'informatique à l'école**, page 129.

[Herman et al., 2010] Herman, G. L., Loui, M. C., and Zilles, C. (2010). Creating the digital logic concept inventory. In **Proceedings of the 41st ACM Technical Symposium on Computer Science Education**, pages 102–106. ACM.

[Hewner, 2013] Hewner, M. (2013). Undergraduate conceptions of the field of computer science. In **Proceedings of the 9th Annual International ACM Conference on International Computing Education Research**, pages 107–114.

[Hewner and Guzdial, 2008] Hewner, M. and Guzdial, M. (2008). Attitudes about computing in postsecondary graduates. In **Proceedings of the fourth international workshop on computing education research**, pages 71–78.

[Hewson and Hewson, 1984] Hewson, P. W. and Hewson, M. G. (1984). The role of conceptual conflict in conceptual change and the design of science instruction. **Instructional Science**, 13(1) :1–13.

[Johnson-Laird, 1983] Johnson-Laird, P. N. (1983). **Mental models : Towards a cognitive science of language, inference, and consciousness**. Number 6. Harvard University Press.

[Jones and Burnett, 2008] Jones, S. and Burnett, G. (2008). Spatial ability and learning to program. **Human Technology : An Interdisciplinary Journal on Humans in ICT Environments**.

[Kaczmarczyk et al., 2010] Kaczmarczyk, L. C., Petrick, E. R., East, J. P., and Herman, G. L. (2010). Identifying student misconceptions of programming. In **Proceedings of the 41st ACM Technical Symposium on Computer Science Education**, pages 107–111.

[Kalelioğlu, 2015] Kalelioğlu, F. (2015). A new way of teaching programming skills to k-12 students : Code. org. **Computers in Human Behavior**, 52 :200–210.

[Karpierz and Wolfman, 2014] Karpierz, K. and Wolfman, S. A. (2014). Misconceptions and concept inventory questions for binary search trees and hash tables. In **Proceedings of the 45th ACM Technical Symposium on Computer Science Education**, pages 109–114. ACM.

[Kessler and Anderson, 1986] Kessler, C. M. and Anderson, J. R. (1986). Learning flow of control : Recursive and iterative procedures. **Human-Computer Interaction**, 2(2) :135–166.

[Ketele and Roegiers, 2009] Ketele, J.-M. d. and Roegiers, X. (2009). Méthodologie du recueil d'informations : Fondements des méthodes d'observation, de questionnaires, d'interviews et d'étude de documents. **Pédagogies en développement. Méthodologie de la recherche**.

[Kinnunen and Malmi, 2006] Kinnunen, P. and Malmi, L. (2006). Why students drop out cs1 course? In **Proceedings of the 2nd International Workshop on Computing Education Research**, pages 97–108.

[Kohn, 2017] Kohn, T. (2017). Variable evaluation : An exploration of novice programmers' understanding and common misconceptions. In **Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education**, pages 345–350.

[Lahtinen et al., 2005] Lahtinen, E., Ala-Mutka, K., and Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. **ACM SIGCSE bulletin**, 37(3) :14–18.

[Lambert, 2015] Lambert, L. (2015). Factors that predict success in cs1. **Journal of Computing Sciences in Colleges**, 31(2) :165–171.

[Lau and Yuen, 2011] Lau, W. W. and Yuen, A. H. (2011). Modelling programming performance : Beyond the influence of learner characteristics. **Computers & Education**, 57(1) :1202–1213.

[Lee and Seshia, 2017] Lee, E. A. and Seshia, S. A. (2017). **Introduction to Embedded Systems. A Cyber-Physical Systems Approach**. MIT Press.

[Leeper and Silver, 1982] Leeper, R. and Silver, J. (1982). Predicting success in a first programming course. **ACM SIGCSE Bulletin**, 14(1) :147–150.

[Lejeune, 2019] Lejeune, C. (2019). **Manuel d'analyse qualitative**. De Boeck Supérieur.

[Linn and Dalbey, 1985] Linn, M. C. and Dalbey, J. (1985). Cognitive consequences of programming instruction : Instruction, access, and ability. **Educational Psychologist**, 20(4) :191–206.

[Lister, 2011] Lister, R. (2011). Concrete and other neo-piagetian forms of reasoning in the novice programmer. In **Conferences in research and practice in information technology series**.

