L'enseignement de savoirs informatiques pour débutants, du second cycle de la scolarité secondaire scientifique à l'université en France : une étude comparative

Claver Nijimbere

Université Paris Descartes

École doctorale 180 : Sciences Humaines et Sociales

Laboratoire Education, Discours et Apprentissages (EDA)

L'enseignement de savoirs informatiques pour débutants, du second cycle de la scolarité secondaire scientifique à l'université en France

Une étude comparative

Volume I

Par ***Claver NIJIMBERE***

Thèse de doctorat de Sciences de l'éducation

Dirigée par **Georges-Louis BARON** et **Mariam HASPEKIAN**

Présentée et soutenue publiquement le 19 juin 2015

Devant un jury composé de :

Bruillard Éric, Professeur des universités, École Normale Supérieure de Cachan, Rapporteur

Komis Vassilis, Professeur des universités, Université de Patras (Grèce), Rapporteur

Drot-Delange Béatrice, Maître de conférences, Université Blaise Pascal, Membre

Baron Georges-Louis, Professeur des universités, Université Paris Descartes, Directeur de Thèse

Haspekian Mariam, Maître de conférences, Université Paris Descartes, Co-directrice de Thèse

Résumé :

Cette thèse de doctorat interroge l'enseignement et l'apprentissage de savoirs informatiques chez des débutants en France. Elle vise à comprendre comment des débutants mettent en œuvre et construisent des savoirs informatiques. Nous avons utilisé une méthodologie qualitative de type ethnographique mobilisant des observations, des questionnaires, des entretiens semi-directifs et des analyses de textes officiels et de manuels. Nous avons aussi procédé par une approche comparative des pratiques des lycéens et des étudiants d'une part, et des enseignants, d'autre part.

Les résultats montrent des pratiques contrastées, entretenues par des tensions dans le prescrit[1]. Au lycée, en dehors d'ISN où l'informatique est rattachée aux mathématiques, les pratiques semblent influencées par quatre facteurs : la motivation (liée aux représentations), la formation continue des enseignants, la jeunesse dans le métier et l'approche pédagogique utilisée. La pratique est focalisée sur l'approche logique de l'algorithmique avec un travail au papier-crayon : la programmation est limitée, et lorsqu'elle a lieu, c'est plus avec une calculatrice mais aussi rarement avec le langage Algobox. Chez les élèves, l'algorithmique est vue comme un nouveau domaine supplémentaire introduit en mathématiques mais différent des mathématiques et de l'informatique. Les très bons élèves en algorithmique sont en général bons en mathématiques. L'ISN accueille des élèves de tous les profils, mais avec des motivations différentes, allant de la découverte de l'informatique dans un contexte formel au refuge des autres spécialités : leurs pratiques sont contrastées. C'est avec l'ISN qu'ils découvrent l'informatique au travers des formes d'enseignement variées et des problèmes de plus en plus complexes. Les pratiques des enseignants restent influencées par leur formation d'origine, avec un manque de recul chez les non-spécialistes d'informatique.

À l'Université, les apprentissages des étudiants en programmation sont avancés, comparativement à ceux des lycéens. Les programmes informatiques ainsi réalisés sont souvent sophistiqués et incorporent des éléments issus de différentes sources externes. Les notions mathématiques investies par les étudiants sont souvent modestes.

Au-delà de la formation des enseignants, la motivation occupe une place fondamentale pour adhérer à cet enseignement/apprentissage et soutenir des pratiques enseignantes comme chez les apprenants.

Mots-clés : savoir informatique, algorithmique, mathématique, programmation des robots, projet, filière scientifique, second degré secondaire, licence de l'université, enseignement, débutant, Burundi, France

1 Les textes officiels et les manuels scolaires

Title :

Teaching computer knowledge to beginners, in scientific secondary school and university in France: a comparative approach

Abstract:

This thesis questions the teaching and learning of computer knowledge to beginners in France. It aims to understand how beginners implement and build computer knowledge. We used a qualitative methodology mobilizing ethnographic observations, questionnaire, semi-structured interviews and the analysis of official instructions and textbooks. We also conducted a comparative study of the practice of both school and university students, on the one hand, and teachers, on the other hand.

Results show contrasting situations between secondary schools and university. In high school, algorithmic curricula exist within mathematic education. In this case, practice is influenced by four factors: motivation (related to representation), professional development for teachers, youth in business and pedagogical approach. The practice mainly focuses on a logical approach to algorithmic using work paper and pencil: programming is limited, and when it occurs, it is often with a calculator but rarely with the Algobox language. Among students, algorithms are perceived as a new domain in the mathematics programs, but different from both mathematics and informatics. Very good students in computing are generally good at math. Another elective course, specifically about informatics, has also been recently implemented for grade 12 students. It welcomes students of all profiles, but with different motivations, from the discovery of computers in a formal context to a shelter against other elective courses: their practices are manyfold.

Within ISN, they discover computers through various forms of education and problems of increasing complexity. Teacher practice is influenced by their original education, with a lack of experience for non specialists teachers.

At the University level, students show more advanced practice. They produce computer programs are often sophisticated and incorporate elements from various external sources. The mathematics knowledge invested by students is often modest.

In addition to teacher's training, motivation is fundamental to adhere to this teaching/learning and support practice, both for teachers and students.

Keywords : informatics knowledge, algorithmics, mathematics, educational robotics, scientific learning, secondary education, higher education, Burundi, France

Dédicace

À mon épouse et mes enfants qui ont supporté mon absence durant toute la période des études de master et de doctorat et m'ont encouragé à aller jusqu'au bout ;

À mes parents, pour l'éducation qu'ils nous ont donnée ;

À mes frères et sœurs, pour l'unité et la solidarité qui les ont toujours caractérisés ;

À mes amis et voisins, pour avoir été proches de ma petite famille en mon absence ;

Je dédie cette thèse de doctorat.

Remerciements

L’aboutissement de cette thèse de doctorat est le fruit du travail et de l'accompagnement de nombreuses personnes que j’ai la joie de remercier.

Mes vifs et sincères remerciements sont particulièrement adressés à Georges Louis Baron, directeur de cette thèse. Votre disponibilité pour accompagner ce travail, vos conseils constructifs et vos encouragements tout au long de son déroulement d’une part et votre rigueur scientifique d’autre part, ont contribué pour un meilleur achèvement.

Mes remerciements s’adressent aussi à Mariam Haspekian, codirectrice de cette thèse. C’est avec vous que j'ai commencé ma première publication. Vos conseils et votre rigueur scientifique m’ont été d’un grand atout pour ce travail et me seront fort utiles pour la suite. Merci d'avoir accepté de codiriger cette recherche.

Je suis très reconnaissant à l’égard d’Éric Bruillard. C’est vous qui m’avez inspiré le goût de la recherche en master 2 et qui m’avez accompagné jusqu’à aujourd’hui. Que cette étape franchie soit votre satisfaction. Merci aussi d’avoir accepté d’être rapporteur de cette thèse de doctorat.

Mes remerciements vont également à l’endroit de Vassilis Komis pour avoir accepté de lire ce travail. Merci d'avoir accepté de faire partie du Jury et d’évaluer ce travail.

Mes remerciements sont aussi adressés à Béatrice Drot-Delange. Vous avez très tôt accepté de faire la relecture de ma thèse. Merci pour votre disponibilité et d'avoir accepté de faire partie du Jury d'évaluation de ce travail.

Je remercie aussi toutes les personnes qui ont contribué d'une façon ou d'une autre à ce que ce travail aboutisse : les enseignants, les élèves et les étudiants qui ont été disponibles pour le recueil des données et tous ceux qui ont gentiment accepté de faire la relecture de cette thèse.

Mes remerciements s'adressent aussi aux conseillers principaux d'éducation du Lycée Gustave Eiffel, y compris Monsieur Védy qui n'est plus dans ce lycée, pour toutes les facilités mises à ma disposition durant cette thèse et, aux surveillants pour leurs encouragements. Je suis très reconnaissant aux laboratoires EDA et STEF pour leurs accompagnements dans mes premiers pas de chercheur.

Enfin, un grand remerciement est adressé au Gouvernement du Burundi pour avoir financé mes études de master et de doctorat.

Les mots me manquent pour exprimer les remerciements que vous méritez. Je préfère le dire en cette courte phrase : **Que Dieu vous bénisse**.

Sigles et abréviations

ACM : Association for Computing Machinery
AFDI : Association Francophone pour la Didactique de l'Informatique
AFSTIS : Association Française des Sciences et Technologies de l'Information et des Systèmes
AUF : Agence Universitaire de la Francophonie
B2i : Brevet Informatique et Internet
BEPC : Brevet d'Études du Premier Cycle
BTS : Brevet de Technicien Supérieur
CAPES : Certificat d'Aptitude au Professorat de l'Enseignement du Second degré
CAS : Computer At School
CDI : Centre de Documentation et d'Information
CE : Computer Engineering
CEPGL : Communauté économique des pays des Grands Lacs
C2i : Certificat Informatique et Internet
CERI : Centre pour la Recherche et l'Innovation dans l'Enseignement
CNDP : Centre National de Documentation Pédagogique
CNRS : Centre National pour la Recherche Scientifique
COMESA : Commun Market for East and South Africa
CONFEMEN : Conférence des Ministres de l'Éducation des pays ayant le français en partage
CPGE : Classes Préparatoires aux Grandes Écoles
CRI : Centre de Recherches et d'Innovation
CS : Computer Science
CSTA : Computer Science Teacher's Association
DEST : Diplôme d'Études Supérieures Techniques
DGPC : Direction Générale de la Programmation et de la Coordination
DUT : Diplôme Universitaire de Technologie
EAC : East African Community
EAO : Enseignement Assisté par Ordinateur
EIAH : Environnement Informatique d'Apprentissage Humain
ENS : École Normale Supérieure
EPI : Enseignement Public et Informatique

ESPE : Écoles Supérieures de Professorat et de l'Éducation
IDH : Indice de Développement Humain
IMO : International Mathematic Olympiad
INRIA : Institut National de Recherches en Informatique et Automatisme
INSP : Institut National de Santé Publique
IOI : International Olympiad in Informatics
IPA : Institut de Pédagogie Appliquée
ISCA : Institut Supérieur des Cadres Militaires
ISN : Informatique et Sciences du Numérique
IUT : Instituts Universitaire de Technologies
Li : Licence i (i := 1, 2 ou 3)
LSE : Langage Symbolique d'Enseignement
MEPS : Ministère de l'Enseignement Primaire et secondaire
MIT : Massachussetts Institute Technology
MIAGE : Maîtrise d'Informatique Appliquée à la Gestion
OCDE : Organisation de Coopération et de Développement Économique
PASEC : Programme d'Analyse des Systèmes Éducatifs de la CONFEMEN
PIB : Produit Intérieur Brut
PIT : Plan Informatique pour Tous
RP : Robotique Pédagogique
SI : Sciences de l'Ingénieur
SIF : Société Informatique de France
SILO : Science Informatique au Lycée : Oui !
SGEN : Syndicat Général de l'Éducation Nationale
SPECIF : Société des Personnels Enseignants et Chercheurs en Informatique de France
STEF : Sciences Technologies Éducation Formation
STIC : Sciences et Technologies de l'Information et de Communication
SVT : Sciences du Vivant et de la Terre
TI : Technologies de l'Information ou Texas Instruments
UB : Université du Burundi
ULC : Université Laval du Canada
ULB : Université Lumière de Bujumbura
ZPD : Zone Proximale de Développement

Sommaire

INTRODUCTION GÉNÉRALE

> *« Ne nous trompons pas de combat : le premier que nous autres informaticiens avons à mener est celui de l'enseignement à tous d'un savoir et d'un savoir-faire informatique suffisants »* (Maurice Nivat, 2009, p. 37)

Le présent travail s'intéresse à l'enseignement et apprentissage de l'informatique pour débutants en France au début du XXI^{e} siècle. Le sujet auquel nous avons consacré ces quatre années de thèse s'est constitué progressivement. Sa genèse se situe en 2010 lors de la rencontre avec Monsieur Éric Bruillard et Madame Françoise Tort en master 2 au laboratoire STEF de l'ENS de Cachan.

Alors que l'enseignement de l'informatique en mathématiques au lycée en France venait d'être institué depuis 2009, la volonté de découvrir les liens entre l'enseignement des mathématiques et de l'informatique et ma posture de spécialiste en enseignement des mathématiques m'ont poussé à un choix d'un sujet de mémoire portant sur une exploration de cet enseignement de l'informatique dans le cours de mathématiques.

Au cours de cette année de Master, les rencontres avec Monsieur Georges Louis Baron dans les séminaires de formation m'ont permis de découvrir son intérêt pour les technologies éducatives et les apprentissages instrumentés, ce qui a fortifié mon ambition d'approfondir le sujet antérieurement commencé dans le cadre d'une thèse de doctorat. L'institutionnalisation d'une discipline de spécialité « informatique et sciences du numérique » (ISN) au lycée en 2012 dans des lycées généraux, a changé la donne avec un nouveau un cadre contextuel. La mise en place de cette spécialité a motivé davantage mon rêve étant donné que ces enseignements - informatique en mathématiques et informatique de spécialité - sont en général assurés par des spécialistes des mathématiques.
Baron a bien accepté d'encadrer cette recherche, en relation avec Mariam Haspekian. Pour cette dernière, j'avais déjà lu ses travaux de recherche sur l'instrumentation de l'enseignement des mathématiques par le tableur, dans le cadre de cours de master.

Au début, je pensais faire cette étude seulement au niveau du lycée, où les élèves sont totalement débutants en informatique. Les difficultés d'accès au terrain ont poussé à un élargissement de cette étude à l'enseignement supérieur. Cet élargissement s'est, par la suite, avéré

intéressant pour l'étude étant donné que les étudiants de licence, eux aussi, débutent en informatique : ils n'ont pas connu une quelconque initiation disciplinaire à l'informatique dans leur scolarité obligatoire. L'existence de deux niveaux différents de l'enseignement de l'informatique pour débutants, a donc permis de conduire la recherche dans une orientation didactique de l'informatique avec l'utilisation d'une approche comparative.

L'informatique est à la fois une science autonome et un ensemble d'instruments. Actuellement, grâce à elle, des progrès spectaculaires sont réalisés dans presque tous les domaines. Pour Maurice Nivat, l'informatique est « une science pleine d'avenir ». Pourtant, plus d'un demi-siècle après sa reconnaissance (Baron & Bruillard, 1996), son enseignement comme discipline dans la scolarité de l'enseignement général n'est pas encore généralisé dans tous les pays, même ceux industrialisés, y compris la France.

Beaucoup de pays manquent de spécialistes d'informatique. Et parfois là où ils en existent, ils sont qualitativement insuffisants : ils ne possèdent pas de compétences nécessaires en informatique pour faire face aux défis de recherches actuelles. Ceci entraîne plusieurs conséquences parmi lesquelles un enseignement précoce de l'informatique. L'absence de l'enseignement de l'informatique très tôt chez des jeunes scolarisés implique le développement tardif des vocations scientifiques nécessaires voire indispensables pour des situations de recherches exigeant des connaissances scientifiques et des compétences technologiques de plus en plus avancées (Nivat, 2009).

L'informatique, devenant une composante de la vie citoyenne, le contexte technologique a finalement suscité une prise de conscience institutionnelle dans beaucoup de pays sur la nécessité d'offrir une culture informatique à tous ses citoyens, des plus jeunes aux plus âgés. Dans le cas de la France, la volonté de développer cette culture informatique pour tous, s'est aussi récemment accompagnée de la mise en place d'un enseignement de savoirs informatiques dans le cours de mathématiques en 2009 et même d'une spécialité informatique optionnelle dans des lycées généraux scientifiques en 2012 (Drot-Delange, 2012).

Cette recherche s'intéresse à la situation de cet enseignement et interroge les apprentissages des étudiants de licence en informatique et des lycéens de filière scientifique.

1. Le numérique comme élément de culture et de formation des scientifiques

La culture numérique entre de plus en plus dans la culture générale des citoyens et tend à en faire une partie composante. Cette culture repose sur l'informatique devenue une science multiforme. En effet, *« au fil des années, un consensus s'est progressivement dégagé sur l'idée que l'informatique était désormais une composante de la culture générale de l'« honnête homme » de notre époque et, à ce titre, un élément de la culture générale scolaire » (*Archambault, 2009). À force d'être partout, l'informatique est devenue, non seulement participante mais aussi incontournable dans la vie quotidienne de tout homme. Au vu de ce contexte, G. Berry (2005)[2] parle de « révolution numérique ». C'est un monde qui exige aux citoyens contemporains un mode de vie un peu particulier : l'acquisition d'un minimum de connaissances informatiques pour ne pas vivre dans « une nouvelle forme d'illettrisme »[3]. Selon Berry, cette révolution bouleverse tous les secteurs de la vie socio-économique, scientifique, culturelle… Loin de s'arrêter, elle va et ira plutôt en s'amplifiant affectant la vie des citoyens en général et celle des jeunes en particulier : *« on n'a encore rien vu, on n'en est qu'au début »*, ajoute-t-il, en soulignant l'urgente nécessité de l'enseignement de l'informatique depuis tôt jusque plus tard. Pour lui, ce n'est pas en utilisant les technologies conçues par d'autres qu'on devient créateur mais en comprenant leurs fondements et le mode technique de la science informatique.

Actuellement, l'informatique n'est pas seulement un élément de culture générale citoyenne, mais elle constitue aussi l'un des plus grands débouchés d'emplois dans le monde. Son enseignement au lycée semble donc une nécessité, tant la place de cette discipline devient de plus en plus grande dans tous les secteurs de la vie, mais aussi comme outils de compréhension du Monde[4].

Néanmoins, les systèmes éducatifs des divers pays ont eu de la peine à prendre la mesure de cette discipline-carrefour, entre technologie et science autonome, en évolution constante et

2 Il est « professeur d'informatique, titulaire de la chaire Algorithmique, machines et langages au Collège de France – première chaire dans le domaine de l'informatique –, et lauréat 2014 de la médaille d'or du CNRS » : http://www.adjectif.net/spip/spip.php ? breve642, site consulté le 20 février 2015

3 Archambault, J.P, se référant aux travaux antérieurs de M. Lévy et J. P. Jouvet, compare actuellement l'incapacité à l'usage des TIC comme une nouvelle forme d'illettrisme aussi handicapante que ne pas savoir lire et écrire

4 http://halshs.archives-ouvertes.fr/docs/00/35/90/25/HTML/a0803d.htm

dont les contours ont beaucoup varié au cours du temps (Baron, 2012). Au sein d'un même pays, diverses approches sont souvent mises en œuvre.

En France, deux courants en tension se sont opposés[5]. D'un côté, il y a un courant de ceux qui prônaient une culture numérique acquise par l'approche « outil » de l'informatique, tournée vers des utilisations de l'informatique dans les disciplines déjà existantes. De l'autre côté, ceux qui défendaient une approche « objet » de l'informatique, mettant en avant une introduction d'une discipline informatique à l'instar des autres disciplines du lycée. Berry (2008), l'un des défenseurs de ce courant, déplore le retard de l'école française dans l'accompagnement des élèves, citoyens et futurs travailleurs, dans l'acquisition de la culture informatique, actuellement incontournable :

> *« L'enseignement en collège, lycée, classes préparatoires, grandes écoles (pas en faculté) est très en retard à la fois par rapport à la réalité informatique mais aussi par rapport à la pratique des jeunes. On est toujours dans la confusion entre apprendre à utiliser l'ordinateur et comprendre les notions et concepts de l'informatique ».*

Après beaucoup de plaidoyers en faveur d'un enseignement de l'informatique au lycée, des réformes de programmes d'enseignement ont été opérées. La première étape a été l'introduction d'éléments d'algorithmique dans le programme de mathématiques en classe de seconde dans les sections scientifiques depuis l'année scolaire 2009. La deuxième étape a été l'ouverture de l'enseignement du numérique au lycée après la classe de seconde (Cabane, 2012). D'abord, il s'agit de la spécialité SIN (Système d'Information et Numérique) de la série STI2D (Sciences et Technologies de l'Industrie et du Développement Durable) et d'autre part, de la spécialité ISN (Informatique et Sciences du Numérique) pour les élèves de la classe de terminale. Des attentes de ces nouveaux enseignements au niveau du lycée sont grandes : ces spécialités ont pour but de répondre au manque de bacheliers qui vont s'orienter vers les études scientifiques de l'enseignement supérieur telles que dans les domaines des sciences et technologies du numérique, de l'information et de la communication, précise Cabane (ibid.). Bien que ces enseignements, déjà en application, n'offrent pas une culture numérique améliorée à tous les élèves avant leur obtention du baccalauréat, une nette impulsion est attendue dans le cadre de l'enseignement du numérique et de l'informatique en France.

5 http://www.letudiant.fr/educpros/entretiens/jean-pierre-archambault-president-d-enseignement-public-et-informatique-cest-un-changement.html, site consulté le 27 octobre 2014

Cette thèse de doctorat s'inscrit dans ce contexte et s'intéresse à l'enseignement/apprentissage de l'informatique chez les débutants[6]. Par une approche comparative, l'étude vise à rendre compte des modalités d'appropriations des savoirs informatiques de base chez les débutants. Elle s'intéresse aux pratiques d'apprentissages chez les lycéens scientifiques (de la classe de seconde à la terminale) et les étudiants de licence informatique en France. Il est intéressant de mieux comprendre ce qui se passe aussi à l'université parce que les étudiants de licence, bien que non débutants complets en informatique, sont proches des lycéens en filière scientifique. Tout comme ces derniers, les étudiants n'ont connu aucune initiation scolaire à l'informatique avant l'entrée à l'université.

2. Quelle informatique et comment l'enseigner aux débutants de niveau lycée ou supérieur ?

Michel Beaudouin-Lafon (2010, p. 52) souligne un manque d'unanimité sur la nature de l'informatique à enseigner aux débutants dans la scolarité obligatoire.

> *« La réflexion sur ce qu'il faudrait apprendre de l'informatique dans les collèges et lycées passe d'abord par une réflexion de ce qu'est l'informatique en tant que science. Je sais d'expérience que les idées évoquées ci-dessus sont loin d'être partagées par l'ensemble de la communauté informatique, notamment francophone, sans doute à cause du poids important de la filiation mathématique de l'informatique dans notre pays, que l'on voit reflété dans les propositions telles que celles de Gérard Berry ou Gilles Dowek »*

Aujourd'hui, l'informatique est une science. Elle comprend des terminologies et des concepts divers. Sa construction, comme d'ailleurs celle de la discipline scolaire informatique, a été longue, progressive et plurielle (Baron & Bruillard, 2001) : la science informatique était d'abord conçue comme un ensemble de méthodes et de techniques, unifiées par l'ordinateur. Quant à son enseignement, il a toujours posé de difficultés, prenant des formes et des approches diverses. Jacques Arsac qui plaide pour une introduction de la science informatique dans l'enseignement général, trouve que, comme toute discipline, *« enseigner l'in-*

6 Nous considérons comme débutants en informatique les lycéens qui sont en situation d'initiation à l'informatique. Ils n'ont jamais connu dans leur cursus de scolarité, aucune formation discipline scolaire à l'informatique.

formatique, c'est faire que l'élève maîtrise ses concepts, et puisse les identifier dans une application concrète » (Arsac, 1993).

Pour Félix Paoletti, l'enseignement de l'informatique doit couvrir trois dimensions (Paoletti, 1993) : une dimension scientifique, une dimension technique et une dimension sociétale. Ces dimensions de l'informatique ont un caractère universel. Récemment, Steve Furber[7] (Furber, 2012) distingue et développe ces dimensions en trois formes d'informatique proposées pour enseignement au Royaume Uni : la science informatique, la technologie de l'information et la culture numérique. La première forme – science informatique – recouvre à elle seule, selon les pays, différentes terminologies (Gander et al., 2013) : « *Computer science »* (CS), comme la terminologie dominante aux États unis ; « *Computer Science* » et « *Informatics* » deux terminologies plus fréquentes en Europe et, *datalogie* en Scandinavie (Arsac, 1989) . La deuxième forme est la technologie de l'information – « Information technology » (IT) – qui consiste en un enseignement spécifique des utilisations des technologies. La troisième forme est la culture numérique – « digital literacy ». Située entre les deux premières formes, elle consiste en l'utilisation des logiciels pour enseigner d'autres disciplines.

La science informatique, comme beaucoup d'autres sciences tant scientifiques qu'humaines et sociales, a ses caractéristiques. L'association britannique CAS (« Computer At School ») la définit comme une discipline avec les cinq caractéristiques suivantes : (1) un corps de connaissances qui comprend un cadre théorique où les idées et les concepts sont circonscrits, (2) un ensemble rigoureux de techniques et de méthodes, (3) une façon de raisonner et de fonctionner, donc un courant paradigmatique, (4) un ensemble stable de concepts et, (5) une indépendance aux technologies spécifiques.

Comme déjà précisé, les formes « CS » et « IT » de la science informatique sont à la fois complémentaires et différentes[8]. Les concepts spécifiques de chacune d'elle permettent de les caractériser et les distinguer. Définie comme une discipline rigoureuse, la science informatique (« CS ») se compose de deux types de concepts stables (Furber, 2012) :

- des concepts qui sont au cœur de la méthode de la science informatique : programme, algorithme, structures de données, architecture et communication ;

7 Il est professeur de génie informatique à l'École des sciences informatiques à l'Université de Manchester : http://royalsociety.org/uploadedFiles/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf, Site consulté le 21 janvier 2014

8 http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf, Site consulté le 21 janvier 2014

- des concepts qui sont à la base de la pensée informatique et qui interviennent lors de la résolution des problèmes : la modélisation, la décomposition, la généralisation et la conception, l'écriture, le test, l'explication et le débogage des codes informatiques.

Définie comme une étude des applications des systèmes informatiques et des usages des logiciels pré-existants, la technologie de l'information, quant à elle, est une partie de la science informatique avec ses caractéristiques propres.

Un enseignement de l'informatique au lycée est une nécessité justifiée par la place et l'intérêt de cette science dans tous les secteurs de la vie. Pour Gilles Dowek (2005), cette discipline aurait pour but principal l' *« enseignement d'un langage de programmation et d'algorithmes de base, avec l'objectif de savoir écrire un programme au moment de passer son bac.»*. Dans cet enseignement, l'apprentissage de l'algorithmique et la programmation serait prioritaire pour leur apport important pour les lycéennes et lycéens dans leur développement intellectuel. Certaines des potentialités permises sont soulignées : une approche de projets, une mise en application des savoirs acquis, un pont entre le langage et l'action et un intérêt pour la rigueur scientifique.

Dans le point suivant, nous abordons la problématique à l'origine de la recherche et qui sera plus développée dans la deuxième partie de la thèse.

3. Contexte et questionnement

Aujourd'hui, un enseignement de notions informatiques est entré au lycée scientifique en France. Mais, tout comme les notions algorithmiques ont été introduites dans le cours de mathématiques, la discipline informatique est en général confiée aux non-spécialistes de l'informatique et particulièrement aux enseignants des mathématiques. Deux problèmes se posent. Le premier problème est que les élèves à qui ces programmes d'enseignement sont destinés n'ont jamais connu une quelconque initiation scolaire à l'informatique. Cela interroge leurs compétences à s'approprier un tel enseignement. Le deuxième problème est relatif aux enseignants à qui cet enseignement est confié. Étant des spécialistes des mathématiques, l'institution suppose que les enseignants de mathématiques sont capables d'enseigner les notions informatiques en général et l'algorithmique en particulier. Ce qui n'est pas toujours vrai car, malgré leur rapprochement, les deux disciplines – mathématiques et informatique- ont les objectifs qui restent assez différents.

Comment des débutants en informatique construisent et mettent en application ces savoirs informatiques dans un tel contexte ?

La situation semble évoluer avec l'ISN : des enseignants volontaires, issus des autres spécialités, ont été choisis pour conduire cet enseignement. Néanmoins, attribuer un enseignement d'une discipline informatique aux non-spécialistes par la simple raison qu'ils sont volontaires et ont une certaine familiarité avec l'informatique, ne serait-il pas une utopie ? Ces critères suffisent-ils pour garantir un succès de son enseignement ?

La situation semble similaire en début d'université. Jusqu'à présent, les étudiants de licence qui commencent l'université, n'ont jamais connu d'initiation scolaire à l'informatique. À la seule différence qu'en licence informatique, leurs professeurs sont des spécialistes d'informatique, les étudiants en début d'université, sont dans les mêmes conditions que les lycéens du point de vue de leurs bagages informatiques. Ils ont été donc tous considérés comme débutants en informatique.

Cette similarité de ces deux publics du point de vue de l'informatique nous a interpellé pour conduire notre étude de façon comparative sur les deux niveaux : lycée scientifique général et licence informatique. Il est question, dans la recherche, de comprendre comment les élèves et les étudiants s'approprient certains savoirs jugés fondamentaux pour les débutants en informatique, mais aussi comment les enseignants se sont approprié l'enseignement de l'informatique.

Nous formulons deux questions générales suivantes. La première concerne les apprenants. Elle interroge comment les élèves et les étudiants, débutants en informatique, construisent les savoirs informatiques lors de leurs apprentissages ?

La deuxième question concerne les enseignants. Elle interroge comment les enseignants, non spécialistes de l'informatique, prennent en compte les savoirs informatiques dans leurs pratiques professionnelles de classe ?

Les deux questions principales sont accompagnées par cinq questions spécifiques.

Quels sont les concepts informatiques jugés incontournables par la littérature de recherche pour une initiation informatique au lycée ? Quels sont les types d'activités proposées dans le prescrit (programmes et manuels) au lycée et quels savoirs informatiques ces activités visent à construire chez élèves ? Quelles sont les tâches données à l'élève en classe par leurs enseignants et quelles sont les motivations de tels choix ? Quelles sont les compétences effectives

des élèves et étudiants à construire un algorithme, sa programmation et son exécution sur ordinateur ? Quelles difficultés les élèves, les étudiants et leurs enseignants connaissent-ils ; quelles grandes erreurs les élèves et les étudiants commettent-ils ? Quelles sont les stratégies utilisées pour les éviter ?

La thèse est structurée en deux parties. La première partie comprendra quatre chapitres. Dans le premier chapitre, nous présenterons notre cadre conceptuel pour rassembler des concepts sur lesquels va se fonder notre réflexion dans le développement de la thèse. Le deuxième chapitre parlera de la problématique qui a motivé notre recherche et du questionnement qui accompagne cette problématique. Dans le même chapitre, il sera aussi précisé et développé la méthodologie utilisée pour mener cette étude.

Après cela, nous retracerons les origines et les étapes qui ont marqué le développement historique de l'informatique éducative et préciserons les différents choix qui ont été effectués et caractérisé le contexte éducatif français. Cela fera l'objet du troisième chapitre. Le quatrième chapitre consistera à réaliser un état de la question, relativement à des recherches en didactique de l'informatique. Cet état de la question nous conduira à pouvoir effectuer une classification thématique des recherches. Ceci nous permettra de comprendre mieux le contexte actuel dans lequel s'inscrit la thèse et de cerner les concepts informatiques jugés fondamentaux par la littérature de recherche pour des débutants en informatique. Ce recueil de concepts nous permettra de faire des choix de ceux qui sont incontournables et sur lesquels pourra porter notre recherche.

La deuxième partie sera consacrée à la présentation des résultats et leur discussion générale à la lumière des hypothèses de recherche qui seront formulées. Elle sera composée de cinq chapitres (V, VI, VII, VIII et IX). Le cinquième chapitre mettra présentera les résultats issus de l'analyse des prescrits (textes officiels et manuels scolaires) qui fondent les nouveaux enseignements de l'informatique en France et ont été mis à la disposition des enseignants. Le sixième chapitre portera sur l'enseignement de l'informatique dans le cours de mathématiques au lycée. Il présentera respectivement et comparativement les pratiques informatiques des enseignants de mathématiques et les apprentissages en informatiques de leurs élèves.

Le septième chapitre s'intéressera à la situation de l'enseignement de la spécialité informatique et sciences du numérique (ISN) en Terminale en France. Il abordera de façon comparative les pratiques des professeurs d'une part et les apprentissages des élèves d'autre part. Le

huitième chapitre présentera les résultats de l'apprentissage de l'informatique par la programmation des robots en licence chez les étudiants. Le chapitre IX qui clôture la partie et par conséquent la thèse, donnera la discussion générale des résultats et la présentation des perspectives de recherche.

Il convient de noter que cette thèse de doctorat reprend trois de nos articles rédigés dans le cadre de cette recherche et déjà publiés ailleurs. Ils seront signalés dans la suite le moment venu.

PREMIÈRE PARTIE : L'INFORMATIQUE ET SON ENSEIGNEMENT CHEZ LES DÉBUTANTS

Cette partie développe l'enseignement et l'apprentissage de l'informatique dans des contextes théoriques d'initiation. Elle sera structurée en quatre chapitres. Le premier chapitre précise et développe les cadres conceptuels qui vont guider notre réflexion. Le deuxième chapitre esquissera la problématique et le questionnement à l'origine de la thèse et aussi la méthodologie qui sera utilisée pour mener cette étude. Le troisième chapitre tracera le parcours historique de l'informatique en éducation en France. Le dernier et quatrième chapitre dressera une revue de littérature portant sur la didactique de l'informatique.

Chapitre I CADRE CONCEPTUEL

Ce chapitre sera développé en six points : les jeunes face au numérique, le choix d'une approche systémique, le concept de l'activité, les différentes théories de l'activité, les systèmes d'activités d'Engeström et le cadre conceptuel proprement dit de la thèse adapté au contexte de projet.

1. Les jeunes face au numérique : quelles appropriations pour quels usages ?

1.1. Problématique de la notion d'appropriation

Les difficultés de l'appropriation des savoirs informatiques entrent dans la problématique générale de l'appropriation d'un savoir. Cette appropriation suscite deux types de questionnement (Blondel & Tort, 2007). D'une part, il y a un questionnement qui concerne les conditions sociales de la circulation des compétences et, d'autre part, un autre portant sur les manifestations de ces compétences dans des interactions, lesquelles interactions peuvent avoir lieu entre les individus eux ou entre les individus et les systèmes techniques. Selon Blondel et Tort, ces questionnements convoquent des mécanismes liés à l'entrée dans la culture juvénile, laquelle s'accompagne d'« *un double processus d'émancipation de l'univers parental et de l'affiliation à des bandes de pairs ; mais aussi à une création d'une autonomie relationnelle et culturelle »*. D'autre part, parallèlement à la transmission familiale d'un capital culturel telle que mise en évidence par Bourdieu, ils questionnent des *« modes de circulations de compétences non seulement au sein des générations mais aussi au sein de leurs familles »*.

La manifestation des compétences techniques est l'autre questionnement suscité par le concept de l'appropriation. Comme ils le montrent, ce questionnement qui engage les schèmes d'usages ou de perception, met en évidence des dimensions individuelles et collectives relativement aux processus de genèses instrumentales dans le cas particulier de l'appropriation d'un outil informatique (Rabardel, 1995).

Pour appréhender les problématiques qui accompagnent le concept d'appropriation, Michelle Artigue (2010) les aborde sous deux facettes. La première facette est sous-jacente aux pro-

blèmes de l'appropriation des outils informatisés et plus particulièrement des ordinateurs utilisés dans les processus d'apprentissage. Dans « *Le numérique dans l'enseignement en France et à l'étranger : les cas des mathématiques »*, Artigue souligne que les travaux sur l'approche instrumentale révèlent des débuts de la fin de l'option informatique des années 1990. Cette dernière *« a d'abord connu une dominance quasi-exclusive de la vision outil avec ses limitations induites dans les apprentissages »*. La deuxième facette est liée à des problèmes cognitifs, logiques et de rigueur qui peuvent trouvent davantage leur place dans les exigences de la construction de la méthode à suivre dans la construction d'un algorithme et dans sa programmation sur un ordinateur. Ces problèmes ont contribué à la suppression de cette option.

1.2. Des rapports à l'informatique limités chez les « digital natives »

a. Des pratiques informationnelles absentes

Les générations actuelles, parmi eux des jeunes collégiens et lycéens, vivent et baignent naturellement dans le numérique. Ils sont souvent présentés comme des « digital natives », pour qui les outils numériques n'ont aucun secret (Tort & Dagiène, 2012). Baron & Bruillard qui ont décortiqué les terminologies de *« digital natives »* et *« digital immigrants »* de Prensky (2001a, b), semblent mitigés quant à l'idée de « discontinuité générationnelle » prônée par leur auteur. Selon eux, *« les élèves d'aujourd'hui ne seraient pas du tout ceux des générations précédentes, ils auraient des styles cognitifs et des modes d'apprentissages différents de ceux qu'on pouvait observer précédemment »*. Pour eux, une situation pareille aurait des conséquences dramatiques pour les systèmes éducatifs. Loin d'être des *« digital natives »* comme le préconise Prensky, ils sont, selon Baron & Bruillard (2008), des *« digital naïves »* et, donc des *« naïfs à former »* : *« leurs utilisations des technologies sont fréquentes, mais dans un spectre très limité et avec un degré d'autonomie très relatif »*. Leurs explications s'appuient sur un exemple d'un logiciel pouvant être avantageusement utilisé pour les apprentissages mais rarement utilisés : :

> « *Avec un logiciel comme Google Earth, ils peuvent accéder au monde entier. Mais s'ils n'y sont pas incités, vont-ils fréquemment regarder au-delà de leur maison ou de leur quartier pour découvrir d'autres horizons ? Vont-ils intégrer le recours à cet outil dans des apprentissages géographiques ? Les compétences qu'ils développent trouvent leur source dans le milieu familial, le groupe de pairs et à l'école »*

Ces travaux allaient dans la ligne de ceux précédents d'Éric Bruillard. Ce dernier avait récemment étudié les pratiques des jeunes et avait trouvé que les technologies préférées et utilisées chez les jeunes sont ceux en rapport avec le jeu ou la communication, conçues pour l'immédiateté et la satisfaction directe de leurs besoins (Bruillard, 2006). Contrairement à celles de communication, les technologies comme le tableur, utilisé depuis longtemps en contexte scolaire, sont faiblement utilisés par les élèves, beaucoup rares au collège qu'au lycée (Blondel & Bruillard, 2007). Après, il s'est développé une informatique grand public où prédomine l'aspect communication, essentiellement caractérisée par des formes de réseautage social (Bruillard, 2012). Selon Dauphin, la primauté des pratiques communications ou de jeu au détriment des pratiques informationnelles est révélatrice des questions de lacunes notamment celles jouant sur la notion de temporalité (Dauphin, 2012) : si les pratiques de communication reposent sur l'immédiateté avec un caractère essentiellement ludique, les pratiques informationnelles demandent du courage pour surmonter des échecs et des frustrations auxquelles on fait face, et donc un temps long pour être maîtrisées. De plus, dans ce cadre communicationnel, dans l'usage juvénile des médias, ce caractère immédiat est renforcé par l'essence même de l'Internet qui est marqué par la suppression ou l'anéantissement des frontières tant spatiales que temporaires. Éric Bruillard (2012) donne deux raisons pour expliquer la migration des jeunes dans les réseaux de communications parascolaires.

La première raison est l'espace entretenu entre le temps scolaire et le temps privé, rendu possible par la décentralisation de l'État et qui a ouvert une large marge de liberté chez les élèves : c'est cette large liberté qui, aujourd'hui, est à l'origine de la création des réseaux qui favorisent le parascolaire et le développement des apprentissages à domiciles : « home schooling ». La deuxième raison est le discours marketing trompeur qui accompagne les outils informatiques actuels. Les jeunes sont actuellement sous la domination des technologies : ils sont dominés, commandés et rendus esclaves par des machines alors que normalement ça devrait être l'inverse (Bruillard, 2012). L'émancipation est attendue de la formation à l'informatique pour être capable et compétent à commander soi-même ces machines.

b. Des pratiques numériques plus limitées à la sociabilité…

Depuis une vingtaine d'années, les jeunes « naviguent » quotidiennement dans un univers de dispositifs et d'instruments informatisés. Chez eux, les technologies numériques semblent « naturelles ». de telle sorte que même « le système éducatif tend à considérer que les élèves

partagent une culture numérique » (Drot-Delange, 2012). Mais, comme le précise cette auteure, les recherches révèlent les limites de cette affirmation.

Depuis le début du 21e siècle, des recherches se sont intéressées sur des questions relatives aux modes d'appropriation de ces outils chez les jeunes, à leurs compétences et aux défis auxquels l'école doit faire face dans ce monde en plein boom technologique (Baron & Bruillard, 2008). Aujourd'hui, ces auteurs parlent, chez les jeunes, de *banalisation* rapide des outils qui peuvent avantageusement être utilisés à des fins d'apprentissage que ce soit dans le cadre formel, donc le cadre scolaire et d'autres dans le cadre informel, hors de l'école. À cet effet, ils distinguent deux catégories d'outils technologiques : une catégorie de ceux qui sont déjà utilisés et stabilisés dans le contexte scolaire tels que les logiciels de traitements de textes, les tableurs… et une autre catégorie de ceux qui n'ont pas bénéficié d'une entrée favorable dans le système éducatif, notamment les logiciels de jeux et ceux intervenant dans les systèmes de communication tels que des blogs, des forums, des messageries instantanées…

Connaître le lieu d'appropriation et comprendre la culture des numérique de jeunes nécessitent la contextualisation de leurs usages des TIC. La tranche d'ages des adolescents, entre 12 et 17 ans, est beaucoup plus connectée : l'autonomie et l'individualisation sont leurs primordiaux mobiles de cette connectivité forte et presque permanente (Dauphin, 2012). Les études faites en matière de l'appropriation TIC chez les élèves dans la scolarité obligatoire révèlent divers lieux de construction et de structuration forte de leurs usages, avec une légère importance de l'école. Cédric Fluckiger (2007) a fait cette étude de l'appropriation des TIC dans le cas général chez les collégiens. À la même période, dans le cadre d'un projet visant à questionner des usages des TIC en éducation et en formation, Blondel & Tort (2007) se sont intéressés à connaître les usages réels chez les jeunes, non pas dans le cas général des TIC, mais dans le cas particulier du tableur au sein d'un projet Didatab (Didactique du Tableur).

Les lieux d'appropriations peuvent être classifiés selon deux types différents : une classification topologique et une classification en fonction des relations sociales. En fonction de la topologie des usages, les recherches reconnaissent et distinguent deux lieux fondamentaux de leur appropriation : Toutes ces recherches distinguent deux types de lieux des appropriations des TIC chez les jeunes : la sphère familiale et la sphère scolaire. Plutôt que de considérer cette classification sous forme de typologie, ces recherches privilégient, en vue d'une bonne réinterprétation des usages, la classification selon les relations sociales dans lesquels ces

usages se trouvent sont « enchassés ». Dans cette catégorie des usages, trois principaux univers d'appropriations ont été mis en évidence (Blondel & Tort, 2007 ; Fluckiger, 2007 ; 2008) : une appropriation générationnelle, liée à la construction identitaire du jeune qui lui permet son inscription dans sa culture, dans la sphère générationnelle et, est donc une ouverture vers son émancipation progressive ; une appropriation familiale, liée et influencée par la distribution des compétences techniques réparties au sein de la famille et au mode et aux conditions de vie de sa famille et, enfin ; une appropriation scolaire. Cette appropriation est en rupture avec les deux univers précédents pour deux raisons : d'une part, le caractère intentionnel des enseignements et, d'autre part, la contrainte qu'impose l'école sur les usages de ces TIC chez les élèves.

c. *Des pratiques en rupture avec l'école !*

Dans le cadre de l'enseignement scolaire en France, le mot « informatique » est rarement associé à des situations d'enseignement ou d'apprentissage et bien plus souvent à l'usage des ordinateurs et des logiciels associés (Cabane, 2012). En revanche, à la place de ce mot informatique, il s'est généralisé les acronymes TIC[9] et TICE[10] lesquels sont fréquemment entendus dans le milieu scolaire français. Si, leurs usages sont théoriquement enseignés et évalués dans le cadre du Brevet Informatique et Internet (B2i) au niveau de la scolarité obligatoire (école élémentaire, collège et lycée), au niveau lycée, en réalité, peu d'établissements se sont réellement engagés dans leur validation et, ont plutôt considéré le B2i comme facultatif. Comme beaucoup d'autres auteurs qui se sont intéressés au B2i et à sa validation dans les lycées (Archambault, 2009, 2010, 2011 ; Bruillard, 2009 ; Tort, 2009, etc.), Robert Cabane (2012, p.10) parle de l'échec du B2i en France. Et face à ce problème, il appelle à une solution de rechange pour donner aux élèves une culture informatique dont ils ont le plus besoin actuellement :

> *« (...) l'échec de la généralisation du B2i Lycée pose une question qui appellera tôt ou tard une réponse dans la mesure où il faudra de plus en plus préparer les élèves à une société dont les acteurs seront les détenteurs d'une solide « culture » numérique ».*

Dauphin (2012) s'est intéressé à l'étude des pratiques culturelles numériques et aux compétences utilisées chez les jeunes dans leur appropriation des TIC. Il souligne une complexité

9 Ce terme qui désigne les technologies de l'information et de la communication

10 Moins claire que le premier, cet acronyme désigne les TIC utilisées ***dans*** et ***pour*** l'acte éducatif (Cabane, 2012)

des acquisitions des compétences numériques, appelées « e-skills » ou « e-competences » comme tous les autres outils technologiques. Dauphin souligne les natures « profane », « ludique » et « communicationnelle » qui caractérisent des pratiques numériques chez les jeunes. Ces pratiques affichent des différences liées aux lieux de leur appropriation qui semblent caractériser et distinguer les compétences chez les jeunes : capital social, culturel et scolaire. Ce caractère « profane » des compétences des jeunes, initialement évoqué par Fluckiger (2008) limite largement les jeunes dans leurs utilisations dans leurs apprentissages scolaires : elles révèlent un certain nombre de lacunes pour une utilisation globale des TIC. Bréda (2010) qualifie les pratiques juvéniles comme des *« pratiques construites sur du sable »*.

Ces limites de l'implication des usages « profanes » dans les apprentissages scolaires semblent être justifiées par les contraintes scolaires (Dauphin, 2012) :

> *« l'école, qui est le lieu pour former à un « usage citoyen » aux technologies de l'information et de la communication, semble prescrire des usages éloignés de ceux vécus comme « naturel » et routines mises en avant par les jeunes générations ».*

À côté de ces contraintes scolaires, d'autres études ont montré une sous-utilisation des TIC au sein des classes. Si on peut imaginer que cela est dû au manque d'équipements informatiques, cela n'est pas vrai. Baron et Bruillard soulignent que les équipements sont relativement bien dans les écoles. En effet, en France, l'informatique en éducation se trouve être une zone de jeu possible dans un contexte fermement régulé par le niveau national et, depuis une dizaine d'années, un certain nombre de départements avaient déjà lancé des opérations visant à équiper les collégiens avec des ordinateurs (Baron & Bruillard, 2008).

Baron et Bruillard soulignent la faible utilisation des ordinateurs et particulièrement une quasi – absence de l'utilisation de l'Internet à des fins de l'apprentissage et de la formation : ils qualifient d'« anecdotique » l'accès quotidien aux ordinateurs et à l'Internet en classe pour les 12-17 ans. Une année après, une étude de type ethnographique menée dans un collège de banlieue de Paris par Fluckiger (2008) semble donner de légèrement amélioration : « s'ils [les collégiens] utilisent beaucoup les ordinateurs et l'Internet, ils ont un spectre d'utilisation limité et peu de maîtrise ». Les disparités des pratiques scolaires et extrascolaires des jeunes élèves n'ont pas seulement été localisées en France. La situation est la même au niveau international comme l'affirme l'étude comparative Médiapro faite au niveau européen en 2006 auquel (Baron & Bruillard, 2008, p. 8) font référence.

1.3. Des compétences moins transférables en apprentissages scolaires

Dans un contexte de prolifération des outils numériques, les études de l'INSEE[11] d'il y a une quinzaine d'années montraient déjà que l'informatique est considérée comme une technique bien assimilée par les jeunes générations par rapport aux plus âgées (Rouquette, 1999). S'il est admis que certaines compétences sont acquises au cours des usages des outils numériques chez les jeunes en général et les scolarisés en particulier, il y a lieu à s'interroger sur le caractère de leur transférabilité dans divers contextes, et particulièrement les contextes d'apprentissage scolaires. Les raisons fondatrices de cette interrogation sont diverses et ont été données par certains auteurs. Fluckiger (2008) évoque le caractère extra-scolaire des apprentissages et compétences des jeunes acquises dans leurs usages des technologies. Selon lui, la culture numérique des élèves est généralement accès sur des pratiques personnelles et intergénérationnelles. Ces dernières pratiques en usage des TIC contribuent à une construction de soi et d'automatisation de leurs pratiques culturelles et communicationnelles.

Acquise en dehors de l'école, cette culture numérique des élèves est généralement accès sur des pratiques personnelles imprimant les usages des TIC dans les processus de construction de soi et d'automatisation des pratiques culturelles et de communication, ajoute Fluckiger. Archambault (2011) appelle à une considération attentive des pratiques des élèves. Selon lui, de telles compétences acquises dans ces conditions, hors de l'école, restent limitées aux usages quotidiens et sont donc fortement à relativiser : acquis dans ce contexte, ce sont des savoir-faire très superficiels et difficilement transférables d'une situation à une autre et, particulièrement dans des contextes scolaires d'apprentissage qui, en général, sont plus exigeants (Fluckiger & Bruillard, 2008).

Tous ces auteurs expliquent un décalage dans la transférabilité des compétences des élèves par le fait que leurs usages reposent sur *« des savoir-faire limités, peu explicables et qui laissent peu de place à une conceptualisation »*. Ce contexte laisse voir le manque ou la très faible verbalisation chez les élèves de ce qu'ils font lorsqu'il s'agit de décrire tant leurs pratiques que les problèmes rencontrés au cours des usages des technologies :

> *« Ce déficit de compréhension et de conceptualisation va de pair avec la très faible verbalisation des pratiques : les élèves ne savent le plus souvent nommer ni leurs actions, ni les objets qu'ils manipulent pourtant aisément »*

11 Institut National de la Statistique et des études économiques

Dans son article *« L'école à l'épreuve de la culture numérique des élèves »*, Fluckiger (2008, p. 1) montre comment, suite à leurs compétences limitées et souvent localisées, l'élève n'a pas une conceptualisation et une verbalisation de ses usages des TIC. Il a des difficultés à intégrer ses savoir-faire techniques dans des situations d'apprentissage en contexte scolaire. Baron et Bruillard expliquent ces limites par des apprentissages superficiels chez les jeunes. Selon eux, s'il est clair que les élèves utilisent souvent l'ordinateur, la simple utilisation, même fréquente, consistant en la simple manipulation des interfaces informatiques, ne semble pas suffire à la construction d'une réelle compréhension du fonctionnement interne de l'ordinateur (Baron & Bruillard, 2001).

De plus, ce n'est pas parce que l'élève est normalement « utilisateur » de l'ordinateur qu'il pourra facilement l'utiliser pour lui faire faire résoudre son problème. C'est dans ce sens qu'Artigue (2010, p. 18), même si elle reconnaît des usages plus ou moins avancés des technologies numériques chez les élèves, déplore l'absence, pourtant indispensable, d'une orientation dans leurs usages, dans le sens de la production.

Maurice Nivat (2005, p. 3) donne plus de lumière sur l'informatique actuellement utilisée par les élèves de collège. Il les compare à un chauffeur qui, facilement, peut très bien conduire sa voiture sans savoir aucune notion de la mécanique et les processus en cours lors de la conduite : *« On ne dira, ni comment marchent les ordinateurs ni comment fonctionnent les logiciels, avant la seconde où nous préconisons d'introduire un enseignement de base de l'informatique, distinct des autres enseignements, notamment de celui des mathématiques »*

1.4. Des savoirs informatiques de base absents chez les élèves

La littérature de recherche souligne la nécessité d'interrelation entre les savoirs théoriques et les savoir-faire. Duchâteau (1994), dans sa communication au quatrième colloque francophone sur la didactique de l'informatique, posait la question suivante qui est jusqu'aujourd'hui d'actualité : *« Faut-il enseigner l'informatique à ses utilisateurs ? ».*

Dans sa communication, il soulignait la tendance de l'enseignement des outils informatiques, confrontée à une double boucle difficile à gérer : mise à la seconde zone des savoirs théoriques alors qu'ils sont indispensables voire incontournables pour l'appropriation de ces outils. De plus, c'est au cours de la manipulation de ces outils informatiques que les savoirs théoriques acquièrent effectivement leur sens et prennent corps. Selon lui, savoirs et savoir-faire sont interreliés et cette interrelation est indispensable pour une meilleure acquisition de l'informatique comme cette illustration l'indique :

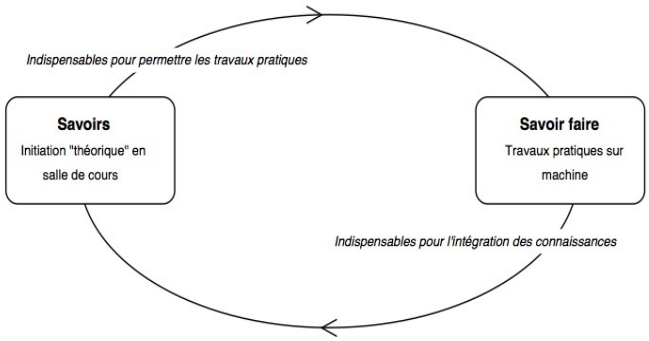


Illustration I.1: Relation en boucle entre savoirs et savoir-faire en informatique (Duchâteau, 1994)

En France, Françoise Tort souligne des situations contrastées à propos de l'enseignement de l'informatique dans la scolarité obligatoire, entre le collège et le lycée. Selon elle, si le cours de technologie au collège permet aux élèves de construire certains savoirs et savoir-faire en informatique, la situation change avec le lycée général où les élèves n'ont aucune occasion d'acquérir des savoirs informatiques. Le contraste est souligné au niveau même du lycée, entre les lycées généraux et les lycées technologiques (Tort, 2009, p. 214) :

> *« Au lycée, on observe des situations assez contrastées. La voie générale est plutôt marquée par la vision de l'informatique outil, notamment dans les séries scientifiques et de sciences économique et social. L'informatique n'est jamais associée à des notions fondamentales. Dans la voie technologique, l'informatique est plus présente dans les enseignements. Les rénovations des programmes des STG et ST2S, renforçant les enseignements théoriques, réduisent la place accordée aux formations aux logiciels. ».*

Si la volonté d'offrir des connaissances en informatiques aux jeunes a été officialisée, à partir des années 2000, Françoise Tort (2009) montre que, par l'instauration du B2i au collège et du C2i au lycée, ce n'est que l'acquisition des compétences qui est visée chez les élèves à la fin de la scolarité obligatoire.

Pourtant, l'informatique représente à la fois un « domaine de la connaissance » et des « objets techniques » fabriqués à l'aide de ces connaissances. Monique Grandbastien (2012) souligne une confusion relevant de ce que c'est l'informatique chez le grand public qui l'assimile souvent à ce qu'on appelle la « bureautique » ou les TIC », des termes non synonymes. Elle déplore le fait que l'Éducation nationale continue à entretenir cette confusion en proposant le B2i au collège et au lycée et le C2i à l'université, parce que dans les deux cas, ce ne sont que des compétences concernées dans l'utilisation du poste informatique et de cer-

tains progiciels, mais *« sans que des concepts fondamentaux de la science informatique y soient explicitement abordés »*.

De plus, Archambault (2009) clarifie le paradoxe du B2i : celui-ci suppose implicitement un apport de connaissances, alors que nulle part n'est précisé où les trouver, dans quelles disciplines. Il s'interroge et trouve paradoxal comment est-il possible d'organiser des apprentissages progressifs sur une durée lorsque les compétences recherchées sont formulées de manière très générale. Et même au niveau de collège, si l'on pense que la situation est relativement mieux, elle est aussi moins rose. Se référant aux travaux antérieurs de Rak (2008), Tort (2009) relève une place beaucoup réduite de l'acquisition des notions informatique dans les nouveaux programmes de 2008 de technologie au collège.

2. Le choix d'une approche systémique

Il existe plusieurs façons d'aborder les problèmes liés au processus d'enseignement/ apprentissage de savoirs et, beaucoup de courants théoriques ont été élaborés pour éclairer des situations éducatives orientées dans ce sens. L'approche socio-constructiviste une forme de révolution paradigmatique dans le monde de l'enseignement/apprentissage scolaire, justifiée par le fait que les connaissances ne sont pas transmises mais sont construites par le sujet à travers les expériences vécues dans son milieu. L'enseignement de l'informatique, pour mettre leurs élèves ou étudiants en activité, se situe dans une approche plutôt socioconstructiviste : l'approche par projets et le travail en groupes fait référence à ce modèle.

Le cadre théorique qui a guidé notre regard et notre réflexion est la théorie de l'activité (TA), que nous allons analyser plus loin à partir de la section 3. Ce choix a été motivé par le fait qu'avec le modèle théorique socioconstructiviste correspondant à l'enseignement de l'informatique, les élèves, les étudiants et leurs professeurs sont tous inscrits dans un système d'activités. Malgré des rôles respectifs, ils sont appelés à entrer en interaction tout en respectant certaines règles en vue de la production d'un résultat attendu qui est l'objectif des projets. Pour toutes ces raisons, la TA nous semble bien appropriée pour cette étude qui, rappelons-le, porte sur les pratiques d'apprentissages des savoirs informatiques chez les élèves ou étudiants débutants dans un contexte d'apprentissage par projets.

2.1. Pédagogie de projets et interactions

La thèse défendue est celle des potentialités de la pédagogie de projets pour l'apprentissage de l'informatique en général et chez les débutants en particulier. Le public est constitué de lycéens scientifiques et d'étudiants de licence en informatique. Quatre éléments principaux interviennent dans les situations d'apprentissage : les élèves/étudiants, le professeur, les instruments et le savoir ; auxquels s'ajoutent leurs interactions liées au fait qu'il s'agit d'un contexte de projets effectués en groupes.

Notre orientation se veut essentiellement didactique : elle concerne la construction de savoirs disciplinaires, essentiellement informatiques. Nos choix ont eu un double objectif :

- d'abord, clarifier le rapport de l'élève (étudiant) en tant qu'apprenant à l'objet de connaissance (Brun, 1994) dans un contexte plus vaste de l'approche par projets ;
- et ensuite, apporter un éclairage sur les pratiques des élèves (étudiants), de leurs professeurs et les interactions en jeux.

La construction des connaissances chez les sujets a lieu, ici, essentiellement en contexte de projets. Rarement utilisée dans la scolarité secondaire, l'approche par projets et par problèmes est en vogue dans l'enseignement supérieur où *« elle constitue un changement de paradigme assez radical »* (Galand & Frenay, 2005, p. 10). C'est une méthode d'apprentissage fondée sur des problèmes pour susciter l'utilisation et la construction de nouvelles connaissances.

Ces auteurs donnent trois caractéristiques de cette approche, sur lesquelles se fondent ses potentialités. La première caractéristique est que, contrairement aux autres contextes d'apprentissage, l'approche par projets n'appartient à aucune discipline en particulier : c'est un apprentissage pluridisciplinaire inscrit sur un processus de longue durée. La deuxième caractéristique est le travail de groupe : si l'apprentissage reste individuel, l'importance de l'approche par projets est liée à son inscription dans un travail de groupe favorisant la coopération et le soutien mutuel des élèves dans leurs apprentissages. Comme forme inhérente au travail groupe, l'apprentissage coopératif suit les principes suivants (Robitaille, 2007, p. 172) : une interdépendance positive, une responsabilité individuelle, un développement des habiletés coopératives et une objectivation qui est un bilan-synthèse des apprentissages fait par l'enseignant pour terminer l'activité. La troisième caractéristique est la nécessité d'un tuteur. Si ce dernier peut être un enseignant ou une autre personne, son rôle est central dans

le travail de projets : il prescrit, conseille, oriente, donne une aide en cas de nécessité… La résolution de problèmes longs et souvent fastidieux nécessite un engagement de l'enseignant auprès de ses élèves/étudiants en apprentissage : ces derniers doivent être soutenus et encouragés pour rester motivés et ne pas abandonner le travail avant la fin.

2.2. Approche par projets et théorie de l'activité

Loin d'être indépendantes, les pratiques des élèves et des enseignants s'influencent mutuellement au cours d'une action didactique (Robert & Rogalski, 2002 ; Rogalski, 2012). L'enseignant est, par exemple, appelé à tenir compte des forces et des faiblesses des pratiques des élèves pour construire les situations didactiques : choix de sujets de projets effectués par les enseignants, sa façon de les encadrer, la nature des aides données, la façon et les moments-clés de les donner… Bref, dans ses pratiques, il prend en compte de leurs pratiques : ce qu'ils sont capables de faire, les questions posées, les tâches déjà effectuées, leurs compétences à aller plus loin, les degrés d'interactions avec les autres, leurs difficultés…

Dans ce sens, l'approche de la théorie de l'activité nous semble adaptée. Les tâches prescrites sont en fonction du sujet. Elles orientent, suscitent et caractérisent les pratiques de l'apprenant. Ensuite, avec l'approche de la théorie de l'activité, les élèves/étudiants apprennent en faisant et leurs apprentissages se consolident dans le travail en synergies. Selon Lewis (1998), la théorie de l'activité peut bien servir d'un cadre théorique de recherche qui s'intéresse aux apprentissages : « *la théorie de l'activité met en évidence les caractéristiques des communautés de travail efficaces* », elle peut aussi servir d'un cadre d'analyse des pratiques pour un travail relatif aux communautés d'apprentissages conjointes. Néanmoins, cette transposition n'est pas automatique. Des conditions doivent être réunies, eu égard aux différences existant entre les deux communautés : la communauté de travail et la communauté d'apprentissage. Dans la communauté de travail à laquelle s'intéresse la théorie de l'activité, les différents membres peuvent travailler sur des tâches différentes en termes d'exigences en fonction de leurs compétences ou leur disponibilité. Dans la communauté d'apprentissage (en groupes d'élèves/étudiants), les tâches sont en général partagées équitablement au sein des membres de chaque groupe[12], ce qui est un objectif difficilement réalisable. Pour que cette approche soit efficacement utilisée en contexte d'apprentissage, les tâches proposées doivent avoir deux caractéristiques (Lewis, 1998) : (1) amener les apprenants à s'y reconnaître pour susci-

12 Même si au sein d'un groupe, les membres peuvent, selon leurs différences de compétences, partager eux aussi inégalement leurs tâches.

ter leur motivation et, (2) nécessiter une approche collaborative/coopérative dans leur exécution pour stimuler les interactions entre les apprenants. Dans le cas qui nous concerne de l'apprentissage de l'informatique en contexte de projets, les tâches prescrites aux élèves/étudiants remplissent-elles les deux caractéristiques de Lewis pour justifier le choix de ce cadre théorique ?