[Lister, 2016] Lister, R. (2016). Toward a developmental epistemology of computer programming. In **Proceedings of the 11th workshop in primary and secondary computing education**, pages 5–16.

[Luxton-Reilly et al., 2018a] Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. (2018a). Introductory programming : a systematic literature review. In **Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education**, pages 55–106.

[Luxton-Reilly et al., 2018b] Luxton-Reilly, A., Becker, B. A., Cao, Y., McDermott, R., Mirolo, C., Mühling, A., Petersen, A., Sanders, K., Whalley, J., et al. (2018b). Developing assessments to determine mastery of programming fundamentals. In **Proceedings of the 2017 ITiCSE Conference on Working Group Reports**, pages 47–69. ACM.

[Ma, 2007] Ma, L. (2007). **Investigating and Improving Novice Programmers' Mental Models of Programming Concepts**. PhD thesis, University of Strathclyde.

[Ma et al., 2008] Ma, L., Ferguson, J. D., Roper, M., Ross, I., and Wood, M. (2008). Using cognitive conflict and visualisation to improve mental models held by novice programmers. **ACM SIGCSE Bulletin**, 40(1) :342–346.

[Maier, 2004] Maier, S. (2004). Misconception research and piagetian models of intelligence. In **Proceedings of 2004 Oklahoma Higher Education Teaching and Learning Conference**.

[Mancy and Reid, 2004] Mancy, R. and Reid, N. (2004). Aspects of cognitive style and programming. In **Proceedings of the 16th Annual Workshop of the Psychology of Programming Interest Group**, pages 1–9.

[Martin, 2004] Martin, C. D. (2004). Draw a computer scientist. **ACM SIGCSE Bulletin**, 36(4) :11–12.

[McCracken, 1957] McCracken, D. D. (1957). **Digital computer programming**. J. Wiley.

[McGrath et al., 2015] McGrath, C. H., Guerin, B., Harte, E., Frearson, M., and Manville, C. (2015). Learning gain in higher education. **Santa Monica, CA : RAND Corporation**.

[Menon et al., 2020] Menon, D., Sowmya, B., Romero, M., and Viéville, T. (2020). Going beyond digital literacy to develop computational thinking in k-12 education. **Epistemological Approaches to Digital Learning in Educational Contexts**, pages 17–34.

[Milne and Rowe, 2002] Milne, I. and Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. **Education and Information Technologies**, 7(1) :55–66.

[Moreno-León et al., 2016] Moreno-León, J., Robles, G., and Román-González, M. (2016). Code to learn : Where does it belong in the k-12 curriculum. **Journal of Information Technology Education : Research**, 15(1) :283–303.

[Nielsen and Molich, 1990] Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In **Proceedings of the SIGCHI conference on Human factors in computing systems**, pages 249–256.

[Norman, 2014] Norman, D. A. (2014). Some observations on mental models. In **Mental models**, pages 15–22. Psychology Press.

[Oman, 1986] Oman, P. W. (1986). Identifying student characteristics influencing success in introductory computer science courses. **AEDS journal**, 19(2-3) :226–233.

[Orey, 2010] Orey, M. (2010). **Emerging perspectives on learning, teaching and technology.** CreateSpace North Charleston.

[Özdener, 2008] Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. **Computers & Education**, 51(3) :1094–1102.

[Özmen and Altun, 2014] Özmen, B. and Altun, A. (2014). Undergraduate students' experiences in programming : Difficulties and obstacles. **Turkish Online Journal of Qualitative Inquiry**, 5(3).

[Pantic et al., 2018] Pantic, K., Clarke-Midura, J., Poole, F., Roller, J., and Allan, V. (2018). Drawing a computer scientist : stereotypical representations or lack of awareness? **Computer Science Education**, 28(3) :232–254.

[Papert and Harel, 1991] Papert, S. and Harel, I. (1991). Situating constructionism. **Constructionism**, 36(2) :1–11.

[Perkins and Martin, 1986] Perkins, D. N. and Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In **Papers presented at the 1st Workshop on Empirical Studies of Programmers**, pages 213–229.

[Piteira and Costa, 2012] Piteira, M. and Costa, C. (2012). Computer programming and novice programmers. In **Proceedings of the Workshop on Information Systems and Design of Communication**, pages 51–53. ACM.

[Piteira and Costa, 2013] Piteira, M. and Costa, C. (2013). Learning computer programming : study of difficulties in learning programming. In **Proceedings of the 2013 International Conference on Information Systems and Design of Communication**, pages 75–80.

[Popat and Starkey, 2019] Popat, S. and Starkey, L. (2019). Learning to code or coding to learn? a systematic review. **Computers & Education**, 128 :365–376.

[Prensky, 2001] Prensky, M. (2001). Digital natives, digital immigrants part 2 : Do they really think differently? **On the horizon**.

[Qian and Lehman, 2017] Qian, Y. and Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming : A literature review. **ACM Transactions on Computing Education (TOCE)**, 18(1) :1–24.

[Ragonis and Ben-Ari, 2005] Ragonis, N. and Ben-Ari, M. (2005). A long-term investigation of the comprehension of oop concepts by novices.

[Renumol et al., 2010] Renumol, V., Janakiram, D., and Jayaprakash, S. (2010). Identification of cognitive processes of effective and ineffective students during computer programming. **ACM Transactions on Computing Education (TOCE)**, 10(3) :10.

[Resnick et al., 2009] Resnick, M., Flanagan, M., Kelleher, C., MacLaurin, M., Ohshima, Y., Perlin, K., and Torres, R. (2009). Growing up programming : democratizing the creation of dynamic, interactive media. In **CHI'09 Extended Abstracts on Human Factors in Computing Systems**, pages 3293–3296.

[Richardson, 1999] Richardson, J. T. (1999). The concepts and methods of phenomenographic research. **Review of educational research**, 69(1) :53–82.

[Robins et al., 2003] Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming : A review and discussion. **Computer Science Education**, 13(2) :137–172.

[Rogalski, 1989] Rogalski, J. (1989). Didactique de l'informatique et acquisition de la programmation. **Publications mathématiques et informatique de Rennes**, (S6) :87–93.

[Rogalski and Samurçay, 1986] Rogalski, J. and Samurçay, R. (1986). Les problèmes cognitifs rencontrés par des élèves de l'enseignement secondaire dans l'apprentissage de l'informatique. **European Journal of Psychology of Education**, 1(2) :97–110.

[Romainville, 1988] Romainville, M. (1988). Une analyse critique de l'initiation a l'informatique : quels apprentissages et quels transferts ? In **Colloque francophone sur la didactique de l'informatique**, pages 223–241. Association EPI.

[Romero et al., 2017] Romero, M., Lepage, A., and Lille, B. (2017). Computational thinking development through creative programming in higher education. **International Journal of Educational Technology in Higher Education**, 14(1) :1–15.

[Rørnes et al., 2019] Rørnes, K. M., Runde, R. K., and Jensen, S. M. (2019). Students' mental models of references in python. In **Norsk IKT-konferanse for forskning og utdanning**.

[Rountree et al., 2002] Rountree, N., Rountree, J., and Robins, A. (2002). Predictors of success and failure in a cs1 course. **ACM SIGCSE Bulletin**, 34(4) :121–124.

[Rücker and Pinkwart, 2016] Rücker, M. T. and Pinkwart, N. (2016). Review and discussion of children's conceptions of computers. **Journal of Science Education and Technology**, 25(2) :274–283.

[Rushkoff and Purvis, 2011] Rushkoff, D. and Purvis, L. (2011). **Program or be programmed : Ten commands for a digital age**. Berkeley, CA : Counterpoint.

[Ruslanov and Yolevich, 2011] Ruslanov, A. D. and Yolevich, A. P. (2011). College student notions of computer science. In **AIP Conference Proceedings**, volume 1373, pages 137–148. American Institute of Physics.

[Saariketo, 2014a] Saariketo, M. (2014a). Imagining alternative agency in techno-society : outlining the basis of critical technology education (en). **Media Practice and Everyday Agency in Europe**, pages 129–138.

[Saariketo, 2014b] Saariketo, M. (2014b). Reflections on the question of technology in media literacy education. In **Reflections on media education futures : contributions to the Conference Media Education Futures in Tampere, Finland**, pages 51–61.