3. Aperçu sur le concept d'activité

3.1. Définition et structure de base d'une activité

Leontiev (1975) conçoit l'activité comme une caractéristique particulière et vitale de l'homme qui lui permet en tant qu'actant à être capable d'initier des projets d'action sur le monde en vue de satisfaire ses besoins. Si ce sont ces derniers qui conditionnent et orientent l'activité du sujet sur le monde, les besoins du sujet deviennent aussi ceux de la collectivité. Ainsi, Leontiev définit l'*activité* comme étant *« une organisation systémique permettant le développement de la conscience »* (Taurisson, 2005, p. 70-75)

Pour comprendre l'activité, il préconise une approche de sa structuration en différents paliers sous une forme hiérarchique à trois niveaux : l'activité proprement dite, les actions associées à sa conduite et les opérations exécutées. Leurs frontières entre eux sont en général difficiles à identifier (Bourguin & Derycke, 2005 ; Beauné, 2010) : elles restent floues et mouvantes.

- **L'activité**. Dans la hiérarchisation des niveaux de l'activité, le nom de l'activité intervient comme le niveau supérieur qui coiffe tous les autres niveaux. Orientée par son objet, l'activité vise une finalité trop vaste pour être complètement perçue et atteinte par une seule personne (Taurisson, 2005, p. 72). Si l'activité est orientée vers une finalité, cette dernière englobe à la fois les intentions d'un sujet et la réalité de la situation dans laquelle l'activité se déroule. En fonction des buts poursuivis et de la motivation suscitée, une activité peut engendrer de nombreuses actions (Kuutti, 1996).
- **Les actions**. Constituant le deuxième niveau de l'activité, les actions ont un objectif précis (Taurisson, 2005) : contribuer au succès de l'activité. Si une activité est réalisée par une série d'actions conscientes orientée chacune par but précis (Bibang-Assoumou, 2013).

Pour Leontiev, le sens de l'action est indirectement orienté vers la satisfaction du besoin. Il est plus lié à la prise de conscience des acteurs qui fondent l'existence des actions complémentaires menées par les pairs. C'est cette prise de conscience des acteurs qui conduit à la recherche de la satisfaction des besoins communs à la communauté. Cette absence de linéarité des actions et la complexité de leurs relations au sein de l'activité a été aussi soulignée par Taurisson. Selon lui, cette prise de conscience de besoins communs conduit à la division du travail laquelle est indispensable pour la réussite de l'activité : chaque groupe aura à s'occuper de certaines actions, différentes d'un groupe à l'autre, mais avec la conscience de leurs rôles dans le succès de l'activité collective (Taurisson, 2005).

- **Les opérations.** Elles constituent le troisième niveau et le niveau le plus bas du modèle. Elles sont à la base de l'activité et correspondent à des actions produites automatiquement à force de réalisations successives et qui respectent un modèle fiable construit en fonction de certaines conditions. Exécutées inconsciemment, les opérations sont sous l'orientation d'une base établie au travers de l'expérience qui apparaît au contact des conditions matérielles concrètes (Bardram, 1997).

« Une activité est associée à un motif, une action à un but et une opération à des conditions nécessaires à son exécution » ((Barma, 2010, p. 683), se référant à Class (2001).

Abibang-Assoumou (2013), s'inspirant de la conception de Leontiev (1978), présente les niveaux d'activité humaine en association avec leur orientation, leur fonction, la ou les personnes intervenant à chaque niveau et l' « état de conscience », comme le montre le tableau I.1 suivant :

Niveau	Orientation vers	Fonction	Réalisé par	État de conscience
Activité	Objet-Motif-Mobile	Incite à l'action	Communauté	Non ou peu conscient
Action	But	Oriente et dirige l'action	Individu ou groupe	Conscient
Opération	Conditions instrumentales – Moyens	Réalisation de l'action	Routines (homme-machine)	Non conscient, automatisé

Tableau I.1: Niveaux de l'activité selon Leontiev (1978)

Pour souligner le caractère mouvant des niveaux de l'activité, Leontiev précise qu'une action peut s'élever au niveau d'une activité ou, une activité peut correspondre à une action dans une activité d'une portée plus générale.

Dans son explication des niveaux de l'activité, Magakian (2009) montre que l'activité se constitue de deux niveaux en correspondance au cours de l'activité. Il indique d'abord que l'activité est liée au motif, matériel ou idée, qui éveille et oriente vers le sujet l'activité en question. De plus, ce motif peut être donné dans la perception comme il peut être imaginé. Ensuite, l'activité n'est observable qu'au travers des actions qui la réalisent. Quant aux actions, leurs réalisations doivent mettre à jour des résultats. Elles sont donc bien soumises à un ou des buts conscients lesquels permettant d'attester si elles ont été clôturées ou pas, si les buts des actions ont été ou pas atteints.

Les opérations elles, se rapportent aux conditions de l'agir. Magakian souligne une éventualité de fluctuation de l'action en fonction des conditions d'accès à l'agir, dues aux objets de la situation. Il Il donne l'exemple d'un possible changement des conditions matérielles susceptible d'occasionner des bouleversements du contexte d'opération. Se référant à Leontiev, Magakian résume cet état de fait comme suit :

> *« Dans le cours général de l'activité qui forme la vie humaine dans ses manifestations supérieures, médiatisées par le reflet psychique, l'analyse distingue premièrement les activités isolées (particulières) d'après le critère de leurs motifs. Puis, on distingue les actions – les processus qui obéissent à des buts conscients. Enfin, ce sont les opérations qui dépendent directement des conditions de réalisations d'un but concret »*

3.2. Exemples d'une activité

Dans le cadre de la concrétisation de la notion d'activité, et surtout son exécution sous sa forme hiérarchique respectant les niveaux, nous présentons ici deux exemples d'activité donnés par Kuutti (1996) et repris par Magakian (2009, p. 61) : la construction d'une maison et la recherche scientifique, présentés parallèlement.

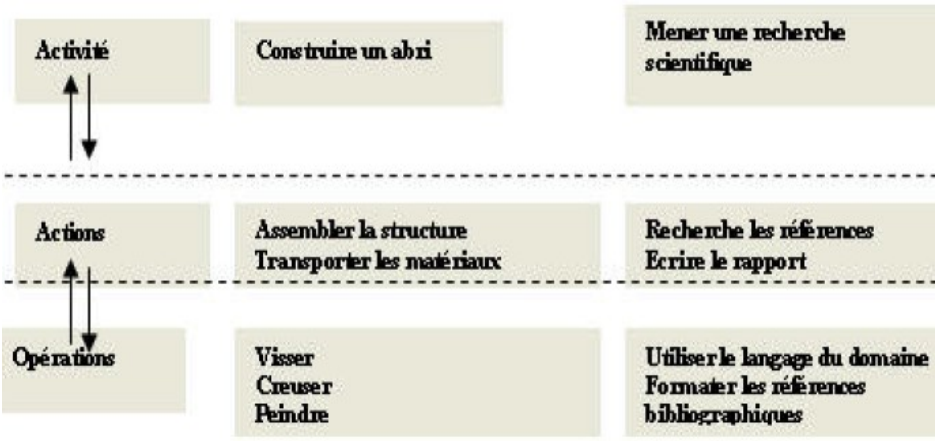


Illustration I.2: Exemples de Kuuti sur des niveaux d'une activité (Magakian, 2009)

Dans l'exemple de la construction d'une maison, l'activité est de « construire un abri ». Dans cette activité, plusieurs actions sont possibles dont celles consistant à concevoir le plan de la maison et donc « assembler la structure » et puis « transporter les matériaux ». Quelques-unes des opérations possibles figurent « creuser » pour la fondation, « visser », « peindre » les murs…

En rapport avec l'exécution de l'activité en communauté, Leontiev introduit deux autres formes. D'abord, il introduit les notions de « rôles » en ce qui concerne la division du travail entre les membres d'une communauté et ensuite, les « rôles et règles » en matière de leur communication. La division du travail est vue comme médian des interactions entre une communauté donnée et son environnement dans lequel elle vit. Elle permet, dans ce sens, aux membres de cette communauté de se fixer des rôles complémentaires en vue de faciliter la réalisation de l'objet commun ainsi poursuivi (Bracewell et al., 2007). De plus, les règles et les normes de la communauté quant à elles assurent les médiations des interactions entre chaque individu et la communauté dans laquelle il vit en permettant de négocier le sens des actions à mener ensemble au sein de la communauté en tant que composante à part entière de celle-ci.

4. Différentes théories de l'activité ?

4.1. Origine et développement

La Théorie de l'Activité (TA) est un courant théorique des sciences humaines et sociales. Elle est un cadre conceptuel pour étudier les différentes formes de pratiques humaines en tant

que processus de développement, combinant des niveaux individuels et sociaux (Bourguin & Derycke, 2005). Né dans les années 1930, ce courant trouve ses origines dans les travaux de Serguei L. Rubinstein et Lev Vygotski (Vygotski, 1934/1985). Trois générations caractérisent cette théorie. La première génération, fondée sur le concept de *médiation*, a été formalisée par Vygotski (1896-1934). Cette génération de la TA est fondée sur le postulat que « *le développement du comportement humain était d'abord et avant tout médiatisé par l'utilisation et la création des artefacts matériels ou symboliques »* (Barma, 2010). C'est donc de la thèse historico-culturelle qu'est née la TA. Cette dernière sera diffusée en France dans les années 1960.

La TA a été vue comme une alternative à la dichotomie entre l'individu et la société sur lesquelles se fondaient les sciences comportementales et sociales (Bourguin & Derycke, 2005) : les sciences du comportement avaient du mal à considérer le contexte dans l'étude des actions humaines d'une part et, les sciences sociales d'autre part. Face à cette dichotomie entre les deux perspectives, un concept intermédiaire – *l'activité* –, a été proposé. Il est défini comme un contexte minimal d'analyse ou l'unité de base pouvant contextualiser l'étude des actions humaines vues sous l'angle de la médiation institutionnelle et sociale (Barma, 2011). L'unité fondamentale d'analyse de la théorie de l'activité est l'activité humaine. Bourguin et Derycke (2005) définissent l'activité comme *« un système cohérent de processus mentaux internes, d'un comportement externe et de processus motivationnels qui sont combinés pour réaliser des buts conscients ».* La T A s'intéresse donc à l'activité humaine socialement contextualisée tel que le monde du travail ou de l'apprentissage (Parks, 2000). Par exemple, *« l'apprentissage, avant d'être le fait d'individus isolés, serait d'abord un phénomène social puisqu'il se déroule dans des contextes culturellement déterminés imprimant leurs marques spécifiques aux groupes humains dans leurs comportements les plus quotidiens »* (Beauné, 2010).

Dans la suite, le développement de la TA a été continué par d'autres auteurs dont Leontiev (1904-1979), Luria (1902-1977) et Engeström (1948-). La conception de la deuxième génération est attribuée à Leontiev, alors collègue et initialement disciple de Vygotski (Barma, 2008). Leontiev (1975/1978) donne une orientation nouvelle de la TA, focalisée sur l'activité et plus particulièrement sur *« la nature collective de l'activité » (*Owen, 2008*, p. 73)*. Si Vygotski s'est plus intéressé au comportement, Leontiev a fait ressortir la faille du modèle de Vygotski. Avec son raisonnement fondé sur la notion de coopération comme une caractéris-

tique de l'activité humaine, il met en évidence comment des objectifs communs peuvent être poursuivis et atteints par le partage du travail. À partir de la complexité des interactions susceptibles de naître autour d'un tel travail coopératif, un contraste a été mis en évidence entre une action individuelle de celle collective (Barma, 2010). Dans la nature de l'activité, il distingue aussi la notion d'activité de celle d'action et d'opération, comme des niveaux qui composent cette activité.

Les buts communs poursuivis par une communauté de personnes poussent à une division du travail au sein des membres. Pour Barma, *« la médiatisation se caractérise par la division du travail et l'instauration des règles qui encadrent les interactions entre les individus faisant partie du système d'activités et partageant le même objet »*.

Malgré l'apport important et reconnu de Leontiev dans la conceptualisation de la TA, des limites ont été signalées et reprochées à son modèle (Barma, 2008) : il *« n'a jamais vraiment concrétisé son modèle d'une façon instrumentale afin de pouvoir le communiquer à la communauté scientifique »*. De plus, il manque à son modèle la clarté et la précision sur les médiations sociales que doivent y avoir entre l'homme et le monde qui l'entoure.

C'est à Y. Engeström (1999) que l'on doit la structure actuelle du modèle de l'activité. Son raisonnement sur la structuration et la formalisation de l'approche de la TA s'est beaucoup inspiré des travaux de ses prédécesseurs tels que Vygotski et Leontiev dont leurs idées ont été synthétisées (Barma, 2010, p. 684) : *« un modèle systémique basé sur les deux premières générations en y ajoutant l'infrastructure socioinstitutionnelle de l'activité, c'est-à-dire les éléments de la communauté, les règles et la division du travail. ».*

4.2. Représentation systémique de l'activité humaine : le modèle d'Engeström

Inspiré de deux premières générations, le modèle actuel de l'activité est basé sur le modèle complexe d'un système d'activités d'Y. Engeström. Dans ses préoccupations à concevoir son modèle, Engeström a recherché davantage comment définir la plus petite unité à partir de laquelle une activité humaine peut être dynamiquement et historiquement caractérisée (Saussez & Yvon, 2010). Pour Engeström, les médiations étant complexes, l'action médiatisée par les seuls artefacts ne constituerait pas une unité d'analyse suffisante pour comprendre tous les contours de la médiation. Partant de ce constat, une expansion de l'unité d'analyse a été initiée par Leontiev (1975), ce qui a permis de formuler des médiations à base de rapports sociaux. Dans la conception du système d'activités, Engeström s'est inspiré de deux

générations précédentes avec une socioinstitutionnalisation de l'activité dans un environnement élargi incluant une communauté, des règles et de la division du travail (Barma, 2010). Le système d'activités devient constitué de six pôles en interrelation : sujet, outil, règles, division du travail, communauté et objet.

Dans la conception dudit système, le sujet et l'objet sont situés en son cœur. Si l'activité est orientée par son objet, le rapport à cet objet est médiatisé par des instruments et participe au développement d'une activité socialement réglée et normée. Au sein du système, le sujet fait partie d'une communauté qui partage avec lui le même objet d'activité. Les relations communauté-sujet et communauté-objet sont médiatisées par les concepts de règles et de divisions du travail véhiculant eux aussi un héritage culturel de la situation (Bourguin & Derycke, 2005). Ainsi Engeström donne la définition d'un système d'activités comme étant *« un système cohérent du point de vue de l'activité orientée vers des objets, des sujets agissant par l'intermédiaire des artefacts, et organisés collectivement au sein d'une division du travail, des règles qu'ils emploient et de leur communauté (d'intérêts, de pratiques ou de cultures) »* (Owen, 2008, p. 76). L'illustration I.3 modélise le schéma actuel de la structure de base d'un système d'activités selon la conception d'Engeström.

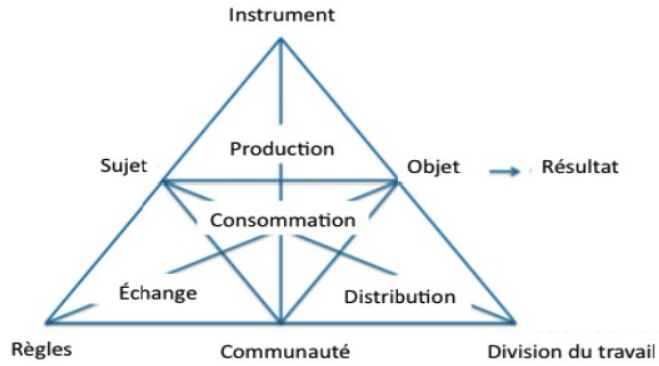


Illustration I.3: Structure de base de l'activité humaine d'Engeström (1987)

Barma (2010) synthétise la génération de la TA d'Engeström en cinq principes :

1. Le système d'activité constitue l'unité d'analyse. Orienté vers l'objet d'étude, ce sont les artefacts qui assurent la médiatisation dans son rapport avec l'objet. Les systèmes d'activités sont inter-reliés : chacun est en interrelation avec au moins un des autres systèmes ;

2. Un système d'activité reflète divers points de vue, divers traditions et intérêts. Cette situation est rendue possible par la division du travail au sein du système, ce qui donne à chaque membre du système un rôle à jouer ;
3. Les systèmes d'activités ne restent pas figés une fois formés : ils se renouvellent et leurs transformations peuvent durer de longues périodes ;
4. Il existe des tensions au sein d'un système d'activités. Souvent de nature structurelle, ces tensions, accumulées depuis longtemps au sein des systèmes d'activités, jouent un rôle essentiel dans la transformation et l'innovation ;
5. Dans un système d'activités, l'innovation se produit lorsqu'il y a une transformation, c'est) dire qu'il y a reconceptualisation de l'objet et du motif de l'activité pour un enjeu plus large comparativement au système d'activité d'avant.

4.3. Une activité médiatisée par des artefacts

La TA est construite autour de l'idée centrale de *médiation.* Dans cette théorie, Vygotski, son fondateur, considère que l'activité humaine est médiée par des artefacts, de leurs origines historico-culturelles et des processus de leur développement (Leong, 2012). Pour Vygotski, si l'activité humaine est à l'origine du développement de son comportement, elle est avant tout médiatisée par la création et l'utilisation des artefacts culturels, matériels ou symboliques (Barma, 2010). Dans l'approche de Vygotski de la théorie de l'activité, l'importance de la dimension culturelle et historique des artefacts est justifiée par le fait que c'est à partir d'elle que les artefacts sont valorisés : ils sont reconnus par la société comme des moyens et des mobiles de leurs actions dans des contextes spécifiques. Cette valorisation est davantage renforcée par leur caractère à la fois normatif, permanent et prescriptif par le fait qu'ils expriment relativement comment, du point de vue de leurs concepteurs, le monde serait ou devrait être (Bibang-Assoumou, 2013, se référant à Ivic (1994)).

L'appropriation de l'instrument (culturel) implique un engagement du sujet dans une activité. Cette activité doit être en conformité relative à celle qui a conduit à son élaboration, mais dont l'objet conscient serait la « signification véhiculée ». L'engagement réel du sujet dans les actions est conditionné par cette « signification véhiculée » par l'objet de l'activité à réaliser qui devrait s'inscrire dans la sphère de ses besoins réels pour donner sens et force à son engagement. Néanmoins, les conditions de l'appropriation de l'instrument vont au-delà du

niveau culturel. En effet, actuellement, dans le contexte d'une prolifération des instruments numériques en éducation, seule une utilisation raisonnée des instruments (numériques) peut permettre des conceptualisations et de la compréhension en vue de renverser la tendance à la domination des outils sur son utilisateur (Bruillard, 2012). Et pour être effectivement profitable et créative, un minimum de connaissances (informatiques) est nécessaire dans son utilisation.

Dans l'approche de la TA, Vygotski distingue deux types d'instruments de médiatisation (Barma, 2008) : les outils techniques et les outils psychologiques tels que le langage, les techniques mnémotechniques, les œuvres d'art, les cartes géographiques, etc. Pour Vygotski, le développement du comportement humain est avant tout médiatisé par des artefacts culturels matériels ou symboliques (Barma, 2010). Ce sont ces artefacts qui participent à la structuration de la relation du sujet au monde et à son activité notamment en mettant à sa disposition des moyens de résolution de problèmes (supports matériels pour l'action, principes opératoires, etc.) et des mobiles d'action (Bibang-Assoumou (2013), se référant à Ivic (1994).

Vygotski présente l'unité de base d'analyse de l'activité comme composée de trois éléments : le sujet, l'objet et artefacts (médiateurs). Si la relation entre le sujet et l'objet était initialement directe, actuellement, l'idée de médiation par les artefacts permet une nouvelle configuration comme la présente l'illustration I.4 suivante :

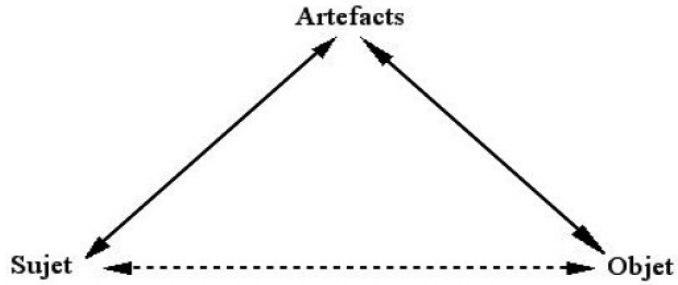


Illustration I.4: Médiation de la relation sujet – objet par des artefacts dans une activité (Rabardel, 1995)

En réalité, contrairement à la configuration de l'activité donnée par Engeström, celle-ci représente une relation individuelle d'activité médiatisée (Kuutti, 1996). Une combinaison d'outils auxquels correspondent des signes précis permet l'atteinte d'une fonction psychologique et d'un comportement supérieurs ; ce qui semble permettre de distinguer l'espèce humaine des autres espèces : soit la création et l'utilisation d'artefacts en tant qu'outils

producteurs de sens et facilitant la reproduction de l'espèce. Et dans ce contexte, les objets ne sont plus considérés comme étant uniquement du matériel à l'état brut, mais comme faisant partie intégrante de l'activité d'apprentissage.

4.4. La zone proximale de développement et le rôle des pairs dans la construction des connaissances

Mise au point par Vygotski, l'expression de zone proximale de développement (ZPD) est connue comme un des principes clés de l'apprentissage et du développement de la pensée au niveau supérieur. Pour expliquer ce que c'est cette zone et son fonctionnement, Lewis (Lewis, 1998) se référant à Vygotski, considère que la connaissance de tout individu est composée de deux types de connaissances. Le premier type est formé de connaissances « noyau » : celles sont considérées comme des connaissances de base et pouvant permettre de réaliser certaines tâches en toute autonomie. Le deuxième type est formé des connaissances non encore bien établies et stabilisées et qui ne permettent l'exécution des tâches que si l'individu est assisté. La ZPD correspondrait donc à cette deuxième catégorie de connaissances : ce sont des connaissances moins intériorisées, moins maîtrisées et qui se trouveraient distribuées autour des connaissances « noyau » de l'individu (illustration I.6).

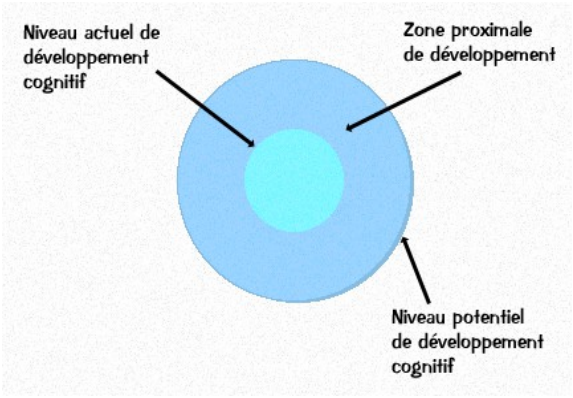


Illustration I.6: Représentation de la zone proximale de développement (Vygotski, 1997)

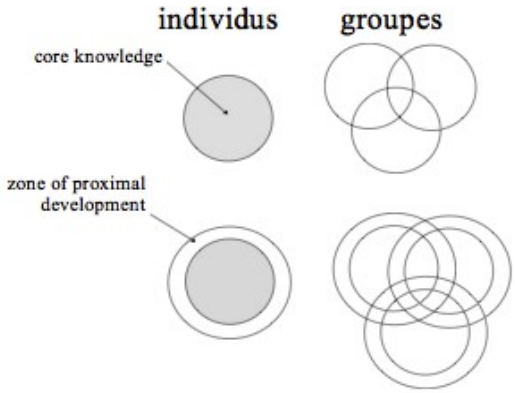


Illustration I.5: Recouvrement des connaissances dans un groupe (Lewis, 1998, p. 16)

La définition de la zone proximale de développement, telle qu'elle est donnée par Vygotski, révèle bien l'apport de la composante sociale dans l'apprentissage d'un individu : *« la distance entre le niveau de développement actuel tel qu'on peut le déterminer à travers la façon dont l'enfant résout des problèmes seul et le niveau de développement potentiel tel qu'on*

peut le déterminer à travers la façon dont l'enfant résout des problèmes lorsqu'il est assisté par l'adulte ou collabore avec d'autres enfants plus avancés »[13].

D'après cette illustration, le niveau actuel de développement cognitif précise ce qu'un individu maîtrise déjà seul et, par conséquent, ce qu'il est capable de faire, le type et le niveau cognitif qu'il est capable de mobiliser pour résoudre un problème donné en toute autonomie.

Quant aux connaissances « noyau » d'un individu, elles ne se limitent pas à celles déjà « internalisées » pour reprendre le terme de Lewis. L'individu bénéficie des connaissances de la communauté dans laquelle il fonctionne et ses connaissances « noyau » forment tout un système de connaissances avec celles des autres individus (Lewis, 1998, p. 17) : *« si l'on considère l'ensemble de la communauté, les connaissances « noyau » des individus se recouvrent partiellement et, surtout, les ZPD des uns coïncident avec les zones « noyau » des autres. De ce modèle, on peut conclure que les connaissances de la communauté sont (comme on pouvait s'y attendre) plus étendues que celles d'un seul individu, mais surtout que chaque membre de la communauté peut contribuer au développement cognitif du groupe en procurant à d'autres un « échafaudage » dans les domaines où leurs connaissances ne sont pas encore disponibles pour un travail autonome. »*

L'illustration I.5 précédente présente ce phénomène de recouvrement. Pour que ces connaissances disponibles et distribuées dans la communauté soient vraiment profitables à tout un chacun des membres, deux conditions doivent être réunies (Lewis, 1998 ; Bibang-Assoumou, 2013). La première est la conscience individuelle de chaque membre de la communauté : la potentialité de la collectivité ne peut être atteinte que si chacun des membres de la communauté a la conscience des connaissances des autres afin de pouvoir en profiter en y trouvant son aide et y contribuer en apportant lui-même son aide ou sa contribution. La deuxième condition concerne le caractère *distribué* des connaissances au sein de la communauté ou du groupe : pour que l'apprentissage conjoint soit effectif, chaque membre du groupe doit réaliser que les connaissances du groupe ne sont pas une « propriété » de certains membres particulièrement distingués du groupe, mais, se trouvent partagées parmi eux.

Dans les processus d'enseignement/apprentissage, relativement à la ZPD, Magakian (2009) considère le discours de l'enseignant comme une façon de cadrer l'espace de raisonnement de l'élève. Bruner (2000) trouve en ce discours de l'enseignant une forme d'impulsion chez

13 http://anaislauraelodie.blogspot.fr/2009/03/la-zone-proximale-de-developpement.html, site consulté le 25 octobre 2013

l'élève qui doit, par la suite, prendre ses responsabilités : *« le discours de l'enseignant exerce une conscience pour les deux : une fois le contexte posé, il ne reste qu'à l'élève lui-même d'agir »*.

5. Retour sur les systèmes d'activités d'Engeström

5.1. Des médiations au sein d'un système d'activités

Comme le montre Rabardel (1995), en recourant aux artefacts comme médiateur dans sa relation avec l'objet de son activité, le sujet s'appuie sur des schèmes d'utilisation pour dégager les potentialités ou les affordances[14] jugées viables pour la réalisation de son objectif : autant le sujet serait apte à dégager les affordances des artefacts auxquels il recourt, autant il pourra se démarquer des prescriptions originales accompagnant leurs usages. Sur cette base, la médiation consisterait en une relation dialectique sujet/artefact et son efficacité serait différente selon la conception que l'utilisateur a de l'artefact médiateur, de sa visée et de son activité (Bibang-Assoumou, 2013).

Dans le processus de production, quatre médiations interviennent entre le sujet et l'objet (Taurisson, 2005) : la médiation par les instruments, la médiation par les règles, la médiation par la communauté et la médiation par la division du travail.

La médiation par les instruments. L'activité humaine n'est ni un réflexe ni une réponse, mais une transformation du milieu à l'aide d'instruments par un sujet. Cela suppose une relation entre le sujet et son milieu. Selon Vygotski, cette relation n'est pas directe (Rabardel, 1995) : toute activité humaine est médiée par des instruments construits à partir d'outils par leur utilisateur au cours de son activité.

La médiation par la communauté. En situation d'activité, la communauté est formée par l'ensemble de tous les acteurs constitués ou non en groupes ou sous-groupe. En situation de classe, Taurisson (2005) précise que cette communauté ne se limite pas seulement à l'enseignant et à ses élèves, mais, est constituée, en plus de ceux-là, de tous ceux qui interviennent dans la réalisation de la production notamment des parents, des amis, des contacts tant phy-

14 Le terme d'affordance apparaît pour la première fois dans les écrits du psychologue James J. Gibson. Il lui donne forme en proposant le terme d'affordance en 1977. Pour ce psychologue, l'affordance est l'ensemble de toutes les possibilités d'action d'un environnement. Celles-ci sont objectives, mais doivent toujours être mises en relation avec l'acteur qui peut les utiliser. C'est en 1988 que Donald Norman, dans ***The Design of Eveyday Things*** réutilise ce terme pour l'interaction Homme-Machine pour désigner les potentialités d'action qui sont perceptibles par l'utilisateur d'un programme : http://fr.wikipedia.org/wiki/Affordance, site consulté le 9 novembre 2013

siques que virtuels. C'est, selon lui, une communauté diverse, constituée par d'entités multiples qui exercent, chacune de sa façon, une médiation différente. Si l'enseignant joue un rôle de médiateur, la classe tout entière offre un certain type de médiation qui est différente de celle offerte par des petits groupes, constitués pour pourvoir assurer des aides plus ou moins individualisées. Comme le précise Taurisson, c'est en se plaçant dans la ZPD (de Vygotski) que ces médiations accompagnent le sujet vers sa production (résultat) qu'est l'apprentissage.

La médiation par les règles : Médiatisant d'abord les rapports entre le sujet et la communauté dans sa diversité telle qu'elle est précédemment définie, ces règles établissent les relations entre le sujet et toutes les entités, et donc les manières de se comporter au sein de sa communauté : clarification des relations entre le sujet et l'objet, précision des rôles de chaque membre de la communauté, clarification du fonctionnement de l'activité, rappel de sa finalité et de son lien avec la production attendue…

La médiation par la division du travail : Ici, la médiation consiste en la division du travail au sein des acteurs. Pour Taurisson, si l'activité est structurée en actions, des buts à atteindre doivent être clarifiés aux acteurs et les actions partagées entre eux afin d'être diffusés au sein du réseau leurs acquisitions. Dans cette division du travail, des élèves de niveaux différents et de compétences complémentaires devraient être mis ensemble au sein d'un même groupe. Cette façon de faire permettrait que des habiletés acquises facilement par certains et difficiles pour d'autres soient partagées. En définitive, la production attendue de l'activité consisterait à une compétence obtenue par la mise en commun de toutes les diverses habiletés acquises.

5.2. Les contradictions au sein d'un système d'activités comme facteurs d'évolution

Une activité n'existe pas de façon isolée. Elle interagit avec d'autres avec lesquelles elle forme un système. Les différents systèmes d'activités sont en interrelation mais aussi en interdépendance. Cette interdépendance peut être une source de contradictions lesquelles engendrent des tensions entre les pôles. À propos, cinq niveaux de contradictions ont été développés. Les quatre premiers ont été développés par Engeström (2001). Le premier niveau de tension concerne l'objet comme production du système d'activités. L'objet produit peut véhiculer une double valeur en opposition : celle liée à son usage et celle liée à son échange en tant qu'une nouvelle forme d'activité au sein du système.

Un deuxième niveau de contradictions peut exister entre chacune des composantes du système. Le troisième niveau de contradictions se manifeste lorsque les motivations liées à la production de l'objet de l'activité viennent en conflit avec les pratiques courantes de l'activité vécues dans le milieu avant l'introduction d'une nouvelle pratique. Elles se manifestent du côté du sujet comme au niveau de la communauté. Les contradictions générées peuvent conduire jusqu'à bloquer le déroulement de l'activité étant donné que l'objet de l'activité initiale a été modifié. En situation d'apprentissage en classe, l'utilisation d'ordinateurs, quand elle demande une approche très différente de celle à laquelle les élèves sont habitués, pose problème.

Le quatrième niveau de contradictions intervient quand il y a des interactions entre le système principal d'activités et des systèmes voisins mais ayant le même objet. Les contradictions de ce niveau sont nécessaires pour la production d'innovation (Barma, 2010).
Le cinquième type de contradiction a été développé par Taurisson (2005) et, est lié à l'évolution des acteurs de l'activité. Selon lui, motivés par l'objet et le résultat (production) attendu, les acteurs sont au travail dans un contexte collaboratif avec de larges interactions. Si le fonctionnement de l'activité dépend de l'évolution des acteurs, l'activité prescrite, des contradictions sont susceptibles de naître. Pour illustration des contradictions pouvant être posées à ce niveau, l'exemple donné est celui du travail des élèves. Quand ils sont principalement habitués à travailler avec un appui rapproché de l'enseignant, ce contexte de travail ne favorise pas la collaboration entre eux ni leur autonomie. Quand la structure de l'activité prescrite nécessite l'autonomie, il y a donc contradiction.

6. Notre cadre théorique : TA adaptée au contexte de projets

La TA se réfère à une activité humaine socialement située, pouvant être associée à un contexte du travail ou de l'apprentissage (Parks, 2000). Nous nous situons plus précisément dans la troisième génération de la TA (Engeström, 1997) et en contexte d'enseignement/apprentissage. Selon la TA, l'apprenant n'est pas singulier dans son activité d'apprentissage. Son activité fait partie d'un système d'activités avec les autres activités de ses pairs. Acteur de ses apprentissages, il interagit sur les apprentissages des autres et vice-versa et les connaissances acquises sont en perpétuelle évolution : la pédagogie de projets en est un exemple. Les élèves/étudiants, dans leurs activités en contexte de projets de programmation, construisent leurs connaissances informatiques dans un environnement technologique. Nous

souhaitons prendre en compte les communications au sein d'un réseau large d'humains en interaction, de savoirs à construire et d'outils à commander.

Inspirées du modèle de la TA d'Engeström et de son adaptation au contexte d'apprentissage en situation de classe (Herviou & Taurisson, 2012), deux modélisations de systèmes d'activités en contexte de projet ont été conçues : une modélisation centrée sur le sujet « apprenant » et celle relative au sujet « enseignant ». L'illustration I.7 modélise le système d'activités de la TA adaptée à la coconstruction des connaissances chez les lycées et les étudiants en projets.

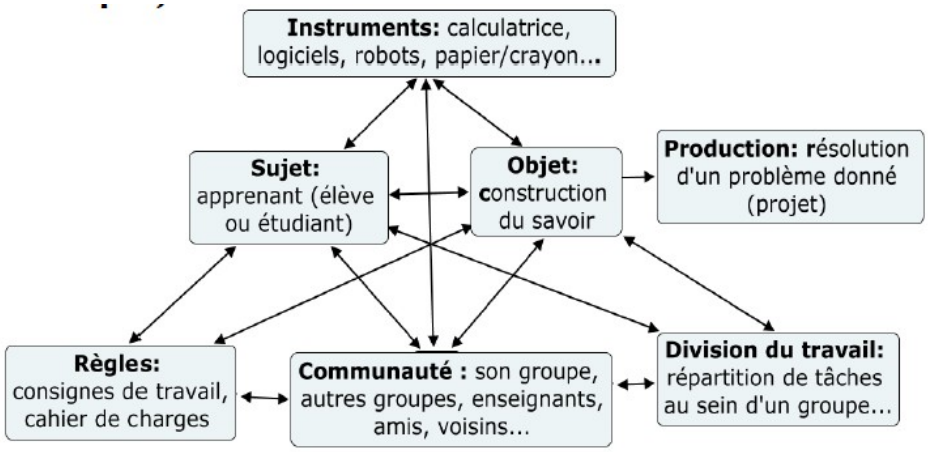


Illustration I.7: Un système d'activités adapté au contexte de projets

Les différents pôles de cette modélisation sont en interaction. Un apprenant construit ses savoirs au moyen des instruments à sa disposition (logiciels, calculatrices, son savoir antérieur…). Œuvrant dans un contexte de projet, ils travaillent en groupe et procèdent par le partage du travail. Chaque apprenant a, au sein de son groupe, ses rôles à jouer. Non seulement, il interagit avec son groupe mais aussi avec une communauté composée de différents groupes d'apprenants, de leur enseignant (encadrant), leurs amis, leurs voisins… Dans leur travail, les apprenants suivent des consignes et un cahier de charges qui sont des règles de travail données et qui doivent être respectées. Toutes ces interactions qui contribuent à la construction de savoirs, visent un produit final qui est la résolution d'un problème prescrit : le projet.

Le chapitre suivant fait un retour pour développer davantage la problématique à l'origine de la recherche.

Chapitre II PROBLÉMATIQUE ET MÉTHODOLOGIE

Ce chapitre esquisse la problématique qui a motivé la recherche et la méthodologie mise en œuvre pour la conduire. Il sera développé en cinq points : les questions récurrentes relatives à l'enseignement de l'informatique au lycée, la problématique sous tendant l'enseignement et l'apprentissage de l'informatique en licence, les risques d'un enseignement d'une spécialité informatique par des non-spécialistes du domaine, la méthodologie choisie et le corpus de la thèse, le rappel du questionnement et des hypothèses de recherche.

1. Enseignements de l'informatique au lycée : des questions insistantes

1.1. Nécessité des savoirs pour construire des compétences

Si la notion de savoir est ancienne, celle de compétence commence à faire parler d'elle vers la fin du XX^{e} siècle et, s'est imposée à l'école. À partir de cette période, savoirs et compétences sont devenus deux notions vivantes de la sphère éducative (Ropé & Tanguy, 1994). Le phénomène de l'approche par compétences a fortement caractérisé beaucoup de systèmes éducatifs au début du XXI^{é} siècle (Baron, 2012). Initialement prôné par des chercheurs canadiens comme Perrenoud (1998), ce phénomène a vite gagné d'autres pays dont la France. Dans la plupart des systèmes éducatifs francophones, cette notion est devenue centrale et les pays ont été amenés à reconstruire les contenus d'enseignement respectant cette approche (Reuter et al., 2007). Des réformes de programmes d'enseignement scolaires basées sur cette approche par compétences ont vu le jour : cette approche était devenue une caractéristique radicale commune (Lebeaume et al., 2007).

La France n'a pas totalement adhéré à cette approche : son entrée était limitée au secteur de l'enseignement technique et professionnel (Reuter et al., 2007) avant d'être massivement reprise par des décideurs politiques. À partir des années 2000, en informatique, à tous les niveaux d'enseignement, cette approche par compétences a été institutionnalisée en mettant en place un B2i depuis l'école primaire jusqu'au lycée et un C2i à l'université, sans référence à un quelconque curriculum (Baron, 2012).

Deux positions étaient en tensions autour de cette approche par compétences. La première est celle des auteurs pour qui, la centration sur la notion de compétences n'était pas justifiée étant donné que la mission première de l'école n'est pas de donner des compétences aux ap-

prenants, mais d'instruire et de transmettre des connaissances. La deuxième position est celle des auteurs qui considéraient que mettre l'accent sur l'acquisition des compétences dans les apprentissages scolaires est justifié : selon eux, il ne suffit pas que l'élève réussisse à l'école, mais faudrait-il aussi être capable de mobiliser ses connaissances acquises dans des situations diverses, complexes et même imprévisibles notamment en dehors de l'école. L'accent mis sur le réinvestissement des connaissances acquises en contexte scolaire répond à un souci d'efficacité de l'enseignement, d'adéquation plus grande des apprentissages scolaires aux situations de la vie hors ou non du travail (Perrenoud, 1995). Cette préoccupation est, selon lui, justifiée par la problématique actuelle du transfert des connaissances ou de la construction de compétences. D'un côté, les savoirs scolaires, pour être utiles, doivent être transférables et, de l'autre côté, la possibilité du transfert n'exige pas seulement une maîtrise des savoirs mais aussi et surtout une intégration de ces dernières à des compétences de réflexion, de décisions et d'action dans des situations complexes auxquelles l'individu est appelé à faire faire face dans la vie.

Philippe Jonnaert justifie la pertinence de cette approche par les compétences par le fait qu'elle contribue à établir des liens entre les apprentissages acquis et le pouvoir de les utiliser en contextes différents (Jonnaert, 2001). Si cette notion de compétence a gagné beaucoup de pays dans la structuration des programmes d'enseignement, les compétences doivent être fondées sur des savoirs stables. En informatique et plus particulièrement dans le cadre des TICE, cette notion de compétences pose problème et occulte même certaines difficultés (Bruillard, 2006). Bruillard donne deux problèmes y relatifs. Le premier est la tendance à ranger tout ce qui est informatique et des télécommunications sous la même appellation et expression TICE. Le deuxième problème est que la tendance à vouloir restreindre les TICE autour des questions d'accès, dans une acception matérielle et de compétences décrites dans les B2i et C2i. Selon lui, la notion de *« compétences cache, sous des expressions parfois sibyllines, une difficulté récurrente à prendre en compte les aspects conceptuels des technologies numériques »*.

1.2. Plaidoyer pour associer savoir et compétence

Le champ de la notion de compétences dépasse le cadre scolaire. Selon Perrenoud (1995), sa place est de plus en plus située dans le champ du travail et des qualifications. Beaucoup d'auteurs se sont intéressés à cette notion de compétence dont la place devient de plus en grande.

Pour Jean-Yves Causer (Causer, 2012), la compétence révèle nos capacités à donner du sens à un cheminement personnel et à y mettre de la cohérence. Se référant à Bellier (1999), il rappelle les quatre points intervenant dans le cadre de l'approche par la compétence. Parmi eux, le lien entre la compétence et l'action. Selon lui, cette relation qui existe entre la compétence et l'action est exprimée par le fait que *« la compétence n'existerait pas indépendamment d'un problème à résoudre en situation »*. Or la résolution d'un problème ne nécessite pas seulement des compétences mais aussi et avant tout une connaissance y relative. Richard Wittorski (1998) justifie son importance de plus en plus grandissante par le fait qu'elle a tendance à remplacer la notion de qualification. Dans sa hiérarchisation, Wittorski donne et définit des notions principales voisines de la compétence et qui interviennent dans la « dynamique de développement des compétences ». Les notions considérées, allant du savoir à la professionnalité, comme le montre l'illustration II.1.

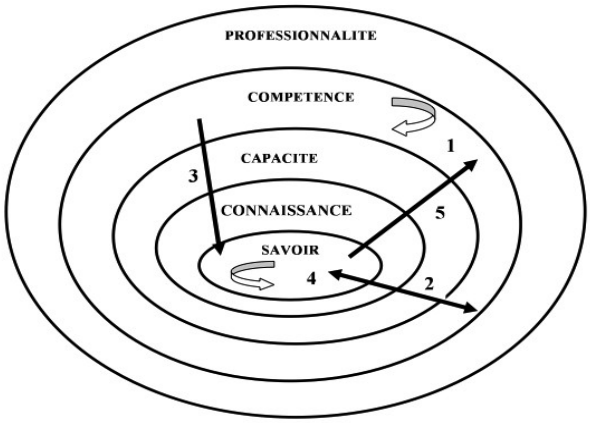


Illustration II.1: Le savoir, au cœur d'une compétence (Wittorski, 1998, p.6)

Il définit un **savoir,** se référant aux travaux menés par un groupe de recherche sur les savoirs d'action au CNAM, comme étant « un énoncé communicable et socialement validé », un énoncé descriptif ou explicatif d'une réalité, établit et reconnue par et dans une communauté scientifique et culturelle donnée à une époque donnée. Des lois de la physique en sont données comme des exemples. Comme le montre cette figure, des compétences n'ont de sens et n'existent pas sans les savoirs sur lesquels et avec lesquels ils sont construits : les savoirs et compétences fonctionnent ensemble en interaction.

Philippe Perrenoud (995) suggère de renoncer à l'idée selon laquelle on peut écarter les connaissances pour développer des compétences ou qu'il y a une exclusion mutuelle entre connaissance et compétences. Selon lui, les connaissances sont des ressources cognitives généralement essentielles dans la construction des compétences. Pour justifier la nécessité de

l'utilisation des connaissances acquises, Perrenoud (1998) compare des connaissances non investies à « des capitaux immobilisés ». Il défend ainsi la thèse de la centration des programmes sur les compétences pour offrir aux apprenants des conditions de transfert des connaissances acquises : les compétences sont, selon lui, un horizon des apprentissages, particulièrement chez ceux qui vont poursuivre des métiers scientifiques et techniques. La recommandation faite est d'articuler des savoirs et des compétences : développer des compétences dès l'école et lier constamment les savoirs et leur mise en œuvre dans des situations complexes, ce qui est possible aussi bien au sein d'une discipline qu'au carrefour des disciplines différentes.

Se situant dans un cadre clinique, Miller (1990), dans son triangle qui porte son nom, *le triangle de Miller,* montre comment les connaissances sont le fondement du développement des compétences, des performances et de toute action.

En informatique, ceux qui ont réclamé son enseignement pour en faire une culture générale de tout citoyen, sont motivés par son importance dans la société d'aujourd'hui. L'idée défendue est que les compétences à valider sont associées aux connaissances pour plus de conceptualisations et de verbalisations des processus mis en œuvre. Après beaucoup de rapports, de déclarations et de conférences publiques et même des audiences auprès des autorités publiques, leurs voix ont été entendues, avec la mise en place des enseignements faisant intervenir des notions informatiques et même la discipline informatique proprement dite.

Ces plaidoyers pour associer connaissances et compétences en informatique ne sont pas nouvelles. Au début des années 1990, les visionnaires sentaient déjà la nécessité que l'informatique fera partie de la culture générale. Le titre d'un atelier d'un colloque, sous la conduite de Jean-Michel Bérard, interrogeait la préoccupation du moment résumée comme suit (Bérard, 1992) : *« Peut-on définir un ensemble minimal de connaissances et de compétences en informatique permettant une utilisation rationnelle de l'ordinateur ? »*. Cette interrogation de la fin du XX^{e} siècle, semblable à celle actuelle prônant que l'informatique doit faire partie de la culture générale du citoyen, était plus ou moins développée, formulée et questionnée en ces termes : *« Informatique, élément de la formation professionnelle des enseignants ? Informatique, élément de culture générale, scientifique et technique de l'homme du vingtième siècle ? Passionnant défi que de contribuer à en définir les contours »*.

Bref, la préoccupation de faire de l'informatique, un élément de la culture générale de tout citoyen, reliant ses connaissances et ses compétences a, aujourd'hui atteint son paroxysme en

France. Cela n'est possible, selon les militants de l'enseignement d'une discipline informatique que si des curricula d'enseignement sont conçus en reliant savoirs et compétences. Monique Grandbastien s'interroge sur la formation qu'il faut pour tous les citoyens, une question à laquelle elle tente de répondre en décrivant les conditions dans lesquelles un enseignement de l'informatique pourrait se passer pour être efficace. Selon elle, il devrait passer par la définition d'un ensemble de connaissances et de compétences à acquérir par divers moyens proposés aux apprenants. Parmi ces derniers, figure une approche par projets (Grandbastien, 2012, p. 95) : *« c'est dans les applications et en interaction avec d'autres domaines que l'informatique intéresse le citoyen et qu'elle intéressera les élèves, et c'est souvent au travers des projets concrets que la formation devra être organisée »*. Cette proposition de présentation du cours d'informatique permettra son accessibilité aux élèves dans leurs différences de compétences.

2. Enseignement de l'informatique en licence : contexte et éléments de problématique

En France, alors qu'on parle de la réapparition de la discipline informatique dans les lycées généraux, après une trentaine d'années d'absence (Archambault, 2011), la situation est différente de l'université. L'informatique y est enseignée depuis les années soixante (Baron, 1987) et, plusieurs approches sont mises en œuvre : approches par projets, programmation des technologies innovantes, etc.

Les étudiants qui fréquentent la licence informatique sont pour la plupart issus des filières scientifiques des lycées généraux. N'ayant jamais acquis une initiation à l'informatique, ils entrent dans les filières informatiques en vue d'une spécialisation. Monique Grandbastien (2012) fait savoir que ce n'est pas parce qu'il n'y a pas de spécialistes pour l'enseignement de l'informatique au lycée qu'ils ne sont formés dans des universités :

> *« Sur le plan de la formation, les universités sont à peu près dotées toutes de départements d'informatique, qui, avec les départements d'IUT et des filières d'écoles d'ingénieurs assurent la formation des étudiants futurs professionnels ; la formation des non-informaticiens reste souvent cantonnée aux usages de la bureautique et se trouve matérialisée par le C2i ».*

Comment les étudiants, vierges de toute expérience en informatique en fin de lycée, construisent leurs savoirs informatiques à l'université ?

2.1. Approche par projets robotiques, une solution aux problèmes de l'apprentissage de l'informatique en licence ?

L'apprentissage de la plupart des disciplines scientifiques souffre d'un manque de motivation chez les jeunes. Un sentiment d'inaccessibilité voire de rejet a d'ailleurs été développé chez beaucoup parmi eux et le caractère abstrait des sciences en serait à l'origine (Nonnon, 2002). Si nous pouvons penser que l'informatique ferait exception, il n'en est malheureusement rien. Janiszek et al., (2011) indiquent que le manque de motivation de l'informatique chez les jeunes a poussé à imaginer et élaborer des parcours pédagogiques plus attractifs, motivants et mieux adaptés aux étudiants pour motiver l'apprentissage de l'informatique : l'utilisation de pédagogies de projets.

Ce contexte d'apprentissage semble entrer dans la ligne suggérée par Pierre Nonnon de mettre les élèves au travail. Selon lui, pour aimer les sciences, il faut en faire. L'instauration de contextes motivants dans les apprentissages des étudiants vise à favoriser l'engagement, la découverte, la curiosité, l'exploration, l'expérimentation et la formalisation. L'importance de ce contexte d'apprentissage est de permettre l'explicitation d'une démarche de projet prescrite aux étudiants surtout en début de formation pour ne pas rendre plus problématique sa mise en œuvre (Lauzier & Nonnon, 2007).

La robotique, depuis les travaux pionniers de Vivet et Nonnon, en particulier, est considérée comme une approche prometteuse (Vivet, 2000 ; Nonnon, 2002). Dans un article récent, Gaudiello & Zibetti (2013) parlent de révolution technologique marquant ce début du XXIe siècle et conduisant à l'« ère du robot ». Selon eux, l'intérêt estimé pour ce genre de technologie a permis, en éducation, l'émergence d'un nouveau champ d'étude spécifique : la robotique éducative. Le choix de cette approche par projets robotiques en éducation est justifié par l'activité des apprenants au cours de laquelle les conséquences de leurs décisions sont clairement et immédiatement visibles. La section 2.2. suivante décrit le processus de développement de la RP.

2.2. Robotique pédagogique : genèse et développement

Si, dans l'enseignement et apprentissage de l'informatique, l'intérêt de la robotique pédagogique (RP) est de plus en plus évoqué actuellement, cette question n'est pas nouvelle. Cette approche trouve sa source à la fin des années 80. Fortement attaché à la formation professionnelle, Martial Vivet (2000) indique les motivations à l'origine de la naissance de la RP : remédier aux limites d'usage de la tortue Logo dans la formation (professionnelle) en vue d'améliorer son efficacité pour les apprentissages. Les outils matériels devraient être enrichis au moyen des bras manipulateurs pour pouvoir travailler, non pas avec la tortue de Logo au sol, mais avec des bras. La tortue logo a donc été dotée d'« un dispositif appelé CARISTO, correspondant au chariot de cariste, point de départ d'un axe de recherche lié à la formation technologique et la formation professionnelle continue. » (Bruillard et al., 2000).

En France, les travaux de recherche dans ce domaine existent donc depuis une trentaine d'années (Leroux, 2005a). Après le premier congrès sur la RP tenu en France en 89, le début de la décennie suivante s'est caractérisé par des recherches accrues dans ce domaine, lesquelles ont confirmé ses potentialités éducatives (Marchand, 1991 ; Baron & Denis, 1993 ; Duchâteau, 1993) : possibilités de renouvellement des pratiques d'enseignement du côté des enseignants et, d'apprentissage chez des apprenants. À partir de cette période, des activités d'apprentissage attrayantes pour les élèves ont été conçues et proposées. Un des atouts majeurs de ces pratiques éducationnelles se situe dans la pratique active de l'élève, où ce dernier est incité à donner une forme – par construction – et un comportement – par la programmation – à un objet de sa propre création.

Pascal Leroux, l'un des pionniers de la RP, s'est intéressé à l'initiation à la technologie et à l'informatique, au pilotage d'automatismes et à l'initiation à la programmation. Il a mis sur pied un logiciel appelé RoboTeach (Leroux, 1995), qui a été reconnu d'intérêt pédagogique par le Ministère de l'Éducation Nationale (Leroux et al., 2005). Destiné aux adultes de faible niveau de qualification et aux élèves de collège, ce système comprend des matériels pédagogiques : des micro-robots modulaires, des fichiers d'activités en automatismes pour les classes à destination des élèves et enseignants. Ce logiciel permet d'aborder des notions algorithmiques telles que des structures conditionnelles (si… alors… sinon) et itératives (tant que…) sous une forme proche du langage naturel.

Malgré la cherté de la RP, Duchâteau (1993) la recommande dans l'apprentissage de la programmation. Selon lui, si les concepts essentiels de l'algorithmique sont importants et si l'exercice de la pensée planificatrice à laquelle ils conduisent est source d'apprentissages essentiels, il importe d'aller au-delà et de proposer des approches non classiques de la programmation : la programmation des robots. Cette importance est justifiée par le fait que l'exercice de la RP peut être un précurseur facilitant l'appropriation de certains savoir-faire et modes de pensée de la programmation, mais aussi comme une étape particulièrement adaptée et sémantiquement riche de la maîtrise de celle-ci. Ainsi, la robotique constitue, selon lui, une des étapes de l'apprentissage de la programmation informatique et, recommande de « consacrer un certain temps à la programmation de robots » au cours de l'apprentissage de l'informatique.

Actuellement, l'approche orientée projet de la programmation des robots est privilégiée dans l'apprentissage de l'informatique (Janiszek et al., 2011b). La RP peut prendre diverses formes *« allant d'un simple ordinateur contrôlant un objet périphérique (une station météorologique, de maquettes de mesures en science physique, un train, des systèmes automatisés) jusqu'à un automate intelligent ou un simulateur d'expérimentation »*. Les robots peuvent être utilisés dans les apprentissages pour tout public et à tous les niveaux d'enseignement, dès le plus jeune âge jusqu'à l'université et même pour la formation professionnelle des adultes (Komis & Misirli, 2015). Nous présentons ici quelques exemples de recherches qui ont été menées en RP à différents niveaux de la scolarité. À l'école maternelle, Komis & Misirli (2011 , 2012 ; 2015) s'intéressent aux processus de construction d'algorithmes et de programmes chez les enfants de bas âge au moyen de jouets programmables de types BEE-BOTS.

Au secondaire (collège) et pour les professionnels faiblement qualifiés, Leroux (2005b) et Leroux & Vivet (2000) ont plus travaillé sur la conception et l'utilisation des micro-robots de type ROBOTEACH pour l'apprentissage de la technologie. À l'université, Janiszek et al., (2011b) et Nijimbere et al., (2015) s'intéressent aux apprentissages de l'informatique par la programmation des robots respectivement de types NAO et LEGO MINDSTORMS NXT en licence.

D'après ce qui précède, le constat est que la typologie des robots change avec l'âge et le niveau des apprenants. Les robots orientés jouets programmables par exemple sont adaptés et

appropriés aux jeunes enfants de la maternelle (Komis & Misirli, 2015) : *« ils permettent d'exercer la programmation de manière ludique, tangible et concret. »*

La section suivante donne un éclairage sur les potentialités de cette approche pédagogique dans les apprentissages.

2.3. Robotique et potentialités éducatives

Depuis les années 90, des recherches se sont intéressées à des approches comparatives entre ordinateur et robot (ou l'informatique et la robotique). Charles Duchâteau (1993) en précise trois principales potentialités communes : d'abord, la potentialité de rapprochements disciplinaires pour passer à l'acquisition de connaissances en privilégiant l'exercice d'une démarche méthodologique ; ensuite, la mobilisation d'une méta-réflexion engagée comme l'un des buts essentiels de tout apprentissage qui vise le développement de l'autonomie des apprenants et, enfin, l'intervention du concept de problème. Si ce concept est commun à la robotique pédagogique et la programmation, ces dernières sont mobilisées dans la résolution des problèmes, une résolution qui est toujours faite en « différé » par un exécutant ad hoc.

À côté de ces ressemblances, des différences ont été aussi relevées. Duchâteau en a souligné quelques-unes qu'il classe en deux grandes catégories. La première catégorie concerne les aspects conceptuels qui s'intéressent à la durée des actions, la façon dont elles se déroulent et la construction de l'exécutant. En rapport avec la durée, dans le cas de l'exécutant-ordinateur, les actions sont conceptuellement sans durée : si la durée est sans importance pour l'écriture de la marche à suivre, dans le cas d'un exécutant-robot, les actions ont souvent une durée limitée de leur déroulement. Un deuxième aspect conceptuel est la programmation parallèle. Ce type de programmation, où des actions peuvent facilement s'exécuter simultanément, est aisément abordé en robotique. Le dernier aspect conceptuel est la construction du robot qui, contrairement à la programmation des ordinateurs, fait partie intégrante de l'activité de la robotique pédagogique et confère ainsi au robot une dimension technologique.

La deuxième catégorie aborde l'apprentissage concerné et les activités cognitives permises. Il distingue d'abord deux types d'apprentissages : la robotique pédagogique et la programmation classique. Par informatique classique en général et programmation classique en particulier, j'entends la programmation qui consiste à manipuler directement l'ordinateur ou un environnement informatisé et non pas un matériel embarqué, donc extérieur à l'ordinateur. Bourguiba (2000) caractérise ce contexte de programmation de « souple » par rapport à celui

de la programmation d'un robot. Pour ce dernier, le code est d'abord construit sur l'ordinateur avant de le tester sur un exécutant distinct et extérieur de l'ordinateur.

D'autres recherches ont comparé le robot et l'ordinateur selon leur état. Deux spécificités du robot ont été relevées. La première est liée à son caractère d'objet réel. Hsu et al., (2007) distinguent le robot de l'ordinateur par sa nature d'objet réel et systématique, en opposition avec la nature d'artefact virtuel et intégré caractérisant le logiciel éducationnel sur ordinateur. La deuxième spécificité est en relation avec leur fonctionnalité. Resnick et al., (1996) montrent qu'un robot, comme un ensemble mécanique et électronique, est un dispositif caractérisé par différents degrés de transparence – programmabilité – et d'interactivité – délai du feed-back – et contrôlable par l'ordinateur. La combinaison de ces deux caractéristiques permises par le robot est, selon eux, le fondement de l'intérêt éducatif du robot chez l'apprenant. Pour d'autres recherches, la robotique pédagogique offre une amélioration de la compréhension et de la maîtrise de la programmation chez les apprenants par rapport à l'utilisation du seul ordinateur (Pap-Szigeti, Pasztor et al., 2010).

Quatre caractéristiques distinguent la robotique pédagogique de la programmation classique (Duchâteau, 1993) : motivation, facilité de vérification des programmes, degré d'abstraction de l'exécutant et vaste champ d'application. Selon lui, la motivation plus grande dans le cas de la RP, résulte des représentations mentales plus immédiates qu'on peut se faire que dans le cas du robot-ordinateur : l'assemblage du dispositif à programmer fait parfois partie de la démarche éducative proposée, la connaissance de l'« intérieur » du robot et la motivation à le programmer sont plus aisées. Cette immédiateté des représentations mentales confère à la programmation des robots un caractère ludique, plus long à mettre en évidence dans le cas de la programmation classique. La deuxième caractéristique dans cette catégorie est la vérification des programmes. Cette dernière est rendue facile dans le cas du robot par le fait qu'on voit le robot agir sous l'emprise de la marche à suivre, conçue et fournie par son concepteur : l'entièreté du déroulement du processus est suivie, la correction et l'adaptation des programmes sont plus aisées. Cette situation est de loin différente du cas de la programmation classique où on ne dispose que de l'écran où s'affichent les résultats du processus alors que tout le processus d'exécution est occulté, ajoute Duchâteau.

En troisième lieu, contrairement au robot construit qui agit et réagit sur des tâches qui lui sont communiquées, l'exécutant-ordinateur est vu comme un robot abstrait et compliqué. Selon (Dowek et al., 2011), ce robot est donc un ordinateur particulier muni des capteurs et

d'actionneurs où une action est exécutée en boucle infinie fermée dans laquelle les capteurs sont interrogés et les actionnaires activés. Ces tâches, dans le cas du robot, deviennent finalement compréhensibles et aisément repérables. La dernière caractéristique qui distingue le robot de l'ordinateur concerne les champs d'applications possibles. Contrairement à la programmation classique des ordinateurs qui se limite aux traitements formalistes de l'information, les robots sont des exécutants en mouvement qui peuvent agir souvent dans un espace, qui manipulent, qui bougent et qui sont dotés d'organes capables de renseigner sur l'état de leur environnement. Si le concept d'information, central en informatique (Dowek *et al.*, 2011), reste le plus souvent une abstraction en programmation classique, Duchâteau précise que les activités de la RP peuvent lui donner corps : la RP apporte une extension du champ accessible à la modélisation et un éclairage supplémentaire sur l'information et son traitement. La section 2.4. suivante montre des types de langages adaptés pour des débutants en programmation.