[Sajaniemi et al., 2008] Sajaniemi, J., Kuittinen, M., and Tikansalo, T. (2008). A study of the development of students' visualizations of program state during an elementary object-oriented programming course. **Journal on Educational Resources in Computing (JERIC)**, 7(4) :1–31.

[Samurçay and Rouchier, 1985] Samurçay, R. and Rouchier, A. (1985). De «faire» à «faire faire» : planification d'actions dans la situation de programmation. **Enfance**, 38(2) :241–254.

[Sanders et al., 2015] Sanders, I., Pilkington, C., and Van Staden, W. (2015). **Errors made by students when designing Finite Automata**.

[Sands et al., 2018] Sands, D., Parker, M., Hedgeland, H., Jordan, S., and Galloway, R. (2018). Using concept inventories to measure understanding. **Higher Education Pedagogies**, 3(1) :173–182.

[Scholtz and Sanders, 2010] Scholtz, T. L. and Sanders, I. (2010). Mental models of recursion : investigating students' understanding of recursion. In **Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education**, pages 103–107.

[Scott et al., 1997] Scott, P., Asoko, H., and Driver, R. (1997). Teaching for conceptual change : A review of strategies. **Connecting Research in Physics Education with Teacher Education**.

[Seppälä et al., 2006] Seppälä, O., Malmi, L., and Korhonen, A. (2006). Observations on student misconceptions—a case study of the build–heap algorithm. **Computer Science Education**, 16(3) :241–255.

[Sharma and Shen, 2018] Sharma, R. and Shen, H. (2018). Does education culture influence factors in learning programming : A comparative study between two universities across continents. **International Journal of Learning, Teaching and Educational Research**, 17(2) :1–24.

[Shuttleworth, 2009] Shuttleworth, M. (2009). Pretest-posttest designs. **Retrieved from Explorable.com : https ://explorable.com/pretest-posttest-designs**.

[Sirkiä and Sorva, 2012] Sirkiä, T. and Sorva, J. (2012). Exploring programming misconceptions : an analysis of student mistakes in visual program simulation exercises. In **Proceedings of the 12th Koli Calling International Conference on Computing Education Research**, pages 19–28.

[Sorva, 2007] Sorva, J. (2007). Students' understandings of storing objects. In **Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88**, pages 127–135.

[Sorva, 2008] Sorva, J. (2008). The same but different students' understandings of primitive and object variables. In **Proceedings of the 8th International Conference on Computing Education Research**, pages 5–15.

[Sorva et al., 2012] Sorva, J. et al. (2012). **Visual program simulation in introductory programming education**. Aalto University.

[Spieler et al., 2019] Spieler, B., Oates-Indruchova, L., and Slany, W. (2019). Female teenagers in computer science education : understanding stereotypes, negative impacts, and positive motivation. **arXiv preprint arXiv :1903.01190**.

[Staggers and Norcio, 1993] Staggers, N. and Norcio, A. F. (1993). Mental models : concepts for human-computer interaction research. **International Journal of Man-machine studies**, 38(4) :587–605.

[Tan et al., 2009] Tan, P.-H., Ting, C.-Y., and Ling, S.-W. (2009). Learning difficulties in programming courses : undergraduates' perspective and perception. In **Computer Technology and Development, 2009. ICCTD'09. International Conference on**, volume 1, pages 42–46. IEEE.

[Taub et al., 2009] Taub, R., Ben-Ari, M., and Armoni, M. (2009). The effect of cs unplugged on middle-school students' views of cs. **ACM SIGCSE Bulletin**, 41(3) :99–103.

[Taylor and Kowalski, 2014] Taylor, A. K. and Kowalski, P. (2014). **Student misconceptions : Where do they come from and what can we do?** Society for the Teaching of Psychology.

[Taylor et al., 2014] Taylor, C., Zingaro, D., Porter, L., Webb, K. C., Lee, C. B., and Clancy, M. (2014). Computer science concept inventories : past and future. **Computer Science Education**, 24(4) :253–276.

[Teague and Lister, 2014] Teague, D. and Lister, R. (2014). Programming : reading, writing and reversing. In **Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education**, pages 285–290.

[Teague et al., 2015] Teague, D., Lister, R., and Ahadi, A. (2015). Mired in the web : Vignettes from charlotte and other novice programmers. In **ACE**, pages 165–174.