2.4. Initier à la programmation : du langage LOGO aux langages à blocs

Mendelsohn (1986) note une spécificité de l'approche LOGO par rapport aux autres approches de programmation. Selon lui, LOGO joue trois rôles à la fois : il est un langage de programmation, une théorie de l'apprentissage mais aussi un dispositif matériel. Populaire dans les années quatre-vingt (Papert, 1994), l'approche LOGO a été caractérisée comme faisant partie de la RP (Duchâteau, 1993). Si l'usage de LOGO était initialement focalisé sur un robot « mécanique », connecté à l'ordinateur par un long « cordon ombilical » pour reprendre le terme utilisé par Resnick *et al*., la prolifération des micro-ordinateurs dans les années 1970 a fait que la communauté LOGO oriente leur approche sur des « robots d'écran », rapides et précis. Cette nouvelle orientation avait pour avantage de susciter chez les jeunes apprenants une créativité et une démarche d'investigation beaucoup plus poussées et complexes. Les limites de l'environnement LOGO dans l'initiation informatique étaient liées au manque d'autonomie (Resnick et al., 1996) : « *it is difficult to think of a machine as an autonomous creature if it is attached by umbilical cord to a computer. »*. Cette limitation de nature structurelle a été prise en compte dans la conception des langages des générations qui lui ont succédé. Ces derniers visaient la correction de ses imperfections et des difficultés vécues par les apprenants dans son utilisation (Leroux, 1995) : *« Grâce à la manipulation directe et à un modèle d'interface, spécifique, qui intègre l'ensemble de fonctionnalités de gestion des pro-*

grammes (édition, exécution, etc.), nous avons limité les problèmes que rencontrent les stagiaires dans nos formations d'initiation technologique et informatique au cours de l'utilisation de l'environnement de programmation LOGO : erreurs syntaxiques et des difficultés liées à l'édition et à l'exécution de programmes ».

Un état de l'art sur les recherches en RP a été donné au Chapitre IV 1.5. de la première partie.

Contrairement à Duchâteau qui assimile LOGO au robot, beaucoup de travaux récents sur la programmation font une distinction entre les deux. Les environnements successeurs de LOGO, conçus pour l'initiation informatique, sont actuellement orientés dans un ensemble de *langages* dits *graphiques à base de blocs* (Muratet et al., 2011). Un premier langage de ce type a été LOGO Blocks, développé au milieu des années 90 par « the MIT Media Lab » (Tempel, 2013). La littérature donne une liste non exhaustive de tels langages : Turtle Art, PICO Blocks, MIT App Inventor, Alice, Scratch (Tempel, ibidem.), Program your robot (Kazimoglu et al., 2012) ; PlayLogo (Paliokas et al., 2011). Ils sont souvent orientés sous forme de jeux, notamment des jeux sérieux, à l'instar de Robocode (O'Kelly & Gibson, 2006), Colobot (Muratet, 2010), Prog&Play (Muratet et al., 2012), etc. Dans ces environnements, l'apprentissage de la programmation se fait en jouant à des jeux centrés sur la résolution des problèmes.

Ces environnements sont porteurs de beaucoup d'avantages potentiels pour des novices en informatique et particulièrement en programmation. Leur structure permet de dispenser certains aspects de la programmation qui sont souvent des sources de difficultés chez les débutants. Dans la programmation de tels environnements, chaque bloc du langage est considéré comme un ou des éléments d'un langage de programmation (Muratet et al., 2011) et diverses notions informatiques sont manipulées : instruction de contrôle, instructions conditionnelles, variables, listes, concepts de séquences, itération, une variable, fonction, opérateur, etc., de telle sorte que la construction d'une programmation informatique se fait par simple glisser-déposer des combinaisons de ces différents éléments de blocs. Deux recherches récentes sur l'initiation à la programmation avec SCRATCH révèlent des potentialités saisissantes de ce langage à blocs (Baron & Voulgre, 2013 ; Tempel, 2013).

Georges Louis Baron et Emmanuel Voulgre montrent les possibilités d'apprendre la programmation de façon séquentielle et se pouvoir se passer de la construction d'un algorithme

pour programmer. Si ce premier avantage permet à l'apprenant une évolution progressive dans son initiation, le deuxième le prive des difficultés de nature algorithmique qui sont importantes et difficiles à franchir chez les novices.

En simulant le jeu entre un chat et une souris, le premier poursuivant ce dernier dans un mouvement aléatoire sur l'écran, Michael Tempel montre comment la structure de SCRATCH est pédagogiquement importante pour un débutant : elle offre une meilleure représentation visuelle du programme, de telle sorte qu'il devient plus compréhensible que le code construit dans un autre langage. De plus, si la forme d'un bloc donné est un indicateur de l'objet de ce programme, la façon dont les blocs sont assemblés et disposés donne une information sur le déroulement du programme. Un autre intérêt important lié à la structure des langages à blocs est la faible probabilité de commettre des erreurs de syntaxe (Tempel, 2013) : « *if we try to put the « point in direction » blocks into the « if » block, it will not go. It's the wrong shape. Similarly, we cannot put the « touching ? » block into the « point in direction » block. In this way, most syntax errors are precluded.* ». Limiter la difficulté liée à la syntaxe, souvent importante chez les débutants, leur permet de ne se concentrer que sur d'autres aspects de la programmation : créer des scripts, y inclure des animations, des sons, etc. Cette expérience initiale dans l'apprentissage de la programmation basée par les langages à blocs offre une base solide pour une transition souple vers des apprentissages futurs de l'informatique, ajoute Tempel.

L'approche de la construction des blocs lors de la programmation est vue comme une clé de prise en main de multiples langages et de leurs caractéristiques (Maloney et al., 2004). Conçus, d'une part pour la promotion d'une « pensée informatique » et d'une « maîtrise pour tous » des compétences informatiques et, d'autre part, pour une motivation des étudiants majoritairement démissionnaires, la programmation par blocs rend l'informatique accessible à beaucoup d'apprenants débutants, susceptibles d'être rebutés par cette science (Temple, ibidem.) : chaque étudiant peut évoluer selon son rythme et acquérir des compétences le permettant la poursuite de la carrière informatique.

Parmi les autres potentialités offertes par la RP, la transversalité occupe une place importante. Si l'approche orientée projet de la RP est privilégiée pour l'apprentissage de l'informatique (Janiszek et al., 2011b) l'intérêt de cette approche par projet de programmation des robots est l'ouverture aux apprentissages pluridisciplinaires. Si beaucoup de recherches se sont intéressées aux apports éducatifs des technologies robotiques, leurs potentialités ont été

étudiées de façon générale, rarement dans le cas d'une technologie spécifique (Gaudiello & Zibetti, 2013).

Au niveau de la licence, une particularité qui fait la singularité de notre étude est de s'intéresser aux deux types de technologies robotiques dans leur spécificité : NAO et LEGO MINDSTORMS NXT. Ces technologies posent néanmoins des problématiques spécifiques.

2.5. Problématique, questionnement et hypothèses

Les étudiants actuellement en licence d'informatique sont en général issus des filières générales scientifiques au lycée. Ils n'ont jamais bénéficié d'une formation scolaire théorique en informatique. De ce fait, ils ne sont finalement pas loin des débutants en informatique. De plus, les recherches montrent que la programmation classique leur pose de nombreuses difficultés en début d'université. Si près d'un tiers d'étudiants échouent dans les premières années d'université, Kaasboll (2002) précise qu'en informatique, la situation est beaucoup plus désastreuse : 25 à 80 % des étudiants sont en situation d'échec en programmation en ce début d'études supérieures. Pour remédier à cette situation, Boudreault & Pregent (2005) proposent un recours aux contextes d'apprentissages attrayants de l'informatique, particulièrement centrés sur des projets intégrateurs consistant au montage et à la programmation des mobiles robotisés. Ces contextes d'apprentissages sont vus comme une solution pouvant susciter la motivation et permettre la réussite de plus d'étudiants et ainsi réduire le nombre d'échecs en informatique en général et en programmation en particulier.

À l'université Paris Descartes, des projets de programmation sont organisés chaque année afin de valider les deux dernières années de licence. Cette section s'intéresse aux apprentissages des étudiants de L3 qui ont choisi la programmation du robot NAO parmi les projets proposés. NAO est un robot particulier. De type humanoïde, il est qualifié de « black box », – non transparent –, et est donc hermétique à la programmation de ses comportements (Kynigos, 2008). Le robot NAO est appelé à exécuter des tâches complexes, généralement confiées aux humains (Espiau & Oudeyer, 2008) : « jouer au basket », par exemple. L'ambition est vaste : le robot doit se déplacer, prendre différentes positions nécessaires, anticiper et prendre des décisions dans des circonstances souvent imprévisibles conditionnées par le contexte environnemental…

La difficulté à proposer des contenus de formation en lien direct avec les applications du monde courant serait à l'origine de la démotivation des étudiants face à la discipline informa-

tique (Muratet, 2010). NAO, l'une des innovations technologiques et pédagogiques, est proposé pour soutenir cette motivation. Des questions se posent : si la programmation classique fait autant problème (échecs et abandons des étudiants en début d'études supérieures), la programmation de NAO arrive-t-elle à faire renaître leur motivation vis-à-vis de l'informatique ? Qu'apprennent – ils et comment les étudiants de licence lors de la programmation des robots ? Ces questions générales visent à identifier les connaissances construites par les étudiants au cours de leurs projets de programmation des robots. Les questions de recherche suivantes se posent : quelles sont les motivations des choix de sujet de projet chez les étudiants ? Quelles sont les connaissances construites au cours des projets de programmation de NAO ? Quelles sont les stratégies utilisées et comment sont-elles mises en œuvre pour gagner la compétition ? Quelles sont les difficultés rencontrées au cours de ces activités ?

3. Une discipline informatique sans enseignants spécialisés ?

Il y a une distance entre le curriculum prescrit et celui réalisé (Martinand, 1994). Cette différence est souvent due à l'espace de son interprétation et par conséquent, sa mise en œuvre chez les enseignants au cours de la transposition didactique[15]. Les acquisitions des élèves en dépendent parce qu'elles sont liées aux pratiques de leur enseignant, et donc de son rapport aux savoirs enseignés (Tiberghien & Malkoun, 2007).

Beaucoup de chercheurs, malgré la mise en place de cette discipline, gardent une inquiétude sur le contexte de sa mise en œuvre et des questions se posent, notamment comment *« peut-on fonder une didactique hors d'un champ de connaissance déjà bien installé, sans corps enseignants bien formé ? »*. Pour eux, le succès de son enseignement n'est pas garanti, faute d'un corps enseignant spécialiste de la discipline, une situation qui interroge sa pérennité.

Les expériences du passé les poussent à s'interroger si la situation ne risque de se reproduire. Par exemple, Jean Pierre Archambault relève que parmi les raisons qui ont influencé la suppression de l'option informatique des années 1980, figure la domination des enseignants de mathématiques qui représentaient le double de ceux de l'option dans le recrutement et la discipline est devenue élitiste (Archambault, 2013). Qu'est-il aujourd'hui ? La situation n'est-elle pas la même ? A-t-elle beaucoup évolué ? Quelles sont les raisons qui poussent à espérer le succès de cet enseignement ?

15 Développé par Chevallard (1985, 1991), le processus de transposition didactique désigne le passage d'un savoir savant à un savoir enseigné

Avec l'absence d'une agrégation d'informatique en France (Beaudouin-Lafon, 2010), les enseignants appelés à dispenser ce cours sont les actuels enseignants du cours de mathématiques et, spécialistes de mathématiques dont la plupart parmi eux n'ont eu aucun module informatique au cours de leur cursus universitaire. Les enseignants sur lesquels on compte pour cette spécialité ISN ne sont pas des spécialistes de l'informatique. Le bulletin officiel du 30 septembre 2011[16] en donne leurs profils : ce sont des enseignants volontaires en exercice qui, normalement dispensent les disciplines scientifiques déjà existantes :

> *« L'enseignement de spécialité ISN (Informatique et Sciences du Numérique) en terminale S peut être assuré par des professeurs de mathématiques, de sciences physiques, de sciences de la vie et de la terre ou de sciences et techniques industrielles »*[17].

Dans le contexte actuel de l'institutionnalisation de l'ISN comme discipline scolaire, des personnes intéressées par cet enseignement attendent impatiemment l'évolution de cette discipline (Baron, 2012, p. 85) que ce soit *« en termes de contenus réellement enseignés, des points de vue des enseignants et des élèves et d'insertion dans l'ensemble de l'offre scolaire »*.

Cette impatience partagée semble liée au contexte un peu paradoxal dans laquelle la discipline informatique a été institutionnalisée. D'une part, il y a cette discipline avec un curriculum ambitieux et des attentes cruciales et, d'autre part, son introduction dans un contexte de manque de ressources humaines telles que des enseignants spécialisés pour son enseignement. Si l'informatique n'est pas des mathématiques, des inquiétudes chez certains auteurs naissent quant aux compétences des enseignants non spécialistes à dispenser cet enseignement. Gilles Dowek, au cours d'une conférence organisée par la Société Mathématiques de France (SMF) déclare que :

> *« malgré la forte interaction entre les disciplines, donner l'enseignement de l'informatique aux mathématiciens pose problème car ceux-ci n'ont pas la même vision des choses qu'un informaticien »*[18]

16 http://www.education.gouv.fr/pid25535/bulletin_officiel.html ? cid_bo=57998, consulté le 27 décembre 2013

17 http://www.ac-paris.fr/portail/jcms/p1_563741/formation-2012-2013-pour-les-enseignants-souhaitant-enseigner-lisn

18 http://smf.emath.fr/files/text_like_files/crtrinfo25septembre.pdf, consulté le 25 mars 2013

Si le profil d'élèves à former n'est pas celui des spécialistes de l'informatique, ce profil d'enseignants pousse à se questionner sur leurs réelles compétences comme d'ailleurs celles de leurs élèves à s'approprier cet enseignement, étant donné que chez les élèves aussi il y a une absence totale de prérequis en informatique.

La situation problématique ne se limite pas au niveau de l'ISN. Les nouveaux programmes de mathématiques contenant des savoirs informatiques au lycée sont confiés aux professeurs de mathématiques pour enseignement.

Nous partons du fait que, les mathématiques et l'informatique sont deux sciences différentes. Elles ont leurs façons spécifiques de poser des questions et méthodologiques d'aborder les contenus dans leur enseignement. Nous nous questionnons sur la prise en compte des l'informatique par les spécialistes des mathématiques dans leurs pratiques professionnelles.

Bref, nous sommes face à des programmes ambitieux de lycées, face à des enseignants non formés ou non suffisamment formés à l'enseignement de l'informatique (Archambault & Dowek, 2009). Si l'introduction de l'informatique au lycée permet d'éviter la rupture actuelle entre la fin du collège, sanctionnée par le B2i et l'enseignement supérieur (Biseul, 2009), étant donné que les enseignants chargés d'un cours devraient en avoir reçu une formation spécifique (Duchâteau & Sass, 1993), attribuer l'enseignement de l'informatique aux non spécialistes de la discipline, risque de conduire à une nouvelle fracture entre les écoles en fonction de la disponibilité ou pas des enseignants formés en informatique.

Un autre problème, non loin du précédent, est celui des équipements en postes informatiques. L'enseignement de l'informatique ne risque-t-il pas de créer davantage d'inégalités scolaires en termes d'équipements entre des écoles d'une part entre les élèves des écoles bien équipées et ceux des écoles qui le sont moins avec le risque de dispenser une informatique théorique même si les enseignants motivés et capables étaient disponibles.

Néanmoins, même si *« l'informatique peut être aussi considérée comme une discipline carrefour développant des liens forts avec d'autres disciplines… »* (Grandbastien, 2012), il n'est pas évident que tout enseignant de n'importe quel profil puisse l'enseignée. Monique Gandbastien précise même qu'elle est exigeante par rapport à d'autres disciplines au lycée : elle demande un corps d'enseignants spécialisés et un corpus de connaissances qui sont des conditions encore non remplies, même si son enseignement ne vise pas à former des spécialistes tel que le stipule le programme défini dans le Bulletin officiel spécial n° 8 du 13

octobre 2011 : son objectif « *n'est pas de former des experts en informatique, mais plutôt de fournir aux élèves quelques notions fondamentales et de les sensibiliser aux questions de sociétés induites. Il s'agit d'un enseignement d'ouverture et de découvertes des problématiques actuelles, adapté à la société d'aujourd'hui, qui valorise la créativité et contribue à l'orientation »*[19].

Le rapport de l'académie des sciences (2013) sur l'enseignement de l'informatique en France s'interroge sur la question de la formation des enseignants à qui serait confié cet enseignement. Une situation un peu paradoxale entoure les circonstances de l'enseignement de l'option ISN : il met en jeu un ensemble d'enseignants qui sont issus de plusieurs domaines disciplinaires, et donc l'ensemble des disciplines, une situation qui contraste avec le fait que l'informatique, comme science et discipline autonome, devrait avoir un corps d'enseignants bien formés et bien spécialisés pour la discipline informatique, comme c'est bien le cas pour tout autre discipline enseignée au lycée. Normalement, comme ce rapport de l'académie des sciences le souligne, comme dans toutes les autres disciplines, un professeur d'informatique au lycée devrait avoir un niveau minimum correspond à une formation en informatique de quatre ou cinq après le baccalauréat. Mais, comme l'indique ce rapport, les raisons historiques ont fait que l'enseignement de l'informatique en France ne soit considéré comme l'enseignement des autres disciplines : c'est que cet enseignement a été confié aux non-spécialistes de l'informatique. Cette façon de procéder est vue par certains auteurs comme une fantaisie qui ne peut pas amener cet enseignement au succès (Abiteboul et al., 2013) :

> *« Il est fantaisiste de penser que l'on peut prendre un enseigeant d'une matière quelconque et le transformer en professeur d'informatique, même s'il sait utiliser un ordinateur. Ce type d'approche – guidé par la volonté d'agir vite et à faible coût – ne peut conduire qu'à des échecs ».*

Selon eux, si des matériels et des logiciels sont indispensables dans la poursuite et l'accompagnement de l'enseignement de l'informatique tout comme les autres disciplines, la formation des enseignants devrait être prioritaire sur le matériel et le logiciel. Ils soulignent que la centration sur la formation permettra d'éviter la prochaine fracture qui n'est pas celle des équipements informatiques : *« la vraie fracture la plus dommageable pour les élèves sera*

19 http://www.education.gouv.fr/pid25535/bulletin_officiel.html ? cid_bo=57572 : Enseignement de la spécialité d'informatique et science du numérique de la série scientifique – Classe terminale.

entre les établissements qui disposeront d'enseignements bien formés à l'informatique et ceux qui n'en auront pas »

Cette situation paradoxale suscite un certain nombre de questionnements. Quelles sont par conséquent les compétences des enseignants face à cet enseignement qui, normalement, devrait être confié aux spécialistes de l'informatique. Malgré la formation qui leur serait donnée, leur spécialité d'origine, ne risque-t-elle pas d'avoir un impact sur l'enseignement de cette option ? Quelles sont les conséquences de l'attribution de la spécialité informatique aux non-spécialistes de la discipline, sur l'enseignement de cette discipline elle-même d'une part, et sur son apprentissage chez les élèves, d'autre part ?

4. Méthodologies et données recueillies : le choix d'une approche qualitative

Nous nous intéressons à l'appropriation des savoirs informatiques chez les débutants. L'interrogation de la littérature de recherche a d'abord montré que le domaine *algorithmique et la programmation* est indispensable pour cette initiation chez les lycéens de l'enseignement général. Ensuite, un certain nombre de savoirs fondamentaux ont été identifiés : variable, algorithme, instruction, boucle… Enfin, nous avons choisi, parmi eux, deux savoirs jugés incontournables pour cette initiation en informatique : variable et boucle. Le tableau donné en annexe (Annexe 1) montre comment l'analyse a été effectuée.

Dans cette section, nous présentons la méthodologie utilisée pour étudier l'appropriation de ces savoirs informatiques en général et les variables et les boucles en particulier.

S'approprier un savoir est tout un processus. Arriver à identifier, par observation le processus en jeu de la construction de connaissances et par conséquent les éléments à observer pour un tel processus discontinu est souvent problématique (Fluckiger, 2007). Cela nécessite un choix méthodologique approprié.

Une méthodologie qualitative de type ethnographique a été privilégiée pour répondre à nos questions de recherche pour ses quatre avantages (Marcel, 2002) :

- la nécessité d'une observation directe et prolongée, la seule à même de permettre d'accéder à la description d'un objet peu connu dans ses éventuelles spécificités ;

- la volonté de combiner des éléments facteurs et observables avec le sens que leur donnent les acteurs, une situation permettant de qualifier le dispositif d'« observation participante » ;
- le recours à la technique de notes de terrain rédigées « à chaud » juste après l'observation et la méthodologie inductive de traitement des données correspondant à la visée exploratoire
- et enfin, la théorisation des matériaux empiriques qui permet d'accéder à la connaissance des pratiques, constitue une étape propédeutique à la stabilisation de l'instrumentation.

4.1. Public

Cette recherche s'intéresse aux apprentissages de deux types d'apprenants, tous débutants, mais de niveaux d'études différents : les lycéens de l'enseignement secondaire scientifique et les étudiants de licence de l'université. Elle s'intéresse aussi aux pratiques de leurs enseignants. Ces choix ont été motivés par le fait qu'ils sont tous en début d'une initiation informatique. Les élèves des lycées scientifiques n'ont jamais connu une discipline informatique au cours de leur scolarité secondaire. Actuellement, avec le début de l'enseignement de l'informatique au lycée, ils sont novices en informatique[20].

De même, les étudiants actuellement en licence d'informatique, ne nous semblent pas, eux aussi, loin des débutants. Au cours de leur scolarité secondaire, ils n'ont jamais suivi, comme les précédents, un quelconque cours d'informatique ni au collège ni au lycée. Bien qu'ils appartiennent à des niveaux d'enseignement différents, les deux populations semblent, à notre avis, de niveau très rapproché en informatique. Il nous semble qu'il y a une faible différence dans leurs rapports à l'informatique. La seule différence est le niveau de formation : les uns sont au lycée, les autres à l'université. D'où le choix d'une approche comparative entre ces deux groupes d'apprenants.

Si, selon nous, les deux publics sont très proches, au point de vue de leurs rapports à l'informatique, ils ont, entre eux, quelques différences. Les étudiants de licence 2 (L2) et de licence 3 (L3) ont eu une année commune de mathématiques ET informatique en licence 1 (L1).

20 Une discipline « Informatique et Science du Numérique » (ISN) a été introduite en terminale S en 2012 et a suivi des éléments d'algorithmique et de programmation, introduits au lycée depuis la classe de seconde en 2009.

Ceci constitue une différence fondamentale avec les lycéens qui eux, sont, au vrai sens du terme, « débutants » en informatique en général et en algorithmique et programmation en particulier.

Au niveau lycée, quatre établissements (notés A, B, C et D) sont intervenus dans cette recherche. Tous sont publics. À l'exception du lycée C qui fait partie de l'académie de Paris, les trois autres sont de la région parisienne. Le tableau suivant présente les principales caractéristiques de ces établissements :

Lycée	Type de lycée	Élèves	Professeurs
A	Polyvalent	950	120
B	Polyvalent	1500	150
C	Général	700	80
D	Général et technologique	825	90

Tableau II.1: Caractéristiques des établissements

Des problèmes de terrains ont fait qu'au lycée B seuls des entretiens avec les élèves d'ISN et leur enseignant ont été conduits après la soutenance du projet en fin d'année. Pour constituer la totalité du corpus, il a fallu alors attendre l'année scolaire suivante, 2013-2014, pour pouvoir utiliser la méthodologie ethnographique à travers des observations de classe avec cette spécialité : ceci a été possible dans le lycée C. Enfin, dans le lycée D, deux séances de cours de deux heures chacune portant sur l'enseignement des notions algorithmiques ont été observées dans deux classes : l'une de seconde et l'autre de terminale. Dans ce lycée, il n'existe pas d'ISN. Les classes sont tenues par une enseignante spécialiste des mathématiques et formatrice. La singularité de ses pratiques est liée à l'approche utilisée : la pédagogie de groupe, non existante dans d'autres lycées visités.

Ainsi, les lycées A et D répondent à nos questions de recherches pour l'informatique et l'algorithmique en mathématiques et, les lycées B et C pour la spécialité ISN.

À l'université, la recherche porte sur des pratiques des étudiants de licence en informatique à l'université Paris Descartes Sorbonne Cité. Durant la première année de la thèse, 2011-2012, le travail de collecte de données a eu lieu au niveau de la licence. Les étudiants de licence 2 et licence 3 (L2 et L3) en informatique, à l'université Paris Descartes, étaient en projets de programmation des robots qui ont lieu au cours du deuxième semestre. Les années

suivantes, 2012-2013, le recueil de données a continué au secondaire dans les lycées A et B et puis, 2013-2014, au lycée C pour la spécialité ISN en terminale.

Durant la première et deuxième année de thèse, 2011-2012 et 2012-2013, dans le cadre de la collecte des données, nous avons suivi des formations à l'algorithmique destinées aux enseignants de mathématiques. Ces formations étaient organisées à l'Université Didérot – Paris 7 et, dispensées par des enseignants-chercheurs en collaboration avec des enseignants – formateurs, spécialistes des mathématiques.

4.2. Instrumentation et déroulement

Trois outils de collecte de données ont été choisis : le questionnaire, les entretiens semi-directifs et les observations directes et prolongées de type ethnographique. Après observation des pratiques de notre population (formateurs, enseignants d'université ou de lycée, étudiants et élèves) en cours d'activité, des entretiens d'explicitation ont été aussi menés pour avoir plus d'informations sur leurs pratiques. Pour répondre à la première question portant sur des savoirs informatiques incontournables pour débutants en informatique, une analyse de la littérature de recherche a été conduite puis complétée par les entretiens auprès des formateurs d'enseignants. La deuxième question de recherche concerne les savoirs informatiques prescrits. Pour y répondre, une analyse des textes officiels (programmes et guide d'accompagnement) et des manuels scolaires pour élèves, mais aussi utilisés par leurs enseignants, a été effectuée. Quant aux questions 3, 4 et 5, relatives aux pratiques des apprenants et leurs enseignants, les outils tels que le questionnaire, l'observation et les entretiens semi-directifs, ont été utilisés pour y répondre. Les outils utilisés ont ainsi été tous complémentaires. Par exemple, en condition réelle d'apprentissage, si l'observation a été utilisée pour voir ce que font des élèves ou des étudiants au cours de leurs apprentissages lors de la résolution des problèmes, essentiellement en contextes de projets, les entretiens réalisés après, contribuent à apporter plus de précision en explicitant ce qu'ils font et comment ils font.

Le recueil de données s'est chaque fois appuyé sur une grille de collecte, préalablement élaborée à cet effet, relative à chaque population. Les différentes grilles ou canevas sont présentés dans la sous-section suivante.

a. Observation directe et prolongée

L'observation, comme outil de recueil de données, a été utilisée pour toute la population (élèves, étudiants, enseignants) sauf au lycée B où, suite à l'avancée de l'année scolaire, seuls les entretiens ont été conduits. Relativement à l'observation, deux grilles ont été utilisées : une grille d'observation des pratiques des enseignants et des apprentissages de leurs élèves au lycée (Annexe 2) et une grille d'observation en contexte de projets des étudiants de licence et leurs professeurs (Annexe 3). Au lycée, la grille est restée la même pour les séances d'informatique en mathématiques que pour celles de la spécialité ISN. De même, à l'université, la grille d'observation est restée la même pour les étudiants, quel que soit le type de technologie robotique programmée.

L'observation directe, un outil essentiel dans l'entreprise de collecte de données dans un contexte ethnographique, a été utilisée dans trois lycées A et D (avec les élèves hors spécialité ISN) et C (avec les élèves de la spécialité ISN). Les trois lycées avaient une configuration différente de la salle de classe/salle informatique. Dans le lycée A, la salle de classe qui sert aussi de salle informatique, les ordinateurs sont disposés à la périphérie et, l'élève sur la machine fait face au mur, des chaises au centre de la classe font face au tableau noir.

Au lycée C, les ordinateurs sont disposés sur les tables des élèves, face au tableau noir. Les élèves sont devant les ordinateurs quelle que soit la nature du cours en question, ce qui favorise la distraction des élèves, tentés d'aller sur Internet sans consignes de la part de l'enseignant.

Au lycée D, la salle de classe ne contenant qu'un ordinateur de l'enseignant mais, non utilisé pour cette séance. Les élèves sont se disposent librement en groupes de quatre, deux à deux face à face. Deux classes de seconde et première avec 32 élèves chacune, ont été observées. Dans chaque classe, un seul groupe au choix a été strictement suivi au cours de toute la séance.

Toutes les séances de cours (théorique ou pratique), durent généralement d'une à deux heures (une séance de 55 minutes ou deux séances d'une heure cinquante minutes (1h 50 min). Lors des observations, des notes étaient prises au fur et à mesure.

Les contextes des observations sont différents pour les étudiants en projets. Les élèves étaient tout le temps avec leurs enseignants au cours des projets. Cette situation a permis beaucoup d'interactions entre l'enseignant et ses élèves : ces derniers étaient beaucoup pris par la main,

contrairement aux étudiants qui étaient presque seuls. Pour les étudiants, les créneaux dédiés aux réunions avec leur encadrant étaient pratiquement les seules occasions de le rencontrer même s'ils pouvaient le contacter au moyen d'un message électronique en cas de nécessité.

Les étudiants en programmation classique travaillaient le plus souvent chez eux. Par contre, ceux travaillant sur la robotique avaient lieu de travail : la salle robotique. Deux moments ont été particulièrement privilégiés avec les étudiants programmant les robots : lors du travail des étudiants en salle de robotique et lors des réunions entre un groupe donné d'étudiants et leur encadrant. En salle de robotique, les groupes avaient deux créneaux différents d'accès. Mais, selon la disponibilité des deux robots, les deux groupes pouvaient se rencontrer et partager la salle, chacun sur son robot et son travail. L'ancrage des observations a porté sur l'organisation des groupes dans leurs apprentissages et leurs interactions : entre les membres d'un groupe, entre les groupes, entre les étudiants et le système ordinateur-robot. Il a aussi été question d'observer les stratégies utilisées par les différents groupes d'étudiants dans la programmation des tâches, les difficultés et contraintes vécues et les connaissances utilisées d'une part et celles construites d'autre part. En réunion entre étudiants-encadrant, les observations s'intéressaient davantage aux types de problèmes soumis par les étudiants à leur encadrant d'une part et au type d'aide et à la façon dont cette aide était donnée par l'encadrant.

Après chaque séance d'observation, un journal de terrain était fraichement élaboré. Son élaboration se fait en deux étapes. À partir des notes prises à la volée sur le terrain d'observation, concernant le sujet en question. Il comprend toutes les informations issues des observations recueillies notamment en rapport avec l'organisation des groupes en contexte de travail, les connaissances mobilisées et/ou construites, les stratégies mises en œuvre, les interactions entre les membres d'un groupe ou entre des groupes ou entre les apprenants et leurs enseignants, les incidents critiques au moment de la séance conduisant à des actions des uns et réactions des autres, les difficultés…

La conception du journal de terrain permet d'organiser les notes prises sur terrain. De ce fait, sa validité dans le corpus est comparable à celle d'une transcription d'un entretien (Fluckiger, 2007).

Néanmoins cette approche qualitative de type ethnographique connaît des limites. Il est difficile de tout retenir ou voir. Par ailleurs, certains enseignants par exemple demandaient d'éteindre les dictaphones pour dire ce qu'ils n'ont pas osé dire étant enregistrés, quant aux

étudiants, ils n'étaient pas très libres dans leurs échanges et semblaient un peu gênés par notre présence dans leur liberté conversationnelle. Avec le temps, une sorte de familiarité s'est établie entre eux et nous. De plus, certains élèves, avant de faire certaines critiques à l'endroit des pratiques du professeur, demandaient d'abord si l'enregistrement sera écouté par le professeur en question[21].

b. Entretiens semi-directifs et groupés

Canevas d'entretien

La méthode d'entretiens semi-directifs a été privilégiée : elle offre des opportunités aux différents interviewés d'exprimer leur point de vue (Darricarrère & Bruillard, 2010). Ces derniers, en répondant librement aux thèmes proposés, révèlent en même temps d'autres points pouvant être intéressants auxquels l'intervieweur n'avait pas pensé dans la construction de la grille d'entretien. L'intervieweur, quant à lui, a aussi les possibilités de relancer des questions pour avoir plus de précisions sur les nouveaux points évoqués qu'il juge importants.

Pour chaque catégorie de population interviewée, un canevas d'entretien a été préalablement conçu : un canevas pour les élèves des lycées en mathématiques (Annexe 4), un autre pour les élèves d'ISN (Annexe 5), un autre pour les professeurs de mathématiques (Annexe 6), un autre pour les professeurs d'ISN (Annexe 7), un autre pour les enseignants des filières industrielles (Annexe 8), un autre pour les étudiants de licence (Annexe 9) et un autre pour les enseignants d'informatique en licence à l'université (Annexe 10).

Entretiens proprement dits

Préalablement aux entretiens, avec les enseignants de mathématiques, un questionnaire léger a été soumis à des personnes en formation à l'algorithmique. Ce dernier a été mobilisé pour le recueil des données auprès des enseignants de mathématiques, après leur stage de formation à l'algorithmique. Cet instrument devait aider à comprendre leurs déclarations à propos du stage. En effet, il est conçu de telle sorte que le répondant décrive ses acquis autant en contenus qu'en méthodologie mais aussi ses représentations et ses pratiques en algorithmique en dressant un parallélisme par une approche comparative de leurs rapports à l'algorithmique avant et après la formation. Cet instrument a été appliqué à chaud, juste après la formation,

21 Ces lacunes sont confirmées par Schwartz (1989) qui souligne les incidences de l'utilisation de certains outils tels que les magnétophones dans la collecte de données d'enquête : l'une des composantes étant l'impossibilité de recueillir certaines informations.

pour permettre d'avoir un retour fidèle sur le stage et ses effets sur les pratiques projetées dans la suite. Le questionnaire est en annexe (Annexe 11).

Dans le recueil des données, les observations ont été complétées par des entretiens. Si les entretiens individuels peuvent être fructueux, les entretiens de groupes semi-directifs ont été privilégiés pour avoir plus de précisions et d'explicitations sur les pratiques observées. L'intérêt de cette méthode repose globalement sur la fécondité des interactions qu'ils engagent entre les participants. Ils sont l'un des meilleurs moyens permettant de co-construire avec les acteurs le sens qu'ils donnent à leurs conduites, pour investiguer la façon dont ils se représentent le monde, ce qui suppose pour le chercheur de reconnaître qu'ils sont les mieux placés pour en parler (Morrissette, 2011).

Dans les deux lycées B et C concernés par l'ISN, cette option est tenue respectivement par un enseignant spécialiste de génie électrique et informatique industrielle d'une part et, une enseignante à double spécialité : mathématiques et sciences des matériaux. Les sujets de projets des élèves sont divers. Tous proposés par l'enseignant au lycée B, plus de la moitié des sujets ont été proposés par les élèves eux-mêmes au lycée C. Ils sont présentés en annexe (Annexe 12).

Ce recueil a eu lieu pendant deux années scolaires différentes mais consécutives : 2012-2013 pour le lycée B et 2013-2014 pour le lycée C. Au lycée B, seul les entretiens semi-directifs avec l'enseignant et ses élèves, ont été utilisés[22]. Au lycée C, les entretiens sont venus compléter les observations de pratiques de classe et de projets initialement effectuées. Dans les deux établissements, les entretiens ont été menés juste après la soutenance des projets et ont porté sur le cours de toute l'année et pas seulement sur les projets. Pour des raisons pratiques, les entretiens avec les élèves étaient individuels. En effet, si les projets étaient souvent faits en binôme ou en trinôme, leurs soutenances étaient individuelles, ce qui rendait impossible la disponibilité des membres du groupe au même moment. Les entretiens avaient pour but d'avoir plus de lumière sur des rapports des élèves aux mathématiques et à l'informatique ; des apports des mathématiques dans leur apprentissage de l'informatique ; les effets de la spécialité d'origine de l'enseignant dans ses pratiques, les pratiques des élèves et les motivations de leur choix de la spécialité ISN…

22 La permission d'accès au terrain a été octroyée au troisième trimestre. Les élèves étaient en fin de projet et se préparaient à leur soutenance

À l'université, les entretiens ont concerné les étudiants puis leurs enseignants-encadrants. Pour chaque groupe d'étudiants en robotique, les entretiens ont été périodiques. Trois moments ont particulièrement été privilégiés : le début, le milieu et la fin des projets. Les premiers entretiens ont eu lieu en janvier, au début des projets. Les entretiens visent à avoir plus d'informations sur les aspects préparatoires du travail à faire, notamment les cahiers des charges et les prévisions organisationnelles anticipées du projet. Les entretiens conduits visent à trouver des réponses aux quatre questions suivantes. D'abord, comment les étudiants ont choisi leur sujet de projet et pour quelles motivations. Ensuite, ils visent à recueillir, selon eux, l'intérêt de ces projets mais aussi pourquoi les enseignants (à l'Université Paris Descartes) ont jugé intéressant de leur proposer de tels projets sur la robotique. Enfin, ces entretiens en début de projet cherchent à rendre compte de leur anticipation dans le travail de groupe à faire : des contenus et des connaissances qu'ils pensent mobiliser, de leur organisation prévue, des difficultés auxquelles ils s'attendent à être confrontés, leur état d'esprit en début ces projets…

À mi-parcours des projets, fin du mois de mars, d'autres entretiens ont eu lieu. Ils visaient l'état des lieux de la mise en œuvre du cahier des charges, de l'évolution de leur travail par rapport à leurs objectifs de départ, des difficultés réellement vécues par rapport à celles auxquelles ils s'attendaient, des stratégies mises en œuvre au niveau de la gestion du projet pour être dans les délais de remise, mais aussi pour gagner la compétition et enfin, l'état d'esprit dans lequel les étudiants se trouvent à ce moment du projet. Les derniers entretiens avec les étudiants ont eu lieu après la soutenance des projets en début du mois de juin. Ils visaient l'établissement d'un bilan chez les étudiants à la fin de tout un semestre de travail en projet : leurs appréciations du travail fourni (satisfaction ou pas), du module proposé en soi, des stratégies mises en œuvre dans la programmation des robots et des motivations de leurs choix, les difficultés réellement vécues, leurs critiques sur ces projets (adéquation sujet de projet/niveau d'enseignement, conditions de travail et éventuelles participations à la compétition robotique qui était prévue en juin de la même année scolaire et stratégies pour la gagner. Pour compléter l'investigation, un autre entretien semi-directif a été mené en groupe, en juin, auprès des deux enseignants d'informatique et encadrants desdits projets, après leur soutenance. Ces enseignants sont notés E1 et E2 dans la suite.

En résumé, des observations ethnographiques, trois entretiens semi-directifs et périodiques menés avec les étudiants en groupes ont été réalisés à des fins d'analyse. Les données re-

cueillies ont été complétées par un entretien semi-directif en groupe avec deux enseignants-encadrants de ces projets.

Au lycée, si les entretiens en groupe ont été conduits avec les autres élèves, cet outil n'a pas été utilisé avec ceux de la spécialité ISN : les entretiens ayant lieu à la fin de l'année après la soutenance des projets. Comme la soutenance était individuelle, il n'y avait que la possibilité de voir un seul élève à la fois. Il en est de même de tous les enseignants du lycée interviewés.

Dans le même lycée A, en vue d'avoir un regard sur leurs pratiques en informatique des enseignants des filières industrielles, trois d'entre eux ont été interviewés. Deux parmi eux ont des statuts particuliers. L'un, en plus d'être enseignant, est à la fois chercheur et formateur au Plan Académique de Formation (PAF). L'autre est aussi formateur des enseignants de mathématiques à l'algorithmique dans l'académie de Créteil. Le statut de chacun n'étant pas connu d'avance, il n'a influencé en rien leur choix.

4.3. Corpus

a. Niveau lycée

Les données recueillies et analysées proviennent de quatre lycées notés A, B, C et D dans la suite. Le lycée A a fourni tout le corpus relatif aux classes de seconde à la terminale, la spécialité ISN non comprise. Les lycées B et C ont fourni des données relatives à la spécialité ISN seulement. Le corpus est donc composé des différents types de données suivantes :

- des textes officiels : un programme d'algorithmique et son guide d'accompagnement de la classe de seconde d'une part, et un programme de la spécialité ISN d'autre part.
- des manuels scolaires (12). Parmi eux, il y a 11 manuels de mathématiques. Trois collections de manuels (transmath, mathsrepères, math'x) sont utilisées par les enseignants interviewés au lycée. Chaque collection comporte trois manuels en raison d'un manuel par niveau, soit 12 manuels. Le manuel de la collection transmath n'était pas disponible dans le lycée en question. Donc, 11 manuels seulement de mathématiques ont été analysés, plus un manuel de la spécialité ISN ;
- des notes d'observations des pratiques des formateurs, des enseignants, des élèves et des étudiants ;
- des productions des groupes d'élèves respectivement de seconde et terminale en travail de groupe au lycée D.

- des données d'enquête par questionnaires avec 28 enseignants de mathématiques au lycée. Parmi eux, il y a respectivement 15 et 13 qui ont répondu aux questionnaires après leur formation en algorithmique, durant des années scolaires 2011-2012 et 2012-2013,
- des transcriptions d'entretiens de :
  - 3 formateurs à l'algorithmique d'enseignants de mathématiques ;
  - 5 enseignants ayant bénéficié du stage de formation en algorithmique ;
  - 7 enseignants de mathématiques n'ayant pas eu de formation continue en algorithmique ;
  - 3 enseignants des filières technologiques et industrielles ;
  - 21 groupes de 60 élèves (40 de seconde, 8 de première et 12 de terminale) issus du lycée A dont les enseignants n'ont pas suivi de stage en algorithmique ;
  - 20 élèves d'ISN issus de deux lycées B et C avec respectivement 9 et 11 élèves. Contrairement aux autres lycéens, les entretiens avec ceux d'ISN ont été conduits individuellement pour des raisons déjà données.

Les effectifs d'élèves des deux classes d'ISN des lycées B et C sont présentés dans le tableau suivant :

Lycée	**Effectif de la classe**		**Effectif interviewé**		**Total interviewé**
	Garçons	**Filles**	**Garçons**	**Filles**	**Garçons + Filles**
B	10	0	9	0	**9**
C	18	1	11	0	**11**
Total	28	1	20	0	**20**

Tableau II.2: Effectifs en ISN par lycée et par sexe

Cette répartition montre une fréquentation presque nulle des filles dans l'option ISN : aucune fille au lycée B, alors qu'elle y en a qu'une seule au lycée C[23]. Cette fille qui refait l'année, a abandonné au cours du premier trimestre, avant le début de notre recueil de données.

23 Cette fille refait l'année selon l'enseignante. Elle a abandonné au premier trimestre : elle n'a jamais suivi les cours après les vacances de fin d'année 2013 et personnellement je l'ai jamais vue en classe.

b. Niveau licence

Le corpus en licence informatique à l'université est composé des données suivantes :

- De notes d'observations des pratiques d'étudiants prises lors de leur travail de groupe en projets de programmation des robots ou lors des réunions d'échanges entre un groupe donné avec son encadrant ;
- Des transcriptions d'enregistrements d'entretiens en groupes avec 20 étudiants de licence. Parmi eux, il y a 13 étudiants de licence 2 et 7 de licence 3, tous en projets de programmation des robots. Ceux de L2 travaillent sur le robot LEGO MINDSTORMS NXT et ceux de L3 sur NAO ;
- Une transcription d'un enregistrement d'entretien en groupe avec deux enseignants d'informatique à l'université Paris Descartes.

Les données ont été analysées. Le point suivant indique comme cette analyse a été effectuée.

4.4. Méthodologies d'analyse

a. Analyse lexicale et sémantique du prescrit

Les textes officiels à analyser sont composés des programmes et guide d'accompagnement en rapport avec l'informatique en mathématiques mais aussi pour la spécialité ISN.

Relativement à l'informatique sur le programme de mathématiques, la procédure de notre analyse est d'abord lexicale. Elle a consisté à repérer dans les textes officiels les notions en rapport avec l'algorithmique ou faisant référence à l'algorithmique. Ces notions ou expressions ont été classifiées dans les catégories, constituées selon leur champ sémantique : technique, activité, outil…

Nous avons aussi repéré dans les textes les termes ou expressions relatifs aux TIC et les noms de logiciels proposés. Nous avons finalement mis en relation la terminologie soulignée précédemment et les savoirs et savoir-faire comme compétences attendues de l'élève selon le prescrit des textes officiels. Cela nous a permis de déterminer la place qui revient à l'enseignement de l'introduction de l'algorithmique dans le nouveau programme de mathématiques au lycée.

Enfin, nous avons repéré des expressions langagières qui montrent l'importance à donner à l'argument proposé. Ces expressions ont un sens qui va de la recommandation à une plus

forte incitation : « *dans la mesure du possible…* », « *le cas échéant* », « *il serait souhaitable…* », « *une piste intéressante est…* », « *il est intéressant de…* », « *on aura intérêt à…* », « *peut avantageusement avoir…* », « *gagne à être mise en œuvre…* », « *l'accent est systématiquement mis sur…* », « *il est indispensable de…* », « *en particulier…* », « *il est important de…* », « *il importe particulièrement que… », « il convient de… », « il conviendrait de… »*, etc.

b. Analyse comparative des manuels scolaires

Notre étude s'intéresse à l'enseignement/apprentissage de l'informatique chez les débutants mais aussi aux rapports à l'informatique des enseignants spécialistes des mathématiques dans leurs pratiques professionnelles. Pour y arriver, une des entrées a été l'analyse des manuels scolaires. Si l'analyse des textes officiels comme curricula prescrits, peut nous permettre de déterminer des rapports institutionnels aux objets de certaines disciplines enseignées, en général, nous supposons qu'ils ne suffisent pas à eux seuls pour offrir toute la lumière à ce sujet, ce qui justifie en quelque sorte notre choix de l'analyse des manuels en tant que curricula potentiels pour reprendre le terme de Jean Louis Martinand (Martinand, 2000).

Dans cette section, nous nous intéressons d'abord aux manuels scolaires utilisés par les enseignants et par les élèves. L'objet est d'analyser le rapport institutionnel à travers les manuels par rapport aux savoirs proposés. Il est question d'analyser comment les éditeurs ont pris en compte l'esprit des programmes et comment ils aident et par quelle forme d'aides ils mettent à la disposition des enseignants, à travers quels discours et quels contenus proposés dans les manuels. Nous avons particulièrement voulu trouver des réponses au questionnement suivant :

- Comment, par rapport au programme, les manuels se positionnent et éclairent l'enseignant dans ses pratiques ? Quelles formes d'éclairages et d'aides sont-ils prévues pour des enseignants et/ou des élèves dans leurs pratiques ?
- Quelles sont les tâches proposées dans le manuel à destination des élèves ? Comment sont – elles abordées ? Avec quelles technologies numériques ou langages ?

Dans notre analyse, une combinaison de méthodes a été adoptée. La première méthode est comparative. Elle est justifiée par l'existence de plusieurs manuels scolaires qui traitent du même contenu dans un même pays, ce qui permet de voir différentes interprétations possibles du même programme pour différents éditeurs (Bernard et al., 2007). Avec cette méthode, une

grille d'analyse, inspirée de la grille de la commission[24] INTER-IREM/APMEP[25] (2000), a été confectionnée. Cette grille de l'IREM/APMEP propose les critères à prendre en compte pour choisir le manuel. La grille a été adaptée pour correspondre à nos questions de recherches (Annexe 13)

Pour chaque manuel, l'analyse a été faite en deux étapes : analyse de la forme et du fond du contenu. L'analyse de la forme a surtout concerné une vue d'ensemble du manuel (présentation générale, organisation des contenus, structures des parties et/ou des chapitres…). L'analyse du fond, quant à elle, a concerné l'aspect épistémologique des contenus à enseigner (notions informatiques prévues, conformité au programme, technologies et langages…), les stratégies didactiques et pédagogiques (présentation des notions, leurs illustrations, les types d'activités et d'exercices proposés à destination des élèves, les objectifs visés par ces tâches, les technologies qui interviennent, les approches prévues pour les pratiques attendues…).

Dans chaque manuel, les tâches relatives à l'informatique ont été analysées. : celles qui font appel aux logiciels et aux calculatrices et celles se rapportant à l'algorithmique et à la programmation pour la résolution des problèmes, proposées par les concepteurs de ces manuels aux élèves. Les tâches présentées dans les manuels d'élèves révèlent l'image des exercices qui sont donnés aux élèves. Si les vraies pratiques de ces derniers avec ces programmes ne sont pas encore connues, des germes des futures prescriptions des enseignants à leurs élèves peuvent plus ou moins être anticipés par les résultats de l'analyse des manuels.

c. Grille d'analyse et/ou analyse logicielle

Les données d'enquête ont été analysées de deux façons : soit avec une grille d'analyse construite par nous-mêmes, soit avec un logiciel. Les données d'entretien et d'observation ont été analysées par des grilles d'analyse conçues à cet effet. Les données obtenues par questionnaire ont été analysées par le logiciel Modalisa.

d. Codage

Avant cette analyse, un codage a été effectué. Dans l'analyse avec le logiciel Modalisa7, le codage a contribué dans le regroupement des variables ou catégories qui sont jugées très proches afin de pouvoir faire des croisements de variables et de faire sortir des résultats intelligibles.

24 http://www.univ-irem.fr/IMG/pdf/Grille-2000.pdf consulté le 5 janvier 2013
25 IREM /APMEP : Institut de Recherches sur l'Enseignement des Mathématiques/Association des Professeurs de Mathématiques de l'Enseignement Public

Après analyse qualitative des données recueillies, les principaux résultats obtenus sont présentés dans la troisième partie suivante de cette étude.

5. Rappel des questions de recherche et hypothèses

Questions générales

1. Comment les élèves et les étudiants débutants en informatique s'approprient-ils les savoirs informatiques enseignés ?
2. Comment les enseignants, spécialistes des mathématiques, prennent en compte les savoirs informatiques dans leurs pratiques professionnelles ?

Questions spécifiques

Nous nous sommes intéressé à trouver des réponses aux questions de recherche suivantes.

1. Quels sont les concepts informatiques jugés incontournables pour une initiation informatique ?
2. Quels sont les types d'activités proposées dans le prescrit (programmes et manuels scolaires) au lycée et quels savoirs informatiques ces activités visent à construire chez élèves ?
3. Quelles sont les activités prescrites aux élèves/étudiants par leurs enseignants et quelles sont les motivations de tels choix ?
4. Quelles sont les compétences effectives des élèves/étudiants à construire un algorithme, à le programmer et à l'exécuter sur ordinateur ?
5. Quelles difficultés les élèves/étudiants et leurs enseignants connaissent-ils ; quelles grandes erreurs les élèves/étudiants commettent-ils ? Quelles sont les stratégies utilisées pour les éviter ?

C'est ce questionnement que cette étude cherche à trouver des réponses.

Deux hypothèses générales ont été formulées.

HYPOTHÈSE I : Les élèves ou étudiants plus ouverts, plus communicants et plus collaboratifs s'approprient facilement les savoirs informatiques construits

La collaboration et la communication sont des qualités majeures d'une pédagogie par projets et par groupe. Nous supposons qu'un élève qui a ces qualités, qui se pose de bonnes ques-

tions, qui demande des explications ou de l'aide en cas de nécessité aura des facilités à s'approprier cet apprentissage qui a lieu en général en contexte de projets.

HYPOTHÈSE II : Les enseignants spécialistes de mathématiques dont leurs représentations de l'informatique sont proches des mathématiques, adhèrent facilement à l'enseignement des savoirs informatiques prescrits dans les nouveaux programmes de lycée.

Si l'informatique et les mathématiques sont des sciences distinctes, elles ont aussi des liens qui les unissent (Dowek, 2010 ; Modeste, 2012b). Partant sur la forte proximité avec l'enseignement des mathématiques et des l'informatique, Bruillard (2009) s'interroge si l'informatique ne serait pas une fille des mathématiques.

Nous supposons donc que les enseignants spécialistes des mathématiques qui, dans leurs représentations, rapprochent les deux sciences plutôt que de les distinguer, adhèrent facilement à l'enseignement de l'informatique (en mathématiques).

À côté des questions et des hypothèses générales de recherche, des questions et des hypothèses spécifiques à certains chapitres ont été formulées. Elles restent valables à l'intérieur de ces mêmes chapitres. Les résultats obtenus au sein de ces chapitres sont d'abord discutés relativement à ces questions spécifiques pour confirmer ou infirmer les hypothèses spécifiques.

Après, une discussion générale des résultats est menée concernant les questions générales pour valider ou invalider les hypothèses générales.

Chapitre III REGARDS SUR L'INFORMATIQUE ET SON ENSEIGNEMENT EN FRANCE : Une histoire ancienne faite d'une série de faux départs

Ce chapitre tracera une brève histoire de l'enseignement de l'informatique en France. Il mettra en évidence les motivations de son introduction, les choix effectués et les dates importantes qui ont marqué son évolution. Il sera développé en six points : ses débuts au niveau de l'enseignement supérieur, son enseignement au second degré de la scolarité secondaire, la place de l'algorithmique et de la programmation dans les débats scientifiques, la situation de l'informatique en dehors des cadres scolaires, le regard sur la situation internationale et il se clôturera par une brève conclusion.

1. Niveau supérieur : débuts et reconnaissance de la discipline informatique

En France, la rencontre de l'informatique comme discipline avec l'enseignement supérieur se situe dans la décennie 1960[26]. Comme l'indique Beaudouin-Lafon (2010), contrairement aux autres pays industrialisés dont les États Unis par exemple où l'informatique est issue de l'électronique, en France, elle est issue en plus grande partie de l'enseignement des mathématiques. C'est d'ailleurs cette origine électronique de l'informatique qui est justifiée par son nom, « computer science », qui veut dire « science de l'ordinateur », précise Beaudouin-Lafon. S'intéressant à l'informatique théorique, Bernard Chazelle trace et met en évidence les différences entre la France et les autres puissances, particulièrement en France et aux États Unis dans leur conception de la science informatique. Selon lui, la discipline de l'informatique théorique est peu pratiquée en France, bien que ce pays ait joué un rôle important en invention de langages de programmation.

Après la reconnaissance de cette discipline informatique supérieure, l'avènement des microprocesseurs vers la fin de la décennie 1970 a contribué à son développement. À cette période, l'informatique était une discipline scientifique autonome dispensée dans beaucoup d'universités de l'enseignement supérieur, mais aussi une discipline technique (Baron, 2012, p. 83).

26 http://www.academie-sciences.fr/activite/rapport/rads_0513.pdf, consulté le 29 décembre 2013

Par exemple, certains diplômes ont été créés avant la fin de la décennie 60 (Pélisset, 1985) : le Brevet de Technicien Supérieur (BTS) a été institué en 1957.

La première reconnaissance officielle de cette discipline est intervenue en 1967, date correspondant à la réforme de Christian Fouchet, qui a institué une maîtrise d'informatique et des départements d'informatique dans les Instituts universitaires de Technologies (IUT) (Baron, 1987). Dès 1967, cinq universités dotées d'offres de formations professionnelles, étaient déjà habilitées à délivrer des diplômes de Maîtrise d'Informatique Appliquées à la GEstion (MIAGE) (Baron, ibid.) : la demande était grande pour de telles formations chez les étudiants.

En conclusion, au niveau de l'université, la période 60-70 correspond à la constitution d'un corps de savoirs savant. La décennie des années soixante-dix, l'informatique était non seulement un objet d'enseignement, mais aussi un objet de recherche et de culture. Tout étudiant était appelé à s'en approprier pour échapper à la marginalisation (Baron, 1987, p. 193).

Malgré le manque de spécialistes de la science informatique, des étudiants en masse affluent vers les nouvelles options instituées en rapport avec l'informatique. Cela va créer une situation de plus en plus préoccupante jusqu'à ce qu'une décision de la séparation des mathématiques de l'informatique soit prise par le ministère ayant l'éducation dans ses attributions : à partir de temps, les deux disciplines mathématiques et informatiques coexistent et l'organisation de la discipline informatique est désormais à part. Depuis les années quatre-vingts, sont nées des sections sous l'appellation informatique avec différentes orientations : « informatique théorique », « informatique des organisations techniques », « informatique industrielle ». Malgré cette évolution, une question cruciale restée sans réponse est celle de l'existence d'un « champ disciplinaire sans corps d'enseignants de professionnels reconnus, dotés par l'institution du privilège de produire et reproduire des connaissances » (Baron, 1989).

C'est à partir de la décennie 1970 que l'informatique est entrée au lycée de l'enseignement général.

2. Regards sur l'informatique dans l'enseignement de second degré

La prise en compte de l'informatique dans le système éducatif français s'est faite en trois grandes phases (Baron, 1987, 2012). Qualifiée de *phase des « premières expérimentations »*, la première phase est située entre les années 1960 et 1970. La deuxième phase, dite *phase de*

première fondation est comprise entre les années 1970 et 1980 et enfin, une phase *d'« expansion-diffusion/socialisation »* d'après les années 1980.

La première phase dite la phase des premières expérimentions se situe dans la décennie 1960. Ds contenus informatiques d'enseignement secondaire dans les filières technologiques tertiaires ont été mis en place à la fin de cette décennie (Baron, 1987, p. 69-70) : un baccalauréat de technicien supérieur en informatique (séries H), une introduction de l'informatique de gestion (séries G) afin d'amener les jeunes à la maîtrise des techniques modernes de traitement de l'information.

Les visions n'ont pas été les mêmes à propos de ce qu'est l'informatique. Se référant à Arsac (1970), Baron (2012) relève des controverses qui ont caractérisé cette phase. Elles portaient sur la considération de l'informatique, entre un ensemble de « techniques » ou une « science », où le mot « techniques » fait référence au traitement automatique de l'information alors que le mot « informatique » avec le suffixe en « ique » fait généralement référence à une science comme c'est le cas pour les mots « mathématique » et « physique ». En reconnaissant cette évolution rapide de l'informatique surtout occasionnée par la multiplicité de techniques davantage sophistiquées qui accompagnent l'informatique, Émilien Pélisset déplore le discours jugé lacunaire autour de l'informatique qui est toujours vue comme une « technologie nouvelle » après plus d'un demi-siècle de son existence (Pélisset, 1985).

2.1. Origine de l'informatique scolaire en France : le colloque de Sèvres

L'origine de l'introduction de l'informatique dans le système éducatif secondaire général français remonte de l'année 1970 et, se place dans un contexte international (Pair, 1987). Antérieurement à cette introduction, deux tendances étaient en œuvre : d'une part, une acquisition des technologies informatiques sous forme d'un enseignement programmé et d'autres, un enseignement de l'informatique focalisé sur un langage de programmation. Cette introduction de l'informatique dans la scolarité secondaire en France a choisi de privilégier une autre approche, en rupture avec les deux approches en vigueur (Baron & Bruillard, 1996) : c'est une approche *outil* consistant à *« rénover toutes les disciplines, grâce aux vertus de la « démarche informatique » : « algorithmique, organisatrice modélisante » »*.

Vue comme la première fondation de l'informatique au lycée, cette deuxième phase a lieu dans la décennie 1970-1980. Si au début des années soixante-dix, les ordinateurs étaient encore des machines à la fois très chères, très rares et destinées aux seuls professionnels, l'in-

formatique était perçue comme un vecteur d'avenir et donc indispensable pour l'éducation des jeunes. C'est ainsi qu'un intérêt pour l'informatique comme discipline d'enseignement secondaire général est né en France après impulsion d'un colloque de Sèvres (Baron, 2012, p. 83).

Ce colloque d'une semaine, intitulé « l'enseignement de l'informatique à l'école secondaire » était organisé au Centre International d'Études Pédagogiques de Sèvres, par le Centre pour la recherche et l'Innovation dans l'Enseignement (CERI), une structure de l'Organisation de Coopération et de Développement Économique (OCDE), créée en 1968. Le séminaire plaidait pour un enseignement de l'informatique, et ce, à toutes les filières de formation du second degré. Se référant aux conclusions des actes de ce séminaire, les participants étaient fortement préoccupés par la naissance de cet enseignement (Baudé, 2010a) : *« l'introduction d'un enseignement de l'informatique du second degré est apparue comme indispensable aux participants et ce, quelles que soient les préoccupations qui peuvent justifier cette introduction : enseignement général du second degré, enseignement économique et commercial, formation des futurs techniciens de l'informatique »*. Pour les participants, l'importance était plus donnée à la démarche informatique mettant donc en œuvre le raisonnement algorithmique, opérationnel, organisationnel plutôt qu'à l'ordinateur. Les conclusions de ce colloque ont incité les pays membres de l'organisation à une introduction de l'informatique dans l'enseignement secondaire (Pélisset, 1985).

Pélisset montre une importance qui devrait être accordée à la formation en informatique des enseignants appelés à mettre en œuvre cette réforme. Trois niveaux de formations ont été prévus : (1) une formation des enseignants qui auront pour tâche d'introduire cet enseignement de l'informatique ; (2) une formation plus spécialisée et plus différenciée pour les enseignants qui auront à développer cet enseignement dans l'enseignement technique, et particulièrement en l'enseignement économique ; (3) une formation générale pour les enseignants de toutes les disciplines. qui peuvent être intéressés par l'informatique.

Le colloque proposait que les deux formes de formation (1) et (2) puissent prioritairement être mises en œuvre, sans pour autant écarter la troisième forme de formation (3). Pour les participants à ce colloque, deux risques étaient signalés une fois la forme de la formation (3) ignorée. Premièrement, écarter cette approche (3) dans la formation, reviendrait à écarter certains enseignants de la formation, ce qui a pour risque d'isoler l'enseignement de l'informatique des autres disciplines enseignées, ce qui a le risque d'accentuer l'aspect « machine » de

l'informatique. Deuxièmement, ignorer cette formation conduirait au risque d'accentuer, même de façon involontaire, l'aspect « machine » de l'informatique. Cette formation des enseignants en informatique était de plus renforcée par d'autres formes d'aides différentes notamment des dispositifs mis à leur disposition.