[Tedre, 2014] Tedre, M. (2014). **The science of computing : shaping a discipline**. CRC Press.

[Tedre and Denning, 2016] Tedre, M. and Denning, P. J. (2016). The long quest for computational thinking. In **Proceedings of the 16th Koli Calling International Conference on Computing Education Research**, pages 120–129.

[Tew and Guzdial, 2010] Tew, A. E. and Guzdial, M. (2010). Developing a validated assessment of fundamental cs1 concepts. In **Proceedings of the 41st ACM Technical Symposium on Computer Science Education**, pages 97–101.

[Treagust, 1988] Treagust, D. F. (1988). Development and use of diagnostic tests to evaluate students' misconceptions in science. **International Journal of Science Education**, 10(2) :159–169.

[Turkle and Papert, 1990] Turkle, S. and Papert, S. (1990). Epistemological pluralism : Styles and voices within the computer culture. **Signs : Journal of women in Culture and Society**, 16(1) :128–157.

[Veerasamy et al., 2016] Veerasamy, A. K., D'Souza, D., and Laakso, M.-J. (2016). Identifying novice student programming misconceptions and errors from summative assessments. **Journal of Educational Technology Systems**, 45(1) :50–73.

[Voogt and Roblin, 2012] Voogt, J. and Roblin, N. P. (2012). A comparative analysis of international frameworks for 21st century competences : Implications for national curriculum policies. **Journal of Curriculum Studies**, 44(3) :299–321.

[Vuorikari et al., 2016] Vuorikari, R., Punie, Y., Gomez, S. C., Van Den Brande, G., et al. (2016). Digcomp 2.0 : The digital competence framework for citizens. update phase 1 : The conceptual reference model. Technical report, Joint Research Centre (Seville site).

[Watson and Li, 2014] Watson, C. and Li, F. W. (2014). Failure rates in introductory programming revisited. In **Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education**, pages 39–44. ACM.

[Webb and Taylor, 2014] Webb, K. C. and Taylor, C. (2014). Developing a pre-and post-course concept inventory to gauge operating systems learning. In **Proceedings of the 45th ACM Technical Symposium on Computer Science Education**, pages 103–108.

[White and Sivitanides, 2002] White, G. L. and Sivitanides, M. P. (2002). A theory of the relationship between cognitive requirements of computer programming languages and programmers' cognitive characteristics. **Journal of Information Systems Education**, 13(1) :59.

[Wiedenbeck, 2005] Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. In **Proceedings of the 1st International Workshop on Computing Education Research**, pages 13–24. ACM.

[Wilson and Shrock, 2001] Wilson, B. C. and Shrock, S. (2001). Contributing to success in an introductory computer science course : a study of twelve factors. In **ACM SIGCSE Bulletin**, volume 33, pages 184–188. ACM.

[Wing, 2011] Wing, J. (2011). Research notebook : Computational thinking—what and why. **The Link Magazine**, 6 :20–23.

[Wing, 2006] Wing, J. M. (2006). Computational thinking. **Communications of the ACM**, 49(3) :33–35.

[Wing, 2014] Wing, J. M. (2014). Computational thinking benefits society. **40th Anniversary Blog of Social Issues in Computing**, page 26.

[Winslow, 1996] Winslow, L. E. (1996). Programming pedagogy—a psychological overview. **ACM SIGCSE Bulletin**, 28(3) :17–22.

[Wittie et al., 2017] Wittie, L., Kurdia, A., and Huggard, M. (2017). Developing a concept inventory for computer science 2. In **2017 IEEE Frontiers in Education Conference (FIE)**, pages 1–4. IEEE.

[Wrubel, 1959] Wrubel, M. H. (1959). **A primer of programming for digital computers**. McGraw-Hill.

[Yadav et al., 2015] Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., and Clayborn, L. (2015). Sowing the seeds : A landscape study on assessment in secondary computer science education. **Comp. Sci. Teachers Assn., NY, NY**.

[Yadav et al., 2014] Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., and Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. **ACM Transactions on Computing Education (TOCE)**, 14(1) :1–16.

[Yardi and Bruckman, 2007] Yardi, S. and Bruckman, A. (2007). What is computing? bridging the gap between teenagers' perceptions and graduate students' experiences. In **Proceedings of the Third International Workshop on Computing Education Research**, pages 39–50.