2.2. De l'expérimentation à la généralisation

a. Phase expérimentale : « expérience des 58 lycées »

Impulsés du colloque de Sèvres, les décideurs politiques étaient convaincus et déterminés de l'intérêt de l'informatique et de la nécessité de donner une formation informatique aux jeunes. Très rapidement, une circulaire ministérielle 70-232 du 21 mai 1970, sortie au BOEN N° 22 du 28 mai. Elle témoignait des intentions objectives et ambitieuses du ministère en rapport avec l'enseignement de l'informatique (Pélisset, 1985). En effet, pour beaucoup d'auteurs, cette décision ministérielle était un témoignage éloquent affiché concernant la prise de conscience d'un phénomène de société que la nécessité d'un enseignement de l'informatique. Ce long paragraphe, extrait de cette circulaire, montre à suffisance cette ambition de l'introduction de l'enseignement de l'informatique pour la haute autorité éducative du ministère (Baron et al., 1981, p. 7.) :

> *« L'informatique est un phénomène qui est en train de bouleverser profondément les pays industrialisés et le monde moderne en général. La mise en place de banques de données, la création de réseaux de communication de l'information, la formulation de nombreux problèmes sans relations apparentes dans un langage unique commun, l'approche synthétique de questions complexes que permet l'informatique, en font un outil scientifique, technique et intellectuel unique. L'enseignement secondaire tout entier et dès la classe de 4e ne peut rester à l'écart de cette révolution. Il doit préparer au monde de demain dans lequel ceux qui ignoreront tout de l'informatique seront infirmes. Il doit apprendre la portée de cet outil, pour éviter les enthousiasmes excessifs et scepticismes étroits. Il doit profiter de la valeur formatrice de l'enseignement de l'informatique, de la rigueur et de la logique qu'elle impose. Il doit faire apparaître la portée économique du phénomène, et faire savoir ce que l'informatique peut apporter dans la vie professionnelle. Enfin, il doit préparer les consciences à affronter les responsabilités nouvelles créées par sa généralisation »,*

La mise en œuvre des recommandations du colloque de Sèvres a commencé par une grande expérience, connue sous le nom de l'« expérience des 58 lycées ». Cette dernière ouvre la deuxième phase de la prise en compte de l'informatique en éducation en France (Baron, 2012) : 1970-1980, la première fondation de l'informatique au lycée. Deux types d'objectifs étaient poursuivis par cette expérience (Pélisset, 1985) : (1) développer une formation de culture générale à l'informatique avec pour but *« non pas d'apprendre l'informatique, mais d'apprendre que l'informatique existe, à quoi elle peut servir, ce qu'elle ne peut pas faire, quelles sont y ses limites, quels sont les aspects économiques qui lui sont associés »* ; (2) « *favoriser par la même occasion une innovation pédagogique en ouvrant l'enseignement secondaire sur le monde contemporain et en amenant les enseignants à se poser des questions sur le contenu de leur enseignement »*.

Si le succès qu'a connu cette « expérience » a été dû à son développement rapide, Pélisset précise certains facteurs qui ont y contribué : des équipements suffisants, un même langage de programmation spécifique, conventionnel et pédagogiquement adapté et, des enseignants motivés et engagés. Dans la mise en œuvre de cette « expérience des 58 lycées », des enseignants étaient formés. Néanmoins, si les contenus de formation étaient les mêmes pour tous les enseignants, l'approfondissement des formations suivies variait en fonction du type de formation suivie. En effet, deux types de formation étaient organisés pour les enseignants (Mercouroff, 1983) : une formation « approfondie » et une formation « légère ». La formation « approfondie » était une formation lourde et à plein-temps suivie durant toute une année scolaire. Elle avait une certaine particularité : la formation suivie par les spécialistes des disciplines scientifiques et particulièrement les mathématiques était légèrement plus poussée par rapport à celle des autres disciplines. Ayant lieu chaque année, elle s'est poursuivie pendant six ans, de 1970 à 1976. Au cours cette période, plus de 1000 enseignants, de toutes les disciplines, ont bénéficié de cette formation.

Contrairement à la formation « approfondie », la formation « légère » était organisée sous forme de cours par correspondance. Cette formation était accompagnée et complétée par un stage de deux à trois jours. Elle a aussi connu un grand nombre d'enseignants qui en ont bénéficié : plus de six mille cinq cents (6522) enseignants. Pour rendre les enseignants disponibles à suivre cette formation, une décharge dans leurs attributions de cours a été instituée.

En conséquence à cette « expérience », les enseignants ont été sensibilisés à l'informatique, et particulièrement à son instrumentation dans l'enseignement des autres disciplines. L'éva-

luation de cette « expérience » a révélé des résultats satisfaisants quant à la maîtrise des outils informatiques chez les enseignants mais aussi à l'introduction de l'informatique dans l'enseignement de toutes les disciplines enseignées (Mercouroff, 1983) : le renouvellement des pratiques pédagogiques chez les enseignants. Pour Mercouroff, cette « expérience » a permis aussi la démystification de l'informatique chez les enseignants et, par conséquent, la passation de certaines notions informatiques fondamentales chez les élèves. Les enseignants les plus motivés ont été jusqu'à l'initiation informatique. Quant à la formation des élèves, un nombre important d'entre eux, ont été formés : environs 50% des élèves ont chaque année utilisé l'ordinateur.

Le succès de cette « expérience » a permis son élargissement. Elle a continué à d'autres niveaux de formation auxquels elle n'était pas initialement destinée (Baron et al., 1981) : *« une classe sur quatre est en classes de premier cycle, ce qui était relativement important dans une expérience qui s'adressait théoriquement aux lycées »*. Les apprentissages des élèves ont eu lieu essentiellement sous deux formes lors des cours proprement dits en classe et lors des clubs informatiques. L'informatique enseignée dans les clubs était beaucoup moins structurée.

b. Choix d'une approche « outil » de l'informatique

La dualité objet/outil informatique est une création de Claude Simon (Baron, 2012, p. 84) : il évoque pour la première fois l'expression d'*« outil informatique »* en tant que *« moyen d'enseignement »* par opposition à l'*« objet d'enseignement »* dans son rapport intitulé « l'éducation et l'informatisation de la société » [27].

Si la France a vite adhéré aux recommandations issues du colloque de Sèvres, elle s'est heurtée à une question importante (Pélisset, 1985) : le manque de précision dans les conclusions du colloque sur l'approche de l'informatique à enseigner. Il « avait laissé ouverte la question de savoir s'il fallait en faire un enseignement autonome ou le placer dans le contexte d'une autre discipline » (Baudé, 2010a). Cela a été vu comme une lacune de ce colloque notamment dans la mise en œuvre des recommandations, bien qu'il soit radical sur l'enseignement de l'informatique. Claude Pair (1987), l'un des défenseurs de l'approche objet d'enseignement de l'informatique, dans ses réflexions, insistait sur ce que l'on peut espérer de l'informatique par rapport aux besoins présents ou futurs de l'école, des élèves et du pays. Selon

27 http://www.epi.asso.fr/revue/histo/h80simon.htm

lui, la question « qu'est-ce que l'informatique peut apporter à l'enseignement ? » n'était pas convenable et propose plutôt de voir les choses et se questionner sous la forme « quels sont les besoins de l'enseignement ? » et par conséquent évoluer par après vers le questionnement « qu'est-ce que l'informatique peut faire pour satisfaire ces besoins ? ».

Selon Pair, la question de l'approche de l'informatique à enseigner était pertinente et sous tendait diverses problématiques. La première problématique était le recrutement des enseignants une fois le choix de la création d'une discipline privilégié. Un tel choix impliquerait aussi la mise en place, au préalable, d'un CAPES, d'une agrégation, d'une inspection spécifique et bien sûr des programmes d'enseignement (les contenus) et des horaires spécifiques. Pour contourner les exigences de l'enseignement d'une discipline, l'approche transversale consistant à un enseignement de l'informatique à travers d'autres disciplines a donc été privilégiée, précise Pair. Si « l'une des caractéristiques de l'informatique est de créer chez les élèves une attitude algorithmique, opérationnelle, organisatrice, laquelle est souhaitable pour bien des disciplines », le choix effectué – enseigner l'informatique à travers des disciplines existantes –, était doublement bénéfique (Baron et al., 1981). Le premier bénéfice est que cette approche permettrait de faire acquérir aux élèves les attitudes attendues de l'apprentissage de l'informatique. Le deuxième bénéfice est d'éviter les exigences d'une discipline nouvelle qui surchargerait un enseignement déjà surchargé et même dit d'« encyclopédisme » pour reprendre le terme de Baron et al., ibidem. Les choix faits ont été fructueux. Si l'« expérience » a été dominée par l'approche « outil » de l'informatique dans les disciplines déjà enseignées, les résultats de son évaluation ont révélé l'atteinte de « la plupart des objectifs initiaux de l'expérience » (Mercouroff, 1983).

Alors qu'une évaluation conduisant à sa généralisation était attendue en 1976, l'« expérience des 58 lycées » a été suspendue par une décision ministérielle en 1975[28]. En conséquence, les équipements en mini-ordinateurs des établissements et les formations approfondies des enseignants ont été arrêtés par la Direction générale de la programmation et de la coordination (DGPC). Par la suite, la gestion de l'« expérience » a été confiée à une Direction des Lycées et son fonctionnement a continué dans de nouvelles conditions (Pélisset, 1985) : *« décharges de service et une partie des postes attribués antérieurement sont conservés ; quelques améliorations techniques facilitent l'usage des matériels en place ». En* l'absence de formations

28 Cette décision a été une conséquence d'un plan d'austérité qui a occasionné un budget de rigueur du ministère de l'Éducation nationale suite aux conditions économiques difficiles du pays.

approfondies des enseignants, celles légères et payantes ont continué à être assurées jusqu'en 1979. En conclusion, après 1976, l'« expérience » a continué mais, de façon marginale (Baron, 1987, p. 73) : *« (elle) se passe largement hors des cadres hiérarchiques traditionnels et des institutions universitaires, dans une relative autogestion par les enseignants eux-mêmes, ce qui lui a conféré une sorte de superbe marginalité »*.

c. Option informatique des années 80 : de l'expérimentation à la généralisation

Après la phase de fondation de l'enseignement de l'informatique dans le système scolaire français (1970-1980), la généralisation de l'« expérience » précédente s'imposait. Cependant, des controverses persistaient quant à son orientation. Deux thèses étaient en tension. La première thèse prônait un enseignement d'une discipline informatique.

Les tenants de l'informatique « objet d'enseignement » proposaient un enseignement de la discipline informatique à tout le lycée. Parmi eux se trouvent les spécialistes de l'informatique dont Jacques Arsac et Jean Claude Simon[29]. Des critiques négatives à l'endroit de l'expérience des « 58 lycées » ont été formulées par Simon (1980, p. 105) dans son rapport (Baron, 1994, p. 74) :

> *« Dans les lycées d'enseignement long, on ne peut pas soutenir qu'il existe une formation à l'informatique. Nous avons déjà parlé des « clubs d'informatique », qui se sont formés autour des centres de calcul de l'expérience des 58 lycées. Ces clubs sont certainement des initiatives louables ; mais, comme nous l'avons dit, ils ne sont en aucune façon un support à une formation à l'informatique. On peut même penser que les élèves risquent de contracter de mauvaises habitudes en programmation qu'il leur sera difficile de perdre ultérieurement. »*

Dans la création d'une discipline informatique, le courant proposait une disposition transitoire (Simon, 1980) : *« création progressive d'une option de « formation à l'informatique », ouverte à tous »* en vue de préparer à une mise en place d' « *un enseignement informatique obligatoire pour tous »*[2].

La deuxième thèse est la continuité de l' « expérience » précédente relative à l'approche outil de l'informatique sans pour autant mettre sur pied une nouvelle discipline scolaire. Les te-

29 Jacques Arsac et Jean Claude Simon étaient de grands professeurs d'universités et spécialistes d'informatique. Ils étaient parmi les grands défenseurs d'un enseignement d'une discipline informatique au lycée

nants de la deuxième thèse – approche outil de l'informatique – prônaient l'enseignement de l'informatique à travers les disciplines existantes. Les partisans de cette approche, en tête desquels Jacques Hebenstreit. défendaient la continuité de l'« expérience » précédente. Tout en reconnaissant l'indispensable nécessité de la discipline informatique, selon eux, le terrain n'était pas encore propice pour son introduction dans l'enseignement, précise Georges Louis Baron. Selon cet auteur, la continuité de l'expérience précédente était donc pour eux, une forme de préparation de terrain afin de réunir d'abord les conditions favorables à l'institutionnalisation d'une discipline informatique scolaire autonome.

Baron, directeur de cette thèse de doctorat, faisait aussi partie de ce deuxième courant à cette époque. Il trouvait que la création d'une discipline scolaire est une entreprise très exigeante qui demande de lourds investissements et des changements de la planification déjà en vigueur. Par exemple, son enseignement exigerait l'amenuisement de l'importance en volume horaire des disciplines déjà existantes pour que cette nouvelle discipline ait sa place sur l'emploi du temps. Tout en reconnaissant l'intérêt d'un enseignement d'une informatique objet orientée programmation des ordinateurs, à cette époque, Baron doutait de l'efficacité innovante d'une nouvelle discipline informatique (Baron, 1994, p. 75) :

> *« Pour ma part, je ne croyais alors pas que la création d'une discipline nouvelle puisse vraiment apporter des innovations majeures et je partageais l'opinion de Jacques Hebenstreit, selon laquelle un des enjeux importants de ces développements était la transmission vers l'aval de méthodes qui avaient fait leurs preuves. La création d'une informatique avec un label ne risquait-elle pas de faire du tort aux autres activités ? Mais ma formation et mon expérience me poussaient à penser que la programmation de l'ordinateur au sens de « faire faire » à l'ordinateur, comme le dit par exemple C. Duchâteau (1990), était une activité pleine d'intérêt, pouvant donner lieu à enseignement. »*

L'approche « outil d'enseignement » de l'informatique a finalement été adoptée pour la phase de généralisation de cette « expérience ». À la suite d'un rapport Nora-Minc[30] de janvier 1978, un plan dit de *« 10.000 micro-ordinateurs »* a été élaboré par le Ministère de l'Éducation nationale en début d'année suivante (Baudé, 2010b). Conçu sur demande du Président de la République, ce Plan avait une orientation d'« accroître l'efficacité et la compétiti-

30 http://www.amazon.fr/LINFORMATISATION-SOCIETE-Rapport-Pr%C3 %A9sident-R%C3 %A9publique/dp/2020049740

vité du système économique » français. Lancé en 1979 pour une durée de six ans, il s'intéresserait à l'enseignement de l'informatique au second degré. Contrairement au précédent, ce plan se limiterait à l'équipement en micro-ordinateurs des lycées d'enseignement général et technologique pour permettre l'accès des élèves aux techniques nouvelles et de leurs usages raisonnés. Intervenu dans une phase de généralisation de l'informatique au lycée, le Plan s'est étendu aussi au collège où il a permis d'amorcer une phase expérimentale.

À cette période, l'informatique prenait de plus en plus de l'ampleur comme discipline scientifique et, un consensus autour de ses approches méthodologiques d'analyses et sa programmation semblait se dessiner petit à petit autour d'elle. En conséquence, à côté de cette approche « outil d'enseignement », un intérêt pour une nouvelle forme commençait à naître (Baron, 1987) : l'informatique comme « objet d'enseignement ». Cet enseignement de l'informatique *objet* était organisé en une expérimentation sous forme optionnelle (Baron, 1994).

Les propositions de Claude Simon ont été bien accueillies par le Ministre de l'éducation nationale pour qui, enseigner l'informatique revenait à exercer une forme de « démocratie éducative »[31]. La mise en œuvre des propositions du rapport Simon a été rendue officielle dans le Plan « *Le mariage du siècle »*, en novembre 1980, lors d'un colloque organisé par la section française de l'Institut International de la Communication et l'Association Télequal. Mais, des changements politiques en 1981 à la tête du pays[32]ont apporté un nouveau ministre de l'Éducation Nationale avait une vision différente de la politique éducative informatique en cours. En conséquence, le plan a d'abord été suspendu avant de reprendre après un petit temps, sur pression du syndicat général de l'éducation nationale (SGEN-FDT) (Baudé, 2010b).

Après cet arrêt momentané de 1981, les formations, articulées en deux types, ont repris au niveau national : celles de « courte » durée pour former de simples utilisateurs de l'informatique et celles de longue durée : des formations approfondies en faveur des enseignants reconvertis en informatique.

Le programme officiel était articulé en trois parties – matériel, logiciel et informatique et société –, qui étaient enseignées généralement de façon séparée et sans la même importance (Brygoo, 1989) : la partie logicielle était privilégiée et celle de l'informatique et société était en général minimisée. Cette option, en général sélective, a connu un grand succès autant chez

31 http://www.epi.asso.fr/revue/histo/h80beullac.htm

32 Élection de 1981 porte à la tête du pays François Mitterrand à la place de Valery Giscard d'Estaing et Alain Savary comme ministre de l'Éducation nationale.

les élèves que chez les partenaires éducatifs. Son succès était dû à la coexistence de deux formes d'enseignement : informatique comme objet et comme outil d'enseignement. Si les conditions tant matérielles qu'humaines, dans lesquelles se déroulait cet enseignement n'étaient pas très bonnes (manque d'ordinateurs suffisants, d'enseignants compétents…), deux principales raisons justifient la motivation des élèves et des parents pour cet enseignement. La première raison est que c'est seulement à travers cette option informatique où ils pouvaient découvrir les ordinateurs jusqu'alors moins nombreux et quasi-absents dans les familles (Baudé, 2010b).

La deuxième raison est liée au fait que cette option était vue non pas seulement comme une « matière moderne prometteuse », mais aussi perçue comme une nouvelle offre de formation par rapport aux autres déjà existantes. Selon Baron, cette dernière raison conférait à la classe de seconde une spécificité particulière. Beaucoup convoitée, cette option informatique était devenue élitiste et beaucoup plus sélective. L'accès largement limité à cette option était dû au faible effectif d'enseignants disponibles mais aussi aux conditions organisationnelles. Sélectionnés sur dossier par les chefs d'établissements, les élèves avec de bons résultats dans les matières scientifiques étaient privilégiés (Baudé, 2010a) : ils étaient essentiellement issus des filières C[33] et à majorité des garçons (70 %). Par conséquent, si l'objectif initial était de démocratiser l'enseignement de l'informatique, l'offre n'a jamais été à la hauteur de la demande (Baron, 1994, p. 78-79).

Préparé depuis les années 1980, cet enseignement optionnel et expérimental de l'informatique était une mise en œuvre concrète de l'« idée de l'informatique comme discipline d'enseignement général et d'introduire à l'école une informatique avec un « label » scientifique » (Baron, 1987, p. 76). Ouverte avec la rentrée de l'année scolaire 1981-1982, cette expérimentation a été limitée à une dizaine de lycées (en tout 12 lycées) avec trente-cinq enseignants. Cette expérimentation sera ensuite généralisée pour devenir une option étendue à plusieurs écoles depuis l'année scolaire 1984-1985 (Baudé, 2010b). Son institutionnalisation, en 1985, l'a faite d'une option ouverte et enseignée à beaucoup d'élèves : elle a été organisée dans 60 % des lycées publics et avec un effectif d'environs deux milles professeurs (Baron, 1987).

33 Aujourd'hui, dans la scolarité française, on ne parle plus de filière ou de série C. Il est question de la série S (scientifique) laquelle a remplacé les séries C, D et E, qui étaient trois séries du baccalauréat général. L'ancien bac C correspond plus ou moins à la spécialité Maths du bac S, là où la spécialité SVT succède à l'ancien bac D. Quant à l'ancien bac E à orientation mathématique et technique. http://fr.wikipedia.org/wiki/Baccalau, r %C3 %A9at_scientifique, consulté le 25 décembre 2013

Les années 1981-1985 sont connues pour être une période d'expérimentation de l'enseignement de l'informatique sous forme d'objet. Depuis 1985, elle a évolué, passant de l'état expérimental à un enseignement optionnel généralisé dans les lycées généraux (Baron, 1985 ; 1994). L'option informatique est passée de la classe de seconde, en 1985 et, a fait objet de sa première épreuve au baccalauréat en 1988 (Baudé, 2010b).

1985 est une année charnière dans l'histoire de l'enseignement de l'informatique en France. Elle coïncide avec le début de la généralisation de l'option informatique au lycée et, est un repère de l'introduction du « plan informatique pour tous » (IPT)[34] (Pair, 1987). Ce Plan était d'une importance considérable vus les moyens mobilisés (Baron, 1987) : un budget de loin supérieur à tous les plans précédents en termes d'impact sociétal. Vu comme une réponse aux enjeux du moment liés non seulement à l'omniprésence de l'informatique mais aussi à son accélération dans tous les secteurs sociaux, ce plan semble avoir convaincu des gens pessimistes de l'« importance de l'informatique pour le système éducatif et au-delà pour l'ensemble de la société »[35]. Ce plan fait parti de la troisième phase d'après les années 1980 : celle où l'informatique est à la fois *objet* et *outil* d'enseignement (Baron, 2012).

Le choix a été d'investir dans la formation des jeunes en informatique, un choix vu comme judicieux pour le développement d'un pays, comme l'indique l'extrait du discours de Fabius tel donné par l'Epi[36] :

> *« La formation est l'investissement le plus important de la Nation, la clef de voûte de la modernisation du pays. L'informatique va devenir de plus en plus une véritable seconde langue. L'objectif du Président de la République, le nôtre, est de faire de cette génération la mieux formée de notre histoire. Grâce à ce plan, la France va être, dès cette année, un des premiers pays du monde, probablement le premier, dans l'enseignement de l'informatique pour tous ».*

Trois objectifs[37] principaux étaient visés par ce Plan : (1) initier à tous les élèves et de tous les niveaux d'enseignement, à l'usage des outils informatiques ; (2) mettre sur pied des ate-

34 Le plan IPT a été lancé par Laurent Fabius, premier ministre d'alors. Il a été justifié par la nécessité de la diffusion de technologies informatiques pour faire évoluer les pratiques numériques de la société. Ce plan consistait en l'équipement de 100 000 mini-ordinateurs et de la formation de 100 000 enseignants pendant une période de 5 ans (1983-1988).

35 http://www.epi.asso.fr/revue/histosom.htm, site consulté le 12 novembre 2014

36 http://www.epi.asso.fr/fic_pdf/b37p023.pdf, site consulté le 8 novembre 2 014

37 http://www.epi.asso.fr/revue/37/b37p023.htm

liers d'informatique, destinés à tous les citoyens mais utilisés dans un cadre bien concerté notamment par la mise en place de contrat négocié entre les collectives locales et les associations ; et enfin, (3) former des enseignants.

Avec ce plan, il a été instauré la politique de décentralisation consistant à mettre dans les mains des collectivités territoriales la responsabilité des remplacements des matériels scolaires après équipements. C'est avec cette politique aussi que la terminologie d'« outils informatiques » a été beaucoup utilisée et généralisée (Baron, 2012, p. 84). À côté de son succès social important, le Plan avait aussi déjà connu des effets considérables sur le système éducatif cinq ans après sa mise en œuvre (Baudé, 2010c) : jusqu'en 1990, *« l'option informatique passera de 74 lycées et 7.000 élèves à environs 550 lycées et 50.000 élèves dans le secteur public ».*

d. Discontinuités de l'option informatique : vers la disparition ?

Après le succès de la phase expérimentale de l'option informatique au lycée, elle a été généralisée à tous les lycées en 1985. Néanmoins, son évolution dans la suite sera caractérisée par des discontinuités dues à des suppressions et des rétablissements successifs. Après son succès, cette option a connu un déclin avant d'être supprimée sans délais (Baron & Bruillard, 1996). La première suppression intervient en 1989, quatre ans seulement après sa généralisation et une année seulement après avoir fait partie des épreuves au Baccalauréat[38]. Rétablie en 1995, elle sera encore supprimée trois ans après, en 1998 (Baudé, 2010c) et, cette fois pour une longue période, avant de réapparaître une décennie après, en 2010.

Deux principales raisons sont données par la littérature de recherche pour justifier sa suppression. La première est liée à son caractère élitiste. L'option était difficile pour les élèves et, le taux important d'abandons par rapport aux autres matières serait la cause d'après Baron et Waiter (1986) à qui se réfère Baron (1994) : ses difficultés étaient assimilées à celles posées par le latin. Selon Baron, à côté de la difficulté de cet enseignement, trop de travaux donnés aux élèves seraient une autre raison justificative de ces nombreux abandons : ce sont surtout les élèves issus des filières scientifiques qui parvenaient à poursuivre. Jacques Baudé (Baudé, 2010c) évolue dans le même sens et donne trois raisons. Il évoque des emplois de temps souvent surchargés mais aussi des horaires imprévisibles qui, selon lui, étaient issus d'un « dérapage en amont ». La troisième raison soulignée est liée à des conditions de son

38 L'informatique est intervenue au Baccalauréat 1987-1988 comme option

enseignement difficiles (Baudé, 2010b) : absence d'un nombre suffisant de professeurs spécialisés. Cela obligeait un recours à ceux d'une « seconde compétence », sans être des spécialistes de la discipline : la majorité d'entre eux était des spécialistes de mathématiques. Pour Baudé, tous ces problèmes qui ont entouré l'enseignement étaient des dérapages qui devraient être corrigés par une formation sur plusieurs années et, par conséquent, contribué à réduire l'élitisme de cette option.

Claude Pair (1987) situe l'échec de cette option sur la focalisation du travail des élèves sur la programmation. Selon lui, les difficultés des élèves en cette option étaient occasionnées par la centration sur l'approche objet de l'informatique : trop à faire faire à la machine ce qu'on faisait à la main, sans avoir mené une *« réflexion suffisante sur la fécondation réciproque entre l'informatique d'une part et les disciplines d'autre part ».*

Dans le même sens, Maurice Nivat explicite davantage la part de l'importance donnée à la programmation dans l'échec de cette option. Selon lui, si cette raison n'était pas exprimée à haute voix, elle aurait influencé la tendance de la généralisation de l'utilisation des applications au détriment de la science informatique en particulier la programmation. Si l'insuffisance d'enseignants d'informatique est vue comme la vraie raison de la suppression de cette option, la Société Informatique de France (SIF), y en ajoute aussi une autre (Alayrangues et al., 2013) : récupérer des postes d'enseignants de mathématiques.

Depuis la fin des années quatre-vingt-dix, l'option informatique a été mise de côté et, a cédé la place à la généralisation des utilisations des technologies. Un nouveau champ construit autour des dispositifs informatiques a émergé dans le système éducatif français (Baron, 1994) : les technologies de l'information et de la communication (TIC).

2.3. Les années 2000 : place aux brevets et certificats

La généralisation de l'informatique à tous les niveaux de formations scolaires, due au Plan IPT vers la fin de 20e siècle, annonçait la naissance d'une nouvelle vision de la présence de l'informatique éducative. Si les deux approches objet et outil d'enseignement de l'informatique sont restées dans le discours officiel ministériel, les nouvelles perspectives ouvertes étaient plus orientées vers les outils informatiques, n'exigeant pas forcément de formation spécifique (Baron, 2012, p. 84). La réforme du curriculum de cours de technologie au collège des années 2000 est un exemple d'illustration minimisant la formation à l'usage des outils informatiques. Elle a consisté à introduire des notions de technologie de l'information, dans la

rubrique consacrée à l'enseignement des TIC en général et des progiciels en particuliers (Lebeaume & Meignié, 2003). Ces auteurs trouvent insuffisants les usages et l'apprentissage des fonctionnalités de progiciels prescrits dans le cours de technologie, face *« aux exigences et aux intentions de la technologie, discipline scolaire d'enseignement général, obligatoire pour tous les jeunes »*.

L'enseignement de l'informatique du début des années 2000 a été marqué par une autre approche de la prise en compte de l'informatique dans le système scolaire français. Cet enseignement était caractérisé par une acquisition des compétences sans référence directe avec un curriculum quelconque, inspiré par une « approche par compétences », née à la fin du vingtième siècle, importée en Europe du nord américain (Baron, 2012).

Cette nouvelle approche consistait à mettre un accent sur des certifications sans référence directe avec un quelconque programme d'enseignement (Baron, 2012) : le brevet informatique et Internet (B2i). Si son introduction a été une surprise pour la communauté scientifique, elle était justifiée par le ministère comme une nécessité dans un contexte de « *banalisation des TIC non seulement dans la vie professionnelle mais aussi dans la vie courante, c'est-à-dire dans des pratiques désormais à la portée des jeunes en dehors de l'école » (Devauchelle, 2004)*. Selon Bruno Devauchelle, l'instauration du B2i dans le système éducatif attestait une rupture totale avec les approches précédentes : une page était tournée pour prendre une nouvelle orientation inspirée du passé. Cet auteur appuie ses propos par cet extrait du discours du ministre de l'éducation d'alors Lang Jack lors du colloque de novembre 2000 pour justifier ce divorce avec les précédentes approches : *« Je crois qu'il faut à ce stade tirer les leçons de l'histoire, et nous pencher sur l'échec des programmes semblables mis en œuvre au milieu des années quatre-vingt. »*

La philosophie qui accompagnait le B2i était que toutes les disciplines étaient supposées l'intégrer. Il n'était donc pas associé à de quelconques savoirs informatiques. Pour Baron (ibidem.), « *l'impasse faite sur les savoirs en jeu et l'idée que les disciplines existantes vont plus ou moins spontanément se saisir de l'affaire de manière harmonieuse »* ont été quelques-unes des singularités du B2i mais aussi une source de sa fragilité.

Michel Laisne montre une nouvelle structuration au cours du temps en niveaux des compétences du B2i depuis son institutionnalisation en 2000 (Laisne, 2004) : un niveau 1 à l'école primaire depuis 2002, un niveau 2 au collège et en seconde (la première classe de lycée) et

un niveau 3 pour les autres classes de lycées générales, les lycées professionnels mais aussi d'autres centres de formations d'apprentis. Laisne conclut sur une préoccupation relative à la nécessité de formation : *« la difficulté de mise en place d'un dispositif de validation des compétences du B2i n'occulte pas la question essentielle de la formation à ces compétences ? ».*

Cette mise en place en l'an 2000 du B2i depuis l'enseignement obligatoire, a été suivie par la même approche à l'université pour des étudiants (Aoudé, 2011, p. 28) : le certificat informatique et Internet (C2i) « *attestant de compétences dans la maîtrise des outils informatiques et réseaux, () institué dans le but de développer et valider la maîtrise des TIC par les étudiants en formation dans les établissements d'enseignement supérieur.* ». En effet, dès son entrée à l'université, tout étudiant est appelé à justifier sa maîtrise personnelle de l'usage des TIC par la validation d'un certificat de niveau 1. Quant aux étudiants qui se préparent à des métiers d'enseignement, ils doivent attester leurs compétences d'usage des TIC dans leur future vie professionnelle par la validation du C2i-nivceau 2. Si le B2i était supposé obligatoire, cette obligation semble n'avoir pas été respectée par tous les établissements. La non concrétisation des attentes du B2i a été justifiée par une absence des pratiques de la part des enseignants dans leur enseignement. Ayant senti des formes de résistances chez les enseignants déjà en fonction, une stratégie qui a été adoptée par le ministère est la prescription du C2i-niveau 2 dans la formation des universités chez les futurs enseignants (Devauchelle, 2004).

a. Limites et fragilités des brevets et certificats

Si l'institutionnalisation du B2i était une façon de tourner la page à une approche de l'informatique objet, un problème important qui s'est finalement posé est celui des savoirs en jeux pour atteindre les compétences attendues. C'est ce que souligne Baron (2012, p. 85)dans l'extrait suivant :

> *« Le B2i reconnaît formellement la nécessité d'acquisition, dès l'enseignement obligatoire, de compétences liées à l'informatique, sans trancher de manière claire sur les modalités nécessaires pour les développer chez les apprenants. Parmi ses singularités, qui sont aussi source de fragilité, est l'impasse faite sur les savoirs en jeu et l'idée que les disciplines existantes vont plus ou moins spontanément se saisir de l'affaire de manière harmonieuse ».*

S'il y a une reconnaissance formelle du B2i dans la nécessité de l'acquisition des compétences informatiques, Baron précise néanmoins que ces acquisitions ne sont pas clairement bien tranchées notamment sur les modalités nécessaires pour leur développement. Pour Baron, la fragilité du B2i est liée à ses singularités (Baron, 2012, p. 85) : *« impasse faite sur les savoirs en jeu et l'idée que les disciplines existantes vont plus ou moins spontanément se saisir de l'affaire de manière harmonieuse »*.

Le B2i, suite aux contraintes de sa mise en œuvre, n'a seulement été une défaite pour ceux qui prônaient que les TIC n'ont pas besoin d'être apprises avant d'être utilisées (Baron, ibidem.) mais aussi a mis en difficultés les professeurs qui en avaient la charge (Béziat, 2012). Sans parler que *« peu d'établissements se sont engagés dans la validation (facultative) du B2i »* (Cabane, 2012, p. 2), alors que d'autres ont été caractérisés par des attributions massives de ces brevets avec l'argument qu'un élève ne peut pas échouer pour n'avoir pas validé ce B2i (Bruillard, 2009), ces choix institutionnels seraient-ils aujourd'hui à l'origine du retard qu'a connu la France dans l'enseignement de l'informatique ?

Jean Pierre Archambault, Gérard Berry et Maurice Nivat (Archambault et al., 2012) sont plus proches de cette hypothèse. Dans leurs tentatives à esquisser des problématiques de l'enseignement de l'informatique dans la scolarité française, le choix de la centration au B2i est classé parmi les principaux blocages, et donc, un enseignement visant la validation des compétences sans référence à des savoirs quelconques connus. Un autre problème souligné est la prescription institutionnelle qui a limité l'enseignement de l'informatique de façon transversale à travers d'autres disciplines déjà existantes par leurs instrumentations aux moyens des TIC. Ces problématiques sont vues comme ayant été des limites à la promouvoir chez les élèves d'une culture informatique :

> *« L'échec du B2i ne signifie nullement pas que l'informatique n'aurait pas sa place dans les autres disciplines. En effet, elle est un outil pédagogique. Elle est aussi facteur d'évolution des objets et des méthodes des autres disciplines enseignées ; par exemple, elle a déjà profondément transformé les enseignements techniques et professionnels. Elle est enfin outil de travail personnel et collectif de la communauté scolaire. Mais, comme l'expérience l'a montré, tous ces usages ne peuvent suffire à donner la culture générale scientifique et technique dont tous les élèves ont besoin ».*

Laurent Bloch, l'un des défenseurs de l'introduction de l'enseignement de l'informatique à tous les niveaux de la scolarité en France s'interroge « où va la France ? » sans cet enseignement disciplinaire (Bloch, 2013). Avec l'exemple de l'Angleterre qui avait fait les choix de programmes de seconde des enseignements des TIC, mais qui, a récemment introduit un véritable enseignement de l'informatique comme discipline à part entière, il recommande de faire pareil en France « de toute urgence », au lieu de continuer à « dissoudre l'informatique » dans la bureautique au collège et dans les mathématiques au lycée.

Bruillard (2006) évoque les problématiques liées aux usages des TIC chez les jeunes. Parmi eux, il souligne la décentralisation de l' État et le désintérêt des élèves envers l'informatique. La décentralisation[39] de l'État est, selon lui, à l'origine de la multiplication des offres privées dans l'éducation et par conséquent favorise des apprentissages extrascolaires des technologies. Cette idée de la décentralisation de l'État, qui ouvre les portes scolaires aux collectivités locales a, par après, été abordée par Khaneboubi (2009) mettant en évidence une série de projets à travers lesquels des ordinateurs portables ont été massivement dotés aux collégiens par certains départements français. Les lois de cette décentralisation ne se limitent pas seulement aux équipements, mais confèrent aux collectivités des prérogatives allant jusqu'aux constructions des infrastructures scolaires (Villemonteix & Baron, 2011), réduisant ainsi le rôle de l'État. Quant au domaine des TICE, le rôle de l'institution ministérielle est la certification et le pilotage des expérimentations des modèles technologiques et pédagogiques. Les auteurs soulignent une fracture en équipements technologiques causée par cette décentralisation entre d'une part, les écoles des départements ou régions plus riches et les moins riches d'autres qui s'ajoute à la « fracture » habituelle entre les familles des élèves.

Concernant la démotivation des élèves, Bruillard part d'un constat d'un changement du rapport des jeunes à l'informatique, un rapport qui semble déconsidérer l'informatique comme objet au profit de l'informatique comme un simple outil de communication et de production, au détriment de son fonctionnement. Selon lui, cette approche limitée à la simple utilisation de l'ordinateur reste un frein pour découvrir les potentialités des machines offertes par leur programmation (Bruillard, 2006, p. 124) :

> *« Alors que la puissance de l'informatique tient à la possibilité de « faire faire » des traitements à une machine, l'illusion du « faire », de l'action directe est certainement*

39 L'État a cédé ses prérogatives aux collectives lesquelles, à des niveaux différents, ont permis l'entrée dans les classes dives types d'outils technologiques. Paradoxalement, rarement utilisés en classe, sont de plus en plus utilisés dans les familles.

un obstacle important à la maîtrise et à la compréhension de ce que peuvent faire les ordinateurs ».

Malgré la prolifération des outils technologiques en éducation, leurs effets sur les apprentissages des élèves font, depuis les années soixante-dix à aujourd'hui, objet de débat (Baron, 1990). Si l'utilisation des TIC a des limites quant à l'acquisition d'une culture générale en informatique, un consensus semble se dessiner : seule un enseignement d'une discipline informatique affirmant son nom est capable de le faire. Cet enseignement vient d'être instauré au lycée en France mais dans un contexte particulièrement difficile d'absence d'enseignants spécialisés en informatique. C'est ce qui est décrit dans le point suivant.

b. Le B2i et C2i : une fausse réponse à une vraie question ?

Pour Jacques Béziat, le choix français de l'institutionnalisation du B2i est une réponse paradoxale par rapport aux réels besoins dans la scolarité obligatoire que sont la formation des enseignants et l'éducation des élèves. Au lieu d'un enseignement sous forme d'une discipline, avec ce brevet, le ministère a voulu plutôt supprimer toute tentative à une disciplinarisation des contenus dont il pourrait être porteur (Béziat, 2012) : *« l'acquisition des connaissances et compétences des référentiels du B2i, à quelque niveau que ce soit du cursus scolaire, ne fait pas objet d'un enseignement spécifique… ».* Si cette restriction signifie en principe qu'« *il n'y a rien à apprendre mais il faut savoir s'en (technologies informatiques) servir »*, elle était en contradiction avec la banalisation de l'ordinateur et des technologies informatiques chez les jeunes et la question cruciale de leur scolarisation.

Malgré les fréquents usages de l'informatique et de ses applications, les élèves, appelés les « digital natives » (Prensky, 2001a, b), ont des problèmes de compréhension des phénomènes en cours lors de l'utilisation de l'ordinateur et des logiciels associés. Ils n'ont *« nulle maîtrise réelle, mais le bidouillage, sans compréhension »* (Bruillard, 2006, p. 124). Pourtant ils sont censés avoir certaines compétences relatives à l'usage des TIC qui sont prescrites dans les textes officiels sous forme de B2i (Béziat, 2005) : « *Le Brevet atteste que l'élève utilise de manière autonome et raisonnée les TIC disponibles à l'école* […] ».

Le problème est l'absence d'un véritable enseignement de l'informatique. Giannoula et Baron, dans une étude de type ethnographique à l'élémentaire, appellent à nuancer et relativiser

le discours des jeunes en ce qui concerne leur maîtrise déclarée des outils numériques (Giannoula & Baron, 2002) :

> *« La maîtrise qu'ils font valoir est avant tout une familiarité avec certains logiciels, à côté de laquelle, il y a une ignorance totale du matériel et des processus de traitements de l'information. À l'égard même des logiciels, ce qui retient leur attention ce sont les procédures de mises en œuvre et d'exécution de tâches plutôt que les fonctions sollicitées. Ce qui fait qu'en cas de problème, la seule issue est la répétition à l'identique de la procédure ou l'abandon au profit d'un autre logiciel ».*

Quelques années après, Fluckiger (2007), lors d'une étude faite dans un collège de banlieue de Paris avec une approche ethnographique portant sur les appropriations des TIC chez les collégiens, obtient des résultats similaires. Ces derniers sont aussi confirmés par Baron & Bruillard (2008) : malgré leurs fréquentes utilisations d'ordinateurs et d'Internet, les pratiques des collégiens sont caractérisées par un faible niveau de conceptualisation et un pauvre vocabulaire chez beaucoup d'entre eux. Finalement que ce soit à l'école, dans les familles ou avec leurs paris, *« si les jeunes générations sont très utilisatrices des technologies informatiques ou numériques qui leur sont proposées, elles s'en servent assez peu à des fins d'apprentissage »* (Bruillard, 2012, p. 25). Ses propos sont illustrés en référence à André (2006) pour le cas du logiciel de traitement de texte et au projet Didatab (Blondel &Tort, 2008) pour le cas du tableur. Ce manque de connaissance et de verbalisation dans l'usage des TIC n'est d'ailleurs pas limité aux seuls élèves, mais aussi à leurs enseignants, comme il le précise ici (Bruillard, 2006, p. 116) :

> *« si les mots peuvent apparaître comme des indices d'une connaissance, d'une compréhension, on constate d'ailleurs (...), autant chez les élèves que chez les enseignants, un manque de mots spécifiques pour désigner des éléments ou des processus intervenants dans l'utilisation des technologies informatiques ».*

Les problèmes posés par l'utilisation des technologies et plus particulièrement l'ordinateur pour apprendre ou enseigner n'est pas nouvelle. Au début du XXI^{e} siècle, Nicolle (2002), dans le symposium sur la didactique de l'informatique, posait la même question pour souligner l'importance d'avoir appris la technologie avant de pouvoir l'utiliser pour l'apprentissage : « *Peut-on enseigner/apprendre avec les TIC sans enseigner/apprendre les TIC ? ».* En

effet, si on peut avoir des difficultés à utiliser, pour l'apprentissage, un outil qu'on sait déjà utilisé, il est de loin difficile de l'utiliser pour l'apprentissage sans s'en être approprié au préalable. De ce fait, s'approprier un savoir informatique, nécessitant explicitement l'usage d'un ordinateur n'est pas automatique même si cet ordinateur est déjà approprié. Pour Georges Louis Baron, affirmant que « la nécessité de former à l'informatique et à ses outils d'autres étudiants que les futurs informaticiens », constate que l'acquisition d'une maîtrise n'est pas toujours l'objectif visé par des formations à l'informatique chez des apprenants. Si selon lui, la maîtrise reste une sorte d'horizon incontournable, il s'interroge si on peut enseigner des compétences dans l'enseignement supérieur en mettant de côté les savoirs, mais aussi « c*omment faire en sorte que les concepts indispensables puissent faire l'objet d'une appropriation et être mis en pratique ? »* (Baron, 2009).

2.4. Les années 2010 : retour aux concepts et un enseignement de spécialité

Dans un contexte de révolution numérique, l'acquisition d'une culture informatique pour tout citoyen est de plus en plus une nécessité. À quelques différences près, le contexte de fin des années quatre-vingts reste comparable avec celui de la période actuelle : cette nécessité de culture numérique se fait aujourd'hui de plus en plus sentir suite à une omniprésence des technologies numériques. Dans les années quatre-vingt-dix, Baron (1989) parlait de la large étrangeté de l'informatique dans la culture de la majorité des citoyens français. Si actuellement le même discours est encore tenu, il semble avoir déplacé le problème : acquisition de l'informatique comme élément de culture pour tout citoyen. Pour remédier à ce problème, le choix de la France a été le rétablissement d'un enseignement de l'informatique et des sciences du numérique comme discipline scolaire.

Depuis 2008, des prises de position demandant un enseignement d'une spécialité informatique et des technologies de l'information et de la communication tout au long de la scolarité[40] et particulièrement lycée ont été observées à grande échelle. Dans ce mouvement, le groupe Informatique et technologies de l'information et de la communication (ITIC) de la fédération des associations françaises de sciences et technologies de l'information et de la communication (ASTI) a été le porte-flambeau. Si le B2i et le C2i sont toujours en vigueur dans le cadre de l'enseignement de l'informatique à tous les niveaux de formation, l'année 2009 a été, en conséquence marquée par d'autres réformes orientées vers un retour à des

40 http://fr.wikipedia.org/wiki/Enseignement_de_l%27informatique_en_France, consulté le 28 décembre 2013

concepts informatiques après une large consultation des partenaires (Baron & Bruillard, 2011). Ces réformes ont d'abord concerné la classe de seconde avant d'arriver en terminale scientifique où, des savoirs informatiques et particulièrement des éléments de savoirs algorithmiques ont été introduits dans les programmes de mathématiques (Balliot et al., 2011, Baron, 2012). Par la suite, des expérimentations qui ont été effectuées dans certains établissements à partir de 2009 ont donné l'espoir d'un terrain idéal pour des pratiques enseignantes intéressantes face à l'informatique et aux sciences du numérique (Cabane, 2012).

En conséquence, une spécialité informatique et science du numérique (ISN) a été créée depuis l'année scolaire 2012 en classe de terminale (Dowek et al., 2011). Dans le souci d'offrir une culture générale informatique à tous les jeunes scolarisés, cette spécialité sera ouverte à toutes les séries de l'enseignement général à la rentrée prochaine 2014, précisent Dowek et al., (ibidem.). Publié dans le Bulletin officiel spécial n° 8 du 13 octobre 2011, le programme de cet enseignement précise le profil des candidats attendus : ce ne sont pas des experts en informatique, mais *« il s'agit d'un enseignement d'ouverture et de découvertes des problématiques actuelles, adapté à la société d'aujourd'hui, qui valorise la créativité et contribue à l'orientation »*[41].

Si la discipline informatique est récemment enseignée dans la scolarité secondaire en France, sa présence dans le système scolaire secondaire de la France date d'il y a longtemps : il est vieux de plus d'une quarantaine d'années à compter du colloque de Sèvres de mars 1970. Son enseignement a connu des suppressions et rétablissements répétitifs, ce que Jacques Baudé (Baudé, 2010c) appelle « *une politique en dents de scie* ».

Les raisons ayant milité pour son récent rétablissement sont nombreuses. Maurice Nivat les résume en un retard de l'industrialisation française lié à la dette de la France envers l'enseignement de l'informatique à la fois comme science et techniques (Nivat, 1985). Dans son article *la dette et l'informatique*, Maurice Nivat montre que l'innovation a, depuis des siècles, une entrée dans l'informatisation. Actuellement, cette dernière intervient non seulement dans des objets produits, dans des processus de leur production et de fabrication, mais aussi dans la gestion des entreprises. Développées par les défenseurs de l'enseignement d'une discipline informatique au lycée, respectivement dans leurs différents rapports (rapport de l'académie des sciences, 2013) et celui de la société informatique de France (SIF) (Alayrangues et al.,

41 http://www.education.gouv.fr/pid25535/bulletin_officiel.html ? cid_bo=57572 : Enseignement de la spécialité d'informatique et science du numérique de la série scientifique – Classe terminale.

2013), les raisons de cette réintroduction de l'informatique dans l'enseignement général sont développées en quatre points :

« (1) *Une prise de conscience des limites de l'industrie française à la compétitivité internationale : Il y a un manque de spécialistes en informatique alors que les progrès technologiques les plus importants sont actuellement des produits directs de l'informatique. Une telle situation rend impossible la mise à disposition des besoins industriels ;*

(2) Une influence de plus en plus grandissante de la science informatique sur les autres disciplines scientifiques : l'informatique est non seulement utilisée, comme outil dans l'enseignement des autres sciences mais aussi, elle transforme de façon plus profonde leur nature notamment leur construction, leur méthode… ;

(3) Une importance de la culture informatique fondée sur des concepts et savoirs homogènes et stables dans un contexte de développement galopant des outils numériques pour être pour s'adapter aux mutations sociétales actuelles et futures ;

(4) Un rôle important du raisonnement informatique dans la résolution de nombreux problèmes, pouvant aider efficacement à la résolution des problèmes mais l'informatique peut remotiver les élèves démotiver ou en échec. »

Chez les décideurs politiques, une volonté de généralisation de cet enseignement semble évidente au-delà du lycée scientifique. Dans les classes préparatoires aux grandes écoles (CPGE), un enseignement de l'informatique fait partie des nouveaux programmes en œuvre depuis la rentrée 2013-2014[42] pour la première année et 2014-2015 pour la deuxième année. La Société Informatique de France (SIF) se dit satisfaite de la généralisation de la spécialité d'ISN et surtout du fait que des contenus d'ISN enseignés au CPGE sont plus larges que ceux du lycée (Alayrangues et al., 2013). Nous présentons, en annexe (Annexe 14), une synthèse des dates clés qui ont caractérisé l'histoire de l'informatique scolaire en France.

En conclusion, la France a été pionnière dans l'adhésion aux conclusions du colloque de Sèvres pour instaurer un enseignement de l'informatique, sous forme optionnelle, au lycée générale. Mais, son caractère élitiste n'a pas permis sa pérennité. L'absence de cet enseignement d'une discipline informatique pendant une période d'au moins 20 ans (1988-2010) a été perçue comme une perte pour la France : les jeunes ont continué a découvrir l'informatique en dehors de l'école. Certaines associations ou initiatives privées sont nées, en marge de

42 http://www.enseignementsup-recherche.gouv.fr/cid66194/renovation-des-programmes-des-c.p.g.e-rentree-2013.html

l'école, pour les accompagner en vue de les aider à construire des savoirs et des savoir-faire structurés. Le point 4. suivant s'intéresse à quelques-unes d'entre elles.

3. Au cœur du débat : algorithmique et programmation

Dans la décennie 1970, l'informatique a surtout été considérée comme une « démarche ». Particulièrement utilisée dans l'enseignement de l'algorithmique et de la programmation, cette démarche était motivée par des compétences à mobiliser. Elle était servie par différentes méthodes, comme la méthode déductive (Médée) de Claude Pair (1988)[43]. Cette méthode permettait la construction systématique des programmes. D'abord, un algorithme était exprimé sur les objets d'un problème donné, ce qui permettait l'introduction des fonctions et des procédures, et par conséquent un choix d'une représentation des objets qui favorisait l'expression de ces fonctions et procédures.

La méthode consistait en une élaboration d'un plan de résolution de problèmes, s'appuyant sur des schémas, avant sa mise en application. Le plan peut être considéré comme une représentation pouvant servir de guide à une activité notamment de résolution de problème. Cela suppose « *la mise en œuvre d'une hiérarchie d'espaces abstraits »* constituée d'espaces de niveau supérieur où un «*problème peut être représenté sous une forme schématique, plus réduite, qui permet d'abstraire les détails traités dans un espace de niveau inférieur »* (Hoc, 1988) : l'objectif est la construction d'un plan qui n'est autre qu'une représentation hiérarchisée, et par conséquent décomposable, d'un objet. Un programme informatique est donné par Hoc comme un exemple d'un tel plan : c'est le programme qui guide le fonctionnement d'une machine.

Le champ de la disciplinaire informatique était en construction au niveau de l'enseignement supérieur entre la fin des années 1960 et le début de 1970. Les mathématiques appliquées ont assuré le lieu de son enseignement avant de sortir de cet habitat pour devenir une discipline autonome sous forme de programmation. Ce sont des contenus de la programmation qui ont donné naissance les contenus de la discipline informatique universitaire qui était enseignée au début dans les premières années du supérieur.

Dans la scolarité secondaire, ce champ informatique a connu des obstacles pour être reconnu comme discipline. Si les objets informatiques ont vite évolué, le système éducatif a lui aussi évolué, mais à une vitesse différente. Ce dernier a eu de la peine à prendre en compte les dis-

43 Claude Pair est l'un des pionniers de l'informatique universitaire en France.

positifs informatiques dans leurs générations. Néanmoins, des réponses, parfois variables et contrastées ont été toujours apportées dans la scolarité secondaire et à l'université où la discipline informatique avait déjà sa place.

En 1993, lors de la troisième rencontre francophone organisée par l'Association francophone de didactique de l'informatique (AFDI), la communauté scientifique réunie à Sion, a trouvé pertinente la question de la place de la programmation dans une initiation informatique[44] (AFDI, 1993, p. 232) : *« Peut-on enseigner l'informatique sans faire une place à la programmation et comment ? Ce qui pose la question de la place de la didactique de l'informatique sans langage (classique) de programmation ».*

Le domaine « *algorithmique et la programmation »* est considéré par la communauté scientifique comme étant au cœur de la discipline informatique (Viallet & Venturini, 2010). Il constitue une base d'un enseignement d'initiation à l'informatique aux lycéens. Jugées inséparables, l'algorithmique et la programmation forment un couple qui devrait être dispensé ensemble pour un enseignement de l'informatique digne de nom (Nivat, 2009 ; Baron & Bruillard, 2011). Pour Nivat, *« la programmation est elle-même une algorithmique un peu particulière : un langage donne des possibilités d'expression que peut-être un autre ne donne pas, et réciproquement »*.

Le contexte d'environnements langagiers diversifiés a été à l'origine du choix privilégié pour l'enseignement de l'informatique dans la scolarité secondaire : l'approche *outil*. Si la question de savoir quelle informatique enseigner aux débutants est récurrente, actuellement, nous vivons une période où *« l'opposition traditionnelle entre les aspects outil d'enseignement/objet d'enseignement s'est affaiblie » (Baron, Bruillard, & Komis, 2011)*. Une convergence semble être sur la nécessité de l'enseignement de l'informatique comme *objet* est internationale (CSTA, 2005 *; Furber, 2012)*.

La notion d'« *outil (informatique)* », développée avec la disparition de l'option informatique des années quatre-vingt au lycée (Bruillard, 1997), s'est davantage amplifiée avec la prolifération des technologies informatiques qui ont envahi tous les milieux économiques, sociaux et éducatifs. Avec cette période, l'algorithmique et la programmation étaient des concepts institutionnellement valorisés, qui ont fait objets d'enseignement avec cette option informatique. Par la suite elles ont été considérées comme des notions de seconde zone non nécessaires que dans le monde technique, au profit des progiciels (Baron et al., 2011).

44 http://www.epi.asso.fr/association/dossiers/d14som.htm, consulté le 18 novembre 2014

Aujourd'hui, un autre rapport vis-à-vis de l'algorithmique et la programmation est né chez les décideurs, les institutionnels et les spécialistes de l'éducation conscients de leur intérêt dans le domaine : ils sont d'une importance capitale dans l'apprentissage de l'informatique. Cette convergence à l'algorithmique et la programmation dans la communauté scientifique informatique et des spécialistes des sciences de l'éducation n'est pas seulement limitée à la France mais, revêt un caractère international. Un rapport[45] (ACM, AIS, & IEEE-CS) des associations qui s'intéressent et s'impliquent dans l'enseignement de l'informatique dans les pays anglo-saxons, donne cinq disciplines informatiques enseignées dans l'enseignement supérieur. fait savoir les composantes sur lesquelles se focalisent ces enseignements de la discipline informatique. Les cinq disciplines informatiques enseignées ont en communs deux composantes informatiques liées à une introduction à la programmation, à savoir : « acquérir les fondamentaux de la programmation » et « être capable de réaliser des programmes simples ».

L'algorithmique et la programmation, deux des composantes de l'informatique, sont considérées par Dowek (2005, p. 2) comme prioritaire pour un enseignement de l'informatique au lycée :

> *« l'enseignement en informatique au lycée doit avoir comme objectif premier que les lycéens sachent écrire un programme au moment de passer leur baccalauréat. On peut concevoir un enseignement de l'informatique en trois temps. L'apprentissage de l'utilisation d'un ordinateur, qui relève sans doute de l'école primaire ou des premières années du collège, l'apprentissage de la programmation et des algorithmes élémentaires au lycée, et l'apprentissage de la science du calcul dans l'enseignement supérieur »*

Cette vision est partagée. Selon Muratet et al., (2008), « *l'apprentissage de la programmation est une clé essentielle et incontournable »* dans l'apprentissage de l'informatique. Or, qui dit programmation dit un langage. Selon eux, chez les débutants, certains logiciels sont plus adaptés pour un apprentissage fructueux d'une initiation à la programmation : ce sont notamment des logiciels *« utilisant des langages graphiques à blocs ou à base de blocs »* (Voir pour plus de détails, la classification des recherches au chapitre IV.).

45 http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf, document consulté le 1er février 2015

Comme étape en amont de la programmation, l'algorithmique ne doit pas être exclue. Cet apprentissage se focaliserait sur l'étude de la conception et de l'écriture d'algorithmes et de programmes, des étapes vues comme à la base de l'informatique (Rivière et al., 2011). Gilles Dowek justifie ce choix de l'algorithmique et la programmation par son intérêt lié à leurs potentialités cognitives chez les lycéens (Dowek, 2005, p. 2).

> *« Cependant, le point de vue que je défends n'est pas uniquement motivé par une telle vision globale. Il est aussi motivé par le fait que l'apprentissage d'un langage de programmation et des algorithmes élémentaires me semble une expérience propre à apporter beaucoup aux lycéens, quant à leur développement personnel et dans leur relation à la connaissance ».*

Si les termes « algorithmique » et « programmation » ont tendance à être distingués par certains auteurs, d'autres auteurs montrent plutôt un fort lien entre eux et, même à les assimiler. Ce que soulignent humoristiquement Élie, Gurerry et Lacroix (Élie et al., 2010) dans cet extrait :

> *« Comment croire que celui qui a prévu de regarder un film et qui passe une heure à réinstaller les codes, convertit, recolle le son et resynchronise les sous-titres, et qui ne regardera pas le film, est l'homme libre ? Le suprême aliéné n'est pas l'homme du tuning ? C'est là ce qu'il faut distinguer deux programmations. Celle qui se fait passer par telle et qui n'est que configuration. Et la vraie programmation, qui est algorithmique »*

Quelques arguments en faveur de la pertinence et des potentialités de cette composante « algorithmique et programmation » dans le cas du lycée sont notamment une acquisition d'une culture scientifique et technologique, une pensée algorithmique et modélisante, une école de la rigueur…

4. Enseignement de l'informatique en marge du système scolaire

4.1. Des initiatives privées sous forme de concours

Avec la prolifération des technologies informatiques, beaucoup de jeunes apprennent leurs utilisations en dehors du cadre scolaire, le plus souvent à la maison et/ou avec des pairs. Beaucoup de recherches ont montré le caractère limité de tels apprentissages (Fluckiger &

Bruillard, 2008 ; Archambault, 2010). La motivation constatée chez les jeunes face aux outils technologiques d'une part et la faible présence institutionnelle de l'organisation d'un enseignement de l'informatique dans beaucoup de pays, a fait naître chez certaines gens, des convictions de l'existence d'« *un véritable enjeu à faire découvrir aux jeunes l'informatique autrement* » (Tort & Dagiène, 2012 ; Drot-Delange, 2013) : différentes initiatives de sensibilisation, d'encadrement et de formation des jeunes à l'informatique ont été créées et développées dans des cadres extrascolaires avec une organisation le plus souvent différente et adaptée aux contextes des pays, des niveaux d'apprenants… Dans beaucoup d'initiatives, un rôle important est joué par des bénévoles passionnés de l'informatique qui s'engagent au côté des jeunes dans les associations pour les soutenir en multipliant exercices et cours pour leur entraînement (Février, Hiron, & Charguéraud, 2011).

Nées dans des contextes locaux, quelques-unes de ces organisations, ont vite conquis les nations et acquis des postures internationales. Nous présentons une liste non exhaustive de quelques-unes de ces initiatives : concours castor-informatique[46], concours IOI[47], concours Prologin[48], concours Algorea[49], concours « Agora programming Contest » (APC)[50], concours FARIO[51], concours USACO[52] (USA Computing Olympiad).

Dans le paragraphe suivant, nous développerons les deux premières initiatives. Deux raisons ont motivé leur choix. D'une part, ce sont des initiatives connues pour être plus actives et, d'autre part, certaines des personnes actives qui y sont impliquées sont accessibles et peuvent nous donner d'informations complémentaires si besoin, que celles trouvées sur le site.

a. Le concours Castor informatique

- **Genèse, objectif et développement**

Connu sous le nom de « Bebras[53] Contest », le concours Castor est né en Lituanie en 2004 sur l'initiative d'une formatrice des enseignants et chercheuse en informatique : Valentina

46 http://www.castor-informatique.fr/, site consulté le 30 décembre 2013
47 http://www.france-ioi.org/ioi/index.php. C'est un concours d'Olympidades internationales d'informatique. Site consulté le 3 janvier 2014
48 http://www.prologin.org/association, site consulté le 3 janvier 2014
49 http://www.france-ioi.org/concours/algorea2013, site consulté le 30 décembre 2013
50 http://www.ginfo.ro/contest/, site consulté le 3 janvier 2014
51 http://orac.amt.edu.au/fario/, site consulté le 3 janvier 2014
52 http://www.france-ioi.org/concours/usaco.php. C'est un « USA Computing Olympiad. Site consulté le 3 janvier 2014
53 Le choix de ce nom Bebras qui signifie Castor en Lituanien a été motivé par les caractéristiques de cet animal qui combine à la fois l'ingéniosité, la ténacité et l'intelligence (Tort & Dagiéné, 2012)

Dagiène. Sa motivation était de « sauver » la science informatique quand son enseignement venait d'être remplacé par un *« enseignement des technologies de l'information »* (Tort & Dagiène, 2012). Un projet inspiré du concours Kangourou[54] de mathématiques, visant la réintroduction et l'apprentissage de l'informatique de façon ludique au travers d'un concours, a été construit et présenté au ministère de l'Éducation, et l'a accepté. Organisé pour la première fois en octobre 2004, le concours Castor a connu un succès quant à son organisation et sa participation (Dagiene, 2005). Le succès de cette « expérience » a vite inspiré d'autres pays qui ont adhéré à cette initiative. Jusqu'en 2013, le concours était organisé dans 21 pays dont la France qui a rejoint l'initiative, sept ans après, en 2011.

Ayant pour but de faire découvrir aux jeunes scolarisés l'informatique et les sciences du numérique, le concours Castor informatique s'intéresse plus particulièrement aux concepts fondamentaux de l'informatique et la maîtrise des méthodes de résolutions propres à l'informatique (Tort, 2011). Tous les aspects de la science informatique[55] y sont abordés : information et représentation des données, pensée algorithmique, utilisations des logiciels, structures des données, jeux logiques, informatique et société. Il s'intéresse à tous les élèves de collèges et lycées sans distinction de filières. Son organisation se fait selon les niveaux scolaires. Les collégiens sont regroupés deux niveaux : niveau 1 (élèves de 6e et 5e) et, le niveau 2 (élèves de 4e et 3e). Comme au collège, les lycéens sont aussi regroupés en deux niveaux[56] : niveau 3 (élèves de 2nde) et le niveau 4 (élèves de 1ère et terminale). Pour chaque classification d'élèves, les problèmes posés font intervenir trois niveaux de difficultés différents : facile, moyen et difficile.

En amont du concours, l'encadrement est assuré par des enseignants issus de diverses disciplines enseignées (Tort et al., 2013) : mathématiques, technologie au collège et des sciences de l'ingénieur au lycée. L'implication des enseignants varie selon les niveaux de scolarité : au lycée, ce sont les enseignants de mathématiques qui sont les impliqués alors qu'au collège, ceux de technologie sont relativement plus nombreux. Le succès qu'a connu le concours est justifié par son intérêt sur la qualité des contenus abordés et l'approche utilisée :

54 http://www.mathkang.org/concours/kangpres2014.html. Consulté le 30 décembre 2103
Le Concours Kangourou est un concours des mathématiques, né en 1991. Plus grand jeu de concours scolaire qui réunit plus de six millions de jeunes au monde

55 http://castor-informatique.fr/index.php, site consulté le 30 décembre 2013

56 Les deux niveaux seront respectivement appelés niveau trois et quatre pour les distinguer des deux niveaux précédents

focalisé sur le programme de formation, les contenus sont abordés différemment qu'en contexte de classe.

Les résultats révèlent une évolution croissante annuelle autant en nombre d'élèves que d'établissements qui participent[57]. Les effectifs d'élèves n'augmentent pas seulement mais, tendent aussi à doubler d'une année en année. Le succès de ce concours témoigne de la motivation et l'intérêt qu'ont les jeunes élèves et les établissements vis-à-vis de l'apprentissage de l'informatique. Organisée hors de l'école, cette initiative constitue une contribution à l'enseignement/apprentissage de l'informatique chez les jeunes.

- **Contenus des épreuves : des problèmes à résoudre**

Plusieurs notions informatiques interviennent dans les épreuves sous forme de petits problèmes à résoudre choisis dans des situations de la vie courante. Si leur résolution nécessite des notions informatiques, il n'y a pas de nécessité au préalable qu'elles soient acquises de façon structurée sous le modèle scolaire ou académique (Tort, 2011). Les problèmes ouverts sont rares, les questions sont essentiellement à choix multiples (QCM). Néanmoins, la réussite à l'un ou l'autre problème exige une réflexion au préalable étant donné que le chemin à suivre dans leur résolution n'est pas immédiat (Cartelli et al., 2010). Comme le souligne Françoise Tort, les problèmes donnés requièrent une double exigence : chez les élèves, d'une part, ils nécessitent de mobiliser des notions et méthodes en rapport avec l'informatique et, chez les encadrants, d'autre part, ils doivent poser des problèmes adaptés et abordables selon les niveaux des « élèves n'ayant pas eu de formation spécifique en informatique ». De plus, les problèmes posés doivent couvrir de façon la plus large possible le champ de la discipline informatique (algorithmique, programmation, structures et représentation des données, logique, utilisation des logiciels.), tout en mettant en relations l'informatique comme science et l'informatique comme usages de logiciels (Dagienė & Futschek, 2008).

b. Concours France-IOI

- **Historique, but et organisation**

L'« International Olympiad in Informatics » (IOI) est une compétition internationale ouverte aux jeunes pour l'algorithmique et la programmation. Inspirée du succès de l'Olympiade Internationale de Mathématique (IMO), l'idée de son organisation date de 1987, avant qu'elle ne soit mise en place (Février et al., 2011 ; Verhoeff et al., 2006). La première compétition

57 http://castor-informatique.fr/resultats.php, consulté le 30 décembre 2013

de ce genre a été organisée en 1989 en Bulgarie et depuis, ce concours a lieu chaque année. Aujourd'hui organisé dans plus de 90 pays (Burton, 2008), ce concours a connu la première participation de la France depuis 1996.

L'objectif principal de ce concours est de faire naître chez les jeunes collégiens et lycéens, un intérêt pour l'informatique, notamment en favorisant les rencontres pour les plus talentueux en vue d'échanger leurs diverses expériences scientifiques et culturelles (Février et al., ibidem.). Son organisation est faite à tour de rôle par chaque pays membre[58]. Chaque pays est représenté par quatre candidats sélectionnés[59]. L'éligibilité est soumise à certaines contraintes : un nombre important d'exercices doivent être faits pour espérer y participer.

- **D'une formation sans limite à un encadrement hybride**

Dans la compétition des olympiades, chaque épreuve comporte trois à quatre sujets d'algorithmique. De plus, les candidats doivent être capables de programmer. Les élèves qui participent à la compétition doivent être capables de programmer dans l'un des trois langages au choix : Pascal, C ou C++. Les problèmes posés contiennent des questions de difficultés différentes et croissantes (Burton & Hiron, 2008).

Malgré les contenus fixés par l'Olympiade (Forisek, 2008), si un accent particulier est mis sur des notions de base de la programmation (notion d'instruction, de variable, de condition, de boucle…), la formation donnée aux élèves n'a pas de limite. La préparation des candidats au concours suit trois étapes (Février et al., 2011). La première étape consiste en un cours et beaucoup d'exercices, orientés dans le sens d'un enseignement formel. Ces derniers sont proposés sur le site[60] de telle sorte que leur succession rende possible un apprentissage en autonomie pour les élèves. La deuxième étape est dédiée aux notions algorithmiques relativement dures : *« types de données, fonctions, logique booléenne et opérateurs binaires, tableaux et structures »*, dans le but d'amener l'élève à les utiliser afin de se familiariser avec les principes de la programmation, pour tout langage utilisé. La troisième étape concerne les aspects algorithmiques et, est plus orientée vers l'acquisition et le développement de la pensée algorithmique.

Dans tous les pays, une approche hybride est utilisée dans l'encadrement des élèves : une approche en ligne et une approche en présentiel. Beaucoup d'exercices sous formes de pro-

58 http://www.france-ioi.org/ioi/index.php, site consulté le 3 janvier 2014
59 http://www.prologin.org/association : Prologin est ue association qui date de 1991 et, est œuvre en faveur du soutien des jeunes passionnés de la programmation
60 http://www.france-ioi.org/, site consulté le 4 janvier 2014

blèmes sont préparés pour les élèves et sont mis en ligne pour permettre leur accessibilité de chez eux. Ces problèmes doivent susciter la motivation des élèves : non seulement, ils sont diversifiés mais aussi, ils sont conçus de telle sorte que l'apprentissage soit amusant et progressif au rythme de chacun avec un accès à des aides et même des corrections sont prévues si jamais l'élève est bloqué, précise Loïc Février.

En présentiel, les élèves sont rassemblés au même endroit pour faciliter leur encadrement. Durant une semaine, ils sont réunis en groupes de huit élèves pour des raisons d'efficacité.

Comme en France, en Belgique, les organisateurs de cette compétition affirment donner une « *formation théorique intensive* »[61] à leurs élèves, une formation justifiée par les exigences de la compétition notamment en programmation alors que l'informatique n'est pas enseignée dans la scolarité secondaire.

c. Conclusion : des apports complémentaires

Bien que totalement différentes en ce qui concerne leur identité et organisation, les deux initiatives fonctionnent de façon complémentaire et permettent l'acquisition des connaissances complémentaires. Elles contribuent toutes à l'acquisition de la culture informatique chez les élèves de la scolarité secondaire. Si le concours castor s'intéresse plus aux élèves plus jeunes essentiellement de collège et focalise ses enseignements sur l'approche algorithmique, le concours France-IOI, lui, s'intéresse aux moins jeunes, surtout des lycéens et se focalise sur l'apprentissage de la programmation. Autant les contenus abordés dans les deux initiatives sont complémentaires, autant les approches utilisées le sont aussi. Il est, néanmoins, intéressant de constater une focalisation sur une compétence commune à tous les élèves dans les deux initiatives : la pensée algorithmique, acquise puis mobilisée dans la résolution des problèmes. Se référant à Janet Wing (2006), Françoise Tort explique le rôle essentiel de cette compétence : « *elle est utilisée dans des choix des modes de représentation, de sélection, d'interprétation et de transformation des informations, considérés comme des modes fondamentaux du raisonnement informatique* » (Tort, 2011, p. 225).

Située à l'interrelation entre les sciences mathématiques et informatiques, l'algorithmique et surtout la pensée algorithmique qu'elle permet de développer au travers des problèmes prescrits permet aux élèves de développer un autre raisonnement en mathématiques et même plus tard en informatique. Simon Modeste qui s'est intéressé aux apports de la pensée algorith-

61 http://www.rtl.be/info/monde/europe/910830/deux-medailles-pour-la-belgique-a-l-olympiade-internationale-d-informatique, site consulté le 13 novembre 2014

mique dans la résolution des problèmes, trouve indispensable la caractérisation de ces problèmes et des champs dans lesquelles cette pensée (algorithmique) peut être pleinement mise en œuvre (Modeste, 2012b). Il souligne aussi l'efficacité de la centration directe sur les algorithmes dans l'apprentissage de la résolution des problèmes plutôt que l'apprentissage direct des méthodes de résolution des problèmes. Il justifie ainsi, selon nous, le pourquoi des problèmes posés par les initiatives Castor informatique et France-IOI autant pour les épreuves que pour les problèmes d'entraînement prescrits aux élèves doivent respecter des critères de choix bien déterminés pour arriver à construire efficacement la compétence attendue chez les élèves.

Les effectifs volontaires toujours croissants qui s'inscrivent à ces formations sont un signe de motivation des élèves à l'apprentissage de l'informatique. Ceci justifie la nécessité d'un enseignement de la science informatique aux jeunes. Depuis un certain temps, cette urgence de l'enseignement de l'informatique est une question récurrente dans des recherches actuelles, demandant la généralisation de l'enseignement de l'informatique (Rapport de l'académie des sciences, 2013).

Si les initiatives en contextes privés apportent leurs contributions à l'acquisition d'une culture informatiques aux élèves, elles ne peuvent pas se substituer à des enseignements scolaires. En attendant la généralisation de cette discipline scolaire informatique à toutes les filières, de telles initiatives privées d'enseignement/apprentissage de l'informatique chez les élèves méritent d'être soutenues et encouragées.

4.2. Un rôle important des associations

En France, à côté des initiatives privées orientées concours informatique chez les élèves scolarisés, et jouant un rôle important dans leur encadrement pour l'apprentissage de l'informatique, plusieurs autres associations militent en faveur de l'enseignement/apprentissage de l'informatique. Nous nous intéressons de trois associations pour les mêmes raisons que précédemment : l'EPI, le SILO, la SIF. Cette section est consacrée au développement succinct de leur identité et contributions.

a. L'Association pour l'enseignement public de l'informatique (EPI)

L'Association « Enseignement Public et Informatique » (EPI)[62] est née en 1971 au lendemain de la mise en place d'un enseignement scolaire d'informatique en France. Son rôle pour la

62 http://www.epi.asso.fr/, site consulté le 3 décembre 2014

création et l'organisation de l'enseignement d'une discipline informatique en France a été est important.

Initialement connue comme une association d'enseignants des lycées formés en informatique, sa structure a évolué avec les différentes phases qu'a connues l'enseignement de l'informatique. Actuellement, l'EPI s'intéresse à tous les niveaux de la scolarité, de la maternelle à l'université[63]. Depuis sa création, elle plaide pour un enseignement de l'informatique, et prône la complémentarité des approches (Baudé, 2011) : informatique objet d'enseignement et informatique moyen d'enseignement.

Son site est devenu une plateforme des ressources pour l'enseignement de l'informatique, une archive des ressources sur l'histoire de l'enseignement de l'informatique et, un cadre de comparaison de la France avec les autres pays du monde, en matière de l'enseignement de l'informatique.

b. Science Informatique au Lycée Oui ! (SILO)

La SILO[64] est une plateforme des ressources pédagogiques. Initialement créée en partenariat[65] par le centre national de documentation pédagogique (CNDP), l'Institut national de recherches en informatique et automatisme (INRIA) et l'association Pasc@line, l'initiative SILO a vite élargi ses partenaires qui ont été associés : la Société Informatique de France (SIF), la Fuscia[66], l'EPI et ePrep, et beaucoup d'autres chercheurs et enseignants du secondaire.

Son objectif est de contribuer à l'enseignement de la discipline informatique au lycée notamment par la conception et la mise à disposition des enseignants et des élèves des ressources nécessaires[67]. Par exemple, un logiciel Java's Cool[68], a été conçu par INRIA sur demande de certains enseignants des lycées. Son intérêt dans l'apprentissage de la programmation chez les débutants est surtout justifié par son aspect ludique :

> *« On peut s'initier à la programmation de manière ludique, puisque les petits programmes créés (dits « proglets »[69]) animent un élément visuel ou sonore. On peut donc*

63 http://fr.wikipedia.org/wiki/Enseignement_public_et_informatique, site consulté le 4 janvier 2014
64 https ://science-info-lycee.fr/une-equipe-a-votre-service/, site consulté le 4 janvier 2014
65 http://isn.ac-amiens.fr/spip.php ? article25, site consulté le 4 janvier 2014
66 https ://fuscia.info/, site consulté le 5 janvier 2014 : c'est un partenaire entre INRIA et les universités numériques
67 https ://wiki.inria.fr/sciencinfolycee/Accueil, site consulté le 4 janvier 2014
68 http://javascool.gforge.inria.fr/index.php, site consulté le 3 janvier 2014
69 Des « proglets » sont définis comme des objets numériques qui font intervenir des concepts clés manipulés

mieux comprendre ces objets numériques à travers une démarche expérimentale concrète, qui mène à la compréhension des notions abstraites des sciences de l'information. Des textes de base sur ce qu'est l'information, comment sont codés les objets numériques et sur les ingrédients des algorithmes, sont référencés »[70].

Selon les informations données par la plateforme, si les versions de ce logiciel évoluent rapidement, sa récente version, version 4, a été conçue par deux élèves de lycée eux-mêmes : elle est une adaptation des versions précédentes adaptée à un apprentissage de la programmation chez les lycéens débutants par la création des jeux à deux dimensions (2D)[71]. Mais, son utilisation peut s'étendre jusqu'au début de l'université.

c. *Société Informatique de France (SIF)*

Créée en 2012 pour succéder à la société des personnels enseignants et chercheurs en informatique de France (SPECIF), datant de 1985, la SIF[72] est une association française régie par la loi de 1901[73]. Elle est un espace d'échanges sur les enjeux de l'informatique et d'actions de la communauté. Son but se résume en quatre points suivants[74] :

- favoriser le développement de l'enseignement et de la recherche en informatique,
- développer les échanges entre les établissements d'enseignement de l'informatique, les organismes et laboratoires de recherche et le monde socio-économique,
- plus généralement, favoriser, par la formation tout au long de la vie et la recherche, l'évolution professionnelle des acteurs, publics ou privés, du secteur,
- *œuvrer à ce que l'informatique et les sciences du numérique contribuent au développement économique et social*.

Pour remplir sa mission, elle procède à une organisation des rencontres scientifiques[75] autour de l'informatique, des correspondances partenariales et des publications d'un bulletin trimes-

« en les programmant et non en cliquant, et qui concourent à la compréhension des notions abstraites au cœur des sciences des sciences informatiques : ce sont des « grains pédagogiques interactifs pour un apprentissage ludique d'une notion » donnée. http://javascool.gforge.inria.fr/index.php ? page=proglets&action=info, site consulté le 7 janvier 2014

70 http://javascool.gforge.inria.fr/index.php, site consulté le 6 janvier 2014

71 http://javascool.gforge.inria.fr/? page=contact&action=credits, site consulté le 7 janvier 2014

72 http://www.societe-informatique-de-france.fr/, site consulté le 5 janvier 2014

73 http://fr.wikipedia.org/wiki/Loi_de_1901 : loi régissant des associations sans but lucratif

74 http://fr.wikipedia.org/wiki/Soci%C3 %A9t%C3 %A9_informatique_de_France, site consulté le 5 janvier 2014

75 Ces rencontres se retrouvent sous diverses formes : séminaires, journées de réflexion et congrès

triel[76] sur l'informatique. Elle rend aussi officiel ses positions par rapport aux décisions prises relativement à l'enseignement de l'informatique en France ou ailleurs. Pour illustration, deux exemples, l'un national et l'autre international, peuvent être donnés. Un exemple, national, est celui d'un communiqué sur un projet d'une éventuelle mention Mathématique-Informatique en Masters en France (SIF, 2013a)[77]: elle plaide pour leur séparation. Un exemple international est sa sortie médiatique dans une posture de non-approbation à propos de l'intention de supprimer l'enseignement de l'informatique au lycée par le Gouvernement grec, officialisée à travers son communiqué du 28 août 2013 (SIF, 2013b).

d. Conclusion : plaider et sensibiliser

Les différentes associations à but non lucratif et/ou leur collectif partenarial affichent des rapports favorables à l'enseignement d'une discipline informatique. Œuvrant en dehors des institutions publiques, leurs points de vue, ne sont pas des décisions en rapport avec l'enseignement de l'informatique, mais peuvent avoir une influence à plus ou moins long terme sur les décisions politiques. Leurs contributions se situent en amont et en aval de la décision politique. En amont, leur contribution est donnée sous forme de plaidoiries à destination des instances de prise de décisions. En aval, elles apportent des soutiens aux enseignants notamment par la mise à leur disposition des ressources dont ils peuvent avoir besoin pour leurs documentations, la création des conditions favorables d'échanges et de collaboration entre collègues enseignants.

Les contributions données sont variées et souvent complémentaires. L'analyse de leurs contributions révèle que toutes les associations ont des points communs : plaider et sensibiliser les décideurs politiques quant à la nécessité de l'enseignement de l'informatique. D'autres, par contre, vont jusqu'à accompagner l'enseignant dans sa pratique professionnelle. C'est le cas de SILO. Dans tous les cas, leurs discours et actions ont un impact sur l'état actuel de l'enseignement de l'informatique en France même si d'autres étapes restent à franchir.

Après cette description de la situation française, nous nous intéressons dans la section suivante à une brève analyse de ce qui se passe au niveau international relativement à l'enseignement de l'informatique au second degré de la filière scientifique.

76 http://www.societe-informatique-de-france.fr/bulletins/1024_001.html, site consulté le 5 janvier 2014
77 http://www.societe-informatique-de-france.fr/actualite/2013.html, site consulté le 5 janvier 2014

5. Situation internationale : analyse comparative

Dans cette section, il ne s'agit pas d'analyser chaque situation, mais de donner une brève vision d'ensemble de l'enseignement de l'informatique comme discipline scolaire au lycée au niveau international.

5.1. Méthodologie

Nous avons utilisé les articles et témoignages sur la situation de l'informatique en France et dans le monde issus du site de l'Association ÉPI[78], du rapport de l'académie des sciences de mai 2013 et, l'analyse comparative des situations de l'enseignement de l'informatique au second degré au niveau international (Tort, 2013). Nous nous sommes proposé de faire une analyse comparative des situations de l'enseignement de l'informatique comme discipline scolaire dans quelques pays du monde. Quatorze pays représentant tous les continents ont été choisis en fonction de la présence d'un enseignement de la discipline informatique au lycée.

En vue d'une analyse comparative, une grille a été conçue. Quatre variables ont été retenues : la terminologie – « (Computer Science » (CS) ou « Information Technology » (IT)) – utilisée pour la discipline informatique ; niveaux scolaires de l'enseignement ; caractère de la spécialité (obligatoire ou optionnel) ; formation initiale des enseignants. Une comparaison des situations de l'enseignement de l'informatique au niveau international a été effectuée (Annexe 15). Les résultats synthétiques sont présentés sur l'illustration III.1 suivante.

L'approche TIC n'a pas été mentionnée : elle est présente dans tous les pays concernés.

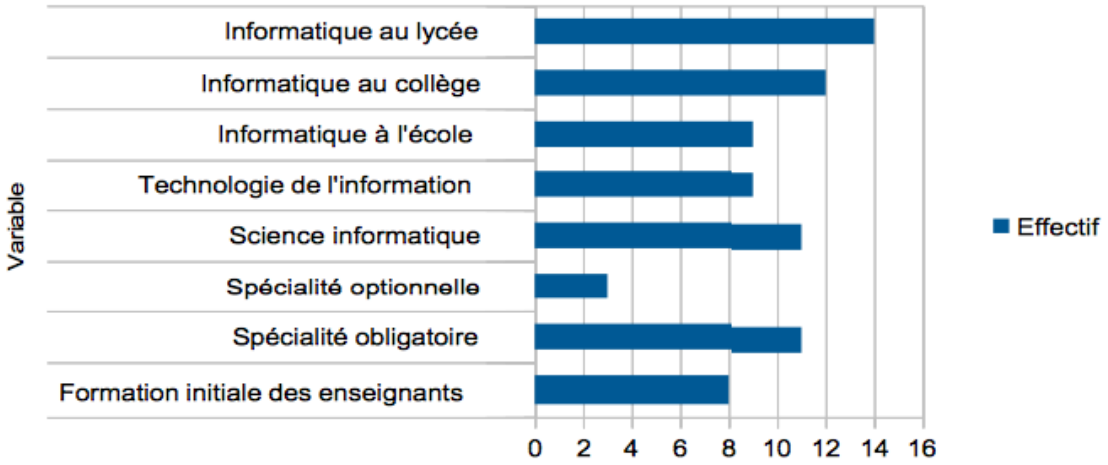


Illustration III.1: Variabilités de l'enseignement de l'informatique (Nbre de pays N=14)

78 http://www.epi.asso.fr/revue/docu/d0912b.htm, Site consulté le 8 novembre 2014

Sur un échantillon de 14 pays ayant institué un enseignement de la discipline informatique dans leur système éducatif, les résultats montrent qu'une discipline informatique est organisée dans 100 % au lycée (tous les pays considérés), dans 85 % de cas (12 pays sur 14) au collège et 64 % de cas (9 pays sur 14) au primaire. Deux formes d'informatique objet sont enseignées : la technologie de l'information (TI) dans 64 % (9 pays sur 14) et la science informatique (CS) dans 78 % (11 pays sur 14) de cas. La spécialité informatique est obligatoire dans 78 % de cas (11 pays sur 14) et optionnelle dans le reste de cas, c'est-à-dire 22 % (3 pays sur 14). La formation initiale d'enseignants d'informatique est organisée dans 8 pays sur 14, soit 57 % de cas.

5.2. Discussion des résultats et conclusion

a. Vers une généralisation de deux spécialités informatiques en parallèle ?

L'analyse des résultats révèle une situation frappante : il n'y a pas seulement une généralisation de la création d'une discipline informatique, mais un parallélisme de deux spécialités informatiques. Pour la majorité des pays, cette discipline est jeune : elle est née après les années 2000 et, a un caractère obligatoire dans plus de trois quarts de cas (78 %). Dans certains d'entre eux, deux approches, à savoir « CS » et « IT », sont associées. Cela semble être justifié par le fait que les deux approches sont complémentaires, comme l'affirme l'Association britannique CAS (*« Computer At School »*). Une autre raison d'un tel parallélisme de deux formes ou approches semble être la volonté d'offrir aux jeunes des compétences informatiques assez larges. Les deux spécialités d'informatique CS et TI constituent toutes des objets scientifiques informatiques. Leurs différences résident au fait que la première – « CS » – désigne un enseignement spécifique d'informatique alors que la deuxième désigne un enseignement spécifique de l'utilisation des applications informatiques. Une troisième approche de l'informatique est celle des usages des TIC[79]. Liée à la culture numérique – « *Digital Literacy* » –, elle désigne un ensemble de compétences pratiques en informatique et renferme un ensemble d'outils intégrés dans les autres enseignements disciplinaires existants (Tort, 2013). C'est cette troisième approche qui est généralisée dans le système scolaire général français.

79 Elle n'apparaît pas dans le tableau comparatif parce que cette approche est présente dans tous les pays de l'échantillon.

Un autre fait frappant est que l'instauration de l'enseignement de cette discipline informatique au lycée va avec sa création au collège (dans 85 % de cas) et même plus de la moitié de pays à l'école primaire (64 % de cas), l'apprentissage de l'informatique étant ainsi pris comme une affaire de tous sans distinction de niveaux scolaires. Le rapport de la SIF le confirme en soulignant le retard que connaît la France dans l'enseignement de l'informatique qui hésite encore à l'instauration de cette discipline à l'école primaire et même au collège (Alayrangues et al., 2013, p. 5) : *« ailleurs, la question semble être « à quel âge faut-il enseigner à programmer ? 12 ou 14 ans ? ».*

Pour d'autres, la question n'est plus non plus l'âge à laquelle l'informatique devrait être enseignée mais celle du matériel adapté à un âge donné pour enseigner l'informatique. C'est le cas du groupe européen « Informatics Europe & ACM Europe »[80], militant pour l'enseignement de l'informatique. Selon lui, il n'y a plus de controverses actuellement que la science informatique soit enseignée depuis l'école primaire et évoluer à tous les niveaux mais, la question qui se pose reste celle du matériel à utiliser (« what material ? ») (Gander et al., 2013).

En France, l'informatique a été récemment réintroduite en Terminale scientifique général, sous forme optionnelle : elle n'existe pas encore ni dans les autres classes de lycées, encore moins au collège et à l'école primaire. Ce retard de la France par rapport aux autres pays est partagé. Le Rapport de l'académie des sciences (2013) montre que beaucoup de pays européens ont privilégié jusque récemment dans le primaire et le secondaire, l'enseignement de l'informatique comme un ensemble d'outils pour acquérir certaines compétences mais aussi les utilisations de ces outils dans l'enseignement des autres disciplines. Cette institution recommande dans son rapport, une généralisation de l'enseignement de l'informatique en France dès l'école primaire : des contenus de programme ont été proposés.

Si la prise de conscience de l'enseignement de l'informatique se généralise dans la scolarité obligatoire dans beaucoup de pays, son introduction semble varier selon les niveaux et les âges des élèves. À l'école primaire, l'informatique enseignée est plus orientée vers le jeu ou l'utilisation de certaines technologies spécifiques telles que des robots. Au collège, la discipline informatique introduite est plus orientée vers la résolution des problèmes, focalisée sur des raisonnements algorithmiques (Tort, 2013). Selon le rapport de l'académie des sciences en France, cette approche se situe entre la réalisation pratique et l'apprentissage théorique

80 http://europe.acm.org/, consulté le 18 novembre 2014

des concepts informatiques (Rapport de l'académie des sciences, 2013). Il propose des orientations des curricula informatiques scolaires adaptés à chacun niveau de scolarité avec des finalités différentes véhiculées dans le mode d'apprentissage.

Une spécificité française est à noter. Si certains pays mettant en place une spécialité informatique aujourd'hui adoptent les deux spécialités disciplinaires (CS et IT) séparément, le choix français semble avoir été celui de combiner les deux spécialités en une : Informatique et sciences du numérique (ISN).

b. Une formation initiale des enseignants d'informatique non encore généralisée ?

Terminons cette analyse par le fait que la ou les disciplines(s) informatique(s) (et) sont en généralisation dans la scolarité obligatoire, il y a toujours un manque de professeurs pour cet enseignement. Pourtant, la formation initiale des enseignants bien qu'elle soit organisée n'a pas suivi le rythme d'institutionnalisation de cette discipline. Un peu plus de la moitié des pays de l'échantillon organisent cette formation des enseignants (8 pays sur 14). Ces résultats ne semblent pas satisfaisants étant donné que la pérennité d'une discipline est fonction de l'existence d'un corps enseignant suffisant et le consensus concernant ses finalités (Baron, 2011).

Pour Charles Duchâteau et François Sass, un des pièges dans lesquels les institutions peuvent tomber est de confier l'enseignement de l'informatique aux non-spécialistes de cette discipline (Duchâteau & Sass, 1993) :

> *« les enseignants chargés du cours doivent avoir reçu une formation spécifique. Il me semble que fort dangereux de confier cette activité nouvelle au dernier arrivé, ou à l'enseignant qui vient d'être mis partiellement en disponibilité suite à une diminution de ses prestations habituelles. Il faut éviter aussi de ne retenir comme critère que la familiarité du professeur avec l'un ou l'autre logiciel ».*

La minimisation ou la mise en seconde zone de la question de formation des enseignants de l'informatique, semble prendre une proportion inquiétante dans un contexte de création d'une nouvelle discipline que l'informatique. Cette situation contraste avec celle qui prévalait au début de la naissance de l'enseignement de cette discipline dans les années soixante-dix. Comme l'indique Jacques Hebenstreit, la formation des enseignants, à cette époque, était une question prioritaire avant toute autre chose même les équipements (Hebenstreit, 1973).

La volonté internationalement affichée dans beaucoup de pays, partout dans le monde, sur l'intérêt de l'enseignement de la discipline l'informatique ne va pas de pair avec la formation des enseignants. Quelles sont les raisons de l'absence de formations qui tend à se généraliser dans beaucoup de pays ?

En conclusion, l'engouement pour l'instauration de la discipline informatique est évident dans beaucoup de systèmes éducatifs. Pour certains pays, cette discipline a existé avant d'être supprimée. Dans les pays où la discipline informatique a été déjà mise en place, elle semble vivre la même difficulté : le manque d'enseignants spécialisés en informatique pour son enseignement. Si les solutions apportées varient, allant d'une formation continue des enseignants déjà en fonction spécialisés dans d'autres domaines dont les mathématiques à une formation initiale des futurs enseignants, cette dernière approche n'est pas encore généralisée. Si cette situation semble trouver son explication dans le coût de la formation initiale, les expériences du passé devraient servir de leçon pour celles actuelles.

6. Conclusion

Récemment l'informatique, discipline jusque récemment exclusivement universitaire en France, a été introduite au lycée scientifique. Cette volonté institutionnelle initialement explicitée par une introduction d'éléments de savoirs algorithmiques dans le nouveau programme de mathématiques de seconde, a déplacé l'attention des intéressés par cette situation de l'enseignement de l'informatique. Ces enseignements, l'un à composante informatique en mathématiques et l'autre totalement informatique, étant confiés à des enseignants non spécialisés en enseignement de l'informatique, ont suscité des inquiétudes quant à leur mise en œuvre. Cette situation pousse, par conséquent, à s'interroger sur les compétences de ces enseignants face à ces enseignements et, par conséquent à la pérennité de cette discipline dont les attentes, sont nombreuses.

C'est sur tous ces quelques problèmes et constats souvent paradoxaux, qu'est bâtie notre problématique. Nous clôturons notre problématisation sur cet optimisme mêlé de questionnement de Michelle Artigue (2010, p. 18) au cours de son exposé, lors de la journée sur « Les enjeux du numérique » :

> *« Dans le système français, après une phase initiale qui a conjugué les deux perspectives (outil et objet), la fin de l'option informatique a vu la domination quasi-exclusive de la vision outil avec les limitations induites, bien réveillées par exemple par les travaux rele-*

vant de l'approche instrumentale (). Aujourd'hui, avec l'introduction de l'algorithmique au lycée, avec l'introduction d'une nouvelle option en terminale S, de nouveaux équilibres semblent se dessiner. Qu'en sera –-t-il dans la réalité des classes ? »

Comme précédemment vu, cette préoccupation de Michelle Artigue est partagée par beaucoup. Elle est même à l'origine de la motivation de notre recherche. Si la situation est ainsi au lycée scientifique, nous nous sommes aussi intéressé à ce qui se passe en licence informatique, étant donné que les jeunes étudiants, proviennent des lycées et n'ont pas aussi connu aucune initiation à l'informatique.

Chapitre IV RECHERCHES EN DIDACTIQUE DE L'INFORMATIQUE : REVUE DE LITTÉRATURE

Dans ce chapitre, nous allons entre en profondeurs pour aborder les concepts informatiques. Dans un premier temps, nous intéressons à une revue de littérature relativement sur le sujet. Les recherches seront classifiées selon des thématiques abordées. Par après, la littérature de recherches sera interrogée pour mettre en lumière les savoirs informatiques fondamentaux pour une initiation informatique chez les débutants. Parmi ces savoirs, des choix seront effectués pour approfondir deux à trois savoirs qui seront jugés comme les plus incontournables que les autres.

1. Recherches et classification thématique

1.1. Introduction

Les premières recherches proprement dites en informatique éducative remontent vers la fin des années 1980 dans la scolarité secondaire, avec une avancée notable des recherches en didactique de l'informatique qui se situe entre les années 1988 et 1996 (Baron, Bruillard, & Pochon, 2009). Elles avaient pour but la mise en évidence de l'existence et la pérennité d'une volonté politique de l'introduction et la scolarisation de l'informatique dans le système éducatif (Baron, 1990). Relativement à l'évolution des recherches, des différences s'observent aussi entre les niveaux d'enseignement supérieur et scolaire.

Au début de la science informatique supérieure, son enseignement portait sur la programmation. Celle-ci, comme une activité de conception des procédures à confier à un exécutant ordinateur pour sa mise en œuvre, était perçue comme un art, et les premiers enseignements de l'informatique consistaient à l'acquisition d'un langage (Rogalski et al., 1988 ; Rogalski, 1988). Comme le soulignent ces auteurs, les « vraies » réflexions sur des pratiques de programmation ont vu le jour avec les années 1960 et, motivées par la volonté d'une structuration du travail. Les études étaient orientées vers l'ergonomie, en vue d'adapter des variations individuelles des programmes produits.

Un peu après, est apparue la nécessité de recherches orientées vers la standardisation des programmes pour l'efficacité et la facilité du travail des concepteurs des programmes informatiques et la naissance d'un consensus sur la mise à jour de la programmation structurée (Dahl

et al., 1972). Ces études ont permis l'évolution de l'informatique en général et, de la programmation en particulier (Rogalski, 1988). Leurs apports ont été, entre autres, la hiérarchisation et la modularisation des programmes, l'élaboration des différentes méthodes de programmation et la naissance des réflexions sur d'éventuelles scolarisations des programmes et les méthodes de programmation qui étaient initialement conçues pour le secteur professionnel.

Malgré cette avancée de la recherche en enseignement de la science informatique dans le supérieur, la recherche en didactique de l'informatique en tant discipline d'enseignement secondaire a connu un retard en France. Une raison donnée pour justifier ce retard semble être l'absence de l'enseignement de cette science comme discipline à l'école secondaire.

Maintenant, nous développons les recherches en sept thématiques : langages de programmation en général et langage LOGO et langages à blocs en particulier, celles portant sur la robotique, les méthodes de programmation, la psychologie de programmation et enfin, les concepts informatiques spécifiques.

1.2. Apprentissage de l'informatique centré sur un langage de programmation

La programmation, partie constituante de l'informatique, a ses caractéristiques propres. Une des particularités de la science informatique n'est pas de *faire* mais de *faire faire* quelque chose à une machine, un exécutant à caractère univers (Duchâteau, 1992) : c'est la programmation. Une machine est universelle et son universalité est liée au fait qu'elle n'est pas conçue pour la réalisation d'une catégorie de tâches spécifiques préalablement définies. Le programme construit, est lui seul, qui, à la place de l'utilisateur, donne les ordres à la machine pour faire telle ou telle autre action. Ce programme est défini comme (Baron & Bruillard, 2001) : « un texte, écrit dans un langage plus ou moins général, plus ou moins sophistiqué, plus ou moins proche de la structure interne de la machine ou des problèmes que l'on cherche à résoudre ».

Éric Batard fait partie de beaucoup qui, actuellement, trouvent qu'enseigner la programmation n'est pas synonyme d'enseigner un langage de programmation. Selon lui, l'enseignement de la programmation peut avoir deux visées (Batard, 1996) : une visée utilitaire et l'autre abstraite. La réduction de l'enseignement de la programmation à celui des langages de programmation serait liée à la seule poursuite d'un but radicalement *utilitaire.* Cette visée se distingue de celle *abstraite,* où l'enseignement de la programmation se situe au-delà de l'en-

seignement des langages de programmation. Dans cette optique, l'enseignement des langages de programmation est un moyen et non un but à atteindre, ce qui, selon Batard, pose le problème de la conception d'un langage de description qui serait au-delà des autres langages de programmation.

Après la naissance de la science informatique, les premiers travaux ont été orientés vers la conception des langages de programmation. En général, comme le précisent Dubois (1975)

et Pair (1989), les compétences d'informaticien et d'un linguiste sont nécessaires dans la conception des langages. Les premiers langages de programmation ont été le fruit de leur collaboration. Pour Claude Pair, cette nécessité est justifiée par deux aspects : un *aspect matériel ou syntaxique* et un aspect *sémantique*. L'aspect *matériel ou syntaxique* concerne la construction des programmes en tant qu'une suite d'instructions bâtie les unes après les autres. L'aspect *sémantique*, lui, justifie les différentes « transformations successives » à faire dans un ordre donné des instructions jusqu'à l'obtention du résultat attendu.

Les premiers langages de programmation conçus sont classés en quatre types de catégories (Denning, 1997) : des langages procéduraux (Cobol, Fortran, Algol, Pascal, Ada, C), des langages orientés objets (Smalltalk, C++, Eiffel, Java), des langages fonctionnels (Lisp, ML, Haskell), langages logiques (Prolog)…

Éric Batard fait une classification des langages de programmation en deux catégories : (1) les langages de programmation universels et, (2) les langages génériques de description. La première catégorie comprend les langages PL/1[81] et Ada. PL/1 est un langage-synthèse des deux langages Fortran et Cobol, qu'il a d'ailleurs dépassés grâce à ses potentialités pour la programmation scientifique. Néanmoins, il n'est pas arrivé, comme pour le langage Ada, à remplacer ses concurrents. Si Ada a remplacé ses concurrents, il n'a pas non plus connu beaucoup de succès : il n'est pas arrivé à synthétiser les éléments de base des langages ayant connu des succès dont Pascal et les conceptions modernes et avancées (types de données abstraits, exécutions concurrentes, etc.). Sa complexité et la lourdeur de sa syntaxe ont limité cette synthèse. Un langage générique de description est un langage naturel. Pour Batard, l'impossibilité pour un langage de programmation d'être universel ne remet pas en question l'existence d'un langage de description, sinon il ne serait pas un langage de programmation.

81 Programming langage number One

La première approche dans l'enseignement de la programmation est donc le choix d'une notation algorithmique permettant la simplification de certains détails jugés lourds gérés par des langages de programmation. Dans un premier temps, l'enseignement de la programmation était vu comme un enseignement d'un ou des langages de programmation, en toute généralité et en se situant au-dessus de toute spécificité langagière de programmation.

Une autre classification des langages de programmation est faite par Nguyen (2005). Elle concerne des langages de programmation modernes dits de haut niveau[82] : ce sont des langages dits *impératifs*. La programmation dans un style de programmation impérative est fondée sur deux sortes de structures (Nguyen & Bessot, 2003) : les structures de contrôle (appel de fonction, le branchement, itération) et les structures de données. Avec ce contrôle, des instructions qui s'exécutent séquentiellement sur des données sont assemblées par transformation de l'état de la mémoire.

D'autres recherches sur les langages ont évolué vers l'étude des effets d'un changement d'environnement de programmation chez les spécialistes du domaine et donc les informaticiens (Brangier & Bobillier-Chaumon, 1997). Les résultats de ces études révèlent que des informaticiens sont souvent confrontés aux changements technologiques qui bousculent leurs environnements de conception. Ils font face à une articulation de trois types de réalités en confrontation : des réalités « opératoires », « cognitives » et « socio-professionnelles ».

1.3. Premières recherches en didactique de l'informatique et le langage LOGO

Jusqu'à la fin des années quatre-vingt, les recherches en didactique de l'informatique étaient encore moins nombreuses. Certains travaux se sont développés autour du langage LOGO (Rouchier 1990). Successeur du LISP[83], un ancien langage de l'intelligence artificielle, LOGO est un langage de programmation orienté objet et réflexif[84], développé par Wally Feu-

82 http://fr.wikipedia.org/wiki/Langage_de_haut_niveau, Site consulté le 28 janvier 2014. En informatique, le terme « langage de haut niveau » ne signifie pas que ce langage est supérieur à celui de bas niveau. Cette notion de profondeur désigne l'écart qui existe entre le langage utilisé sur la machine et le travail de cette machine. Des langages « haut niveau », plus proches des langages naturels, se caractérisent par un haut niveau d'abstraction : ils font abstraction des caractéristiques techniques du matériel utilisé dans l'exécution du programme. Ils permettent d'écrire de programmes avec des mots couramment utilisés issus des langues naturelles et des symboles mathématiques familiers

83 LISt Processing, en abrégé, LISP, est un des anciens langages impératifs et fonctionnels des années soixante-dix-80. Il a été conçu comme modèle pratique en opposition au modèle théorique des machines de Turing.

84 http://fr.wikipedia.org/wiki/Logo_%28langage%29#La_p.C3.A9riode_d.E2.80.99incubation_:_.281966_.C3.A0_1980.29_:_une_tortue_.C3.A0_petits_pas, site consulté le 28 janvier 2014

zeig, Danny Bobrow et Wallace Feurzeig en 1966, en collaboration avec Seymour Papert (Bruillard, 1997 ; Guzdial, 2004). Dans le système éducatif, LOGO a été l'un des langages ayant connu plus de succès et, en conséquence, plus diffusé. Relativement à ce langage, beaucoup de travaux de recherches, essentiellement centrés sur des enfants, ont été effectués (Mendelsohn, 1985 ; 1986 ; Dupuis et al., 1988 ; Duchâteau, 1992). Pour Charles Duchâteau, l'environnement d'apprentissage LOGO était rendu possible par ses aspects graphiques ; Il exécute des ordres provenant de son utilisateur, ici l'enfant (Duchâteau, 1989) :

> *« Ceux qui prétendent que LOGO permet aux enfants de dessiner sont vraiment à côté du vrai processus. Quel intérêt y aurait-il d'ailleurs à dessiner des cercles, des carrés ou des maisons ? À travers LOGO, l'enfant ne dessine pas, il fait dessiner ! Ce qu'il a devant lui, ce n'est pas un ordinateur, ni non plus une panoplie d'outils graphiques, c'est un exécutant-tortue à qui il va pouvoir donner des ordres des déplacements, pour qui il va concevoir des marches à suivre. Il y a un monde, à nouveau, entre « dessiner un cercle » et « concevoir à l'avance, pour l'exécutant – tortue, la marche à suivre comportant toutes les indications indispensables pour lui faire tracer un cercle »*

Il avait certaines potentialités qu'à son époque, Seymourt Papert recommandait aux éducateurs à faire profiter aux enfants. Ce sont notamment ses potentialités procédurales (Papert, 1980) :

> *« le mieux que nous puissions faire est ce que nous suggère la métaphore déjà évoquée : permettre à l'enfant de « faire connaissance » avec ces questions de méthodes, comme on « fait connaissance » avec un nouvel ami. En tant qu'éducateur, c'est une chose que nous pouvons favoriser en donnant aux enfants des occasions d'exercer leur pensée procédurale de manière efficace et joyeuse. Et nous pouvons les aider en leur offrant l'accès à de nombreux concepts liés à la notion de procédure. C'est précisément ce que peut offrir le contenu conceptuel de l'environnement LOGO ».*

Baron & Bruillard (2001) justifient l'origine de son succès par « *sa capacité à piloter un mobile »*. Patrick Mendelsohn (Mendelsohn, 1986) distingue l'approche LOGO des autres approches de pratiques de programmation, et lui confère plusieurs statuts. Selon lui, LOGO joue à la fois trois rôles : il est un langage de programmation, une théorie de l'apprentissage mais aussi un dispositif matériel. Dix ans plus tard, Bruillard (1997) confirme ces propos et lui confère des qualités d'outil et d'objet d'apprentissage et de courant de l'informatique en

éducation. Il distingue trois capacités qui confèrent à LOGO certaines les potentialités éducatives importantes : création de nouvelles procédures de façon interactive ; possibilité de simulation chez l'enfant des mouvements de la tortue avec les mouvements de son propre corps, permettant d'apprendre des notions géométriques et, enfin, création par l'enfant lors de son apprentissage des procédures représentant ses propres idées.

Avec LOGO, les séances d'enseignement étaient très intéressantes pour les enfants. Cela était justifié par des connexions riches et diversifiées construites par certains enseignants. Probablement suite à son succès, ce langage a connu beaucoup de descendants conçus pour des novices en programmation (Guzdial, 2004). Ses successeurs, intervenus une dizaine d'années après, étaient caractérisés par plus de performance et de potentialités éducatives : ils étaient plus souples, plus intuitifs avec une utilisation facile. Ce langage n'a pas seulement rendu service aux enfants. Des milliers d'enseignants et enseignantes de l'école primaire en ont profité pour « *s'approprier l'informatique tout en conservant leur façon à eux d'enseigner »*. Les enseignants étaient motivés et engagés pour arriver au changement dans leurs pratiques professionnelles, malgré leurs difficultés à concevoir, avec cet environnement, des projets pédagogiques satisfaisants (Papert, 1994).

Les limites de ce langage sont justifiées par la façon dont il était conçu : certains aspects didactiques étaient favorisés (Laborde & Vergnaud, 1994) : il était très orienté vers l'approche constructiviste piagétienne au détriment d'autres aspects qui, pourtant devraient être importants et bénéfiques pour l'apprentissage. L'exemple donné est le rôle de l'enseignant qui, en situation d'apprentissage, n'a pas été considéré suffisamment et valorisé dans cet environnement. Un exemple donné par Laborde et Vergnaud, est celui de l'apprentissage de la notion d'angle en mathématiques. Cette notion, difficilement conceptualisable chez les élèves, qui devrait être acquise par la commande *tourne* par un changement d'angle au cours de son orientation, n'a pas pu prendre de sens avec la seule utilisation de ce langage.

Si par facilité, les premiers apprentissages de l'informatique se sont focalisés sur l'apprentissage du langage, cette centration a, par la suite, trouvé ses limites (Papert, 1981). Pour Seymour Papert, deux raisons ont contribué au changement de cette orientation : une évolution rapide des environnements langagiers de programmation qui rendait caduque et rapidement obsolète les langages déjà acquis d'une part, et, un impact négatif des anciens langages pour

permettre une adaptation aux nouveaux langages venus. Il était alors plus que temps de distinguer l'apprentissage de l'informatique avec l'apprentissage d'un langage de programmation, : la communauté informatique s'est finalement mise d'accord avec le fait l'apprentissage d'un type donné de langage de programmation ne facilite forcément l'apprentissage d'un autre langage. Après le constat que l'*« apprendre l'informatique n'est pas apprendre un langage, l'apprentissage de l'informatique s'est orienté vers l'apprentissage d'une certaine méthode de traiter les problèmes () avec pour but le développement d'une « pensée algorithmique » par la formulation d'une méthode de résolution de problèmes »* (Chaguiboff, 1985, p. 31).

1.4. Du langage LOGO à SCRATCH

Avant de nous focaliser sur des recherches relatives à des méthodes de programmation, parlons des recherches sur SCRATCH. Ce langage revêt un intérêt particulier chez les débutants en programmation. Le choix de développer les recherches relatives à SCRATCH a été motivé par trois raisons. La première est qu'il est un langage conçu pour les novices en programmation comme c'est le cas de LOGO. La deuxième raison est que ce langage, à scripts, a des liens forts avec les langages de la famille LOGO dont il fait partie. Sa construction est d'ailleurs fortement inspirée du principe et des idées des environnements de programmation antérieurs dont LOGO et son successeur LogoBlocks (Maloney et al., 2008). La troisième raison est de nature institutionnelle. En effet, ce langage est prescrit dans les nouveaux programmes de lycées en France sur lesquels se fonde cette thèse de doctorat : depuis 2009[85], il est présenté comme un langage d'enseignement de l'algorithmique et de la programmation au lycée scientifique général en France.

Développé par *the Lifelong Kindergarten Group,* un groupe de chercheurs du MIT[86] (Massachussetts Institute Technology) Media Lab, SCATCH a été conçu pour l'enseignement et l'apprentissage. Généralement destiné aux élèves du primaire au secondaire, ce langage permet aux apprenants la création d'histoires interactives, la construction des dessins animés, la composition musicale, la simulation numérique, etc. Il permet aussi l'apprentissage collaboratif, l'acquisition des notions mathématiques et informatiques et particulièrement des notions de programmation (Muratet, 2010).

85 Ressources pour la classe de seconde : algorithmique, Document consulté le 13 février 2014 à l'adresse : http://media.education.gouv.fr/file/Programmes/17/8/Doc_ress_algo_v25_109178.pdf

86 Manchester Institute Technology

Il peut aussi contribuer à initier à la programmation pour les plus âgés notamment des étudiants de licence et même ceux de master n'envisageant pas une spécialisation en informatique (Baron & Voulgre, 2013). N'ayant jamais eu de formation préalable en informatique et en particulier du cours de programmation, les étudiants de master en science de l'éducation ont été initiés à la programmation avec SCRATCH, un langage de programmation à scripts, caractérisée par une interface distincte de celle des langages de programmation classiques. Les résultats ont révélé que ce langage constitue un bon vivier pour une initiation à la programmation. De petits programmes informatiques relativement intéressants, nécessitant des assemblages de certaines briques et faisant intervenir des notions informatiques de base telles que les structures de contrôle, les animations et les sons, ont été construits par ces étudiants. Des difficultés ont été aussi relevées. Elles sont surtout liées à l'utilisation des variables, à la gestion et manipulation des divers types de blocs. On note aussi que leurs limites ont été vite atteintes à cause de la complexité du contexte de programmation. Leur tendance était le changement d'orientation de langage pour migrer vers environnements plus ludiques et distrayants : l'animation narrative par exemple, précisent Baron et Voulgre.

Les différentes notions informatiques (affectation, boucle, test) sont manipulées chacune sous forme de bloc graphique, une manipulation facilitée par le fonctionnement de ce langage SCRATCH. Son fonctionnement est basé sur le principe de construction des blocs qui sont des métaphores sur lesquels les apprenants peuvent se baser pour construire des scripts par simples déplacements de blocs graphiques en pièces (Maloney et al., 2004). La construction d'un programme demande juste des assemblages et juxtapositions des blocs entre eux selon un ordre dicté par une tâche à lui faire faire à la machine.

Son interface comporte huit blocs rassemblés par la nature de données, caractérisée chacune par une couleur spécifique : un bloc de mouvement en bleu, un bloc d'apparence en mauve, un bloc de sons en magenta, un bloc de stylo en vert, un bloc de contrôle en orange, un bloc de capteurs en cyan, un bloc d'opérateurs en turquoise et un bloc de variables en rouge. La figure suivante en donne une illustration :

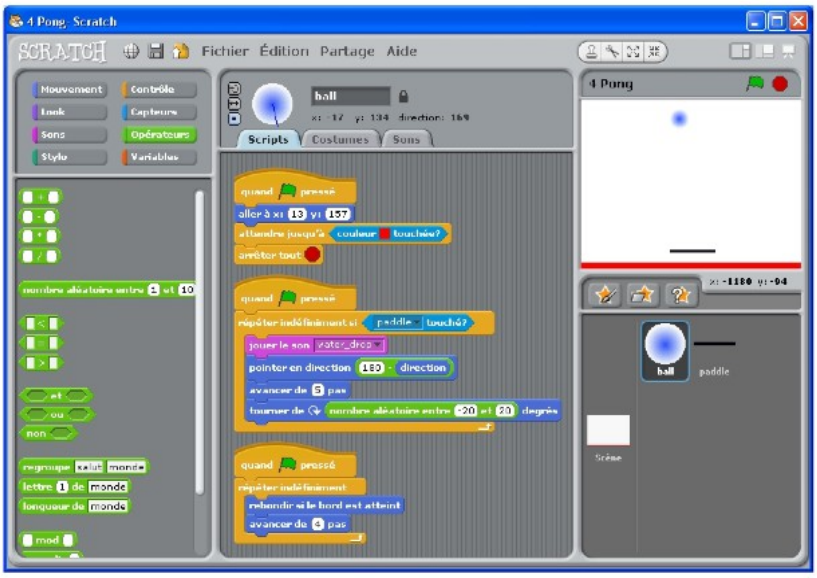


Illustration IV.1: SCRATCH et ses scripts pour la programmation (Muratet, 2010, p. 61)

Dans cette illustration, les blocs de menus à gauche permettant l'extraction des briques pour les rassembler en vue de construire le programme voulu dans la fenêtre du milieu. Ce programme s'exécute dans la fenêtre blanche en haut à droite. Trois types de blocs interviennent et caractérisent la programmation en langage SCRATCH (Muratet, 2010) : les *blocs de pile*, les *chapeaux* et les *rapporteurs*. Chacun des types de blocs a un rôle spécifique pour la conception d'un programme SCRATCH. Par parallélisme, si les blocs de pile sont rapprochés aux structures, les rapporteurs se rapprochent des fonctions dans la programmation classique.

Comme SCRATCH, l'avantage des langages à blocs, généralement conçus pour les débutants en programmation, est que la place pour les erreurs de syntaxe dans la programmation avec de tels langages est largement réduite voire absente (Tempel, 2013). Ce langage n'a pas que des potentialités, des défauts ont été aussi relevés. Templel parle d'un défaut de présentation à l'écran :

> *« There are limitations to blocs programming. Because of the graphical representation, a large program takes up a lot screen real estate and can be hard to hard follow. But as with text-bases programming languages, on can create procedures (functions) that allow body of code to be collected and called by name. »*

1.5. Recherches en programmation des robots

Un autre courant de recherches en informatique éducative est orienté dans la programmation robotique. Ces recherches sont motivées par la place de plus en plus grande des technologies robotiques en éducation. Leur côté ludique justifie souvent le succès des robots dans les contextes scolaires. Les robots sont vus comme des viviers pédagogiquement efficaces pour l'apprentissage de l'informatique par la programmation (Janiszek et al., 2011a). En France, les premières recherches sur les potentialités cognitives des environnements robotiques pédagogiques (RP) sont connues depuis les années quatre-vingt. Les auteurs majeurs de la RP sont Martial Vivet et Pascal Leroux. Au début des années 1990, Charles Duchâteau souligne déjà leur importance dans les apprentissages (Duchâteau, 1993) :

> *« elles constituent une des manières exemplaires de changer au sein de l'école les relations au savoir et ses modes d'appropriations : elles sont un puissant levier pour modifier les pratiques pédagogiques et faire passer l'enseignant d'un rôle de haut-parleur à celui de gestionnaire d'environnement pour apprendre, de complice et de co-explorateur ».*

La métaphore recommandée par Duchâteau pour initier à la programmation des débutants est le fondement de la conception et du fonctionnement des environnements de type LOGO et de ses successeurs. Signalons que cet auteur classe LOGO parmi les robots.

La démotivation de l'apprentissage de l'informatique chez les apprenants a été l'une des raisons d'introduction de l'approche par projet robotique (Janiszek, et al., op. cit.). Cette approche est justifiée par l'activité des apprenants au cours de laquelle les conséquences de leurs décisions sont immédiatement visibles, justement et clairement sanctionnées par un feed-back direct. Vue comme stimulante pour des étudiants déjà intéressés par la discipline informatique, cette approche pédagogique entraîne un effet positif sur l'assimilation des connaissances (Arnaud, 1999). Arnaud rejoint ainsi le point de vue de Tardif (1997). Selon ce dernier, cette assimilation est rendue possible par le fait qu'« *ils [les étudiants] ne vont pas seulement appliquer une série de procédures et de contenus mémorisés mais procéder à une recherche de solution en prenant appui sur les acquis théoriques qui seront réinvestis dans cette situation-problème »*. Cette liaison théorie-pratique permet ainsi de mobiliser et construire un ensemble de connaissances issues de beaucoup de domaines : informatique, ingénierie, mathématiques, physique… La RP est donc inscrite dans une approche constructi-

viste de l'apprentissage en utilisant un ordinateur connecté au monde physique (Nonnon, 2002). D'un point de vue didactique, elle est, selon lui, un dispositif technologique supposant un processus pédagogique de résolution systématique des problèmes et, permettant de capitaliser sur le goût pour le concret. S'intéressant à l'apprentissage par la programmation des micro-robots, Vivet (2000) présente les potentialités offertes par des outils technologiques dans l'initiation à la technologie et à l'informatique. Un intérêt souligné de cette approche est notamment le « *pilotage de micro-robots sous un langage comme Logo pour aborder le problème de la formation de base de personnels de bas niveau de qualification* ». Le contexte de la RP est, selon lui, adapté comme une solution pour résoudre le problème de formation professionnelle en informatique et en technologie des personnels pas ou peu qualifiés initialement.

Leroux (2005b) a expérimenté un type particulier de logiciel : la RoboTeach, un assistant logiciel pédagogique. Conçue au milieu des années quatre-vingt-dix en vue d'une initiation technologique et informatique chez les enfants, la RoboTeach est *« un environnement informatique support des activités menées dans le cadre d'une démarche de projet en micro-robotique pédagogique »* (Leroux, 1995). Elle permet la découverte des notions technologiques, techniques, de programmation, etc., mais aussi le pilotage d'un micro-robot. Un avantage important de ce type de langage est la possibilité de limiter d'abord, chez les débutants, l'apprentissage des notions informatiques (structures algorithmiques d'itération et de répétition, etc.) par une approche algorithmique dans une forme proche du langage naturel. Avec ce langage, l'approche de programmation qui permet d'acquérir d'autres notions informatiques un peu plus avancées, telles que des structures, est abordée après acquisition d'une certaine maturité chez les apprenants. En RP, l'intérêt de la pédagogie de projet est souligné (Leroux et al., 1996 ; Leroux, 2005b) : des groupes de 2 à 3 apprenants travaillent en coopération pour la résolution des problèmes prescrits par l'enseignant, le rôle de ce dernier étant d'en assurer le guide.

Les recherches sur les kits robotiques dont fait partie le robot LEGO MINDSTORMS NXT (Nijimbere et al., 2015), révèlent leurs potentialités à une initiation à la programmation comme les langages à blocs. Leurs potentialités éducatives sont offertes par le caractère à la fois interactif et transparent de ce type de technologie (Kynigos, 2008) et, les activités de construction qui offrent l'occasion à l'apprenant de devenir auteur plutôt que consommateur de la technologie.

1.6. Enseignement/apprentissage de l'informatique par des méthodes

Après avoir initialement focalisé l'enseignement de l'informatique sur un enseignement d'un langage de programmation, le centre d'intérêt s'est déplacé à un enseignement d'une méthode de programmation. Il a été constaté qu'enseigner la programmation c'est se heurter à une difficulté cruciale et incontournable (Arsac, 1989) : *« comment fabriquer une méthode pour résoudre un problème ? ».*

Beaucoup de recherches ont été développées sur ce point. Parmi elles, sans être exhaustif, citons : Guéraud et Pyrin (1987) ; Gram (1986) ; Hoc (1977) ; Pair (1989) ; Rogalski (1988) ; Rogalski, Samurçay, & Hoc (1988)…)… Beaucoup d'entre elles proposent et/ou justifient l'intérêt de l'entrée par une méthode rigoureuse de programmation pour enseigner ou apprendre l'informatique et notamment susciter un raisonnement algorithmique. Rogalski, Samurçay et Hoc émettent cette hypothèse : *« une programmation méthodique dès les premiers apprentissages évite de prendre des habitudes de programmation empirique et désordonnée, tout en facilitant la résolution des problèmes de programmation » (Rogalski et al., 1988)*. Comme ils le précisent, elle doit être adaptée aux destinataires selon leur âge. Une méthode de programmation destinée aux élèves pour leur permettre de résoudre des problèmes liés à l'acquisition des contenus scolaires sera différente d'une méthode destinée aux professionnels pour leur *« faciliter la gestion des connaissances déjà acquises et l'abord des problèmes nouveaux »,* précisent-ils.

Au sens plus large, Rogalski et al., considèrent une méthode comme *« un guide explicite et systématique pour la recherche et la gestion de stratégies de résolution de problèmes d'une certaine classe. (Elle est) le plus souvent élaborée par des spécialistes d'un domaine d'activité pour expliciter ce qui se dégage de commun dans des pratiques efficaces de résolution et pour rationaliser ces pratiques. ». S*i « l'acquisition d'une méthode permet de mettre en œuvre des stratégies nouvelles dans la résolution des problèmes du domaine considéré », ils montrent que les chez les élèves novices en informatique, des méthodes enseignées sont considérées par les élèves comme leur contrat avec l'enseignant plutôt qu'un outil au service de la résolution des problèmes de programmation. Ils contrastent une méthode d'une technique ou procédure selon la largeur de leur domaine de validité : une méthode a un domaine de validité plus large que les deux autres, ce qui lui confère une diversité de situations à gérer et une nécessité d'un niveau d'abstraction plus élevé qu'une technique ou procédure.

En particulier, les méthodes de programmation consistent en une explicitation des étapes de résolution des problèmes par la programmation. En programmation, Rogalski et al., parlent de deux principaux types d'activités qui interviennent : (1) analyser le problème pour construire un algorithme adéquat et, (2) implémenter le programme dans un langage de programmation. Dans ces activités, les méthodes permettent de gérer deux aspects. D'une part, décomposer le traitement et, d'autre part, représenter les objets du problème. D'une étape à une autre, les méthodes de programmation sont utilisées en tant qu'*« outils d'aide à la résolution des difficultés de différents niveaux qui se présentent au cours du passage de l'énoncé initial d'un problème à sa solution informatique rédigé sous forme d'un programme avec sa documentation. » (p. 311)*

Certaines d'entre elles orientent la méthode de programmation vers une expérimentation. Pour Viviane Guéraud et Jean-Pierre Peyrin, le temps de l'enseignement de la programmation centré sur le langage chez les élèves est révolu (Guéraud & Peyrin, 1989) : un consensus dans la communauté scientifique des spécialistes de l'informatique a été établi pour la centration à la méthode de cet enseignement. Deux approches sont données pour l'enseignement de la programmation. Une première approche est orientée vers un choix entre la présentation des concepts de base pouvant être concrétisés par de simples exercices ou, un choix permettant d'aborder des concepts de base par un problème concret et ouvert. La deuxième approche se situe dans le cadre d'une programmation impérative et, est jugée favorable pour des débutants, où un choix d'activités leur fait prendre conscience de la nécessité de l'approche méthodique.

Charles Duchâteau justifie l'intérêt d'entrée par la méthode dans l'apprentissage de la programmation. Selon lui, *« ce n'est pas par hasard si la formation des informaticiens comporte une grosse partie consacrée à l'acquisition de méthodologie d'exploration du « quoi faire » pour de grandes catégories de tâches : gestion, conception de base de données… Il s'agit là de la première étape souvent la plus ardue u travail de l'informaticien sur le chemin qui conduit de la tâche à son traitement par ordinateur.* » (Duchâteau, 2002, p. 11).

Contrairement à la précédente approche qui a tendance à minimiser le raisonnement algorithmique pour ne centrer l'enseignement de l'informatique que sur un simple enseignement d'un langage de programmation, les tenants de l'approche centrée sur la méthode, avancent l'idée qu' « *un apprentissage correctement conduit de l'algorithmique développe des capaci-*

tés cognitives et métacognitives et, promeut des attitudes de créativité, de rigueur et d'organisation » (Duchâteau & Sass, 1993, p. 21).

Pour Rogalski (1988), l'efficacité d'une méthode est jugée pour deux faits : sa mise en place et son apport d'une part et, son caractère général, d'autre part. Concernant sa conception, elle évoque une nécessité de collaboration des professionnels pour confronter des compétences collectives afin que la méthode arrive à une explicitation des *« invariants dans des pratiques efficaces de résolution d'un certain type de problèmes »*. L'efficacité de la méthode dépend de sa généralité qui est liée à la plus grande largeur de la classe de situations qu'elle permet de résoudre. La généralisation de la méthode minimise la nécessité du passage par l'expression directe des procédures à suivre dans la résolution des problèmes. En rapport aux modes d'acquisitions de méthodes en classe, Janine Rogalski fait deux hypothèses. La première stipule qu'*« une présentation explicite des concepts de la méthode suivie d'une mise en œuvre par les élèves sur des problèmes particuliers – pour lesquels la méthode est fonctionnelle – leur permet de s'approprier la méthode comme outil pour guider une démarche efficace de résolution »*. Selon cette hypothèse, l'acquisition de la méthode se fait en deux étapes : une étape d'explicitation de la méthode faite par l'enseignant et une autre qui consiste en la pratique des élèves pour se l'approprier. La deuxième hypothèse prône que *« l'appropriation de la méthode est le résultat de l'élaboration par l'élève de la procédure de pour résoudre des problèmes représentatifs d'une classe de situations ; l'élève lui-même ou l'enseignant, dégage ensuite la démarche commune qui peut guider la recherche de ces procédures. ».* Visant l'appropriation de la méthode, cette hypothèse s'intéresse à l'élaboration de la procédure de résolution de problèmes par les élèves en vue de la généralisation de la démarche utilisée : elle devient valable pour une classe de problèmes.

Fournier & Wirz (1993) utilisent un logiciel ALLOGÈNE pour enseigner l'algorithmique aux débutants. Selon eux, l'*« enseignement de l'algorithmique se propose de fournir les outils et les méthodes permettant de trouver une solution applicable dans le monde spécifique »*. Pour résoudre un problème donné en langage naturel, cette méthode leur permet la modélisation de sa solution décrite d'abord en langage de description, puis sa modélisation en langage machine pour validation de l'algorithme ainsi construit.

Rogalski et al., distinguent trois types de méthodes informatiques fondamentales sur lesquelles s'appuie l'enseignement et la formation de la programmation par méthode : la programmation structurée, la programmation descendante et la programmation modulaire.

Claude Mardirossian (1989) souligne une similarité entre les exigences de la mise en œuvre d'une méthode dans la résolution de problèmes informatique et celui de la dissertation sur un sujet donné. Selon lui, *« il n'y a pas une différence fondamentale entre le travail intellectuel qui consiste à analyser un problème informatique et celui que l'on fournit pour étudier un sujet de dissertation ; dans les deux cas, il faut lire avec soin, reposer clairement le problème, en avoir une vue d'ensemble qui permette aussi d'en faire apparaître les limites (il est très important de définir ce qui fait partie du problème que ce qui en être exclu), enfin le résoudre en parties de plus en plus petites (paragraphes ou procédures). »… « L'effort qui consiste à faire faire par la machine ce que l'on réalise parfois intuitivement, c'est aussi comme quand on lit un texte, entrer dans un autre mode de fonctionnement intellectuel et, en programmant, mettre au clair des modèles de raisonnement. Tout ce qui entretient une réflexion sur le mode de pensée ne peut être que formateur. On peut même penser que c'est l'un des chemins vers une pensée plus abstraite. »*

Rogalski (1988), se référant à Claude Pair, définit la résolution d'un problème par la programmation comme *« le passage d'un problème sur des « objets du monde » au texte d'un programme exécutable par un dispositif informatique donné »*. Partant de cette définition, elle modélise les dimensions du processus suivi lors de la résolution par une méthode de programmation par l'illustration IV.2 suivante, où les tirets doublement fléchés indiquent des interactions symétriques entre les nœuds :

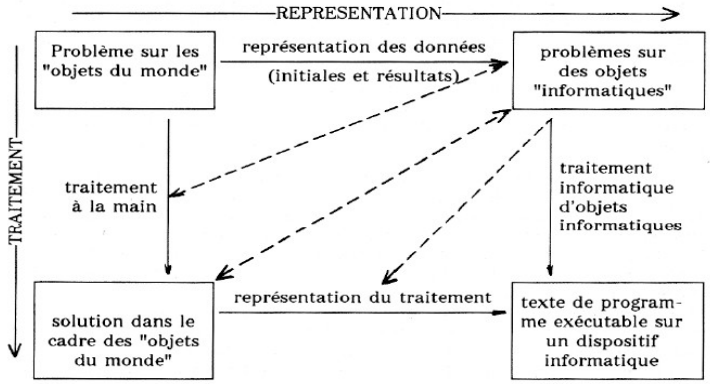


Illustration IV.2: Dimensions d'un processus de résolution d'un problème de programmation (Rogalski, 1988, p. 66)

Rogalski *et al*., (1988) donnent une perspective orientant des recherches futures à analyser selon une orientation psychologique des méthodes de programmation mises sur pied, notam-

ment en privilégiant la recherche d'invariants par rapport aux classes de problèmes y relatives. Si l'intérêt d'une orientation psychologique a été très tôt vu comme une thématique importante (c'est l'objet la section suivante), la centration des recherches sur des invariants dans les méthodes de programmation mises sur pied semblait être une nouveauté dans les recherches à cette époque (Dupuis et al., 1988). Enfin, Guibert et al., (2005) font l'hypothèse que chez les débutants en programmation, *« les difficultés sont () renforcées par les moyens d'interactions avec l'exécutant-ordinateur employés classiquement (ils sont non interactifs, abstraits) et requièrent l'apprentissage d'une syntaxe difficile. Ils nécessitent la construction, l'animation d'une représentation mentale complexe de l'état de la machine. »*. Ils suggèrent finalement une méthode de « programmation sur exemple », non focalisée sur le langage et qui permettrait un apprentissage séparé des concepts de la programmation.

Dans leur étude sur la planification des actions de programmation chez les lycéens, Rouchier & Samurcay (1985) font deux hypothèses. La première est que pendant « *la phase d'initiation à la programmation, le sujet ne se met pas spontanément dans une situation de production de procédure mais plutôt dans une situation de production de résultat »*. La deuxième stipule que même en situation de production de procédure de résolution, *« les premières procédures qu'il met en œuvre sont celles qui se rapprochent le plus à des « schèmes familiers » de la programmation d'actions ».*

Dans l'enseignement de la méthode de programmation, en vue de cette acquisition de la représentation mentale du système informatique, des pratiques stratégiques utilisant des métaphores se sont observées depuis les années soixante-dix : « épluchage de pommes de terre » pour évoquer des notions de base de l'algorithmique (Dijkstra, 1971 cité par Guéraud & Peyrin (1987) ; « casier », « tiroir » pour parler de « variable » mais aussi « remplissage » pour parler d' « affectation », « étagère » pour parler de « tableau » (Duchâteau, 1993) ; « bureau » et de « boîte » pour mieux faire comprendre respectivement le « fonctionnement interne de l'ordinateur » et le concept de « variable » (Guibert et al., 2005). Charles Duchâteau qualifie ces métaphores de stratégies pédagogiques (Duchâteau, 2002) *« C'est toujours avec mauvaise conscience que nous nous éloignons des choix de vocabulaire et de notations consacrés par usage. Pourtant, il est peut-être préférable d'attendre que des images mentales pertinentes soient installées avant d'introduire les termes habituels ».*

Ces stratégies sont vraisemblablement motivées par la volonté d'adopter un langage déjà familier aux novices en programmation pour ne pas être perdus mais, bien se représenter

l'image de ce qui est évoqué. Dans leurs pratiques, Guéraud & Peyrin (ibid.) utilisent des métaphores, qu'ils appellent « changement de domaine », souvent mêlées à des activités sous forme de jeu pour ajouter un aspect ludique et motiver les élèves afin qu'ils ne soient pas déroutés par ce contexte nouveau (programmation) qui ne leur est pas familier. L'intérêt de cette métaphore, est justifié par des apprentissages qu'elle rend possibles chez les débutants.

1.7. Recherches sur la psychologique de la programmation

Nous développons cette thématique en s'inspirant de la présentation de Janine Rogalski au colloque DidaPRO5-Dida&STIC 2013[87], publiée dans l'ouvrage qui a été dirigé par Baron, Bruillard & Drot-Delange (2015). Rogalski (2015) donne quatre générations qui ont caractérisé le développement de l'*« histoire de la psychologie de la programmation et de la didactique de l'informatique »*. La première génération commence vers la fin de la décennie 60 avec la première publication en psychologie de programmation (Rouanet et Gateau (1967), cités par Rogalski). La deuxième génération, composée par ceux que Rogalski appelle les « théoriciens de l'informatique », commence dans la décennie 70, avec des auteurs comme Dahl, Dijkstra et Hoare qui se sont intéressés à une « approche normative » fondée sur les concepts puissants de la programmation et, ont ouvert beaucoup de recherches d'expérimentation d'informaticiens. La troisième génération est intervenue dans la décennie 80. Elle a été caractérisée par une entrée importante d'autres chercheurs dans le champ de la psychologie de la programmation. Se référant à un ouvrage « Psychology of programming » de Hoc et al., (1990), elle donne, au niveau international, trois chercheurs qui ont joué un rôle important : Elliot Soloway aux États Unis dont les travaux étaient centrés sur les programmeurs novices, Thomas Green à Cambridge en Angleterre qui a contribué dans la constitution de l' « European Association of Cognition Ergonomics (EACE) et, enfin, Jean Michel Hoc, en France, qui a œuvré depuis tôt dans les recherches en programmation en France et au niveau européen, en collaboration avec Thomas Green notamment dans le cadre de l'EACE. La quatrième *génération* se situe au XXI^{ème} siècle avec des recherches en programmation orientées vers l'enseignement d'une science informatique universitaire au sens du « computer science ». Un intérêt particulier de cette dernière génération est accordé aux recherches en programmation plus orientées sur des concepts informatiques.

87 http://acte.univ-bpclermont.fr/article329.html, site consulté le 18 février 2015

En France, les premières recherches en psychologie de programmation des années quatre-vingt ont été développées par les organismes CNRS et l'INRIA. Comparativement aux autres enseignants-chercheurs de cette époque, les chercheurs de ces organismes avaient plus de facilités, précise Rogalski. L'exemple donné de leur production est le numéro spécial, produit par Hoc et Mendelsohn – *« Les langages informatiques dans l'enseignement »* – et publié dans la revue Française de Psychologie. Si ce numéro a abordé beaucoup de problématiques de l'heure relatives aux apprentissages, leurs cadres de référence ont, jusqu'aujourd'hui, gardé leur saveur théorique, ajoute Rogalski.

D'autres exemples de recherches existent dès le début des années soixante-dix (Hoc, 1972). Certaines d'entre elles se sont intéressées à l'analyse psychologique des activités des apprenants débutants en programmation depuis l'école primaire (Hoc (1977b, 1983) ; Mendelsohn (1985 , 186) ; Rouchier, & Samurcay (1985).

Se situant dans le cadre de la psychologie du travail, Hoc (1981) s'intéresse aux activités de conception qui consistent en la planification d'ensembles du programme avant l'introduction des instructions de façon détaillée. Son étude fait suite à Hoc (1979) qui consiste en une analyse séparée d'une élaboration d'une procédure et de son expression dans la résolution d'un problème de programmation chez des sujets débutants. Pour Hoc, trois conditions essentielles doivent être réunies pour mettre en œuvre une stratégie de planification bien hiérarchisée : la bonne connaissance du fonctionnement du dispositif d'exécution, la capacité d'élaboration des représentations statiques de structures de procédures et la décomposition facile de ces représentations en niveaux hiérarchiques.

Les études concernant cette thématique semblent groupées en deux catégories, relativement au niveau du programmeur : des *novices* et des *experts* en programmation. Relativement aux novices, Mendelsohn (1986) fait une analyse des caractéristiques psychologiques de l'enfant en situation de programmation. Selon lui, l'activité de programmation est une activité qui motive et, est en rapport avec ce que l'enfant apprend réellement. Deux types de résultats émergent. D'une part, chez l'enfant, l'activité de programmation offre un travail d'analyse et d'organisation des objets sur lesquels porte cette activité. D'autre part, sans toute fois s'intéresser aux contenus, la programmation est vue comme une activité structurante chez l'apprenant. Relativement aux experts, Brangier & Bobillier-Chaumon (1996) s'intéressent à leur psycho-ergonomie en programmation. Selon eux, un informaticien n'est pas seulement orienté vers la résolution des problèmes : il joue plusieurs rôles et, est, à la fois « auteur » et « ac-

teur » dans son activité de programmation. Il planifie et spécifie des besoins. Son activité n'est pas limitée aux seules données du problème à résoudre par la programmation, mais combine, au contraire, un ensemble de paramètres sociaux, ergonomiques et organisationnels du milieu dans lequel il œuvre.

1.8. Des recherches centrées des concepts informatiques spécifiques

Contrairement aux autres thématiques, les recherches qui se sont intéressées aux concepts informatiques ont été tardives et rares. Si cette situation était visible dans les années quatre-vingt (Samurçay, 1985), la rareté de ce genre de travaux s'observe encore aujourd'hui en didactique de l'informatique. Il est intéressant de constater que, depuis longtemps, des concepts informatiques sont plus étudiés en didactique et par des spécialistes des mathématiques. Comme le précise Rogalski, *« la didactique de l'informatique en France s'est développée autour de chercheurs déjà impliqués en didactiques des mathématiques »*. On trouve, depuis même les années quatre-vingt, beaucoup de recherches portant sur les concepts informatiques, en didactique de l'informatique mais par des spécialistes des mathématiques. Citons sans être exhaustifs Samurçay (1985.) ; Rouchier & Samurçay (1985) et Dupuis & Guin (1988, 1989) sur les notions de variable et de boucle ; Laborde et al., (1985), Mejias (1985) et Dagdilelis et al., (1990) sur le concept d'itération ; Rouchier (1990) sur le concept de récursivité associé au langage LOGO ; Lagrange (1992) sur l'appropriation des données codifiées chez les débutants en programmation.

Leurs orientations étaient plus focalisées sur des repérages des difficultés associées à l'acquisition de savoirs informatiques fondamentaux plus chez les élèves, notamment par la conception, la réalisation et l'analyse des expériences en situation d'apprentissage.

Les recherches effectuées par des spécialistes de l'informatique étaient rares. Nous retrouvons par exemple celle de Pair (1989) portant sur la programmation et faisant intervenir les notions de variable et de boucle et celle d'Arsac (1992) sur l'itération et la récursivité.

Depuis la dernière décennie du XX^{e}, les recherches portant sur les notions informatiques et conduites par les spécialistes des mathématiques ont sensiblement diminué. Cette situation est justifiée par le fait que *« l'intérêt des chercheurs en didactique des mathématiques s'est centré sur l'utilisation de systèmes informatiques pour l'enseignement des concepts mathématiques (tableurs pour l'algèbre, les logiciels de géométrie dynamique…) »* (Rogalski, 2015).

D'autres travaux portant sur les concepts informatiques ont été menés par des spécialistes des mathématiques se situent au XXI[e] siècle. Nguyen & Bessot (2003) étudient la prise en compte des concepts informatiques de variable et de boucle, dans différents programmes de mathématiques en France. Cette étude a été suivie par celle de Nguyen (2005), portant sur les mêmes concepts, mais dans une approche comparative entre la France et le Vietnam. Modeste (2012a) fait une étude épistémologique du concept d'algorithme avant de présenter une expérimentation de comment cette notion peut-être questionnée en situation de classe. Enfin, pour ne parler que de celles-là, dans un contexte d'une absence d'un cadre théorique focalisé sur la notion de savoir dans le cadre de l'informatique, Viallet & Venturini (2010) s'intéressent à l'épistémologie du concept de boucle. Janine Rogalski (2013) justifie cette augmentation de recherches par la nécessité de généralisation de la science informatique universitaire orientée « computer science » :

1.9. Conclusion et nouvelles orientations de la recherche

La centration sur un apprentissage d'un langage de programmation dans l'enseignement de l'informatique n'a pas, au regard de la littérature porté des fruits. L'approche centrée sur l'enseignement de la méthode a été depuis une certaine période mise au centre de la formation, en tant qu'outil et support du raisonnement et de la réflexion. Le peu de travaux portant sur des savoirs informatiques dans les recherches en didactiques de l'informatique semble révéler un manque d'importance qui a été accordée aux concepts informatiques dans leur spécificité. L'absence d'une discipline scolaire informatique aurait aussi été une raison.

Dans la suite de notre recherche, nous nous proposons de focaliser notre attention sur l'appropriation de concepts informatiques de base chez les apprenants débutants. Les choix de variables et de boucles comme concepts informatiques fondamentaux sont motivés par les questions de recherches auxquelles nous cherchons à apporter des réponses.

Une variable est l'objet informatique élémentaire qui intervient dans la construction de tout algorithme (Modeste, 2012c) : elle est incontournable dans cette construction que cet algorithme soit utilisé comme outil ou objet.

Quant au concept de boucle, si tous les algorithmes ou programmes n'en contiennent pas, elle intervient dans beaucoup de raisonnement notamment des raisonnements nécessiteux par la résolution des problèmes qui demandent des répétitions des séquences devant être exécu-

tées plusieurs fois. L'automatisation de la résolution de tels problèmes demande l'intervention de telles notions de boucles pour construire des algorithmes simplifiés.

2. Concepts informatiques incontournables pour débutants

Comme l'enseignement de toute science, la question qui se pose en amont pour l'enseignement de l'informatique est l'identification des savoirs fondamentaux qu'un débutant doit acquérir. Dans la suite, nous explicitons les concepts jugés fondamentaux par la littérature de recherche disponible. C'est cet ensemble qui donnera lieu aux concepts dits incontournables pour l'initiation à l'informatique pour des débutants.

Nous définissons d'abord ce qu'un *concept,* une notion qui va être souvent évoquée dans la suite. Ensuite, nous précisons notre méthodologie de constitution de corpus pour ressortir des savoirs informatiques incontournables indispensables pour une initiation à l'informatique. À partir d'une sélection de deux à trois savoirs, nous allons creuser pour savoir davantage sur les concepts informatiques ainsi choisis par une étude conceptuelle et les difficultés posées par leurs constructions chez les novices.

2.1. Notion de concept

Yves Reuter et al., (2007) définissent un concept comme une construction qui rend compte de caractéristiques communes à une collection d'objets. La notion de concept joue un grand rôle dans la construction des connaissances. Relativement à leur sphère de construction et d'utilisation, ils relèvent trois types de concepts : des concepts scientifiques produits dans des disciplines de recherche, des concepts scolaires construits dans les milieux scolaires et des concepts quotidiens utilisés dans la vie courante. Vygotski auquel Reuter et al. font référence, affirme qu'à la différence d'autres concepts, les concepts de type quotidien manquent de structuration : leur utilisation précède la définition, une situation qui contraste avec le contexte scolaire de construction des conceptualisations, où un concept est formalisée et défini avant sa mise en situation. Les concepts scientifiques font référence à des « unités de discours caractérisées » notamment par leur mode de construction et leur définition, leur fonctionnement en réseaux et servant d'outils au travail théorique. Ces derniers sont toujours définis en liaison avec les autres. Vergnaud (1990) définit le concept scolaire en le situant dans trois ensembles : un ensemble des situations qui lui donnent du sens, un ensemble des formes langagières ou non, et enfin, un ensemble d'invariants opératoires. Ce dernier en-

semble d'invariants opératoires peut être considéré comme stable dans les conduites des élèves, et mis en œuvre d'une situation à l'autre (Reuter et al., 2007) : cette stabilité de notions n'est possible que dans les apprentissages organisés dans des cadres formels, dont des apprentissages scolaires.

2.2. Méthodologie et corpus

Cette partie s'intéresse aux savoirs informatiques fondamentaux pour les élèves des lycées débutants en informatique. Ces savoirs sont à situer dans la composante algorithmique et programmation comme partie de la science informatique requérant un consensus pour constituer une initiation à l'informatique pour les lycéens débutants.

En vue d'identifier ces savoirs, une analyse d'un échantillon de dix-huit ressources sélectionnées de la littérature de recherche a été faite. Cet échantillon comprend de présentations de colloque, d'articles scientifiques, de rapports de recherches, de thèse de doctorat, de livres scientifiques. Si les articles en langue française dominent, ils proviennent d'un niveau international. L'analyse a consisté en une extraction des concepts jugés fondamentaux par les auteurs des recherches pour une initiation en algorithmique et programmation.

2.3. Synthèse des résultats

Les résultats présentés dans le tableau 1.1 de l'annexe 1 montrent un certain nombre de savoirs informatiques, considérés comme fondamentaux par la littérature de recherche : algorithme, variable, instruction, structure de contrôle (alternative, itérative), boucle, itération, programme, branchement, fonction, information. La section 2.4. suivante présente nos choix pour la suite de l'étude

2.4. Choix effectués : variable et boucle

Les savoirs recueillis sont considérés comme fondamentaux en informatique par les auteurs. Certains d'entre-eux cachent des notions même si elles ne sont pas explicitement nommées, sont sous – entendues. En effet, il est impossible de construire un algorithme en ignorant les notions de données, d'entrée/sortie, du traitement et de la sortie. Aussi, par exemple, quand on dit qu'un algorithme est un concept incontournable pour un débutant en informatique, même si une variable n'est pas explicitement nommée, elle est directement concernée en tant qu'élément fondateur d'un algorithme. On ne peut non plus parler d'itération tout en ignorant le concept de variable. De plus, la notion d'itération est un concept qui n'a pas de sens en de-

hors d'une boucle. Les deux sont donc intimement liés : une itération est incluse dans une boucle. Aussi, dans notre cas, peut-on parler du concept d'affectation en dehors d'une instruction ? Et pourtant, cette notion d'affection n'est pas autant évoquée parmi les concepts fondamentaux autant que le concept d'instruction.

Les concepts sont donc inter-reliés même s'il y a qui sont des incontournables plus que d'autres. Par exemple des concepts de variable, d'affectation et d'instruction sont trois concepts reliés entre eux : ils se retrouvent ensemble au sein d'une même instruction, et enlever l'un ou l'autre de sa place, fait qu'il y a une perte de sens. Cette situation est expliquée par la notion de champ conceptuel de Vergnaud (1990). Le cas de boucle engendre davantage de relations. Cette situation d'interrelation des concepts au sein d'un algorithme a été illustrée par Samurçay (1985) :

> *« Il est nécessaire pour des besoins de l'analyse de dissocier les concepts de boucle, de variable et d'affectation, mais ce sont des concepts indissociables du point de vue de processus de conceptualisation par le sujet. Les situations problèmes que rencontre le sujet font nécessairement appel à ces concepts dans leur interaction ».*

Dans la suite de l'étude, nous nous intéressons aux notions de variables et de boucles, deux concepts unanimement reconnus comme étant fondamentaux à la base de la science informatique et particulièrement en algorithmique et programmation. Chez les élèves débutants en informatique, qu'il s'agisse d'un travail purement algorithmique ou d'une activité de nécessitant la programmation, et donc qui nécessite un langage de programmation, ces deux notions restent essentielles voire incontournables pour une initiation à l'informatique.

Le choix de variable est aussi justifié par le fait que cette notion est familière chez les élèves des lycées généraux particulièrement dans le cours de mathématiques. La signification et les propriétés du concept de variable en mathématiques diffèrent de celles de la variable informatique. La familiarité aux élèves de cette notion de variable en mathématiques contribue –-t-elle à son appropriation ou au contraire la freine en informatique ?

Contrairement à la notion de variable (mathématique), déjà familière aux élèves, la notion de boucle (informatique) est nouvelle. Les situation-problèmes qui peuvent la mobiliser en mathématiques sont celles nécessitant un raisonnement par récurrence auquel les élèves ne sont pas encore suffisamment entraînés. S'il n'est pas difficile de comprendre que telle action se

répète, la grande question peut être le choix des élèves : quelle boucle pour quelle répétition de séquences étant donné qu'une boucle à utiliser varie d'un problème à l'autre.

3. Zoom sur les concepts de variable et de boucle informatiques

Comme nous l'avons déjà vu (point 2.3. précédent), un certain nombre de concepts informatiques sont mis en avant par la littérature de recherches comme étant des concepts fondamentaux de la science informatique : algorithme, variable, affectation, instruction, boucle… Nous ne prétendons pas passer détailler tous ces concepts dans cette recherche. Des choix ont été faits. Les notions de variable, d'affectation, de boucle, d'instruction… interviennent dans la construction d'un algorithme : elles sont essentielles dans la construction d'une méthode de résolution de problèmes. Les liens forts qu'elles entretiennent entre eux, les rendent indissociables du point de vue de processus de conceptualisation : dans les situations-problèmes prescrites au sujet, elles agissent toujours en interaction si bien que rares sont les circonstances de leurs dissociations (Samurçay, 1985). Nous nous inscrivons totalement dans cette vision de Samurçay. Leur dissociation ici ne vise qu'à présenter dans la suite une analyse de quelques-unes d'entre elles : variable et boucle.

Nous faisons le choix des concepts de *variable* et de *boucle*, même s'ils ne sont pas les seuls concepts informatiques fondamentaux pour une initiation informatique en algorithmique et programmation. On ne peut pas prétendre apprendre l'informatique, en général, et l'algorithmique et la programmation en particulier, en mettant de côté ces deux notions.

Avant d'entrer dans le cœur du concept de variable et de boucle, il est important de parler de ce qu'est un algorithme.

3.1. Notion d'algorithme

L'encyclopédie Wikipédia définit en 2014 un algorithme comme *« une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème »*[88]. Ce nom d'algorithme revient au mathématicien perse Al-Khawarizmi (789 – vers 850), reconnu comme le fondateur de l'algèbre. Quels que soient les points de vue, mathématique ou informatique, deux caractéristiques communes émergent des différentes définitions que peut avoir un algorithme (Nguyen, 2005) : la *séquentialité* et l'*effectivité*. La séquentialité se réfère à l'ordre des actions dans l'exécution d'un algorithme. Selon Nguyen, la séquentialité est liée au fait que la description d'un algorithme est faite par une succession d'actions rangées dans un

88 http://fr.wikipedia.org/wiki/Algorithme, Site consulté le 1 février 2014

ordre bien définie et ordonnée. L'effectivité quant à elle fait référence au résultat. Un algorithme donné est effectif s'il donne toujours le résultat attendu.

Pour Donald Khun[89], un algorithme a cinq propriétés caractéristiques : la finitude, la précision, les entrées, les sorties et le rendement. Si la *finitude,* traduisant qu'un algorithme, dans son exécution, doit se faire et se terminer en un certain nombre toujours fini d'étapes, sa sa *précision* est une propriété stipulant que chacune de ses étapes doit être rigoureusement définie et sans ambiguïté. Quant aux *entrées,* elles sont définies comme des données choisies dans un ensemble d'objets initialement bien identifié et fournis à l'algorithme pour être exécutées ; les *sorties qui* sont toujours en relation bien spécifiée avec des données entrées, sont les résultats de cette exécution ; enfin, le rendement d'un algorithme est constitué par *« toutes les opérations que l'algorithme doit accomplir doivent être suffisamment basiques pour pouvoir être en principe réalisées dans une durée finie par un homme utilisant un papier et un crayon »*.

Un algorithme serait d' « une importance égale dans la constitution des sciences mathématiques et informatiques ». Dans les deux domaines, il reste un concept fondamental (Dowek et al., 2011 ; Nguyen, 2005). Si cette notion d'algorithme est prescrite depuis les années soixante-dix dans les programmes d'enseignement des mathématiques au lycée, elle a rarement eu le statut d'objet d'enseignement/apprentissage (Modeste, 2012c). Associer un algorithme aux notions de variables et de boucles reste normal étant donné que les deux dernières fonctionnent au sein d'un algorithme et en sont des composants essentiels.

3.2. À propos du concept de variable en informatique

La notion de variable est centrale en informatique. Elle intervient essentiellement dans la construction des algorithmes et des programmes informatiques. Nguyen & Bessot (2003) définissent une *variable* comme *un nom que l'on donne à une adresse mémoire*. Pour Samurçay (1985), la définition d'une variable associée à une adresse est lacunaire parce qu'elle ne rend pas clairement compte du caractère invariant de cette notion. L'invariance d'une variable dépend de plusieurs facteurs : elle devient évidente non seulement par une attribution d'un nom à la variable, mais aussi par un contrôle sur l'ensemble des valeurs qu'elle est susceptible de prendre, par une précision d'un type de données permettant de prévoir les opérations permises sur les variables dans un environnement langagier donné.

89 Donald Khun est professeur de l'université de Stanford aux États Unis http://fr.wikipedia.org/wiki/Algorithme, Site consulté le 1 février 2014

a. Variables et instructions élémentaires

Il est impossible d'écrire un algorithme ou un programme sans utiliser de variables et une suite d'instructions. Charles Duchâteau distingue trois sortes d'instructions élémentaires (Duchâteau, 1993) instructions d'affectation, instructions d'entrée de données et instructions d'affichage (sortie) de résultat. L'information (donnée) entrée ou sortie peut-être une constante, un nom ou une expression.

La construction d'un algorithme ou d'un programme informatique nécessite la combinaison de trois types d'instruction interviennent (Nguyen & Bessot, 2003) : instructions d'affectation, instructions conditionnelles et instructions itératives. Et, à la base de toute opération qui a lieu au sein d'un algorithme se trouve une *variable*. Par rapport à la notion d'initialisation, ils distinguent deux principales opérations portant sur une variable : initialisation par lecture et initialisation par affectation. Ces opérations permettent une attribution des valeurs initiales ou au remplacement d'une valeur préexistante dans les variables.

Au cours de la résolution d'un problème nécessitant la construction d'un algorithme, les sujets font face à un certain nombre d'activités à faire. Samurçay (1985) en distingue deux types :

- la construction des significations et des opérations sur des variables : déclaration de variable, affectation de valeurs, entrées et sorties respectivement de données et résultats ;
- le contrôle de valeurs particulières prises par des variables au cours de l'exécution du programme. Il intervient ici des structures de contrôle telles que les répétitions, les conditions et les séquentialités. Leur planification est pensée au préalable.

b. Types et statuts de variables

Il existe plusieurs types de variables : entier, réels, chaîne de caractère, tableau d'entiers, tableau de réels, etc. Les statuts fonctionnels des variables varient selon le problème à résoudre. Dagdilelis, Balacheff et Capponi montrent que, lors de la résolution de problèmes, les élèves de fin de collège font intervenir deux types de variables dans une boucle itérative (Dagdilelis et al., 1990) : un variable compteur et une variable d'accumulation. Ils définissent un variable c*ompteur* comme un générateur d'entiers, obtenus par valeur successive par rap-

port à la valeur initiale et peuvent être calculés par une instruction de récurrence de la forme[90] : « *compteur : = compteur +1 » ;*

La variable d'accumulation quant à elle permet l'actualisation des calculs par une instruction de récurrence suivante : « *somme : = somme + compteur » ;*

Par exemple, quand les élèves ont à construire un algorithme qui calcule la somme de 100 premiers nombres entiers, il y aura dans leur algorithme une variable qui compte si les 100 premiers entiers sont atteints : c'est un variable compteur. Il y aura aussi une autre qui compte leur somme. C'est la variable d'accumulation.

c. Opérations sur des variables

Il existe plusieurs types d'opérations possibles sur les variables. Ces opérations se font à travers des instructions élémentaires. Les plus couramment utilisées sont : la déclaration, l'affectation et l'initialisation.

Déclaration : Une variable doit être déclarée avant d'être utilisée. Déclarer une variable consiste à l'énoncé et donc à lui réserver une place dans l'espace mémoire.

Affectation : Elle consiste en l'attribution de la valeur à la variable déjà déclarée. Cette opération se fait à travers une instruction élémentaire appelée : *instruction d'affectation.*

L'instruction d'affectation est fondamentale : elle porte sur les variables. Elle peut se noter : E**xpr → Variable** ou **Variable ← Expr.**

Cette instruction d'affectation consiste à changer la valeur de la variable **Variable** et indique qu'elle doit prendre une nouvelle valeur, celle de l'expression **Exp** et se lit : « **Variable** prend la valeur de **Exp** » (Nguyen & Bessot, 2003, p. 9). Les *entrées* et *sorties* de données sont d'autres types d'opérations fonctionnant sur des variables.

Dans la dualité variable/fonction, Claude Pair distingue trois principales instructions d'affectation simples sur lesquelles portent une variable (Pair, 1989, p. 81) : donner une valeur à une variable par lecture[91] ; donner une variable à une variable par affectation et, écrire un résultat.

90 La syntaxe utilisée varie d'un environnement langagier donné. Par exemple avec une calculatrice, on peut avoir somme ← somme + 1 ; avec le langage C, on a somme = somme + 1 ; avec le langage Pascal, on a : somme : = somme + 1, etc.

91 L'endroit où se fait la lecture doit être indique : ça peut être dans un fichier déjà constitué ou ailleurs.

Initialisation : Une initialisation d'une variable est une opération qui consiste en l'attribution d'une valeur initiale requise à la variable donnée. En général, l'opération d'initialisation suit deux règles (Samurçay, 1985) :

- initialiser toute variable susceptible de voir sa valeur modifiée dans la boucle ;
- tenir compte dans l'initialisation des variables des types d'opérations intervenant dans leur mise à jour.

Initialiser une variable à zéro fait savoir qu'en mémoire, une variable n'est pas « vide ». L'idée d'initialisation révèle donc une volonté de partir, avant tout calcul pouvant se faire sur la variable donnée, de la mémoire correspondante à la variable qui est vide. Dans la gestion des variables, deux informations importantes doivent être coordonnées : une information sur les mises à jour des variables et une information sur la valeur de la variable qui correspond à la valeur de réalisation de la condition d'arrêt (Samurçay, 1985) : *« un accord implicitement admis sur la valeur de la variable consiste à arriver à rendre au moins une fois l'expression booléenne VRAIE au sein de la condition »*. L'évaluation de la condition est effectuée par l'instruction *test,* dont l'importance est capitale surtout au sein d'une boucle pour opérer des choix de branchements.

Cette notion d'initialisation est conceptuellement difficile chez les débutants en informatique (Nguyen & Bessot, 2003). Sa difficulté est surtout d'ordre cognitif (Samurçay, 1985).

d. Variable mathématique/informatique

Une variable mathématique se distingue d'une variable informatique. Contrairement à la variable informatique, une variable mathématique est, par définition, *« un symbole représentant un élément non spécifié ou inconnu d'un ensemble »* (Samurçay, 1985). De plus, dans la manipulation de variables, certaines instructions changent de propriétés selon qu'elles sont utilisées avec une orientation mathématique ou informatique, ajoute Samurçay. Par exemple, en mathématique, la *relation d'affectation* d'une valeur à une variable est une *égalité* régie par des propriétés d'une *relation symétrique* alors qu'en informatique, cette relation est *asymétrique*.

La propriété de symétrie/antisymétrie de la notion d'affectation a été explicitée par Claude Pair (1989). Il souligne une rupture fondamentale avec son usage en mathématiques en se situant dans le cas d'une variable informatique : « *aucune variable ne peut être utilisée, à*

droite d'une affectation ou dans une écriture, avant d'avoir reçu une valeur.et donc sans se trouver déjà dans le lexique. ».

En résumé, la notion de variable mathématique est différente de celle informatique : les différences sont liées à sa définition, sa structure et ses propriétés fonctionnelles. Il reste à voir, dans les pratiques des élèves, si ces différences sont une aide ou une entrave dans sa construction en vue de la résolution des problèmes.

3.3. À propos du concept de boucle informatique[92]

a. Notions d'itération

Si le déroulement ordinaire de l'exécution d'un programme sur ordinateur se fait instruction par instruction, il arrive souvent qu'une même séquence d'instructions, au sein du programme, soit exécutée plusieurs fois de suite. On parle d'une *itération.* Le branchement avec l'instruction « GOTO » jadis utilisée permettait de dispenser la réécriture des séquences à répéter mais connaissait aussi des limites : elle rendait difficile la lecture du programme par une personne tierce que le concepteur du programme mais aussi occasionne certaines erreurs de programmation. Pour arriver au bout de ces limites, C'est avec le courant de la programmation structurée que ces problèmes ont été résolus en mettant en place des programmes régis par des structures de contrôle construites à base d'une itération et caractérisées par une seule entrée et une seule sortie (Mejias, 1985).

La notion d'itération est, en informatique, une notion de base constituée pour permettre la construction des algorithmes qui nécessitent des répétitions des séquences d'instructions identiques au sein des boucles (Laborde et al., 1985). Ces auteurs distinguent trois parties qui interviennent dans la construction d'une itération : un avertisseur d'itération, une condition et un corps de l'itération. L'*avertisseur de l'itération* est une expression qui annonce une itération. La condition, qui comme son nom l'indique, conditionne une fois vérifiée, l'exécution de la partie de la boucle à répéter et, enfin, le *corps de l'itération qui* est délimité par la condition et ce qui sera répété une fois la condition vérifiée. Il se situe entre la condition et la fin de la boucle.

Dans la résolution des problèmes, diverses situations peuvent se présenter. Parmi elles, certaines nécessitent des actions qui se font avec répétitives ou sous une certaine condition. De

92 Dans la suite, nous utiliserons seulement le terme de boucle. Le lecteur pourra comprendre qu'il s'agit d'une boucle dans le sens informatique

telles actions nécessitent la construction des algorithmes qui font intervenir des itérations. De façon générale, une structure itérative est sous la forme suivante (Laborde et al., ibidem., se référant à Soloway et al., 1983) :

Itérer <corps d'itération1> <condition> < corps d'itération2 > finItérer (1)

Les « Corps d'itération1 » et « Corps d'itération2 » sont des ensembles d'instructions qui vont respectivement ensemble et qui sont répétées un certain nombre de fois donné sous une certaine condition. De façon générale, une structure itérative comporte deux éléments principaux (Mejias, 1985) :

- un ensemble d'instructions à répéter un certain nombre de fois : c'est le corps de l'itération ;
- une expression conditionnelle qui autorise la répétition du corps de l'itération et qui contrôle que cette répétition se fasse correctement : c'est la condition d'arrêt.

Dans certaines situations particulières, l'expression (1) peut donner naissance à trois sortes de boucles suivantes : : Tant que Faire, Pour, Répéter… Jusqu'à ce que.

b. Boucles

Dans l'expression générale de l'itération (1), il arrive qu'un corps d'instructions soit vide, dans ce cas, cette expression peut s'écrire comme suit *:*

Itérer <condition> <corps d'itération2 > FinItérer (2)

L'expression (2) devient équivalente à une boucle appelée ***Tant que Faire*** qui s'écrit comme suit :

Tant que <condition> Faire < corps de l'itération > FinTantque (3)

Quand le corps de l'itération2 est vide, l'expression (1) devient équivalente à :

Itérer <corps de l'itération > <condition > FinItérer (4)

L'expression (4) donne une boucle **Répète…** ***jusqu'à ce que****, qui* s'écrit :

Répéter < corps de l'itération > jusqu'à ce que <condition> (5)

Dans ce genre d'itération, une autre situation particulière possible intervient lorsque le nombre de répétitions du corps de l'itération est connu d'avance. Dans ce cas, la condition va porter sur un compteur dont sa valeur sera incrémentée d'une unité à chaque fois que le corps de l'itération est exécuté.

L'expression (3) devient équivalente à une boucle ***Pour*** qui s'écrit comme suit :

Pour < condition > faire <corps d'itération> FinPour (6)

Autant les boucles sont structurellement différentes, autant elles diffèrent dans leur fonctionnement. C'est l'objet du point suivant.

c. Comparaison et fonctionnement des boucles

Après ce descriptif schématique, il est important de faire un bref comparatif de fonctionnement de ces boucles. Les trois boucles ont des ressemblances et des différences. Leurs ressemblances résident dans le fait que qu'elles sont toutes annoncées par des avertisseurs que sont les expressions *Tant que, Répéter et Pour.*

Il existe deux différences fondamentales entre les boucles *Répéter…* et *Tant que.* Elles sont à la fois structurelles qu'algorithmiques (Laborde et al., 1985). La différence de structure est liée à la délimitation du corps de l'itération. Dans la boucle *Répéter,* le corps de l'itération est délimité par les mots R*épéter* et *Jusqu'à ce qu'*alors que dans *Tant que*, le corps de l'itération est délimité par les mots *Faire et Fin Tant que.*

La deuxième différence liée à leur signification algorithmique est due au fait que le corps de l'itération est au moins exécuté une fois dans la boucle Répéter… jusqu'à ce qu'alors qu'il peut ne pas être exécuté même une fois dans *Tant que* : il suffit que la condition qui autorise cette exécution ne soit pas vérifiée. Si l'effet de la boucle *Répéter* peut-être obtenu à partir de la boucle *Tant que* par changement de place de la condition, *Répéter* est considérée comme la boucle la plus générale par rapport à *Tant que, soulignent* Laborde, et al. De plus, *le mot jusqu'à ce que annonce la condition qui, une fois réalisée arrête la répétition du corps de l'itération alors que* la boucle *Répéter* est souvent présenté sous forme implicite ou accompagnée de tournures linguistiques (Duchâteau, 2002).

Dagdilelis, Baracheff et Capponi soulignent deux aspects justifiant l'importance des éléments d'une structure itérative (Dagdilelis et al., 1990) : un aspect informatique et un aspect didactique. Les différents éléments sont des « paramètres didactiques même si leur prise en compte reste transparente chez les informaticiens ». Leur aspect didactique est justifié par le fait qu'un choix d'un type donné de boucle a un impact sur les conduites de l'apprenant mais aussi sur le sens et la complexité d'un problème à faire.

Une structure de contrôle est complexe. Elle comprend beaucoup d'éléments : avertisseur, délimiteur, condition, variable, corps d'itération, compteur, etc. Ces derniers entretiennent

entre eux des relations fortes pour son fonctionnement. Si un avertisseur d'itération annonce le début d'une boucle, le corps d'itération qui est un ensemble d'instructions, est exécuté avec la véracité de la condition : le résultat du *test* de la condition donne une orientation à prendre. Laborde et al., soulignent l'importance de l'instruction test dans une boucle en précisant que c'est grâce à elle qu'elle existe : l'instruction test joue deux actions importantes. D'une part, elle évalue la condition et d'autre part, décide le branchement à faire : sortir ou pas de la boucle, arrêter ou continuer l'exécution (Laborde et al., 1985).

3.4. Difficultés liées à l'apprentissage des variables et boucles chez les novices

a. Difficultés de planification

Les difficultés liées à la planification de ce qu'il faut faire pour résoudre un problème occupent aussi une place importante. Si les phases d'analyse et de compréhension des problèmes à résoudre sont très formatrices d'un point de vue intellectuel (Lévy, 1987), elles posent aussi des difficultés importantes aux novices en informatique. Les principales difficultés se situent plus au niveau des phases de planification, d'analyse et d'explicitation des procédures à suivre dans la résolution des problèmes qu'au niveau de la programmation proprement dite (Chaguiboff, 1985 ; Rouchier & Samurcay, 1985 ; Guéraud & Peyrin, 1989 ; Duchâteau, 2002).

En contexte d'un apprentissage de l'algorithmique et de la programmation, les novices se heurtent à un certain nombre de difficultés. Leur grand problème se trouve dans la planification des actions de l'exécutant (Rouchier & Samurcay, 1985). Face à ces difficultés, Rouchier et Samurçay font deux hypothèses : *« (1) dans la phase d'initiation, à la programmation, le sujet ne se met pas spontanément dans une production de procédure mais plutôt dans une situation de production de résultat ; (2) les premières procédures qu'il met en œuvre () sont celles qui se rapprochent le plus à des « schèmes « familiers » de la programmation d'actions. »*

Si l'apprentissage de l'algorithmique et la programmation exige « une maîtrise de la complexité » pour reprendre l' expression de Jacques Arsac (1980), justifiée par la nécessité de faire face à trop de détails à la fois, les difficultés sont souvent liées à l'acquisition de certains concepts, des expressions spécifiques, de certaines procédures, qui nécessitent la mobilisation de beaucoup de notions nouvelles chez les élèves : variables, instruction, boucles, affectation, etc. Certains auteurs trouvent qu'*« apprendre à réaliser des programmes, c'est*

apprendre à « faire fonctionner » des concepts » (Hoc, cité par Chaguiboff (1985)). Or, cette activité suppose d'abord leur planification puis de leur traitement. Si cette activité n'est pas gagnée chez les élèves débutants, elle est significativement riche dans la conceptualisation en général et des concepts de variables et de boucles particulièrement (Samurçay, 1985). L'apprentissage de ces concepts se fait le plus souvent au cours de la résolution des problèmes. Au Durant cette activité, en rapport avec la gestion de la notion de variable, Claude Pair souligne trois étapes de complexité croissante (Pair, 1989) : recours à des valeurs de la variable totalement différentes, nécessité de l'ancienne valeur pour trouver la nouvelle et enfin, utilisation d'une ancienne et une nouvelle valeur dans des instructions différentes d'une même itération.

b. Difficultés conceptuelles

Dans la résolution des problèmes nécessitant la programmation, si la planification du traitement à communiquer à la machine est une activité cognitivement constructive, elle est aussi la principale difficulté des apprenants novices (Samurçay, 1985).

Les concepts de variable informatique et de boucles posent souvent des difficultés conceptuelles aux novices. Dans le cas d'une variable, la littérature de recherche justifie la difficulté des élèves débutants en informatiques par l'absence du concept de variable informatique dans leur « passé cognitif » (Rogalski, 1988 ; (Duchâteau, 1989) ; Pair, 1989). Pour Duchâteau et Pair, les difficultés posées par une variable informatique sont justifiées par l'absence de véritable précurseur dans l'expérience de l'élève. Si la variable mathématique est initialement connue chez les élèves qui débutent en informatique, la notion de variable informatique est nouvelle. Les deux types de variables ont des statuts et rôles différents. Dans le cas présent de mon travail, par rapport aux élèves de lycées, la variable mathématique constitue un précurseur à celle de variable informatique. ce qui être une probable source de difficultés chez eux. Nous nous interrogeons si le fait d'être d'abord vu en mathématique favorise son appropriation en informatique. Notre hypothèse suppose que oui.

Rogalski (1988) considère la variable informatique comme « une fonction de l'exécution », une situation qui renvoie à la « dualité variable/fonction », non spécifiquement informatique, une relation mal maîtrisée au lycée. Cette absence de maîtrise de cette dualité engendre une difficulté de compréhension de la nation de l'invariant de l'itération.

Selon leur statut, les variables posent des difficultés différentes. De façon comparative, si le variable compteur est moins difficile, la variable d'accumulateur pose plus de difficultés aux élèves débutants. Ces différences de difficultés sont justifiées par le fait que la gestion d'une variable d'accumulation *« demande une représentation fonctionnelle particulière »* (Samurçay, 1985), alors que le variable compteur est plus proche de la notion d'énumération d'entiers déjà connue chez les élèves (Dagdilelis et al., 1990). De plus, à la différence de la précédente, la variable d'accumulation est suscitée par le découpage du problème à résoudre : elle est une notion spécifiquement liée au raisonnement algorithmique.

Une autre difficulté conceptuelle en rapport avec la variable concerne les différents états des variables et les transformations qu'elles sont susceptibles de subir, dès son état initial à son état final. Les difficultés de même ordre sont perceptibles sur des traitements des expressions dans lesquelles interviennent des variables (Samurçay, ibidem.) : construction et traitement d'un corps d'itération, d'une condition d'arrêt et d'une expression booléenne.

Rogalski souligne deux obstacles conceptuels « profonds » qu'il faut absolument surmonter dans l'apprentissage de la programmation. Le premier obstacle est l'appropriation d'une itération et des méthodes d'écriture de programmes comportant des boucles. Selon elle, deux éléments, souvent difficiles chez les débutants, doivent être maîtrisés. Le premier est l'appropriation de l'exécution d'actions avec son caractère dynamique et (la) succession de situations avec son caractère statique. Le deuxième élément, en rapport avec la variable informatique, est sa conception sous forme de fonction, c'est-à-dire la comprendre dans son statut dual « variable/fonction ».

Le deuxième obstacle est lié à la conception de la méthode de résolution de problèmes. Pour Rogalski, la méthode construite par l'élève doit être orientée non pas vers la description des calculs à faire mais dans le raisonnement permettant de comprendre ce *« que réalise le programme »* plutôt que *« comment agit le programme »*. Elle recommande l'utilisation des fonctions et des procédures, vues comme des « briques » dans la construction d'un programme : elles orientent les apprenants à se focaliser sur le but du programme. Chercher à contourner des obstacles conceptuels posés par l'apprentissage de l'informatique chez les débutants est impossible, Janine Rogalski suggère que la solution est de les faire confronter aux élèves dans des situations didactiques appropriées (Rogalski, 1988).

Comme les variables, les structures posent des difficultés différentes chez les débutants. Si les structures itératives sont vues comme celles posant moins de difficultés que celles conditionnelles (Hoc, 1982, cité par (Chaguiboff, 1985), même entre ces premières des différences de difficultés existent. Parmi les structures itératives, la boucle, **Répéter…** ***jusqu'à ce que*** est vue comme celle qui est plus abordable par les élèves novices en informatique. Elle serait plus proche de leurs représentations initiales que les autres boucles (Mejias, 1985).

Renan Samurçay relève trois conceptions des élèves en rapport dans une activité de programmation. D'abord, c'est la priorité de certaines valeurs particulières des variables, certaines pratiques par rapport aux autres : une initialisation par lecture et des actions simultanées sur des instructions de lecture et d'affectation pour une même variable. Cette situation laisse supposer les compréhensions des instructions chez les élèves dans le sens des interactions d'entrée et sortie. Ensuite, les élèves se préoccupent de vider les cases mémoires de variables dans chaque programme quelle que soit l'évolution future des variables dans le programme. Enfin, pour faire face à la difficulté de la construction de la condition d'arrêt, les élèves se focalisent sur la boucle *Répéter… jusqu'à ce que* qui leur semble compréhensible et l'utilisation des répétitions non efficace pour l'obtention d'un résultat.

Une autre justification de se focalise à cette boucle chez les débutants est donnée par Claude Pair (Pair, 1989) : c'est celle de familiarité. Selon lui, si *Répéter… jusqu'à ce que* est très proche de l'itération « *si… alors recommencer telle partie »,* une telle structure est déjà familière chez les élèves. Son utilisation permet d'éviter des itérations imbriquées.

Par ordre de difficulté croissante des boucles chez les novices en informatique, un panorama de la littérature de recherche semble donner une classification suivante :

- Répéter… jusqu'à ce que
- Pour… jusqu'à… faire
- Tant que… faire

D'autres raisons données sont en rapport avec la place de la condition d'arrêt dans la boucle. Les débutants en programmation. Ils ont des difficultés à comprendre ou à évaluer la condition qui vient avant l'action. Selon les schémas possibles ***action/test et test/action*** qui peuvent intervenir dans la boucle, c'est le deuxième schéma qui pose de difficultés aux novices. Les justifications données pour expliquer ces difficultés concernent l'anticipation né-

cessaire chez les débutants pour évaluer la *condition avant l'action* (Mejias (1985) : alors que les boucles *Répéter… jusqu'à ce que* et *Pour… jusqu'à ce que suivent le* schéma *action/test,* la boucle *Tant que fait le schéma contraire, test/action, ce qui* justifie leur classement précédemment donné.

Les difficultés posées par l'apprentissage de l'informatique ont plusieurs sources. Les savoirs informatiques peuvent être acquis lors d'une approche purement algorithmique au papier-crayon, notamment par la construction d'algorithmes. Mais, ces derniers sont conçus pour être programmés afin de résoudre un problème donné par une machine. Cette approche de la programmation apporte des nouvelles difficultés non seulement liées au langage mais aussi à la représentation de l'exécutant notamment comment se fait le processus de résolution à l'intérieur de la machine. Le point 4. suivant présente les difficultés que peuvent vivre les apprenants novices en informatique en général et particulièrement en programmation.

4. Représentation mentale des systèmes informatiques

La résolution d'un problème par un exécutant comme un ordinateur fait intervenir une interaction entre trois niveaux du système informatique formé par le système matériel, le système d'exploitation et le langage de programmation (Rogalski & Samurçay, 1986). L'absence de représentations mentales suffisantes adéquates de ce système informatique[93] est le principal handicap du sujet débutant face à une tâche de programmation (Lagrange, 1992). Se référant aux travaux antérieurs de Hoc (1977), Lagrange indique que la programmation est très exigeante en capacités de représentations mentales à la fois dans la généralité du dispositif mais aussi dans ses particularités et, de façon imagée, compare le débutant en programmation à quelqu'un qui *« serait dans une situation où il est appelé à faire un plan d'une maison sans connaître les fonctions des pièces, les mœurs des habitants, les contraintes de construction »*

Dans une situation de programmation, les problèmes des élèves tiennent pour partie aux difficultés qu'ils éprouvent à se construire des représentations globales du système informatique qu'ils utilisent en raison d'utilisations partielles menées dans des contextes disciplines (Delamotte, 2009). Ici, il faut entendre le système informatique dans le sens que lui donne Lagrange (1992) : l'ensemble « ordinateur + langage de programmation ».

93 Différentes appellations en rapport avec l'expression « dispositif informatique » sont données dans la littérature : dispositif informatique, système informatique… Dans notre travail, nous comprendrons cette expression dans le sens de Lagrange (1993) qui la définit comme l'ensemble formé de l'ordinateur et du langage de programmation utilisé

La problématique de représentation lacunaire ou erronée du système informatique des apprenants débutants en informatique en général et en programmation en particulier, a été soulevée dans beaucoup de recherches déjà faites en France depuis une trentaine d'années. Sans être exhaustif, nous pouvons citer Rogalski et Samurçay (1986), Samurçay (1985), Mejias (1985), Duchâteau (1994), Lagrange (1993), etc.

Ces auteurs et bien d'autres montrent que l'absence de représentations mentales adéquates du dispositif informatique est le principal handicap du sujet débutant dans l'apprentissage de la programmation par la résolution des problèmes. Rogalski et Samurçay situent ce manque de représentation mentale du système informatique dans le champ conceptuel de l'informatique. Elles distinguent deux niveaux de représentations erronées du système informatique. Le premier niveau est en rapport avec une confusion ou une mauvaise différentiation des fonctions attribuées à l'utilisateur ou à l'ordinateur. Un exemple donné est celui de la confusion des instructions de lecture qui, normalement appellent une entrée au clavier, c'est-à-dire une écriture de la part du sujet et, celle des instructions d'écriture qui ont pour fonctions d'afficher un résultat sur l'écran.

Le deuxième niveau de confusion consiste pour le sujet à attribuer à l'ordinateur des fonctions et des propriétés « sémantiques » normalement confier à l'utilisateur notamment en croyant que *« l'ordinateur attribue une signification à tous les éléments du texte du programme. En particulier, l'ordinateur est alors supposé « comprendre » le sens des messages qu'il doit afficher »*. Selon elles, une représentation erronée du fonctionnement du système a des conséquences sur la qualité des relations entre l'ordinateur, le programme et l'utilisateur (Rogalski & Samurçay, 1986) :

> *« Les représentations erronées vont interférer avec les processus liés à l'algorithmique sous-jacente à la programmation ; elles vont aussi jouer un rôle dans la signification attribuée par le sujet aux résultats de l'exécution des programmes, ainsi que celle des messages d'erreur éventuels »*

Ce déficit de représentations mentales suffisantes du système informatique n'est pas seulement chez les élèves mais aussi chez les étudiants (Poisseroux et al., 2009). Selon lui, l'existence d'un écart de représentation entre le système informatique et la perception chez eux, est à l'origine de nombreuses catachrèses. Passeroux et al. donnent l'exemple de la plupart des démarches et des productions liées aux TIC qui conditionnent d'autres démarches et d'autres

productions. Si l'insuffisance de représentations est une source d'usages moins productifs, une maîtrise des concepts informatiques est nécessaire pour être réparée, soulignent Poisseroux et al.,

Rogalski et Samurçay évoquent des anticipations nécessaires pour pouvoir résoudre un problème de programmation. Selon elles, si la programmation suppose des représentations du système de relations en jeu, ces dernières ne sont pas d'office installées chez les élèves (Rogalski & Samurçay, 1986, p. 102) :

> *« Dans l'écriture d'un programme, le sujet doit se représenter quelles seront, au moment de l'exécution, les relations fonctionnelles entre le programme écrit, l'ordinateur sur lequel ce programme tourne et l'utilisateur qui entre les données nécessaires. »*

Selon elles, se représenter ce partenariat, et surtout savoir les fonctions de chacun des partenaires, est vraiment indispensable pour l'apprenant débutant en la programmation, étant donné qu'une absence de ces représentations ou encore une mauvaise représentation de ces relations peut être dommageable sur l'activité de l'élève qui s'initie à l'algorithmique et à programmation.

La nécessité de faire acquérir une représentation mentale suffisante du système informatique n'est pas seulement nécessaire pour les élèves ou étudiants se destinant à la programmation informatique n'est pas nouvelle, mais date du début des années quatre-vingt-dix. Se positionnant contre le discours alléchant et trompeur des professionnels qui met un accent sur le caractère à la fois simple et convivial de l'utilisation des environnements informatisés, il précise que la réalité est toute autre chose et que leur utilisation raisonnée et réussie doit absolument passer par l'acquisition des représentations mentales à la fois mentalement adéquates et opératoires de leurs caractéristiques (Duchâteau, 1994). Cette nécessité d'acquérir une culture de représentations mentales est motivée par la particularité des caractéristiques communicationnelles en évolution récurrente qui s'établissent, souvent sous forme imagée ou métaphorique, entre les machines automatiques à commander et leurs utilisateurs appelés à les commander.

On peut donc se demander si les élèves de lycée ont ces représentations adéquates du système informatique. Si oui jusqu'où ils arrivent à se représenter ces relations, sinon quelles sont leurs difficultés.

Programmer un ordinateur pour résoudre un problème n'est pas toujours facile pour les apprenants débutants. Rogalski (1992, p. 4), sans distinction d'âge, donne le déficit de la repré-

sentation mentale du dispositif informatique comme un grand problème pour comprendre les processus en jeu à l'intérieur du système informatique pour une activité de programmation pour les débutants :

> *« utilisateurs « naïfs » (profanes en informatique) ou apprenants visant l'apprentissage de la programmation, à titre culturel ou professionnel, tous – adultes ou jeunes élèves – rencontrent le problème de la représentation du dispositif informatique : quels sont les « objets » manipulés par le logiciel, quels types d'opérations sont effectuées sur ces objets : « où » est-on – par rapport au dispositif informatique – quand on ouvre, lit, modifie, sauvegarde, copie, un travail… »*

Dans leur article concernant l'étude des difficultés cognitives dans l'apprentissage de la programmation chez des élèves du secondaire, Rogalski et Samurçay montrent qu'une présentation erronée du système informatique et de son fonctionnement, est l'une des principales difficultés pour les débutants en programmation. Selon elles, ces représentations erronées peuvent se situer à deux niveaux (Rogalski et Samurçay, 1986, p. 102) :

> *« À un premier niveau, le sujet différencie mal (ou pas) les fonctions d'exécutions de l'ordinateur et les actions de l'utilisateur ; à un second niveau, le sujet prête à l'ordinateur des propriétés « sémantiques » de l'utilisateur »*

Selon Arsac (1989) les élèves et étudiants savent résoudre beaucoup de problèmes mais le fait de faire faire, donc faire résoudre ces problèmes à un exécutant-ordinateur est très délicat pour eux. Deux raisons sont données pour justifier cette délicatesse. D'une part, parce que l'ordinateur est une machine universelle conçue pour le traitement de l'information, non construit de façon spécifique pour ce travail-là. D'autre part, l'exécutant-ordinateur nécessite une méthode de résolution de ce problème qu'eux-mêmes savent déjà résoudre. Le problème auxquels ils sont confrontés est moins la méthode à appliquer pour la résolution que la formalisation de cette méthode :

> *« (...) nous savons faire, mais nous savons mal dire comment nous faisons. On essaie alors d'expliciter notre propre façon de faire. On manifeste l'enchaînement de nos actions au moyen d'un organigramme, puis on code le résultat dans un langage de programmation : l'analyse informatique part du vécu ».*

Éric Bruillard (2009) considère les algorithmes et la programmation comme le cœur de l'informatique. Selon lui, en apprenant à classer, trier, ranger, chercher, mettre en ordre…, opérations complexes nécessitant la mise en œuvre d'algorithmes, les élèves sont appelés à décomposer une tâche en une suite d'opérations simples qui seront réalisées selon un ordre donné pour résoudre ce problème. Ces compétences sont acquises en algorithmique et programmation. Malgré la relative convergence sur la vision algorithmique et programmation comme forme de l'informatique à dispenser au début du lycée, des divergences persistent sur la qualification des enseignants, mathématiciens ou informaticiens, qui doivent piloter cet enseignement pour sa pérennité et son succès (Blanchet, 2000). Dans une table ronde organisée par la Société Mathématique de France (SMF), la question des contenus et des enseignants est posée. Dowek, un des intervenants évoque la nécessité pour la discipline informatique des enseignants informaticiens, tout en soulignant une difficulté liée à leur formation rapide et de manière efficace[94].

5. Conclusion du chapitre

Les savoirs informatiques fondamentaux de variable et de boucles sont quelques-uns de ceux qui sont prescrits en algorithmique dans les nouveaux programmes du cours de mathématiques au lycée scientifique en France. Les difficultés qui accompagnent leur appropriation sont de plusieurs ordres. En plus des difficultés habituellement posées par l'acquisition des savoirs mathématiques, ces savoirs informatiques peuvent apporter non pas aide mais des problèmes et des difficultés supplémentaires. Cela interroge les conditions de leur enseignement et apprentissage.

Il sera question dans la suite de voir comment ces notions sont enseignées par les enseignants et acquises par les élèves. Certaines questions guideront la suite de la recherche. Quelles sont les notions enseignées aux élèves de lycées ? À travers quels types d'activités, ces notions sont abordées en classe ? Comment et par quelle méthodologie, elles sont abordées ? Comment les élèves s'en approprient-ils ? Quelles difficultés rencontrent-ils ? Comment varient leurs difficultés en fonction de ces notions ?

Ce sont les quelques questions auxquelles nous tenterons de trouver de réponses dans la suite. Avant de développer la méthodologie utilisée, nous présentons d'abord notre cadre de

94 http://smf.emath.fr/files/text_like_files/crtrinfo25septembre.pdf, idem.

référence. Après nous présentons notre problématique. Il sera question de revenir sur les quelques éléments des problèmes précédemment présentés pour les développer davantage. Enfin, la partie sera clôturée par la méthodologie qui va guider notre recherche.

DEUXIÈME PARTIE : PRÉSENTATION ET DISCUSSION DES RÉSULTATS

Cette partie présentera les résultats de la thèse, leur discussion et suggestions des perspectives de recherche essentiellement focalisées sur le contexte éducatif burundais. Elle sera développée en cinq chapitres, allant du V^{ème} chapitre au IX^{ème}. Le chapitre V donnera des résultats issus de l'analyse des textes officiels et des manuels scolaires de lycée. Le chapitre VI mettra en exergue des pratiques des enseignants et des apprentissages de leurs élèves dans le contexte de l'enseignement de l'informatique en cours de mathématiques.

Le chapitre VII présentera les résultats relatifs à l'enseignement de l'option ISN en classe de terminale au lycée. Le chapitre VIII donnera les résultats de l'apprentissage de l'informatique en contexte de projets au niveau licence de l'université. Le chapitre IX clôturera la partie : il concernera la discussion des résultats présentés précédemment en fonction des hypothèses de recherche.

Chapitre V TEXTES OFFICIELS ET MANUELS CONCERNANT L'INFORMATIQUE

Dans ce chapitre nous présenterons les résultats d'analyse des documents officiels à la disposition de l'enseignant mais aussi des manuels scolaires les plus utilisés par les enseignants et les élèves.

> *Ce chapitre reprend les résultats issus de l'analyse des textes officiels, déjà publiés dans notre article (Haspekian & Nijimbere, 2012). Ils sont complétés par ceux obtenus par l'analyse large de manuels scolaires de mathématiques et informatique de tout le niveau lycée général, filière scientifique.*

1. Les textes officiels

Les textes officiels analysés comprennent les nouveaux programmes du lycée général (programme de mathématiques[95] et celui d'ISN[96]) et le guide d'accompagnement pour l'algorithmique en seconde[97]. Nous présentons les résultats suivant deux entrées : informatique en mathématiques et informatique comme spécialité ISN.

1.1. En mathématiques

a. *Une argumentation soutenue en faveur de l'introduction de l'algorithmique en mathématiques*

Dans le programme et le guide d'accompagnement, une argumentation est proposée, pour justifier l'introduction de l'algorithmique et l'étude des algorithmes dans l'enseignement de mathématiques. On note dans cette argumentation, le souci de relier le domaine de l'algorithmique aux mathématiques (et non à l'informatique) en utilisant divers points de vue. Le programme montre du point de vue historique, un lien originel entre ces deux champs, du point

95 http://www.ac-paris.fr/portail/upload/docs/application/pdf/2009-11/pgm2nde2009_109201.pdf, site consulté le 25 novembre 2010

96 http://www.education.gouv.fr/pid25535/bulletin_officiel.html?cid_bo=57572, Site consulté le 13 février 2012

97 http://media.eduscol.education.fr/file/Programmes/17/8/Doc_ress_algo_v25_109178.pdf, site consulté le 25 novembre 2010

de vue de la pratique des mathématiques une omniprésence des algorithmes, et du point de vue de l'enseignement et de l'apprentissage, les vertus et apports de l'algorithmique pour l'apprentissage du raisonnement dans toutes les autres parties du programme de mathématiques.

Cependant, malgré ce souci, les programmes mettent également en avant le lien entre l'algorithmique et l'usage des technologies dans notre univers actuel. À ce titre l'algorithmique pourrait être vue comme étant un domaine plutôt à rapprocher de la programmation et de l'informatique. Cette tension qui existe dans les programmes entre « algorithmique-informatique » et « algorithmique-mathématiques » se révélera être une difficulté pour les enseignants ainsi que pour les formateurs en charge de concevoir des formations sur l'algorithmique.

- **Un lien originel entre mathématique et algorithmique**

Plusieurs expressions renvoient à un lien originel entre les mathématiques et l'algorithmique :

> *« L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme (fonctions, géométrie, statistiques et probabilités, logique) mais aussi avec les autres disciplines ou la vie courante » (Programme, p. 11, souligné par nous).*

> *« La démarche algorithmique est, depuis les origines, une composante essentielle des mathématiques. (Programme, p. 11, souligné par nous).*

C'est un lien originel, que l'on retrouve dans l'enseignement de mathématiques lui-même :

> *« Dans le cours de mathématiques, les algorithmes apparaissent très tôt dans la scolarité » (Guide d'accompagnement, p. 3, souligné par nous).*

> *Au collège, les élèves ont rencontré des algorithmes (algorithmes opératoires, algorithme des différences, algorithmes d'Euclide, algorithme de construction en géométrie) » (Programme, p. 11, souligné par nous).*

- **Une omniprésence des algorithmes**

Une autre justification repose sur l'omniprésence des algorithmes dans les mathématiques de la vie courante. Un paragraphe intitulé : « *présence universelle des algorithmes* » l'explique :

> *« les mathématiques sont partout présentes dans la vie courante : traitement de données, statistiques, codage, simulation numérique… Mais cette présence qui se renforce, est souvent occultée aux yeux du public qui ne voit que le produit fini. Cette observation s'applique parfaitement aux algorithmes dont on voit plus souvent les résultats que les principes fondamentaux » (Guide d'accompagnement, p. 3).*

Cependant, un autre paragraphe explique la « présence d'algorithme » en la reliant cette fois à notre « univers technologique » dans la vie courante (Guide d'accompagnement, p. 3) :

> *« La présence d'algorithmes dans l'univers technologique qui nous entoure n'est plus à démontrer. Depuis l'automate le plus simple jusqu'aux systèmes les plus complexes, les algorithmes ordonnent beaucoup de nos gestes quotidiens ».*

b. Des capacités transversales et des compétences clairement identifiées

Ces deux arguments sont renforcés par le fait que, dans le nouveau programme, l'introduction de l'algorithmique ne constitue pas une section à part entière du programme mais, est intégrée dans chacune des trois parties (fonctions, géométrie, statistiques et probabilités). Cette forme est justifiée par un argument de transversalité des capacités attendues en algorithmique (Programme, p. 4) :

> *« Les capacités attendues dans le domaine de l'algorithmique d'une part et du raisonnement d'autre part, sont transversales et doivent être développées à l'intérieur de chacune des trois parties ».*

Un dernier argument est plutôt pratique et tourne là encore les algorithmes vers leur implémentation dans des machines. Les méthodes manuelles dans la résolution de certains problèmes peuvent en effet s'avérer insuffisantes, d'où le besoin de méthodes plus performantes dont l'utilisation de la machine (Guide d'accompagnement, p. 4) :

> *« l'introduction de chaque nouvel élément (variable, boucle, itération, etc.) devrait apparaître lors de la résolution des problèmes pour lesquels les démarches habituelles*

sont trop longues ou peu performantes : par exemple dans le cas de répétition d'une tâche, ou dans le cas d'un traitement trop long pour être envisagé « à la main ».

La présence de « machines » et de la programmation se confirme par la suite dans les activités auxquelles les élèves doivent « *être suffisamment entraînés* » que les textes officiels listent (Programme, p. 11) :

1. décrire certains algorithmes en langage naturel ou dans un langage symbolique ;
2. en réaliser quelques-uns à l'aide d'un tableur ou d'un petit programme réalisé sur une calculatrice ou avec un logiciel adapté ;
3. interpréter des algorithmes plus complexes.

- **Des notions à aborder transversalement pour tout le lycée**

Contrairement aux autres domaines mathématiques, l'enseignement de l'algorithmique ne doit pas constituer un chapitre à part entier : elle intervient comme une méthode de résolution de problèmes à l'intérieur des autres chapitres du programme de mathématiques à travers lesquels les notions algorithmiques seront transversalement abordées. De plus, si les objectifs de l'algorithmique sont présentés à part dans le programme en classe de seconde[98], (et même dans les manuels de seconde), ils sont à étendre sur tout le lycée (pour les trois classes de seconde, première et terminale). Précisons qu' aucune précision n'est donnée sur la limitation de contenus à aborder pour chaque classe.

- **Une description des contenus à enseigner**

De même, les contenus à enseigner naviguent là encore entre un apprentissage purement mathématique de l'algorithmique et conjointement une traduction « machine » de ces algorithmes, sans faire la part des choses claire entre ce qui relève des mathématiques et ce qui relève d'une « extension », laissant l'enseignant dans un flou au final sur ce qu'est l'algorithmique. Dans le cadre de l'algorithmique, les objectifs prescrits et poursuivis par le programme sont les suivants (programme, p. 11, c'est nous qui soulignons) :

« - formalisation en langage naturel propre à donner une traduction sur une calculatrice ou à l'aide d'un logiciel ;

98http : //eduscol.education.fr/D0015/pgm2nde2009.pdf

— familiariser les élèves avec les grands principes d'organisation d'un algorithme : gestion des entrées, affectation d'une valeur et mise en forme d'un calcul »

Il en est de même dans le guide d'accompagnement. Sur les 33 pages qui le composent, 25 sont réservées aux contenus à enseigner dans lesquelles le guide :

- rappelle les définitions, les principes et les notions de base de l'algorithmique : « instructions », « variables », « structures de contrôle », etc. Bref, il donne un cours synthétique et complet.
- donne des illustrations (d'affectation des données dans des variables, de lecture (entrée) des données et d'écriture (sortie) des résultats) dans un pseudo-langage.
- donne des illustrations dans deux des logiciels préconisés dans le programme : Scratch et Xcas.

Ces illustrations concernent les notions d'affectation de données dans des variables, de sortie de résultats et de structures de contrôle.

« Avec Scratch cela prend l'allure suivante : (...) » (Guide d'accompagnement, p. 7).
« dans Xcas, l'instruction « input (A) ; » va affecter dans la variable nommée A un nombre ou une expression tapé au clavier » (Guide d'accompagnement, p. 7).

- donne une illustration imagée de l'affectation (Guide d'accompagnement, p. 7) :

« On peut comparer l'affectation de valeur à une variable comme le rangement d'un objet dans un petit tiroir (ne pouvant contenir qu'un objet à la fois) ; sur la façade du tiroir figure un nom, c'est l'identificateur qui permet de parler du tiroir lui-même. Cette notion est très proche de celle de variable au sens mathématique. »

On voit aux citations précédentes que le souci d'un rapprochement avec les mathématiques est noyé dans des instructions propres à un langage machine (« *input* », *lecture* à l'entrée de l'algorithme, *écriture* à la sortie…) qui côtoient pourtant l'introduction d'un pseudo-langage dans les textes officiels :

Le guide d'accompagnement introduit en effet l'utilisation d'un pseudo langage français pour la description formelle des algorithmes (Guide d'accompagnement, p. 7) :

*« identificateur **prend la valeur** valeur » (...). « Ainsi l'instruction « A **prend la valeur** 2 » affecte la valeur 2 à la variable dont A est l'identificateur et ceci quelle que soit la valeur contenue au préalable dans la variable A (laquelle sera perdue) »*

Mais dans le même pseudo langage, est précisé que l'entrée et la sortie des données peuvent respectivement être traduites (Guide d'accompagnement, p. 7) :

*« **Saisir** identificateur » (...) « **Afficher** identificateur »*

Or ces instructions n'ont strictement aucune utilité d'un point de vue « algorithmique » (càd au sens mathématique). Elles sont par contre nécessaires bien entendu d'un point de vue « programmation » de l'algorithme. Les termes « Saisie » et « Afficher » proviennent du monde « calculatrice » et « programmation » mais ne se justifient pas du point de vue strict de l'algorithme. Ils constituent d'ailleurs une erreur fréquente d'écriture des algorithmes chez les étudiants de première année universitaire. Notre propos n'est pas de contester la présence de ce point de vue, un enseignement uniquement formel serait sans doute déraisonnable, mais de montrer que les textes officiels n'aident pas les enseignants à faire la part des choses précisément entre ces différents points de vue, à délimiter clairement ce qui relève de l'algorithmique au niveau des raisonnements de ce qui relève des nécessités de son implémentation technologique.

- **Un foisonnement de logiciels et langages *recommandés non obligatoires***

Il est souligné dans le programme (p. 11) que : « *Aucun langage, aucun logiciel n'est imposé »*, mais les langages et outils logiciels sont néanmoins largement introduits dans les textes, avec là encore, sans aucune précision sur le fait qu'on entre là dans un travail distinct de celui du travail de l'algorithme pur, travail qui engendre des questions propres et nouvelles purement liées à la programmation (interaction avec un utilisateur, capacités de calcul, limitations de mémoire, complexité en temps, etc.).

Les textes donnent une liste de logiciels libres et de langages, pouvant être utilisés : « Scratch », « Xcas », « Linotte », « Maxima », « Python », « Scilab », « Execalgo » et il est précisé qu'il est possible d'employer d'autres logiciels (Guide d'accompagnement, p. 31) :

- *« cette liste n'est pas limitative et rien n'empêche que d'autres logiciels existants ou à venir puissent être employés pour illustrer l'algorithmique »*

Le choix est laissé libre aux enseignants, aucun logiciel ne fait l'objet d'une plus forte préconisation (Guide d'accompagnement, p. 31) :

- *« La liste ne suit pas un ordre particulier (mais le premier logiciel [Scratch] est un peu à part) » (...). « L'environnement SCRATCH se distingue de ceux qui suivent par sa capacité à gérer la programmation événementielle voire parallèle : un projet SCRATCH ne se réduit pas à un seul algorithme, il inclut généralement des éléments multimédias (sons, images animées) ainsi qu'une multiplicité d'algorithmes s'exécutant tour à tour »*

Les logiciels utilisés sont variés et semblent répartis selon les domaines mathématiques sans qu'il n'y ait d'outil logiciel ou langage strictement réservé à une notion ou à un domaine. En effet, certaines classes d'instruments sont transversales aux domaines mathématiques. C'est le cas :

- des calculatrices (TI et CASIO) qui sont mentionnées dans tous les trois domaines ;
- du tableur qui est dans les fonctions comme en statistiques et probabilités ;
- du langage Python en géométrie comme dans les fonctions ;
- Scilab en statistique et probabilités comme dans les fonctions ;
- Scratch dans les fonctions comme en géométrie ;
- Xcas en statistique et probabilités comme en géométrie[99].

Enfin, les calculatrices programmables peuvent être utilisées également, mais au final si le choix de l'outil est à la liberté de l'enseignant, il serait néanmoins impensable, vu l'insistance et le foisonnement d'exemples des préconisations officielles, de ne pas en choisir un. Le guide donne même diverses informations pratiques de sites où on peut télécharger gratuitement des logiciels et des systèmes d'exploitation (Guide d'accompagnement, p. 31-32) :

« Les logiciels proposés sont « libres » au sens où leur téléchargement et leur installation sont autorisés sans aucune restriction »

99 Par exemple dans les fonctions, les exemples suggérés sont issus des domaines divers, allant de la géométrie plane à l'actualité. Les logiciels ou langages proposés sont : « tableur », « calculatrice TI », « calculatrice CASIO », « Python », « Scilab », « Scratch ». En géométrie, les logiciels ou langages mentionnés sont : « calculatrice Casio », « calculatrice TI », « Xcas », « Linotte », « Scratch », « Python ». En statistique et probabilités, les logiciels proposés sont : « tableur », « calculatrice », « Xcas », « Scilab ». Il ressort de ce qui précède qu'il n'y a pas

Par ailleurs, il est précisé que certaines activités d'élèves nécessiteront l'usage de l'ordinateur (Guide d'accompagnement, p. 5) :

> *« Les calculatrices graphiques programmables seront exploitées grâce à leur commodité d'usage en classe entière. Cependant, leurs limites dues à leur petite taille et leur capacité mémoire incitent à proposer aux élèves des activités s'appuyant sur des logiciels utilisables sur ordinateur. »*

On voit ici clairement soulevées de réelles problématiques liées à la programmation et non plus à l'algorithmique, telles que la rapidité d'exécution : certains logiciels ou langages de programmation présentent l'intérêt d'être plus rapides que d'autres notamment s'ils doivent effectuer un grand nombre de calculs (Guide d'accompagnement, p. 28) :

- *« Jeu du lièvre et de la tortue : (). L'intérêt d'un langage de programmation devient évident : l'itération est très rapide aussi bien à écrire qu'à exécuter (ce qui n'est pas le cas avec un tableur). On pourra noter, à cette occasion, que certains langages sont beaucoup plus rapides que d'autres »*

Le seul endroit où une nuance est évoquée est non pas pour distinguer clairement les aspects liés à la programmation mais pour souligner simplement que l'algorithmique ne consiste *pas seulement* en l'écriture de programme (ce qui laisse entendre que l'écriture de programmes fait partie intégrante de l'activité algorithmique). Le guide d'accompagnement précise en effet que les activités à proposer aux élèves dans le cadre de l'algorithmique doivent être variées et ne pas consister *seulement* à rédiger des programmes :

> *« La pratique de l'algorithmique ne se résume pas à l'écriture de programmes ; il serait même judicieux de ne pas commencer par là. Il convient donc de proposer aux élèves des situations, activités et organisations pédagogiques variés (). Les travaux pratiques seront conçus dans une perspective d'action de l'élève et lui seront présentés le plus souvent possible dans un cadre plus large que celui de la simple réalisation isolée d'un programme. Ce sont notamment des travaux qui s'inscrivent dans la durée et dans une organisation individuelle et/ou collective. »* (Guide d'accompagnement, p. 4, souligné par nous).

c. *Un discours du changement des méthodes pédagogiques*

Enfin, l'expression « algorithme » ou « algorithmique » est souvent accolée dans les textes officiels à une idée de changement dans les rapports de l'élève aux mathématiques mais si l'on regarde bien, ce n'est pas par l'algorithmique elle-même qu'il y a changement mais parce que l'algorithmique est dans les instructions immanquablement accolé aux outils technologiques :

> *« Le programme de seconde a été conçu pour être enseigné et mis en œuvre en s'appuyant assez largement sur les progrès de la science et de la technique informatique, qu'il s'agisse de logiciels ou de la pensée algorithmique » (Guide d'accompagnement, p. 3, souligné par nous).*
>
> *« l'algorithmique modifiera profondément le rapport entre l'élève et les outils ou instruments auxquels il sera confronté dans son environnement scolaire et particulièrement ceux habituellement identifiés comme issus du monde des TIC dans l'enseignement (calculatrices, ordinateurs, logiciels mais aussi les divers objets comme les appareils photos numériques, etc.) » (Guide d'accompagnement, p. 4, souligné par nous).*
>
> *« L'utilisation de logiciels (calculatrice ou ordinateur), d'outils de visualisation et de représentation, de calcul (numérique ou formel), de simulation, de programmation développe la possibilité d'expérimenter, ouvre largement la dialectique entre l'observation et la démonstration et change profondément la nature de l'enseignement » (Programme, p. 4, souligné par nous).*

d. *Conclusion sur les textes officiels*

Les nouveaux programmes au lycée affirment explicitement l'apparition de l'algorithmique dans le cours de mathématiques. L'analyse des textes officiels montre une argumentation soutenue en faveur de son introduction. Mais, ce souci d'inscrire l'algorithmique pleinement dans une activité mathématique est noyé dans une autre visée officielle, celle d'utiliser les technologies informatiques. Le domaine de l'algorithmique subit alors une tension entre « pensée algorithmique » et problématiques de programmation sans que les textes officiels n'aident les enseignants à faire la part des choses entre ces enjeux de natures différentes. S'il est évident qu'il y a des recommandations d'amener des élèves à implémenter des algo-

rithmes, il n'est pas bien prescrit le contexte technique dans lequel cela doit se faire notamment par quel langage explicite de programmation.

D'une part, le contexte institutionnel dans lequel les enseignants de mathématiques au lycée sont appelés à prester est particulier : enseigner l'algorithmique de façon transversale et donc sans y dédier des séances spécifiques. Il nous semble que pour un enseignant non expert de l'informatique qui situerait l'algorithmique complètement du côté informatique, les textes officiels ne constituent pas une aide et contribuent au contraire à entretenir l'amalgame. Constitués et structurés comme un cours mis à la disposition des enseignants, ils modélisent fortement un enseignement de l'algorithmique incapable de détacher de l'implémentation informatique. Donc, l'enseignant fait face à deux approches en tension : l'algorithmique orientée logique mathématique et l'algorithmique orientée programmation informatique, et cela sans que des clarifications suffisantes lui soient données. Nous nous interrogeons sur les pratiques des enseignants de mathématiques en contexte d'enseignement dans un contexte en confusion totale.

Comment initier les élèves à l'algorithmique ou à défaut aux notions algorithmiques sans faire de cours d'algorithmique ? De plus, les notions mathématiques posent souvent des difficultés aux élèves. Comment, avec la transversalité de l'algorithmique, est-il possible de faire apprendre les notions algorithmiques à travers des notions mathématiques encore compliquées pour des élèves ?

Enfin, la mise en œuvre de cet enseignement est contrainte au large champ d'interprétation laissé aux enseignants. Comment alors ces derniers s'approprient cet enseignement ? Comment abordent-ils les concepts informatiques avec leurs élèves ? Quels types d'activités prescrites aux élèves ? Quelle(s) approche(s) méthodologiques adoptées ? Quelles sont les motivations de ses choix chez les enseignants ?

Nous tenterons de trouver des réponses à ces questions dans le Chapitre VI relatif à l'enseignement de l'algorithmique au lycée.

1.2. La spécialité ISN

a. Pas pour former des spécialistes en informatique

Considéré comme un enseignement d'ouverture et de sensibilisation, l'objectif de l'enseignement de spécialité ISN en terminale de la série S n'est pas de former des experts en informatique (souligné par nous). Publié dans le bulletin officiel spécial du 13 octobre 2011, le

programme[100] est enseigné à titre optionnel. L'objectif visé est de réduire la fracture qui a été induite par la généralisation des sciences du numérique qui dominent la société, la vie personnelle et professionnelle. Pour cela, elle vise à fournir aux élèves, citoyens et travailleurs de demain, des notions fondamentales de l'informatique et des sciences du numérique. Elle vise aussi à ce que les élèves arrivent à maîtriser les mécanismes fondamentaux, ignorés par la plupart des populations, qui régissent les mutations profondes des sociétés actuelles induites par les sciences numériques et en apprécier les enjeux de société (programme, 2011, p. 1).

b. Des contenus à enseigner diversifiés, des capacités attendues précisées

Le programme est structuré autour de quatre notions – information, algorithmique, langage et programmation-, dites fondamentales, qui en constituent les parties. Pour chaque partie, le choix est justifié avant d'en esquisser les contenus à enseigner et les capacités attendues des élèves :

> *« La découverte de l'architecture de (ces) machines constitue une étape essentielle d'une initiation à l'informatique. De plus, mieux comprendre cette organisation est nécessaire pour programmer de manière efficace, en tenant compte des capacités et limitations des machines numériques » (programme, 2011, p. 6).*

Pour chaque partie du programme, les savoirs proposés et capacités attendues chez les élèves sont décrits tout en précisant leur caractère optionnel ou obligatoire.

1. **Représentation de l'information** : représentation binaire, opérations booléennes de base, numérisation, compression de données, formats, structuration et organisation des données, persistance de l'information, non rivalité de l'information ;

2. **Algorithmique : algorithmes simples, algorithmes de tri**. Différents algorithmes de tri par fusion, recherche d'un chemin d'un graphe, recherche d'un plus cours chemin sont considérés comme des algorithmes avancés. Pour ces savoirs, trois capacités sont attendues chez les élèves. Si comprendre et expliquer, à l'oral et à l'écrit, ce que fait

100 http://www.education.gouv.fr/pid25535/bulletin_officiel.html ? cid_bo=57572, consulté le 28 décembre 2012

un algorithme donné est une capacité obligatoire, la notion d'efficacité d'un algorithme est vue comme une capacité optionnelle.

3. **Langages et programmation** : types de données, fonctions, correction d'un programme, et langages de description. Pour la notion de correction de programme par exemple, quatre capacités sont attendues : mettre au point un programme, le tester, l'instrumenter et corriger des erreurs de ce programme construit. L'usage d'un outil pour exécuter ce programme est considéré comme une capacité avancée.

4. **Architectures matérielles** : les savoirs sont développés selon qu'il s'agit d'abord d'une machine simple, où les savoirs développés sont les composants de base d'un ordinateur et les instructions simples **;** ensuite des machines en réseaux pour lesquelles les savoirs sont : Pour le cas de machines en réseaux, savoirs prescrits et capacités attendues sont les suivants : la transmission point à point, adressage sur un réseau, routage et la supranationalité des réseaux. Enfin pour le cas d'un robot, il s'agit de la découverte d'un système robotique et de sa programmation. Dans ce sens, les savoirs prescrits sont dits introductifs et les capacités attendues chez les élèves sont toutes optionnelles où les activités consistent à décrire des systèmes à événements simples au moyen d'une machine à états finis mais aussi de programmer un mini-robot en vue d'exécuter une tâche complexe.

Contrairement aux autres parties du programme, l'algorithmique est la seule partie qui est mise en relation avec les notions antérieures du lycée et donne les types de capacités attendues en terminale (programme, 2011, p : 5) :

> *« Les programmes de mathématiques des classes de seconde et première contiennent une initiation à l'algorithmique sur laquelle il convient de s'appuyer. À travers l'étude de quelques algorithmes, on développe la faculté de lire et de comprendre un algorithme conçu par d'autres, puis d'en concevoir de nouveaux. Ces algorithmes sont exprimés dans un langage de programmation et exécutés sur une machine ou bien définis de manière informelle.*

Des notions de savoirs et de capacités, optionnelles et jugées avancées, sont aussi précisées et mises en évidence par le programme (programme, 2011, p : 3) :

> *« Une partie des savoirs et capacités, repérés par le signe distinctif, sont optionnels et seront traités en fonction des équipements disponibles ainsi que des orientations pédagogiques choisies par les enseignants »*

c. Des orientations pédagogiques multiples

À côté des savoirs à enseigner et capacités attendues du programme d'ISN, des orientations pédagogiques, sous forme d'observations, sont données. Les différentes expressions utilisées peuvent être classifiées en trois groupes ou niveaux selon l'information visée à apporter à l'enseignant : une introduction d'un savoir, son explicitation et les limites à ne pas dépasser dans la façon d'aborder. Leur analyse montre que certaines d'entre elles indiquent à l'enseignant la façon d'introduire le savoir. Une approche de présentation parallèle des notions en faisant référence aux autres est mise en évidence. Ce sont les expressions telles que : « On introduit la notion de… », « On introduit ces notions en comparant… », « Il est utile de faire référence à… », « En parallèle avec… », « On propose des activités adaptées à… », « On propose des activités sous forme de… », « On présente simultanément… », « On fait appel à… »…

D'autres expressions montrent comment à partir d'une circonstance donnée, on en profite pour s'interroger sur telle ou telle circonstance ou pour faire découvrir d'autres notions. Ce sont des expressions telles que : « On peut ici étudier… », « On peut expliquer la différence entre… », « On met en valeur… », « On met en évidence… », « On s'interroge sur… », « On découvre… »…

Les dernières expressions préconisent des frontières tracées pour l'enseignant pour ne pas aller trop loin dans son enseignement de certains savoirs. Ce sont les expressions telles que : « L'objectif se limite à… », « On se limite à… », « Au delà de… ». Ces expressions de limitations concernent essentiellement les savoirs dits « avancés ». Chacune de ces expressions précédemment évoquées peut intervenir plusieurs fois et ce, pour chaque savoir proposé. Ici, on oriente l'enseignant en le suggérant comment distinguer deux notions qui semblent liées et pouvant susciter une confusion (programme, 2012 : p. 5) :

« On présente simultanément les notions d'algorithme et de programme, puis on les distingue »

- **Une mise en activités fréquente des élèves dans des contextes variés**

En vue de motiver l'élève, le programme d'ISN recommande de centrer son enseignement sur l'activité de l'élève et avec des approches d'apprentissage diversifiés (programme, 2011, p. 2) :

« Afin de refléter le caractère scientifique et technique propre à la discipline et de développer l'appétence des élèves en faveur de cet enseignement nouveau pour eux, il convient de les mettre en situation d'activité aussi souvent que possible. »

« la progression peut suivre un rythme annuel construit autour de périodes spécifiques favorisant une alternance entre différents types d'activités (acquisitions de nouveaux savoirs, exposés, projets) permettant d'entretenir l'intérêt des élèves et de développer leur autonomie ».

Le programme ne propose aucun langage. Mais, il recommande à l'enseignant de faire le choix d'un seul langage quelle que soit l'approche utilisée. Des orientations sur des critères de son choix sont données (programme, 2011, p. 6) :

« L'enseignant choisit un langage de programmation selon les critères suivants : simplicité d'utilisation, liberté d'installation, présence d'outils associés, existence d'une communauté d'utilisateurs et de bibliothèques facilitant le développement ».

- **Une pédagogie de projet au centre de l'apprentissage**

L'approche par projet semble promue pour la mise en œuvre de ce programme d'ISN : elle est vue comme pouvant y contribuer efficacement. Différentes raisons ont été données pour justifier son efficacité et ses potentialités dans les apprentissages : apprentissages pluridisciplinaires liés au caractère complexe des problèmes prescrits pour résolution, apprentissages en groupes, utilisabilité des connaissances acquises précédemment, construction de nouvelles connaissances… comme les quelques extraits le témoignent (programme, 2011, p. 2) :

« Une pédagogie de projet est à privilégier pour favoriser l'émergence d'une dynamique de groupe »

Néanmoins, malgré les potentialités de cette approche, le programme recommande de l'associer aux autres approches dans son utilisation avec les élèves pour rentabiliser son d'efficacité (programme, 2011, p : 2) :

> *« L'informatique étant connexe à de nombreux domaines, il est utile d'envisager un travail pluridisciplinaire : la complémentarité des approches, associée à la richesse d'un travail collaboratif, joue un rôle stimulant pour les élèves et les équipes pédagogiques ».*

- **Un rôle central du professeur dans les projets**

L'esprit du programme est de mettre les élèves en apprentissages par projets. Dans ce cas, le professeur a un rôle central. Si ce n'est pas lui qui fait le projet, son rôle dans les apprentissages des élèves en général, et dans la réussite de ce projet en particulier, est essentiel. Il joue plusieurs rôles : impulser, coordonner le travail des élèves, guider les élèves dans leurs choix, animer les débats, réguler le travail des élèves, évaluer les productions…, bref, il a un rôle central.

- **Une évaluation tout au long de l'année**

Dans cette approche de la pédagogie par projet, l'enseignant n'a pas seulement un rôle de guide, il doit aussi enseigner mais aussi évaluer le travail des élèves. Si ce rôle d'évaluateur n'est pas nouveau, le programme demande une évaluation régulière qui tient compte des attentes de ce programme chez les élèves et qui diffère de ses habitudes dans les autres matières.

Une particularité semble attribuée à l'évaluation de ce programme d'ISN. Si ses modalités n'y sont pas précisées, elles sont ultérieurement sorties dans d'autres bulletins : un bulletin[101] spécial du 6 octobre 2011 et celui du 6 avril 2012[102]. Ce dernier éclaire le précédent et présente de façon détaillée les modalités d'enseignement et d'évaluation de ce programme : *« La prise en compte des progrès des élèves et de leurs acquis à l'issue de cet enseignement s'appuie sur une grille de compétences et de capacités, dont le détail est présenté en annexe I de la présente note de service. Cette grille est en cohérence avec les compétences attendues dans le livret scolaire. Elle est également proposée aux enseignants comme outil de suivi pédagogique des progressions des élèves et peut servir pour l'établissement des bulletins sco-*

101 http://www.education.gouv.fr/pid25535/bulletin_officiel.html ? cid_bo=57489
102 http://www.education.gouv.fr/pid25535/bulletin_officiel.html ? cid_bo=59864

laires trimestriels. La fiche d'évaluation, placée en annexe II, reprend les compétences et les capacités décrites dans la grille. Cette fiche fixe le cadre national de l'évaluation orale en cours d'année comptant pour l'examen du baccalauréat ».

d. Conclusion sur le programme d'ISN

- **Un enseignement aux ouvertures multiples**

Offrir aux élèves une culture générale informatique permettant de comprendre les problématiques liées à la généralisation de l'informatique et des sciences connexes dans la société est l'enjeu de cet enseignement. Pour y arriver, il a été choisi de structurer le programme d'ISN sur des contenus à enseigner qui englobent tous les contours de l'informatique et fondés sur ses concepts dits fondamentaux *d'information*, de *langage*, d'*algorithme* et d'*architecture.* Si l'enjeu n'est pas la formation des spécialistes de l'informatique, un bagage de connaissances informatiques de base nécessaire pour affronter des orientations diverses (vers la spécialisation en informatique ou vers d'autres domaines nécessitant des bases en informatique…) semble avoir été fourni aux élèves bénéficiaires de cette formation : des notions allant de celles de base à celles dites « avancées ». Les contenus de cette discipline satisfont plus d'un. C'est le cas de l'Académie de Versailles[103] qui, selon laquelle, les contenus de cet enseignement d'ISN sont suffisamment riches pour permettre à tout élève de tirer profit de ces programmes quelle que soit son orientation future. Pour elle, cette spécialité au lycée est une occasion de permettre aux élèves de s'assurer un solide bagage pour tout le parcours après le bac S, quels que soient l'institution ou le parcours choisi : université, classes préparatoires aux grandes écoles, des écoles spécialisées…

- **Une interaction entre contenus enseignés**

Au niveau didactique, au-delà des observations présentées comme des guides pédagogiques, le programme d'ISN a une place importante à l'apprentissage de la programmation, avec une obligation d'un langage de programmation au choix.

Deux approches sont bien spécifiées : l'algorithmique et les langages et programmation. L'algorithmique occupe à elle seule une partie importante du programme, avec un accent particulièrement mis sur le travail généralement fait au papier crayon. Les élèves sont familiarisés à ces deux types d'approches. La référence faite dans ce programme aux notions vues dans les années précédentes en classes de seconde et première (cf. deuxième partie : algorith-

103 http://euler.ac-versailles.fr/actualites/2012/isn/Depliant_ISN.pdf

mique), peut permettre leur approfondissement : effet, comme le précisent Chaachoua & Comiti (2007) l'appropriation d'une notion peut s'étendre sur plusieurs années de formation. Cette référence aux notions antérieures est pédagogiquement importante : maîtriser davantage les concepts algorithmiques pour une entrée facile des élèves dans la partie suivante en rapport avec la programmation. Ceci est davantage justifié par Wozniak et al. (2008) pour qui *« il n'existe pas de connaissance isolée, toute connaissance est un agrégat »*.

- **Une orientation socioconstructiviste des apprentissages**

Le programme d'ISN est plus pratique que théorique. Il met l'élève dans les conditions idéales d'une approche socioconstructiviste dans ses apprentissages : accompagné de ses pairs dans le groupe de travail, l'élève peut s'approprier avantageusement les contenus de cet apprentissage. L'approche de la pédagogie par projet, est pour nous adaptée pour cet apprentissage, ce qui permet aux élèves de travailler en groupes, donc en interaction pour bénéficier les compétences des uns et des autres. D'une part, la complémentarité permise par le travail en groupe est au service de cette approche socioconstructiviste. Ce sont les élèves qui sont appelés à concevoir eux-mêmes leurs projets : « les projets ne sont pas proposés *aux* élèves mais plutôt *par* les élèves, ces projets ne correspondent pas à l'apprentissage de savoirs ou de savoir faire fondamentaux (ce sont pas des "exos") mais aux mini-projets réalisés au cours de l'année dans le cadre de cet enseignement. »[104]. Cela est, selon eux, porteur de motivation étant donné qu'ils vont travailler sur des projets issus de leur création.

En conclusion, le programme de la spécialité ISN en terminale S ouvre aux élèves la porte à une culture informatique élargie, une ouverture permise par des contenus abordés suffisamment diversifiés. L'approche de la pédagogie par projet adoptée semble, selon nous, adaptée pour permettre aux élèves l'appropriation efficace de cet enseignement dans un cadre socioconstructiviste. Si ce programme révèle des pratiques potentielles attendues chez les élèves, comment seront les pratiques réelles des élèves ? Compte tenu des écarts qui existent entre les savoirs prescrits, savoirs enseignés et ceux appris (Chevallard, 1991), notre perspective est d'aller voir de prêt comment les enseignants s'approprient cet enseignement d'une part et, comment les élèves s'en approprient les contenus prescrits, d'autre part. C'est l'objet du Chapitre VI

104 https ://wiki.inria.fr/sciencinfolycee/PartageDeProjets

2. Manuels scolaires de lycée

Nous avons analysé deux types de manuels : les manuels de mathématiques et celui de la spécialité ISN. Les résultats sont présentés en suivant le type de manuels.

2.1. Manuels des mathématiques

Transmath[105], Mathsrepères[106] et Math'x[107] sont trois manuels les plus utilisés par les enseignants et les élèves au lycée. Relativement à l'informatique, nous avons effectué leur analyse (Annexes 16 et 17). Nous présentons dans la suite une synthèse des résultats obtenus.

a. Questions de forme : beaucoup de ressemblances

- **Une présentation uniforme**

Les manuels ont une présentation identique, conservée pour chaque collection et à chaque niveau d'enseignement – seconde, première et terminale-. Les notions informatiques sont présentées à part avant d'intervenir de façon transversale à travers les chapitres. Des notions en rapport avec des technologies (TICE) sont séparées de l'algorithmique.

- **Des notions informatiques mises en évidence**

Les pages des notions informatiques sont clairement identifiées par une numérotation et coloration spécifique. Les notions informatiques – algorithmiques et TICE – ne forment pas un chapitre comme les autres, mais sont présentées à part. Les TICE sont aussi présentées sur les rabats au début et à la fin des manuels. Dans chaque chapitre de mathématiques, une activité ou un exercice qui fait intervenir une notion informatique ou un logiciel, est chaque fois étiqueté par un post-it nommé « TICE » ou « ALGORITHMIQUE » selon le cas.

Pour chaque thème, les notions algorithmiques suivent une même présentation en trois ou quatre étapes : « activités », « vocabulaire », « exemple » et « exercices ». Dans la première étape – « activités »-, une ou des activités, choisies, soit dans le programme, soit dans la vie courante, est/sont présentée(s) en fonction des notions informatique qu'on veut introduire. Cette étape permet de rebondir à la deuxième étape – « vocabulaire » – pour ressortir la ou les notions(s) en question et le vocabulaire associé.

105 Barra, R., Barros, J. M., Bénizeau, P., Liorit, K., Morin, J., Nivaud, D., & Ricomet, V. (2010). *Mathématiques Seconde*. Italie: Nathan.

106 Choquer-Raoult, A., Hanouch, B., Cocault, M., & Joffrédo, T. (2010). *Programme de mathématiques en classe de seconde*. Paris. Hachette éducation.

107 Gastin, H., Guignard, M., & Guillemet, D. (2010). *Mathématiques seconde*. Italie. Didier.

Les notions algorithmiques sont développées, sous forme de cours théorique, illustré par des exemples de la vie quotidienne ou dans le domaine des mathématiques. À travers les chapitres, les notions informatiques sont transversalement réutilisées dans le développement des chapitres.

Trois étapes sont successivement suivies dans la présentation des notions informatiques :

- Des calculatrices à utiliser sont décrites sur les rabats des couvertures ;
- Des notions algorithmiques sont développées sur des pages spécifiques, facilement distinguées des autres par coloration spécifique ;
- Une présentation des TICE est faite sur des pages portant des couleurs qui le distinguent des autres.

Des illustrations des logiciels sont données sur des rabats des manuels avec des exemples choisis dans le programme. Comme pour l'informatique, les notions de logique sont présentées à part des chapitres, à la fin du manuel, avec une numérotation qui lui est propre.

b. Questions de fond : tension algorithmique-programmation

- **Algorithmique sous forme de fiches de cours**

Les contenus informatiques suivent une structuration identique. Les notions algorithmiques sont présentées à part, au début du manuel avant de revenir de façon transversale dans les chapitres du cours et ce, dans tous les manuels. Leur présentation laisse transparaître des intitulés faisant référence à une initiation à l'algorithmique : « Débuter en algorithmique » pour transmaths, « À la découverte des algorithmes et de la calculatrice » pour mathsrepères et « Premiers pas en algorithmique » pour maths'x. Si ce titre ne change pas pour le manuel mathsrepères, les titres choisis par d'autres manuels montrent une évolution progressive de sens selon les niveaux, en passant de la classe de seconde aux classes suivantes. Par exemple, le titre d' « Algorithmique » est choisi en première et celui de « Rappels d'algorithmique » en terminale pour la collection math'x et, « Pratiquer l'algorithmique » en première et terminale pour transmath.

De façon générale, au-delà de cette uniformité dans la présentation, les contenus algorithmiques sont quasiment les mêmes pour tous les manuels et toutes les classes, comme s'était d'ailleurs annoncé dans le programme : *« Algorithmique (objectifs pour le lycée »* (transmath, p. 9 ; math'x, p. 3 et mathsrepères, p. XXXI).

Utilisés par les élèves et les enseignants, les manuels à leur disposition sont conçus de façon à faciliter leur travail : non seulement, l'enseignant est guidé dans son travail de préparation de cours mais aussi presque toute la préparation semble avoir été déjà faite.

- **Contenus développés et termes définis**

Les notions informatiques développées sont celles de base : algorithme, variable, affectation, boucles, etc. Ces notions restent quasi les mêmes dans tous les manuels, elles sont présentées sous forme de cours et se distinguent par les détails qu'y apporte chaque manuel. Par exemple, dans transmath, elles sont présentées en thèmes, répartis successivement comme suit : Qu'est-ce qu'un algorithme ? Variable et affectation ; L'instruction conditionnelle ; La boucle itérative ; la boucle itérative, la boucle conditionnelle, et Programmer des algorithmes. Chaque thème apporte des notions ou une étape supplémentaire par rapport aux thèmes précédents. Un vocabulaire concernant un algorithme et les étapes qui le composent – entrée des données, traitement des données et sorties des résultats – sont définis. Dans leur présentation, ils sont accompagnés des exemples d'illustration d'un algorithme. Dans le deuxième thème, sont définies la notion de variable et les instructions de base y relatives : saisie, affectation et affichage de la valeur de la variable. Chaque vocabulaire est concrétisé par une mise en relation de deux thèmes précédents à partir d'un exemple par la construction d'un algorithme et ses étapes commentées. Dans cet exemple donné, cette construction d'un algorithme fait suite à la présentation de son protocole.
Chaque notion est définie, expliquée et illustrée par des exemples issus des domaines variés. Par exemple, dans *Mathsrepères*, une variable est définie comme *« une boîte, repérée par un nom, qui va contenir une information. Au sein d'un algorithme, pour utiliser le contenu d'une variable, il suffit de l'appeler par son nom »* .

Le manuel Math'x donne plus d'explications sur les boucles itératives et leurs illustrations dans leurs usages. Ce manuel regroupe les boucles en deux ensembles. Le premier ensemble concerne la boucle « Pour ». Selon ce manuel, les conditions de son utilisation sont les suivantes : lorsque le nombre de répétitions à faire est connu d'avance. Le deuxième ensemble présenté concerne les boucles pour lesquelles un test à faire est seulement connu mais pas le nombre de répétitions de la boucle. Il s'agit des boucles « Tant que » et « Répète ».

Comme l'algorithmique, les manuels analysés contiennent aussi des notions de logique, présentées de façon similaire mais, en fin des manuels.

- **Des illustrations dans différents langages**

Dans tous les manuels, la partie « cours » est systématiquement présentée de façon similaire Chaque notion est introduite par une à trois activités le plus souvent choisie de la vie courante. Après, un cours théorique synthétique est donné suivie d'exercices proposés pour son approfondissement, avant de l'utiliser pour résoudre un problème ouvert selon le cas. Souvent appliquée avec un logiciel ou un langage, une série d'étapes est suivie pour cette illustration : protocole à suivre, construction d'un algorithme et son implémentation sur la technologie (logiciel, langage ou calculatrice). Dans le manuel transmath, par exemple, chaque notion (algorithme, variable, boucle…) introduite par une activité, est concrétisée sur un exemple qui suit trois étapes commentées, à savoir : le protocole, la construction de l'algorithme et la programmation de cet algorithme sur un logiciel ou langage : Calculatrice et Algobox. Si le manuel transmath semble se limiter sur l'illustration des notions sur les calculatrices et rarement sur Algobox, d'autres manuels procèdent par des illustrations sur plusieurs langages. En voici un exemple à l'illustration V.1:

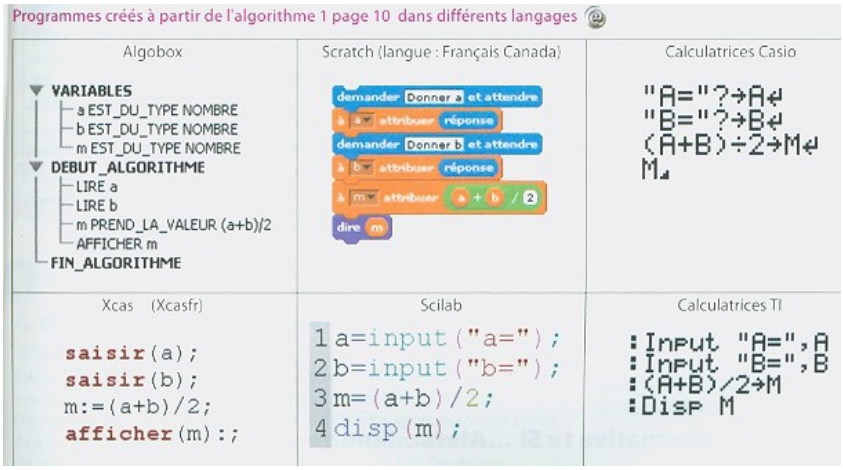


Illustration V.1: Un programme illustré sur plusieurs logiciels (math'x, seconde, p. 11)

Contrairement aux autres manuels, la boucle « Répéter » est absente dans le manuel transmath. À tous les niveaux, de la seconde à la terminale, tous les manuels reviennent sur les mêmes notions informatiques en général et algorithmiques en particulier. La différence selon les classes porte sur le type d'exemple utilisé pour illustrer la notion en question : il est choisi dans le programme de la classe.

c. Discussion et conclusion

- **Des manuels, un reflet fidèle du programme officiel ?**

Nous avons essayé aussi de comparer les manuels avec les programmes. Les résultats de cette comparaison montrent une certaine conformité entre les deux : les manuels traduisent fidèlement l'esprit des programmes, notamment en ce qui concerne le pseudo-langage :

« Ainsi, l'instruction « b prend la valeur b +2 » déclenche le calcule de b +2 pour le « Julie a programmé cet algorithme avec le logiciel Xcas » (transmath, p.199)

En rapport avec les contenus algorithmiques, les manuels contiennent les mêmes notions algorithmiques que les programmes (instructions élémentaires, boucle et itérateur, et enfin l'instruction conditionnelle). Ils précisent qu'en rapport avec l'algorithmique, les objectifs définis s'étendent sur tout le lycée, comme le souligne le programme. Ceci se remarque par le fait que les mêmes notions algorithmiques s'étendent de la classe de seconde à la terminale. Le foisonnement de logiciels est plus développé dans les manuels de seconde.

Des notions de complexité d'un algorithme, de structures sous forme de tableau ou de liste et même de fonction, n'interviennent dans aucun manuel. Mais, on trouve, dans tous les manuels, des exemples de programmation de petits algorithmes sur les mêmes logiciels, comme s'était dans le programme.

Si le programme recommande de ne pas donner un cours d'algorithmique, tous les manuels commencent par une structuration des notions algorithmiques sous une forme synthétique de cours, avant de les utiliser transversalement dans les différents chapitres du manuel. Nous émettons l'hypothèse que cette façon de présenter les manuels, en commençant de mettre en évidence les notions algorithmiques, peut influencer les pratiques des enseignants en suivant le même modèle dans leur cours avec les élèves : faire un cours théorique avant de les utiliser dans les chapitres. Ce qui contrasterait avec les injonctions institutionnelles.

Malgré cette convergence, des particularités dans la présence des logiciels dans les manuels se font remarquer. Les simulations qui se trouvent concentrées plus en géométrie que dans d'autres domaines des mathématiques, font plus appel aux logiciels de géométrie dynamique et aux tableurs. Les activités orientées vers les expérimentations, habitent plus en statistique mais aussi en géométrie et mobilisent aussi des tableurs et des logiciels de géométrie dynamique. En dehors de ces domaines d'autres logiciels sont utilisés dans certains manuels et

manquent dans d'autres. Pour le manuel math'x, les logiciels Xcasfr, et Scilab sont utilisés de la classe de seconde à la terminale et le logiciel, Scratch se limite en seconde. Scilab est rarement utilisé et intervient seulement dans le manuel math'x, et ce, de la seconde à la terminale.

La conformité dans la présentation de la logique par rapport à l'informatique pousse à penser à la complémentarité entre les deux domaines dans les apprentissages des élèves.

Pour conclure, nous pouvons dire que malgré quelques différences de détails, les manuels de mathématiques au lycée sont quasiment identiques. Ils affichent une juxtaposition de deux approches objet et outil de l'informatique avec une dominance de l'approche outil. Le fait d'associer à chaque notion algorithmique une illustration sur plusieurs logiciels ou langages de programmation laisse penser à une programmation systématique. Cela risque d'entretenir une confusion chez les enseignants qui peuvent penser qu'à chaque notion d'algorithmique, il faut obligatoirement programmer.

Il est intéressant de voir comment les enseignants gèrent ces tensions entre algorithmique et programmation dans leurs pratiques de classe. Quels sont leurs choix ? Quelle est la place réservée à chacune des deux approches ? Quels logiciels sont-ils privilégiés ? Quels types tâches prescrites aux élèves ? Quels en sont les motivations de leurs choix ? Nous tenterons de répondre à ces questions, dans le Chapitre VI

- **Informatique au service des mathématiques ?**

Les résultats de l'analyse des tâches proposées dans les manuels montrent que les notions informatiques mobilisées dans les exercices semblent être plus au service de l'apprentissage des mathématiques plutôt que faire objets d'enseignement et d'apprentissage de l'informatique. Environs 70 % des enjeux des notions informatiques sont au service de l'apprentissage des mathématiques, donc servant d'outils pour apprendre les mathématiques. Des enjeux tels que : découvrir une notion mathématique, mieux comprendre une notion mathématique, traduire une notion mathématique et encore aller plus loin en mathématiques sont les plus mis en en valeur. Seulement, les notions informatiques dont les enjeux sont orientés vers l'apprentissage des objets informatiques seraient estimées à environ un tiers. Les tâches dont les enjeux militent pour l'apprentissage des objets informatiques sont : la découverte d'une notion informatique, appliquer une notion informatique et aller plus loin en informatique. Ces tâches orientées vers l'apprentissage des objets informatiques sont plus limitées à l'approche

de l'apprentissage de l'algorithmique avec le plus souvent un travail limité « à la main » ou exécuter des programmes proposés sur des langages, généralement, au choix.

La tension entretenue par les manuels scolaires entre l'algorithmique et la programmation dans le sens où chaque notion algorithmique implique chaque fois son implémentation sur un logiciel, un langage ou une calculatrice. pousse à penser qu'on ne peut pas faire l'algorithmique sans la programmation. Certains manuels semblent vouloir dissocier les deux approches en considérant quelquefois la programmation comme une activité pour « aller plus loin ». Cette façon de donner l'information peut davantage contribuer à entretenir la confusion ou à donner raison aux enseignants moins motivés qui ne veulent pas faire la programmation.

Nous supposons que mentionner la programmation comme une activité pour « aller plus loin », est limitatif des pratiques. De plus, le fait que l'informatique prescrite en classe de seconde s'étend à tout le lycée peut-être un prétexte pour certains enseignants moins motivés pour l'informatique et particulièrement la programmation, ce qui a le risque de toujours mettre à plus tard certaines notions jugées plus difficiles pour soi. Cela peut davantage avoir lieu si ce n'est pas lui qui aura les mêmes élèves les classes suivantes.

Ces résultats étaient prévisibles au regard des objectifs assignés au programme des mathématiques de 2009 en classe de seconde. L'apprentissage des objets informatiques n'est pas vraiment et clairement très visible et souligné en tant qu'objets d'enseignement des notions informatiques pouvant aller dans la programmation, comme faisant partie des objectifs poursuivis dans le manuel de math'x dont les tâches ont été analysées, (math'x, seconde, p : 3, soulignés par nous) :

> *« modéliser et s'engager dans une activité de recherche ; conduire un raisonnement, une démonstration ; pratiquer une activité expérimentale ou algorithmique ; faire une analyse critique d'un résultat, d'une démarche ; pratiquer une lecture active de l'information (critique, traitement), en privilégiant les changements de registre (graphique, numérique, algébrique, géométrique) ; utiliser les outils logiciels (ordinateur ou calculatrice) adaptés à la résolution de problème ; communiquer à l 'écrit et à l'oral. »*

2.2. Manuel « Informatique et science du numérique »

Comme les autres manuels, l'analyse de celui-ci s'est intéressée à sa forme et au fond. L'analyse a été faite en annexe (Annexe 18). Nous présentons dans les points suivants les principaux résultats.

a. Description générale

Sorti des éditions Eyrolles, le manuel « Informatique et Science du Numérique »[108], est un ouvrage collectif d'académiciens et de spécialistes de mathématiques et d'informatique, en tête desquels Gilles Dowek. Développé sur 22 chapitres, le manuel de 301 pages donne les bases d'un cours d'informatique pour débutants. Le manuel est structuré autour de quatre concepts fondateurs de l'informatique, sur lesquelles se fondent les quatre parties le composant : langages, informations, machines et algorithmes.

Dans la suite, les différentes notions sont développées à travers les quatre parties dans lesquelles les chapitres sont développés. Les notions informatiques abordées sont réparties en deux groupes : des notions informatiques de base, et celles dites d' « un niveau plus avancé ». C'est notamment des notions d'ingrédients de programme tels que les variables, les instructions…, les types de base de variables, la récursivité, les notions de langage formel, les nombres entiers ou à virgule, les caractères, et textes, les fonctions booléennes, les structures d'information, la mémoire, l'organisation d'un ordinateur, les réseaux, les composants des robots et leur programmation, les tris, les graphes… Toutes ces notions sont définies, développées et concrétisées dans des exercices dont certains sont corrigés.

En marge, au début de chaque chapitre, est présenté un ou des chercheurs (noms, photo, année de naissance et de décès…) ayant joué un rôle clé, aussi bien dans la conception des langages de programmation, longtemps avant la naissance de l'informatique, que dans son développement.

L'ouvrage se termine par une présentation de 34 idées de projets à la disposition de ses utilisateurs. Ces projets peuvent servir de support aux enseignants dans leurs choix des prescriptions à leurs élèves et aux autodidactes soucieux d'apprendre l'informatique.

Après cette première lecture globale, nous présentons dans la suite une analyse détaillée de l'ouvrage visant les contenus abordés, le discours pédagogique et la place que peut jouer cet ouvrage dans l'acquisition des connaissances informatiques aux débutants.

108 http://www.editions-eyrolles.com/Livre/9782212135435/, site consulté 13 janvier 2013

b. Des contenus de base à ceux avancés

Les notions abordées dans cet ouvrage sont organisées autour de quatre concepts dits fondamentaux pour une science informatique : langage, information, machines et algorithmes. Au sein de chacun concept sont développés deux types de notions : celles dites de base et celles d'un niveau avancé par rapport aux débutants. Ces dernières sont marquées d'un astérisque au travers de l'ouvrage pour être identifiées.

Les notions considérées comme de base interviennent dans toutes les parties et dans tous les chapitres. Dans la première partie construite autour du concept de langage, les notions de base présentées sont : les ingrédients d'un programme (instruction, affectation, séquence, test), boucles (« For », « While » et leurs imbrications), types de variables (entiers, doubles, booléens, chaînes de caractères, déclarations, portée d'une variable, initialisation de variable, tableau) et notion de langage de programmation. Les notions avancées interviennent autour de chaque concept. Par exemple, pour le concept de langage, les notions de fonction, de récursivité, de langage formel, sont données comme des notions avancées.

c. Types d'activités prescrites

Différents types de tâches sont proposés dans le manuel. Dans le chapitre en rapport avec les ingrédients des programmes, les principales tâches proposées sont : comprendre et expliquer ce que fait un programme donné, tester un programme donné, modifier un programme existant pour avoir un résultat différent. Relativement aux boucles, les tâches suivantes sont proposées : écrire des programmes utilisant de boucles « For » et « While », imbriquer deux boucles « For », indenter un programme, commenter un programme, choisir entre les boucles « For » et « While » pour écrire un programme… Concernant les types de variables, les tâches suivantes sont proposées : différencier les types de base, changer le type d'une expression, déclarer des variables avec des types et des portées appropriées, initialiser les variables, utiliser un tableau dans un programme, calculer avec des chaînes de caractères, mettre au point un programme en l'instrumentant… Dans les fonctions, des tâches telles que : isoler une instruction, passer des arguments, récupérer une valeur, écrire l'entête d'une fonction ou une fonction, identifier la portée d'une variable dans un programme comportant des fonctions, choisir une portée adaptée aux différentes variables d'un programme comportant une fonction, choisir entre un passage par valeur et une variable globale, définir une fonction récursive.

Dans la deuxième partie de cet ouvrage, il est possible de distinguer les tâches proposées suivantes : représenter en base donnée un entier naturel donné, trouver la représentation binaire sur n bits d'un entier naturel donné en décimal, représenter en base binaire un entier à virgule donné, représenter en ASCII binaire un texte donné, écrire une page en HTML, identifier des formats d'images, numériser une image sous forme d'un fichier, comprendre les tailles des données et les ordres de grandeur, choisir un format approprié par rapport à un usage ou un besoin, à une qualité, à des limites, trouver une expression symbolique exprimant une fonction à partir de sa table, classer des fichiers sous forme d'une arborescence, utiliser un logiciel de compression, comparer des images enregistrées sous des formats de compression différents.

Dans la troisième partie – les machines – les tâches proposées sont les suivantes : exécuter une séquence d'instructions, trouver les adresses MAC des cartes réseau d'un ordinateur, trouver l'adresse IP attribuée à un ordinateur, trouver l'adresse IP du serveur par lequel un ordinateur est connecté à Internet, écrire un programme pour commander un robot, construire un circuit réalisant une fonction booléenne donnée, modifier un circuit construit pour avoir un résultat différent.

Enfin, dans sa quatrième partie, – les algorithmes –, l'ouvrage propose les quelques tâches suivantes : créer une image, transformer une image en couleurs en une autre image, augmenter le contraste d'une image, modifier la luminance d'une image, changer la taille d'une image, fusionner deux images données, évaluer l'efficacité d'un algorithme, comparer le temps d'exécution d'un algorithme avec deux méthodes différentes, lisser une image pour éliminer ses petits défauts et en garder les grands traits, s'interroger sur l'efficacité d'un algorithme, évaluer le temps d'exécution d'un algorithme donné.

d. Discours en faveur de l'apprentissage de l'informatique

Dans chaque chapitre, des savoirs à construire, des savoir-faire à acquérir, des exercices d'applications et des évaluations… sont bien mis en évidence et à la disposition de l'utilisateur. Dans chaque chapitre, les notions abordées sont d'abord illustrées au sein des exercices non corrigés, précédés de ceux corrigés. À la fin de chaque chapitre, une évaluation composée de trois questions théoriques clôturent le chapitre dans un encadré intitulé « Ai-je bien compris ? ». Les témoignages des ingénieurs œuvrant dans différents domaines sont suffisamment documentés et mis à la disposition des utilisateurs de l'ouvrage.

Des orientations pédagogiques, notamment le parcours susceptible d'être suivi lors de l'enseignement de cette discipline, ont été proposées. Selon les auteurs, l'ordre des chapitres tels qu'ils sont présentés dans l'ouvrage n'est pas obligatoirement à suivre comme tel dans son enseignement : certains chapitres d'une partie, jugés compliqués aux débutants, peuvent être vus plus tard après les autres chapitres se trouvant dans les autres parties. Ces chapitres sont marqués d'un astérisque. Dans cette préface, un appel est particulièrement mis sur l'acquisition, assez tôt, des bases des langages de programmation : les élèves sont appelés à être mis en situation réelle de programmation et être confrontés à ses vraies difficultés, très formatrices notamment pouvoir écrire, par eux-mêmes, de petits programmes et les tester sur un langage. Au sein de chaque chapitre, une méthodologie et un style pédagogique ont été adoptés dans la présentation et la promotion d'une progression cohérente et coordonnée des contenus pour une meilleure appropriation. Selon un ordre régulièrement suivi au sein de l'ouvrage, trois types de contenus y interviennent : une partie pour le cours, une partie pour des « Savoir-faire » visant à faire acquérir des capacités essentielles et une partie pour des exercices dont certains sont corrigés. Des orientations sont données pour les passionnés de l'informatique qui voudraient approfondir et se perfectionner dans leurs apprentissages dans des encadrés intitulés « Pour aller loin », qui ouvrent des recherches sur Internet. Après les exercices, chaque chapitre se clôt, chaque fois, avec trois questions se rapportant au cours dans un encadré intitulé « Ai-je bien compris ? » permettant à s'évaluer.

Le Java est le langage utilisé dans l'ouvrage. Si certains autres langages de programmation tels que Python, C, Camel, Fortran et Cobol sont mentionnés dans l'ouvrage, il est précisé que c'est seulement à titre indicatif : il n'y a aucune motivation particulièrement relative à ce choix de Java du fait qu'avec tous ces langages, des transpositions d'un langage à l'autre se font très aisément. Par contre, des concepts communs, très importants à connaître parce qu'intervenant dans l'organisation de chaque langage, sont soulignés : déclaration, affectation, séquence, test.

3. Conclusion sur les textes officiels et les manuels de lycée

Les manuels scolaires reflètent fidèlement les textes officiels. Dans les manuels de mathématiques, les contenus informatiques sont identiques. Quant aux technologies numériques présentées, elles semblent varier 'un manuel à l'autre et d'un niveau d'enseignement à un autre. Les notions informatiques comprennent l'algorithmique et les TICE, initialement présentées

séparément mais qui, après, interviennent comme des outils au service des mathématiques. Les résultats révèlent une faible proportion de l'informatique en tant qu'objet d'apprentissage. En effet, en mathématiques, de l'orde de 30 % des tâches proposées sont orientées vers l'apprentissage de l'informatique et 70 % vers l'apprentissage des mathématiques.

Les manuels de mathématiques au lycée affichent deux approches en tension : les mathématiques et l'algorithmique liée à la programmation. Chaque algorithme est relié à la programmation comme si les deux sont inséparables. : dans l'esprit des manuels, on ne peut pas faire l'algorithmique sans la programmation. En effet, à chaque notion algorithmique est parallèlement associée son implémentation, soit sur un logiciel, un langage ou une calculatrice. Néanmoins, certains manuels laissent voir qu'il est possible de dissocier les deux approches en considérant quelquefois, et ce, pour certains exercices, que la programmation une activité pour « aller plus loin ».

Dans le manuel d'ISN par contre, le discours est l'informatique, objet construit et structuré sur les contenus informatiques diversifiés allant des notions de base à celles dites « avancées » développées sur quatre concepts fondamentaux : algorithme, information, langage et machine. Contrairement au foisonnement de logiciels en mathématiques, dans ce manuel, il n'y a aucune trace de TICE : un seul langage est recommandé et, est utilisé.

L'analyse des manuels de mathématiques révèle des tensions algorithmique mathématiques – algorithmique programmation. Comment ces tensions sont-elles gérées par les enseignants qui, dans leurs pratiques de classe, utilisent ces manuels avec leurs élèves ? Comment sont choisies les tâches à prescrire aux élèves ? Quelles sont leurs motivations ? Comment les élèves et les enseignants de mathématiques se représentent l'informatique en général et l'algorithmique en particulier ?

Ce sont ces questions auxquelles les chapitres VII et VIII cherchent à répondre.

Chapitre VI ENSEIGNEMENT DE L'ALGORITHMIQUE AU LYCÉE

Le présent chapitre s'intéressera à la situation de l'enseignement et apprentissage de l'informatique en cours de mathématiques au lycée. Il sera développé en deux points. Le premier point concerne les enseignants et le deuxième leurs élèves. Dans chaque point, nous présenterons leurs représentations de l'algorithmique, les pratiques, les apprentissages et les difficultés des uns et des autres. Enfin, nous clôturerons le chapitre par une discussion synthétique des résultats obtenus.

> *Dans ce chapitre, nous utiliserons aussi les résultats de notre article (Haspekian & Nijimbere, 2012), complétés par ceux issus de l'analyse des pratiques en informatique d'une population plus ou moins diversifiée d'enseignants de mathématiques.*

1. Les enseignants

Les enseignants peuvent être classés en deux catégories : ceux qui ont suivi un stage de formation en algorithmique et ceux qui n'en ont eu. Leurs pratiques sont contrastées. Elles sont développées en six points : un sentiment d'illégitimité face à l'algorithmique, une absence de cours théorique d'algorithmique, une focalisation sur les aspects logiques, des choix de tâches et des difficultés de manque de recul et enfin, des effets de la formation.

1.1. Un sentiment d'illégitimité

La première représentation des enseignants de mathématiques qui semble bloquer les pratiques est un sentiment d'illégitimité chez certains envers l'enseignement de l'algorithmique. N'ayant pas eu de stage de formation en algorithmique, la plupart se sentent « mal à l'aise » au cours de cet enseignement : *« Moi, je ne suis pas compétente pour enseigner l'informatique. Nous, on nous demande d'être polyvalents au moins bivalent, informatique et mathématique, ce qui n'est pas possible » (A_L).*

En conséquence, les enseignants trouvent risquer d'enseigner une matière pour laquelle ils ne sont pas formés. Comme professeurs de mathématiques, ils disent avoir été institués en pro-

fesseurs d'informatique : *« ça nous a été imposé. (...) On a été forcé de s'y mettre… »* (D_P). Cependant, ils ont suivi des stages pour d'autres parties de mathématiques et notamment l'usage de logiciels. Maintenant, avec l'algorithmique, ils se voient contraints d'enseigner l'informatique pour laquelle ils ne sont pas formés et donc sans compétences.

> *« Initialement, je pensais enseigner les maths. Je n'ai jamais pensé à enseigner l'informatique. Moi, je découvre au fur et à mesure qu'on nous a imposé de faire l'algorithmique. On n'a pas été effectivement formés pour ça… mais maintenant, on a pas d'autres choix… » (A_L)*

1.2. Absence de cours théorique d'algorithmique

Conformément aux instructions officielles, les enseignants sans stage ne donnent pas de cours théorique pour les notions algorithmiques. Une seule enseignante, ex-formatrice et motivée, considère l'algorithmique comme une partie des mathématiques qu'il faut aborder comme objet d'enseignement. Ces notions sont introduites à l'intérieur des autres chapitres de mathématiques comme elle le précise ici : *« nous n'avons pas à avoir des cours d'algorithmique. Nous devons introduire l'algorithmique au fur et à mesure sur le contenu du programme. »* (F_D).

Pour tous, les contenus algorithmiques sont répartis sur tout le lycée. Cette répartition semble dépendre de chaque enseignant en fonction de l'appréciation qu'il fait de sa classe. C'est ce que dit celle-ci pour certains contenus jugés difficiles :

> *« le deuxième contenu qu'on a, qui est assez compliqué, moi je ne l'ai pas fait l'an dernier avec ma classe parce que j'avais lu que c'était pour le lycée et que ma classe n'avait pas le niveau suffisant pour que je cours après ça : on a aussi la boucle itérateur c'est-à-dire faire faire des petites instructions en boucle et puis les instructions conditionnelles, donc les élèves dans le cas d'une résolution de problème c'est pas en cours c'est clair que l'élève doit être capable de programmer un calcul itératif donc le nombre d'itération est déjà donné, de programmer une instruction conditionnelle un calcul itératif avec fin de boucle conditionnelle voilà ça c'est pour le lycée or je vais probablement cette année dans les secondes, je ne sais pas si je vais aller jusqu'à l'instruction de boucles. »(F_D)*

Donc, les enseignants prennent un peu l'algorithmique « à la carte » en l'adaptant à leurs classes. Le rapport des enseignants à l'algorithmique et plus généralement à l'informatique est aussi un facteur déterminant pour s'intéresser à ces notions dans sa classe : ceux dont ces rapports ne sont pas positifs, ont tendance à retarder l'enseignement de l'algorithmique. De plus, leurs pratiques diffèrent de celles des enseignants ayant eu le stage de formation : ces derniers donnent des éléments théoriques à leurs élèves.

a. Une alternance travail papier-crayon/machine

Lors des séances de mathématiques faisant intervenir l'algorithmique, les élèves sont en demi-groupes. Deux formes de travail sont généralement demandées : un travail sur papier-crayon, suivi d'un travail sur calculatrice/ordinateur. La configuration de la salle de classe y est adaptée : une partie de la salle de classe est pour le cours et une autre pour les ordinateurs. Néanmoins les ordinateurs ne sont pas systématiquement utilisés chaque fois qu'un travail d'algorithmique a lieu : la calculatrice reste centrale lors des activités de programmation en algorithmique même si Algobox est aussi quelquefois utilisé. En général, les ordinateurs sont utilisés pour des notions mathématiques autres qu'algorithmique.

Le choix du langage Algobox est, selon les enseignants, dicté par son intérêt pédagogique : il est proche du langage courant et les difficultés liées au vocabulaire et à la syntaxe sont réduites. Néanmoins, il lui est reproché de ne pas afficher toutes les étapes du déroulement de l'algorithme lors de son exécution. En général, une fiche de cours initialement bien préparée est distribuée aux élèves. Ces derniers, avec cette fiche, ont un petit travail à faire « à la main » puis sur machine ou directement sur machine. C'est le cas de l'exécution d'un algorithme déjà construit choisi dans les manuels.

b. Quasi-absence du travail en devoir maison

Certains enseignants ne donnent pas aux élèves de devoirs à faire à la maison en rapport avec cet enseignement. Différentes raisons sont données. Pour les uns, certains élèves n'ont pas d'ordinateurs chez eux. Pour d'autres, ces exercices demandent beaucoup de temps. D'autres disent ne pas donner ces devoirs parce qu'ils savent que leurs élèves ne vont pas les faire.
Par contre, il y a d'autres qui le font. Il est souvent question de continuer et terminer un exercice commencé en classe ou de faire le même exercice, mais de façon différente à celle faite en classe.

Les devoirs maison donnés en algorithmique sont limités sur un travail papier-crayon ou sur calculatrice.

1.3. Une focalisation sur les aspects logiques

Les enseignants de mathématiques se sentent ainsi illégitimes face à l'enseignement de l'algorithmique et plus généralement de l'informatique. Dans leurs pratiques, le côté logique de l'algorithmique en mathématiques les intéresse : *« Moi, ce qui m'intéresse en algorithmique, c'est la logique. L'algorithmique est très logique… »(P_M).*

Pour beaucoup, l'intérêt de l'algorithmique est lié au raisonnement en jeu et, par conséquent à la compréhension des mathématiques : Les activités proposées aux élèves (cf. paragraphe des pratiques des élèves) révèlent cette centration.

1.4. Choix de tâches

L'algorithmique en mathématiques au lycée est essentiellement abordée comme un outil d'enseignement des mathématiques notamment pour la résolution de problèmes. Quant aux choix de ces tâches chez les enseignants, ils sont orientés par la posture de l'enseignant. En algorithmique, les tâches choisies doivent avoir un intérêt mathématique, comme le témoigne cette enseignante :

> *« Moi, je suis un enseignant de maths, je reste dans un cadre mathématique. Je ne suis pas du tout compétente et prof d'informatique, pas du tout. (...) Je fais de l'algo dans le cadre du cours de maths. Ce qui est intéressant, c'est le niveau maths, je trouve. On lie le raisonnement maths à l'enchaînement des calculs et à l'enchaînement du raisonnement maths. Ça peut aider à la compréhension du raisonnement maths. C'est pour cela, pour moi, que ça reste maths. (…). L'objet est de former des profs polyvalents insidieusement. Je pense que l'on nous demande de devenir des professeurs d'informatique…, je pense que ç'est ça la dérive… » (A_L)*

Chez les enseignants sans stage de formation en algorithmique, les notions algorithmiques sont abordées au travers des chapitres du programme comme le demandent les textes officiels auxquels ils sont très attachés. Certains chapitres semblent plus propices pour eux pour aborder certaines notions. Par exemple, les notions de boucles sont plus abordées dans le chapitre sur les fonctions et puis sur les suites où elles sont plus utilisées. Si les tâches en rapport avec la construction d'un algorithme sont prescrites aux élèves en terminale, les

autres types de tâches (comprendre ce que fait un algorithme ; modifier un algo donné ; tourner un algorithme donné pour une valeur donnée sur une calculatrice ou Algobox) sont proposés depuis la classe de seconde.

1.5. Manque du recul

La non-participation au stage de formation chez les enseignants a des conséquences sur leur travail dont l'une est le manque de recul envers les contenus à enseigner. Si certains, par leur expérience, se sentent légèrement un peu au-dessus de leurs élèves, beaucoup parmi eux disent se sentir au même niveau qu'eux (A_L) :

> *« Je n'avais jamais pensé enseigner l'informatique. L'algorithmique nous est imposée sans formation, maintenant, on n'a pas de choix. (…) On manque complètement de recul. Moi, j'ai l'impression d'être au même niveau que les élèves, de découvrir comme eux ce qui est à faire, d'avoir autant de difficultés… c'est ça le problème. Moi, je trouve que c'est pas une bonne chose de faire enseigner quelque quand on ne domine pas un sujet. Avec les maths, j'ai beaucoup d'avance par rapport aux élèves alors qu'en informatique ce n'est pas le cas. En informatique, je suis très mal à l'aise, je manque de recul pour voir leur raisonnement, leurs erreurs, je dois l'avouer… »*

Beaucoup d'enseignants, face à cet enseignement se sentent gênés voire complexés devant les élèves. Souvent collés aux instructions officielles, les enseignants sans stage en algorithmique choisissent les activités à prescrire aux élèves parmi celles proposées, avec souvent des difficultés de faire la part des choses entre celles qui sont plus efficaces que d'autres. Ce manque de recul de l'algorithmique chez certains enseignants, fait qu'ils tâtonnent dans leur enseignement :

> *« Je trouve que l'algorithmique est quelque chose de particulier en mathématiques. Après deux ans, je n'ai pas encore trouvé comment faire. Je suis encore en train de tâtonner. » (S_C)*

Les enseignants de mathématiques éprouvent beaucoup de difficultés avec l'activité de programmation qui est essentiellement limitée à la calculatrice. Comparant ce qu'ils vivent en mathématique, l'enseignement de l'informatique leur pose d'énormes difficultés. Si, en mathématiques, les enseignants peuvent facilement anticiper des erreurs éventuelles des élèves,

en informatique, il n'est pas souvent le cas : *« en informatique, je regarde ligne par ligne pourquoi ça n'a pas marché, ça ne fonctionne pas. On n'anticipe pas les erreurs qu'ils sont susceptibles de faire. On est obligé de reprendre ligne par ligne… rires. C'est pas impossible, je ne sais rien mais… » (A_L.)*

Ce manque de recul engendre aussi deux autres conséquences : trop de temps de travail pour essayer de comprendre et une difficulté dans l'évaluation.

Une autre enseignante, indique quelques-unes des exigences de l'informatique en général et de la programmation en particulier : *« la capacité de localiser l'erreur dans le programme, savoir où pour pouvoir la corriger… et de conclure qu'« en informatique, je ne maîtrise rien »* (D_P). Leur collègue C_S, dit avoir *« des difficultés partout »*. Selon elle, le manque d'expériences et de recul ne le permet pas de dire si telle ou telle boucle est plus difficile que telle autre ou pose plus de difficultés aux élèves.

a. Trop de temps de travail

Les préparations demandent excessivement de temps. Le manque de formation est donné par les enseignants comme étant le problème majeur qui handicape leurs pratiques en algorithmique, car ils doivent apprendre d'eux-mêmes pour apprendre d'eux-mêmes pour comprendre le cours au moment de la préparation, ce qui demande beaucoup de temps. A_L s'inscrit en faux contre une étiquette de réticence qui leur est collée et accompagnée d'un discours de dévalorisation (A_L) :

> *« On est toujours présentés comme des gens peu réactionnels, attachés aux anciens programmes, incapables d'évoluer. Je ne crois pas que ce soit le cas. Il y a des gens intéressés par de nouvelles choses mais, il faut des moyens et des conditions pour que ça puisse se faire. »*

Une leçon doit être comprise par l'enseignant avant d'être enseignée. Cette autoformation leur prend beaucoup de temps (S_R) et certains utilisent des mots très forts pour caractériser le travail vécu (F_D) :

> *« Moi, j'ai vraiment une pratique professionnelle de 35 ans, l'introduction de l'informatique et de l'algorithmique nous a, en tant que professeur de mathématiques, aug-*

mentés considérablement le temps de préparation. Donc, actuellement, nous sommes quasiment étranglés ».

Comme cette enseignante, les autres enseignants sans stage en algorithmique insistent sur le fait que l'algorithmique demande un lourd investissement, au détriment des autres notions mathématiques à enseigner et il leur est « difficile de trouver un juste équilibre ». L'institution, au courant de leur manque de formation, fournit des documents volumineux mis à leur disposition pour une auto-formation, mais ceci semble contribuer à semer davantage la confusion au lieu de servir d'aide (F_D) :

« un certain nombre de documents qui sont les commentaires du programme et pour l'algorithmique on en a un qui fait 40 pages, c'est un document qui est assez ambigu parce que sachant que les enseignants n'étaient pas du tout formés en algorithmique alors que d'habitude les commentaires du programme ont pour but de nous montrer en fait, de nous développer un peu les objectifs de ce qu'on doit faire en classe, etc le déroulement de l'algorithmique il est particulièrement à la fois délicat puisqu'il est à la fois du côté de ce qu'on pourrait faire avec les élèves sans se rendre compte de ce qu'un élève de collège et puis aussi de la formation des maitres c'est-à-dire qu'il est très étonnant c'est-à-dire à la fois qu'il est insuffisant pour se former et si on a ça avec nos élèves on ne fait que ça. »

b. Difficultés d'évaluer

L'évaluation pose de difficultés avec l'algorithmique. Si les notions algorithmiques sont abordées en situation de demi-groupe, seul le travail des élèves fait au papier-crayon est évalué par les enseignants. La pratique de programmation, faite souvent avec la calculatrice ou sur ordinateurs, n'est pas évaluée. Pour les travaux évalués, cette évaluation prend plusieurs formes. Pour certains enseignants, il n'y a pas de note chiffrée donnée, précise F_D : les élèves sont notées au moyen d'appréciations avec des lettres et les autres, au départ n'évaluaient pas cette partie. Petit à petit, l'évaluation chiffrée du travail papier-crayon s'améliore en posant des questions à trous qui le facilitent.

c. Autres difficultés et contraintes des enseignants en algorithmique

Bien d'autres difficultés et contraintes sont soulevées par les enseignants de mathématiques en algorithmique : des problèmes de gestion de classes à effectifs pléthoriques, absences de

postes informatiques en cas de nécessité même si les élèves sont mis en demi groupes, absence de mainteneur informatique dans un contexte de pannes incessantes des postes informatiques…
Les pratiques des enseignants s'améliorent selon certains, d'autres pessimistes disent ne rien attendre de cet enseignement qui ne se contente que des initiatives privées et en autodidactes des enseignants sans formation.

Le contexte de la programmation est vécu comme plus exigeant que les mathématiques : *« Je trouve que finalement, la programmation c'est très compliqué. Ça nécessite à la fois de comprendre parfaitement ce qu'on fait en maths et, en plus, être capable de le restituer avec un langage machine. C'est pas simple… »* (A_L).

1.6. Effets du stage sur les pratiques enseignantes

Les pratiques présentées ici sont celles des enseignants de mathématiques qui ont bénéficié de stage de formation à l'algorithmique. Rappelons que la formation que nous avons suivie a été organisée sur deux années successives. Elle était la même et assurée par les mêmes formateurs : un enseignant-chercheur appuyé par des formateurs spécialistes des mathématiques.

Le stage de formation semble avoir été bénéfique pour les enseignants et influe sur leurs pratiques. Cela s'observe dans leurs déclarations que dans leurs pratiques de classes. Nous présentons leurs représentations et pratiques en trois points : conduite de classe, exemple d'activité introductive et leur état d'esprit vis-à-vis de l'algorithmique.

un petit cours théorique introduit par une activité ouverte et complexe, un exemple d'activité introductive et une algorithmique démystifiée.

a. Conduite de classe et stratégies d'enseignement

L'enseignement des notions algorithmiques suit trois étapes : une activité introductive, un petit cours théorique et des problèmes mathématiques à résoudre en utilisant les notions acquises.

- **Une entrée en douceur avec une activité introductive ouverte et complexe**

Les enseignants avec stage entrent en algorithmique subtilement en algorithmique, sans en parler, avec une activité introductive ouverte et complexe sans rapport direct avec l'informatique. Avec cette activité, l'approche du travail de groupe est privilégiée pour susciter des interactions entre élèves en groupes. Cette activité est améliorée chaque année, pour susciter les échanges fructueux entre élèves. Par exemple, B_M explique que, si l'activité de tri de cartes est celle souvent utilisée, le guidage des élèves dans leur travail est de moins en moins assuré, pour donner plus de place à leurs réflexions.

L'enseignant B_M témoigne de ses pratiques en algorithmique qui étaient initialement focalisées sur la programmation avant son stage comme d'ailleurs beaucoup d'autres motivés :

> *« j'ai commencé à l'heure. Je me suis rendu compte que ça n'allait pas. C'est vrai que si on fait quelque chose sans le stage ça peut ne pas marcher. Donc, le stage nous permet de donner la bonne orientation à notre enseignement. Quand j'ai commencé l'algorithmique, j'ai commencé par des activités qui peuvent être pas du tout intéressantes et puis, on se rendait compte que ça n'allait pas parce que je n'étais pas dans la bonne direction et donc, on se rend compte que ce que je faisais n'était pas bien. » (B_M)*

Un autre enseignant, initialement motivé, dit aborder certaines notions informatiques par des métaphores :

> *« je l'ai exposé comme un procédé de cuisine. J'ai jamais parlé d'affectation, j'ai parlé de ce qu'on fait quand on prépare un plat à cuisiner. Voilà, c'est vraiment dédramatiser. J'ai jamais parlé d'ordinateur comme tel mais des armoires où on prend des étagères et on étiquette… l'affectation des variables, voilà vous prenez une boîte, vous collez une étiquette dessus et vous mettez ce que vous voulez dedans. » (S_S)*

Sur ce, la qualité des activités introductives est importante. Pour eux, il faut une activité-problème, choisi de la vie courante qui nécessite un travail en amont des élèves en petits groupes, conduisant vers l'introduction des notions algorithmiques. L'exemple de tri (de cartes), acquis au cours du stage de formation, est celui qui a été utilisé comme activité introductive des notions algorithmiques. Elle est décrite dans le point suivant.

- **Un petit cours théorique d'algorithmique**

Une pratique préconisée en stage de formation est de faire un petit cours théorique au début. L'enseignante observée, ayant suivi cette formation, fonde ses pratiques sur ces suggestions. Si les notions algorithmiques sont, selon les textes officiels, à étendre sur tout le lycée, en laissant à l'enseignant de faire sa répartition, après le stage, les enseignants prévoient *« apporter rapidement dès le début du premier semestre de seconde, toutes les notions algorithmiques comme outils à utiliser quitte à les retravailler et les consolider le reste du lycée »,* comme le déclare A_NG au cours de notre entretien. En effet, le fait de ne pas donner un cours d'algorithmique fait qu'ils se heurtent, à un certain moment de l'enseignement, à un problème d'avoir besoin d'une notion pour la résolution d'un problème alors qu'elle est non encore abordée. Le choix de donner un petit cours théorique permet donc de prévoir des outils au préalable mais aussi d'initier les élèves aux notions informatiques.

- **Réinvestissement en mathématiques**

Après ce petit cours d'algorithmique de 3 à 4 séances, les notions seront réinvesties dans les différents chapitres de mathématiques notamment pour servir à la résolution de problèmes.

b. Un exemple d'activité introductive de l'algorithmique : tri de cartes

Selon les textes officiels, l'introduction des notions algorithmiques se fait par une activité. Généralement choisie de la vie courante chez ceux qui ont suivi un stage de formation, elle permet une entrée en algorithmique sans en dire le nom :

> *« J'ai pas commencé l'algorithmique en faisant l'informatique. J'ai commencé avec des séances de réflexion en groupe mais pas sur la programmation, pas sur l'écriture de l'algorithme, mais sur la recherche de l'algorithme pour la résolution d'un problème complexe… » (B_M)*

Ici est présenté le déroulement d'une séance introductive de l'algorithmique en classe de seconde par une activité de tri de cartes.

Une classe de seconde observée compte 32 élèves. C'est au cours d'une séance de cours de 2 heures, ils se sont librement répartis en huit groupes de quatre élèves chacun. Une activité de tri de cartes leur a été proposée par l'enseignante : 32 paquets de cartes aléatoirement numérotées. Chacun élève reçoit un paquet de 10 cartes à trier par ordre croissant. Un certain nombre de consignes à respecter étaient données aux élèves :

- ne pas retourner que deux cartes à la fois au maximum ;
- ne pas garder en « mémoire » de leur valeur en passant d'une comparaison à l'autre ;
- un travail de recherche individuel d'environs 20 minutes : un moment de lecture en vue de comprendre le sujet et s'approprier les consignes.
- chaque élève rédige dans son cahier la méthode de tri utilisée au moyen des mots tels que : d'abord, puis, ensuite, enfin, jusqu'à ce que, tant que, alors. La description donnée doit permettre à tout lecteur de pouvoir trier, sans problème, tout paquet de cartes en utilisant cette méthode ;
- mettre en commun leurs productions au sein de chaque groupe et rédiger sur une feuille portant les prénoms des auteurs ;
- La feuille commune est à remettre à la fin de la séance (de 2 heures)

Avec cette méthode, il leur ai demandé de donner le nombre maximum de comparaisons effectuées pour faire ce tri de 10 cartes et, celui pouvant être engendré pour trier 20 cartes ; 100 cartes et N cartes (N >1) (Généralisation de la méthode de tri) mais aussi inventer d'autres méthodes de tri pour faire le même travail.

Cette activité a mobilisé trois professeurs : la titulaire du cours et deux autres y ont pris part, un professeur de français et une autre de mathématiques. Celui de français aide les élèves dans la rédaction et/ou l'argumentation en respectant les consignes, notamment la bonne utilisation des expressions précédentes dans les phrases, celle de mathématiques qui se prépare à utiliser cette activité dans sa classe, suit sa mise en œuvre.

Pendant le travail des élèves, les enseignants circulent, chacun individuellement, dans chaque groupe. Ils peuvent répondre aux questions des élèves, les aident à la compréhension de ce qu'ils doivent faire. À la fin de la séance, les copies sont ramassées.

Beaucoup de notions algorithmiques sont implicitement abordées au cours de cette activité. Dans la suite, elles font objet de synthèse-bilan pendant au plus quatre séances suivantes de cours théorique comme le précise l'enseignante : l'algorithmique et les notions y relatives sont à haute voix annoncées cette fois-ci et elles seront dans la suite utilisées comme outils dans l'apprentissage des mathématiques. Affirmant faire de l'algorithmique mais pas de l'informatique, l'enseignante dit passer presque toute l'année sans aller en salle informatique avec les élèves de secondes : le travail papier-crayon est central. L'usage de la salle informa-

tique, surtout avec Algobox, n'a lieu qu'à la fin de l'année avec cette classe de seconde. Mais, elle est plus utilisée pour les autres classes de première et de terminale. Bien que l'écriture d'algorithmes commence en seconde, c'est en première que les élèves sont capables d'écrire des algorithmes qui fonctionnent. Les calculatrices sont utilisées comme des outils de vérifications et de test des algorithmes construits.

Avec cette approche, certaines tâches initialement difficiles à prescrire en contexte papier-crayon, sont maintenant possibles : « *résoudre une équation dichotomique, ce sont des choses qu'on ferait pas « à la main ». Donc, là où on devait les utiliser c'est difficile « à la main ». Ça apporte quelque chose pour nous et pour les calculatrices, les logiciels, l'algorithmique…* » (B_T)

c. De l'algorithmique démystifiée ?

Considérée comme de l'informatique, l'algorithmique n'a pas été bien accueillie par certains enseignants de mathématiques. Ceux qui ont participé au stage de formation en algorithmique étaient des enseignants initialement motivés par cet enseignement : avant le stage, 75 % des enquêtés (N=28) avaient déjà enseigné l'algorithmique. L'adhésion rapide à cet enseignement aurait été aussi soutenue par cette motivation et par la connaissance de certaines notions de base en informatique : un tiers parmi eux (1/3) avaient déjà eu un module informatique à l'université au cours de leur formation initiale. Ces deux conditions, la motivation et la possession de certaines notions de base en informatique, semblent avoir permis d'oser affronter cet enseignement. Une fois l'algorithmique démystifiée, le sentiment d'illégitimité qui engendrait la peur pour certains a cédé la place à la décomplexation, à la motivation et à la prise d'initiative. Certains extraits d'entretiens avec les enseignants illustrent bien ces évolutions dans leurs représentations :

- *« je dois vous dire que j'avais peur de l'algorithmique, mais maintenant ça me paraît plus simple enfin, c'est pas plus simple que ça mais…* » (S_S)
- « *j'ose plus à faire des choses qu'avant je n'osais pas faire » (*A_NG)
- « *Moi, je vois bien le prof de maths s'occuper de l'algo. Ça me semblerait étrange si ces notions sont enseignées par un spécialiste d'informatique. En tant qu'enseignant de maths, je les utilise pour illustrer, consolider, concrétiser, visualiser, quoi ! » ; « avant la formation, moi, c'était le tâtonnement ! »* (M_J_M)

Les enseignants ayant reçu le stage de formation témoignent des potentialités de l'algorithmique en mathématiques. Dominique Baroux, enseignante de mathématiques et formatrice, affirme avoir été initialement réticente envers cet enseignement. Elle affirme maintenant avoir découvert l'intérêt formateur de l'algorithmique dans le programme de mathématiques du lycée.

D'autres enseignants, après formation, projettent de la suivre chaque fois qu'elle est organisée. C'est le cas d'A_NG. Après le stage, d'autres enseignants motivés trouvent dommage qu'il y a des enseignants qui ne suivent pas ces formations. C'est le cas de M_J_M. Face à l'impossibilité d'obliger les enseignants encore réticents à participer à ces formations, il donne quelques propositions :

- réduire la période des vacances scolaires d'été au profit des formations payées ;
- valoriser l'algorithmique dans les épreuves de baccalauréat. Pour l'enseignant M_J_M, l'inscription de l'algorithmique aux épreuves de bac ne suffit pas mais aussi il faut valoriser les exercices posés. Il fait référence à la notation minime de l'exercice à l'épreuve de bac 2012[109].

Après la formation, beaucoup d'enseignants projettent de changer leurs pratiques et d'adopter de nouvelles stratégies dans leur enseignement d'algorithmique. Quelques exemples peuvent être donnés :

- A_NG pense visualiser des boucles en les délimitant au sein d'un algorithme pour aider les élèves à les localiser et les comprendre ;
- S_S pense déconnecter l'algorithmique de la programmation : *« avant la formation, il fallait tout de suite illustrer un algo sur une calculatrice, maintenant il faut insister sur le travail à la main… ».*

Certains d'entre eux affirment procéder par l'approche papier-crayon pratiquement toute l'année en classe de seconde. En cas de nécessité de vérification ou d'illustration, le recours est fait à la calculatrice. L'ordinateur n'est rarement utilisé qu'en classe de première et terminale. Ceci s'observe dans les activités proposées qui restent essentiellement manuelles, mais avec une évolution vers la mutualisation d'approches papier-crayon et ordinateur-langage plus tard au lycée avec une prédominance de la première approche. Une synthèse des pra-

109 Un exercice d'algorithmique était noté sur 2 points alors que d'autres exercices étaient plus valorisés.

tiques enseignantes est donnée sous forme comparative en annexe (Annexe 22), selon qu'ils ont eu ou pas un stage de formation en algorithmique.

En conclusion, les pratiques des enseignants sont contrastées. La formation en est responsable, mais elle ne paraît pas être la seule. D'autres facteurs comme la motivation associée au rapport personnel de l'enseignant à l'informatique a aussi une certaine influence sur leurs pratiques.

Dans la section suivante, nous développerons les représentations et les pratiques des élèves en algorithmique.

2. Le point de vue des élèves

Les élèves concernés ici sont ceux de la classe de seconde à la terminale, en dehors de la spécialité ISN. Leurs pratiques sont développées en quatre points : représentation de l'algorithmique, savoirs informatiques construits en mathématiques, approche utilisée par les élèves et enfin, leurs difficultés dans cet apprentissage.

2.1. L'algorithmique : ni des mathématiques, ni de l'informatique

Si les élèves sont en général motivés par les utilisations de l'informatique dans leurs apprentissages notamment les logiciels de géométrie dynamique en mathématiques, ils ne considèrent l'algorithmique ni pour des mathématiques ni pour l'informatique mais, pour un autre domaine scientifique ou une discipline supplémentaire à découvrir : pour eux, l'informatique, c'est avec un logiciel ! Interrogés dans le cadre de cet enseignement de l'algorithmique, au cours des entretiens sur les notions déjà vues en informatique, ils n'évoquent rien en rapport avec l'algorithmique. Leurs réponses étaient directement orientées vers des logiciels utilisés en classe. Cela est aussi confirmé par leurs enseignants qui semblent être impressionnés par le fait que leurs élèves trouvent en algorithmique une autre discipline qui, au lieu d'être une aide pour comprendre les mathématiques chez eux, occasionne une difficulté supplémentaire en mathématiques :

> *« Pour le moment, j'ai l'impression que c'est encore pour eux… ils ont le sentiment que c'est une discipline supplémentaire qu'on leur demande d'apprendre, c'est-à-dire au lieu que ça représente une aide, il y a encore en plus à apprendre et à savoir l'utiliser. Donc, pour eux, on a l'impression que c'est une notion supplémentaire. Nous, pour le moment, on le voit comme ça. Peut-être après, avec du recul, on ne verra plus*

de cette façon-là, mais pour les élèves, je ne sais pas si ça constitue une aide pour le moment. » (A_L)

Selon Anne, l'enseignante, *« quand on enseigne l'algorithmique, c'est comme si on enseigne quelque chose d'autre… Ça pose une vraie difficulté ! »*. En général, l'algorithmique pose au départ de sérieuses difficultés mais, les élèves « bons » en maths se retrouvent plus rapidement que les autres. Néanmoins, les enseignants soulignent qu'il y a quelquefois des exceptions. Certains élèves qui, normalement sont faibles en mathématiques, sont quelquefois plus motivés par l'algorithmique alors que parfois même les plus forts en mathématique ont du mal à comprendre en algorithmique. À ce propos, Serge, l'un des enseignants très motivés en algorithmique, témoigne :

« Oui, les matheux purs, parce qu'eux préfèrent les équations, les vecteurs… Sinon passer leur temps à l'algorithmique, ils n'aiment pas trop, ils ont l'impression que ce ne sont pas des maths. » (S_S).

Les élèves se voient en cours d'informatique lorsqu'ils utilisent une machine tel qu'un ordinateur ou à défaut une calculatrice. Ils disent apprendre l'informatique en mathématiques avec les logiciels tels que ceux de géométrie dynamique ou les tableurs : ils sont très motivés et contents de voir une figure bouger, tourner à l'aide d'un logiciel, etc. et dans ce, ils comprennent mieux les propriétés des figures géométriques dans l'espace. C'est dans ce sens que certains élèves affirment que l'informatique[110] contribue à donner du sens aux mathématiques et ainsi sert à leur compréhension. Par contre, si beaucoup d'élèves ne sont pas motivés par les mathématiques, l'algorithmique, à cause des difficultés conceptuelles qu'elle occasionne chez les élèves, leur ajoute des difficultés supplémentaires dans la compréhension des mathématiques : pour eux, les mathématiques c'est « à la main », précisent les enseignants de mathématiques autant formés en algorithmique que ceux non formés. Ceci semble être un peu contradictoire à ce que.

En général, les élèves bons en mathématiques sont aussi bons en algorithmique. Néanmoins, des cas d'exception existent. Certains exemples peuvent être donnés : certains élèves forts en mathématiques ont des difficultés en algorithmique, comme le témoigne l'enseignante A_L :

110 Il s'agit en général de l'utilisation des logiciels

« Madame, en algo, je suis complètement perdue je ne comprends pas ce qu'on me demande, je ne vois pas ce qu'il faut faire… »

Paradoxalement, d'autres élèves, normalement moins « bons » en mathématiques, sont motivés par l'algorithmique, comme le précise cet autre enseignant (S_S) :

« parfois, les plus forts ont du mal à comprendre ! Oui, les matheux purs, parce qu'eux préfèrent les équations, les vecteurs… Sinon passer leur temps à l'algorithmique, ils n'aiment pas trop, ils ont l'impression que ce ne sont pas des maths. ».

De façon générale, les élèves qui sont à l'aise en algorithmiques sont ceux qui, normalement, sont à l'aise en mathématiques. En général, les enseignants soulignent des effets contrastés de l'algorithmique chez leurs élèves :

1. Pour certains, l'algorithmique contribue à donner du sens aux mathématiques et, par conséquent, à leur compréhension.
2. Pour d'autres, l'algorithmique pose des difficultés supplémentaires : des difficultés conceptuelles à l'instar de celles posées par des mathématiques et particulièrement l'algèbre et la géométrie au collège. Certains ne comprennent pas la place de l'algorithmique en mathématiques : selon eux, les mathématiques, c'est avec le papier/crayon.

2.2. Des savoirs informatiques construits à travers des activités mathématiques

Certains élèves semblent souvent motivés dans l'utilisation des logiciels en mathématiques. En algorithmique, la situation n'est pas toujours pareille. Souvent faite en classe au papier/crayon, l'algorithmique n'est pas perçue comme de l'informatique chez les élèves.
Ne voulant pas chercher par eux-mêmes notamment lorsqu'il est question de construire leur algorithme, ils sont souvent démotivés par cette approche de travail papier/crayon au profit de celui à la machine : beaucoup d'élèves parmi eux ne veulent qu'un programme tout près et tout fait pour l'entrer sur une machine ou une calculatrice. Mais, comme le dit cet enseignant des génies, « *il est difficile de programmer quand on ne sait pas comment fonctionne le matériel : ce qu'on programme, ça va quelque part, il faut donc connaître le matériel »*. Ces difficultés d'utilisation des langages ou des logiciels sont fréquentes chez les élèves : des

séances de prise en main des logiciels sont demandées avant de les utiliser dans les apprentissages.

Les élèves sont motivés quand ils travaillent sur des ordinateurs. Mais, en mathématiques, comme on ne se contente pas de leur donner un programme juste à rentrer sur la machine, mais aussi de mobiliser un raisonnement mathématique pour justifier les résultats observés là, leur enthousiasme tombe. Quand l'exercice est en tout peu difficile tel que celui qui demande une modélisation, ou une construction de programme, certains se découragent tout autant. La situation devient problématique s'il s'agit des élèves en difficultés : les mettre au travail n'est plus possible. Même s'il s'agit d'un travail sur machine, les motivations sont limitées à des choses simples. Mais un problème demandant des étapes un peu plus longues et un peu plus complexes, leur désintérêt sur machine est le même que quand ils travaillent en papier et crayon : même l'ordinateur ne les raccroche plus.
La motivation au travail sur machine en algorithmique semble liée à l'intérêt du feed back direct qu'il lui apporte, ce qui peut permettre des essais et erreurs, souvent non possibles en contexte de travail « à la main ». Comme le précise l'enseignante A_L, c'est pourquoi le travail sur la machine est souvent préféré chez eux :

> *« sur écran, ils bidouillent jusqu'à ce que ça marche sur calculatrice ou Algobox et un feed bac leur permet de réessayer »* (A_L).

La machine devient une sorte de support visuel aidant l'élève à « évaluer » si ce qu'il a fait est correct ou pas. Dans ce contexte, si la calculatrice est centrale en algorithmique, les autres langages, à l'exception d'Algobox, ne sont pas connus en dehors de la spécialité ISN. Même Algobox, il est rarement utilisé comme le soulignent les élèves de la terminale eux-mêmes.

> *« Algobox, on l'a utilisé cette année une seule fois, en début d'année, sur les suites numériques »*

En classe de seconde, les pratiques des élèves se limitent à :

- lire et comprendre un algorithme donné (dire ce que fait un algorithme) ;
- compléter un algorithme à trous donné ;
- exécuter un algorithme avec une ou des valeurs donnée(s) (dire le résultat retourné) sur papier crayon ;

- programmer et exécuter un algorithme donné sur une machine (sur calculatrice ou avec un langage sur ordinateur) ;
- modifier un algorithme donné pour qu'il fasse autre chose ;

a. Un exemple de structure conditionnelle

Au lycée A, les enseignants donnent quelquefois des contrôles communs aux élèves. Durant les années scolaires 2012-2013 et 2013-2014, les contrôles comprenaient des algorithmes portant sur la structure conditionnelle. Nous présentons ici un algorithme de 2012-2013.

ABCD est un parallélogramme ; On donne l'algorithme suivant :

Saisir[111] x_A, y_A, x_B, y_B, x_C, y_C, x_D, y_D
S prend la valeur $(x_C - x_A)^2 + (y_C - y_A)^2$
T prend la valeur $(x_D - x_B)^2 + (y_D - y_B)^2$
  Si S = T
    Alors, *afficher*
      « le parallélogramme…»
    Sinon, *afficher*
      « …… »
  Fin Si

Illustration VI.1: Exemple d'algorithme en devoir commun en classes de seconde au lycée A (Mars, 2012-2013)

Il a été demandé aux élèves de :

- citer la propriété utilisée
- compléter les phrases à trous
- modifier l'algorithme donné pour tester si le parallélogramme ABCD est un losange.

Dans cet algorithme il est clair que c'est la logique qui est visée avec l'instruction conditionnelle « Si… Alors… Sinon ». Les élèves n'y ont pas réussi. Cette instruction alternative semble poser des difficultés aux élèves de seconde, même si elle est fait référence à la discipline mathématique elle-même. De plus, les deux exemples confirment les propos des enseignants selon lesquelles leur pratiques s'intéressant plus à la logique mathématique dans l'enseignement des notions algorithmiques.

111 Les italiques et les soulignements sont faits par nous

De plus, les enseignants confondent dans leur prescription un travail de l'utilisateur de la machine à celui de la machine. Par exemple, les formulations telles que « afficher… » ou « saisir… » qui interviennent dans les activités de contrôle sur table au papier crayon n'ont pas de sens en algorithmique : ces notions font référence à la programmation et à l'utilisation d'interface-machine. Comme l'indiquent Baroux & Prouteau (2013), deux enseignantes et formatrices de mathématiques, ce vocabulaire n'est pas correct pour un travail papier-crayon : il évoque l'utilisation d'une machine avec des périphériques d'entrées et de sortie. Selon elles, une la conséquence de tels vocabulaires est d'*« entretenir dans les esprits des élèves une confusion entre algorithme « idéal » et programme concret »*. Cette confusion entre terminologies et vocabulaires est aussi entretenue dans les manuels scolaires utilisés.

b. Un exemple de boucle « Pour »

Dans le cadre de nouveaux programmes de mathématiques du lycée, deux types de boucles sont essentiellement enseignés : les boucles « Pour » et « Tant que ». De façon générale, la boucle « Tant que » semble être plus difficile chez beaucoup d'élèves par rapport à la boucle « Pour ». Nous sommes en seconde, une classe tenue par une enseignante, ex-formatrice des mathématiques. Elle est motivée pour l'enseignement de l'algorithmique qu'elle déclare vouloir l'aborder non seulement en tant qu'outil pour résoudre des problèmes mathématiques mais aussi comme objet de savoir.

L'exemple présenté est un algorithme d'abord fait en devoir maison avant d'être corrigé en classe : il consiste à déterminer les diviseurs entiers d'un entier naturel non nul n donné. Ici est présentée une production d'un des élèves pour la valeur n= 4 :

Instructions	État des variables		
VARIABLES : *n, c, i* entiers, $n \neq 0$			
ENTRÉES : Saisir *n* INITIALISATION : *c* prend la valeur 0		*n* : 4	*c* : 0
TRAITEMENT : Pour i allant de 1 à n Faire Si i divise n Alors c prend la valeur c+1 FinSi FinPour	*i* : 1	*n* : 4	*c* : 0,25+1
	i : 2	*n* : 4	*c* : 0,5+1
	i : 3	*n* : 4	*c* : 0,75+1
	i : 4	*n* : 4	*c* : 1+1
SORTIES : Afficher *c*.			

Illustration VI.2: Exercice avec une boucle "Pour" et « Si… alors » imbriquées (Classe de 2nde)

On constate que cet élève a des difficultés pour exécuter la boucle *Pour*. Cela se complique davantage quand il s'y imbrique l'instruction conditionnelle *Si… alors*. L'évaluation de la condition constitue une grande difficulté : qu'elle soit vérifiée ou pas, l'affectation « c prend la valeur C+1 » est exécutée. De détail des difficultés des élèves sont développées dans la section 2.4. de ce chapitre.

c. *Un exemple de boucle « Tant que »*

Les textes officiels recommandent un enseignement transversal de l'algorithmique à travers les chapitres mathématiques. Dans les pratiques enseignantes, certains chapitres leur semblent mieux adaptés pour introduire certaines notions. Le chapitre sur les suites est privilégié pour la boucle « Tant que » comme le montre l'illustration VI.3 suivante.

Illustration VI.3: Exercice avec la boucle « Tant que » (classe de Terminale)

Avec ce programme il est demandé de l'expliquer, de le tester et de le modifier pour déterminer un rang 'seuil) à partir duquel la suite Un est supérieur à un nombre donné : 50 ; 1000.

À l'exception de la calculatrice qui accompagne le travail « à main », Algobox est le langage préféré par les enseignants : sa syntaxe est plus proche du langage courant.

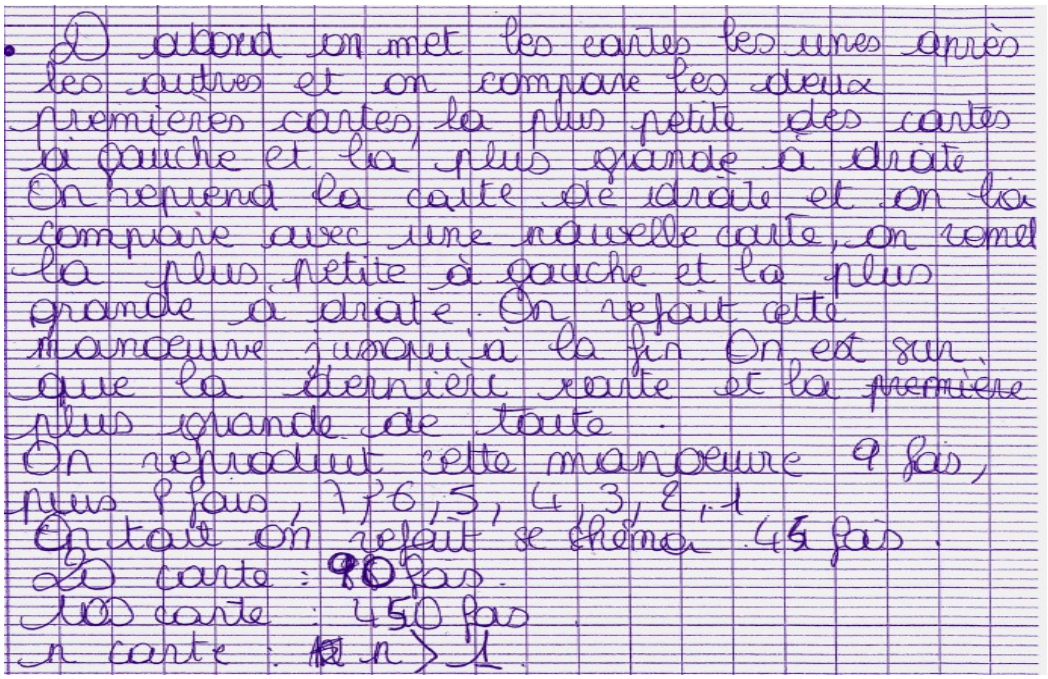
2.3. Algorithmique et approche de groupe

La pédagogie de groupe en algorithmique est utilisée par des enseignants ayant reçu stage de formation. Nous présentons ici deux exemples d'activités proposés par une enseignante à ses élèves : tri de cartes et approximation d'une racine carrée par la méthode de Bombelli.

a. *« Algorithme » de tri de cartes*

L'activité de tri de cartes est proposée aux élèves de seconde pour introduire l'algorithmique. Il leur est demandé de décrire une méthode qui leur a servi au tri. Après un travail individuel,

ils mettent en commun au sein d'un groupe de quatre. La production est le fruit d'un travail collaboratif au cours duquel les membres d'un groupe échangent, discutent, après un travail individuel. Après avoir discuté, enrichi et convenu sur la meilleure méthode, cette dernière est rédigée et remise au professeur. Au cours de cette activité, les élèves ne semblent pas être en séances informatiques comme on peut le constater dans les intitulés des documents produits par deux groupes respectivement : « exercice de français-maths », « mathématiques – français ». Ces intitulés semblent être choisis en référence aux domaines de spécialités des professeurs présents à ladite séance. Le professeur de français était invité pour aider les élèves à la rédaction afin de respecter les consignes. L'illustration VI.4 donne la méthode de tri construite par un groupe d'élèves :



llustration VI.4: Méthode de tri de cartes conçue par un groupe d'élèves de seconde

L'activité est abordée comme un problème dans toute sa généralité sans aucune référence à l'informatique. Pourtant, toutes les notions algorithmiques semblent indirectement intervenues sans s'en rendre compte. En effet, tout au long de cette séance, nulle part n'est écrit ou évoqué ni les mots algorithme ou algorithmique ni les notions de la même famille (variable, instructions, boucles…) :

- une carte représente une **variable** ;
- lors du tri, le remplacement d'une carte par une autre correspond à une **affectation**

- l'opération de tri continue **jusqu'à ce que** toutes les cartes soient triées : autrement dit on commence avec une nouvelle carte pour
- **Tant qu'**il reste encore une carte non triée (non à sa place), le tri continu ;
- **Pour** une carte non encore triée et donc, non encore dans sa place, on réitère le processus de tri.

b. Algorithme de Bombelli et approximation d'une racine carrée

Comme en seconde, nous présentons ici un extrait d'échanges d'un groupe de quatre élèves, de la classe de première, au cours d'un travail de groupe dirigé. Le travail consiste à déterminer, « à la main », des approximations (rationnelles ou décimales) de la racine carrée de 13 en s'inspirant d'une méthode de Bombelli[112]donnée. Trois tâches sont à accomplir :

1. Faire fonctionner « à la main » un algorithme pour des valeurs données en se servant d'un tableau à compléter ;
2. Modifier l'algorithme pour renvoyer une approximation à la nième étape du déroulement de Bombelli ;
3. Écrire un algorithme qui respecte une condition donnée

Nous présentons plutôt pour un groupe suivi, un extrait de leurs interactions au cours de cette activité :

…

Elève 1 : Question suivante : modifier l'algorithme afin qu'il renvoie l'approximation obtenue à la n^{ième} étape du procédé

Élève 2 : A la nième étape, ça veut dire qu'on va continuer mais ça dépasse pas 13. Cette fois, I ne va pas de 1 à 3 mais de 1 à n. Il doit renvoyer 13 en sortie

Élève 3 : Non, mais on ne peut pas le modifier. Ça dépasse 13

Élève 2 : Non, qu'est ce qui te dit que ça dépasse 13 ?

Élève 4 : Attends, on demande à la prof

Élève 1 : Elle est là !

112 Bombelli est un mathématicien du 16^{e} siècle.

Élève 3 : On ne voit pas comment le modifier, vous pouvez nous aider (demande-t-elle à l'observateur) ? Rires ! Est-ce que vous êtes fort en algorithmes ? Rires. C'est dur !

Observateur : Je n'ai pas le droit d'intervenir en classe.

Élève 4 : Madame ! La cinq…

Prof : Est-ce que vous avez compris là ? Je ne vous demande pas de continuer l'algorithme, mais de le modifier parce que même si vous le faites 30 mille fois, il va vous donner quoi ?

Élève 2 : La valeur approximative de racine de 13

Prof : Laquelle exactement ?

Élève 2 : La plus proche

La prof : La première que vous avez trouvée avec Bombelli ?

Élève 2 : Non, la dernière

Élève 1 : La troisième ?

Prof : Oui. Mais vous voyez que le processus pourrait continuer ?

Élève 1 : Oui.

Prof : Moi, telle que je l'ai écrit, c'est pas la peine de faire fonctionner mille fois ! Il va vous donner les mêmes approximations. Moi, ce que je voudrais ce que l'utilisateur puisse dire : moi, je vais la 10^{e}.

Élève 2 : Ça veut dire alors que I va de 1 à 10 ?

Élève 4 : Alors, il y a combien de nombres entiers n ?

Élève 1 : Allant de 1 à n

Prof : Il va être où n ? Dans votre algorithme, si vous dites… Imaginez que vous l'implémentiez sur une machine. La machine va dire, moi, je ne connais pas ce que c'est n. Il va falloir que l'utilisateur l'entre. Alors, vous allez le mettre où ?

Élève 2 : Avant

Prof : Avant, vous le mettrez où n ?

Élève 2 : Avant « Pour », il faut l'initialiser…

Observateur : L'enseignante s'en va vers d'autres groupes. Il y a un petit silence. Les échanges portant sur la résolution d'un problème vont continuer entre élèves.

Les élèves recourent à l'enseignante en dernier lieu après avoir investi toutes leurs compétences. L'intervention de l'enseignant leur donne des orientations permettant d'avancer. Cette pédagogie de groupe semble fructueuse pour les apprentissages des élèves : elle valorise des échanges entre les élèves et l'aide de l'enseignant constitue un déblocage de situations difficiles à surmonter. C'est leur dernier recours.

2.4. Difficultés des élèves

a. Rigueur algorithmique

En algorithmique, les élèves sont confrontés à sa rigueur. Comme le dit G_L, l'un des enseignants au lycée, *« mettre en ordre la pensée, toujours arriver à construire un schéma rigoureux pour pouvoir arriver au résultat demandé »* est l'une des difficultés majeures des élèves. Cette rigueur qui commence par le travail au papier-crayon, s'accentue lors de la programmation et donc sur un langage. La première étape de la rigueur algorithmique réside dans le fait que chaque étape doit être détaillée. Ils ne comprennent pourquoi et l'intérêt de tous ces détails : ces derniers, au lieu d'être une aide pour les élèves, ils leur posent de difficultés. C'est ce qu'exprime manifeste cet élève de seconde : « *C'est trop détaillé. Pourquoi on a mis tout ça jusqu'à End ? (Il pointe du doigt sur un groupe d'instructions sur sa fiche de cours donnée par le prof ?) Je comprends rien du tout… ».*

Selon une des formatrices, enseignante et spécialiste des mathématiques, cette rigueur est fondamentale en algorithmique : un petit détail non ajusté, non corrigé empêche tout le programme de fonctionner. Par son exigence, l'algorithmique, au lieu d'être une aide chez les élèves, constitue une autre difficulté supplémentaire, précisent les enseignants.

b. Difficultés conceptuelles

Au lycée, beaucoup de difficultés des élèves sont liées aux concepts. À l'exception des élèves d'ISN, rares sont les autres qui arrivent à construire des algorithmes. Ces derniers ne sont prescrits qu'en terminale ou par des enseignants particulièrement motivés. Dans la construction d'un algorithme, les difficultés vécues par les élèves sont la gestion des variables, le manque d'anticipation quant aux variables à déclarer, les opérations sur les variables lors de l'exécution d'un algorithme, la syntaxe qui varie selon le langage…

Dans la résolution des problèmes faisant intervenir des boucles, l'essentiel de leurs difficultés réside dans la gestion de la condition, la mise à jour des variables et l'instruction à exécuter pour un résultat donné de la condition.

Trois structures sont enseignées au lycée : la structure conditionnelle « Si… Alors… Sinon ». la boucle « Pour » et « Tant que ». Si la structure conditionnelle n'est pas acquise, elle semble la moins difficile chez les élèves. Une fois la condition vérifiée, les élèves n'ont pas de difficultés avec cette structure : les instructions sont effectuées successivement dans leur ordre de présentation. La difficulté se pose pour eux si la condition n'est pas vérifiée : ils sont perdus.

L'illustration VI.2 montre par exemple que l'élève a connu deux difficultés majeures : le type d'une variable et le test de la condition au sein de la boucle. Déclarée comme entière et initialisée à 0, la variable c est transformée en un réel : il y a une confusion de type. La deuxième difficulté est que l'élève exécute les mêmes instructions selon que le test est vérifié ou pas. La condition semble avoir été modifiée : au lieu de considérer l'instruction « Si i divise n Alors » comme une condition, il est considéré comme une simple division à faire « diviser i par n ».

L'illustration VI.3 concerne la boucle « Tant que ». Si les deux boucles « Pour » et « Tant que » abordées au lycée, posent des difficultés aux élèves, « Tant que » leur est, en général, plus difficile que « Pour ». Sa difficulté réside dans le fait qu'il est difficile de comprendre et évaluer la condition de sa réalisation alors que pour la boucle « Pour », la borne supérieure est connue d'avance : elle est relativement abordable chez les élèves.

3. Conclusion du chapitre

L'analyse des résultats de l'enseignement de l'informatique en mathématiques au lycée scientifique révèle des pratiques et des avis contrastés des effets de l'algorithmique. Si certains enseignants motivés pour cet enseignement affirment le caractère formateur de l'algorithmique en mathématiques surtout du côté logique, d'autres trouvent que l'algorithmique est très peu utilisée en mathématiques pour pouvoir évaluer et juger ses effets.

Les approches utilisées par les enseignants sont choisies parmi celles prescrites et, leurs choix semblent justifiées par leurs compétences et leurs facilités. Si les textes officiels prescrivent un enseignement transversal de l'algorithmique et des savoirs informatiques à la

fois comme objet et outil d'enseignement, les enseignants qui ont bénéficié d'un stage, ont un petit cours théorique d'algorithmique à donner au début avant que l'informatique ne revienne pour servir d'outil dans les autres chapitres mathématiques. Ils ont du recul vis-à-vis des textes officiels par rapport aux autres enseignants. Si certains enseignants passionnés de l'informatique avaient tendance, bien avant la formation, à focaliser leurs pratiques sur la programmation, l'orientation à l'approche papier-crayon lors du stage de formation a influencé un changement de pratiques de classe. Non seulement le stage a décomplexé les enseignants pour adhérer volontiers à cet enseignement, mais aussi il a ramené les enseignants passionnés de la programmation à revoir leurs pratiques orientées programmation pour se focaliser sur l'informatique plus ou moins débranchée. Néanmoins, la pratique n'est pas celle de l'informatique totalement sans ordinateur au sens que lui donne Drot-Delange (2013), mais de son usage modéré notamment pour des illustrations non systématiques.

Les résultats confirment nos hypothèses (Haspekian & Nijimbere, 2012) selon laquelle *« l'approche adoptée quant aux contenus de formation en algorithmique serait alors cruciale pour que les enseignants adhèrent et par suite s'approprient cette partie du programme. Une formation axée sur des aspects techniques, des langages de programmation, sans mise à distance avec l'informatique, ni mise en exergue de ce qui relève de l'activité algorithmique mathématique par rapport à ce qui relève du monde de la programmation et des machines serait a priori moins efficace à « décomplexer » les enseignants. ».*

Par contre, les autres enseignants sans stage semblent le faire par contrainte institutionnelle. Se sentant illégitimes d'aborder des problèmes de programmation, ils évitent de prendre des risques de discrédit devant les élèves : leurs pratiques en informatique sont orientées vers la logique mathématique. Leur choix est orienté par la facilité : *« On enseigne le mieux ce qu'on maîtrise le mieux »* (Arsac, 1992).

Chez les élèves, l'algorithmique est perçue comme un nouveau domaine à découvrir, déconnecté des mathématiques et de l'informatique. Leurs pratiques bien qu'elles aient lieu en mathématiques sont construites en dehors de toute référence disciplinaire.

En conclusion, la référence des élèves reste les pratiques de leurs enseignants : les pratiques des élèves varient en fonction de celles de leurs enseignants et donc selon leurs rapports à

l'informatique. Les pratiques de ce dernier jouent un rôle important pour impulser de modèles aux élèves. Mais, est-ce la formation seule qui influence les pratiques des enseignants et partant celles des élèves ? Nous tenterons de montrer que non dans la discussion générale de cette thèse au Chapitre IX . Ce chapitre s'est occupé de l'informatique en mathématiques. Qu'en est-il de l'informatique de spécialité en ISN ? C'est l'objet du chapitre VII suivant.

Chapitre VII ENSEIGNEMENT ET APPRENTISSAGE DE L'OPTION ISN

Ce chapitre concerne l'option ISN en France. Les résultats présentés sont issus de deux lycées B et C seulement. Le développement du chapitre suivra deux entrées : une entrée par les enseignants d'une part et une autre par les élèves d'autre part. Avant d'aborder l'une ou l'autre de ces deux points, nous rappelons d'abord le questionnement à l'origine de ce chapitre et les hypothèses de recherche formulées.

1. Rappel des questions et hypothèses de recherche

En France, l'enseignement de la spécialité « informatique et les sciences du numérique » (ISN) date de l'année 2012 au lycée. Il est en général assuré par des spécialistes des autres disciplines, recrutés sur dossier parmi les enseignants volontaires. Après recrutement, ces derniers ont suivi une formation de 60 heures en vue de cet enseignement.

Quant aux élèves d'ISN, ils sont débutants en informatique. Exceptés des éléments d'algorithmique de base (algorithme, variable, boucle…) acquis depuis la classe de seconde dans leur cours de mathématiques, ils n'ont connu aucune autre formation à la science informatique au cours de leur scolarité.

L'étude a lieu dans les deux lycées B et C et s'intéresse aux pratiques de deux enseignants d'ISN et aux apprentissages de leurs élèves, essentiellement en projets de programmation. Les projets comptent pour l'épreuve d'ISN au baccalauréat. L'étude ambitionne de répondre à la question générale suivante : comment les enseignants et leurs élèves se sont-ils approprié cette spécialité d'ISN ?

De façon plus spécifique, elle cherche à répondre aux questions suivantes :

Quelles sont les motivations de choix de la spécialité ISN chez les élèves ? Quelles en sont leurs attentes ? Quelles sont les compétences des élèves à résoudre un problème par la programmation sur un ordinateur ? Quelles difficultés rencontrent-ils et pour quelles notions informatiques en particulier ? Quelles stratégies sont mises en œuvre par les élèves pour surmonter ces difficultés ? Comment les enseignants d'ISN préparent et dispensent leurs leçons ? Quels sont les types de tâches prescrites aux élèves et les motivations de leurs choix chez les enseignants ? Quelles difficultés rencontrent-ils ? Quelles sont les différences

de pratiques des enseignants ? En quoi ces pratiques sont-elles en rapport avec leurs spécialités d'origine ?

Pour cet enseignement-apprentissage, trois hypothèses suivantes ont été formulées.

Hypothèse 1 : Les choix de la spécialité ISN et des sujets de projets chez les élèves sont motivés par leurs visées professionnelles futures.

L'informatique est actuellement omniprésente et incontournable dans tous les secteurs socio-professionnels. Nous pensons que les élèves jugent que la spécialisation en informatique est une garantie de l'obtention du travail en fin des études.

Hypothèse 2 : La diversification des situations et des formes de séances d'enseignement est un atout pour l'appropriation des savoirs informatiques chez les élèves

Les concepts informatiques posent des difficultés conceptuelles différentes chez les débutants (Mejias, 1985). Nous supposons qu'en multipliant les situations dans lesquels ces savoirs sont présentés aux élèves mais aussi, les occasions dans lesquelles ces élèves manipulent ces savoirs, on maximise les chances de se les approprier.

Hypothèse 3 : Les pratiques des enseignants d'ISN sont influencées par leur spécialité d'origine.

En effet, nous supposons que le domaine de sa discipline d'origine est mieux maîtrisé que tout autre domaine. Cette maîtrise peut faire que l'enseignant est le plus souvent poussé à utiliser les connaissances et compétences issues de sa spécialité pour orienter son enseignement : des modèles à convoquer et des exemples à donner proviendront le plus de sa spécialité disciplinaire.

Nous présenterons les résultats en deux points principaux: les élèves d'une part et les enseignants d'autre part.

2. Apprentissages des élèves en ISN

Les apprentissages des élèves d'ISN ont été analysés (Annexes 26, 27 et 29). Les principaux résultats sont structurés dans la suite en sept points : les motivations de choix de l'option ISN, l'organisation des élèves en groupes de travail, les connaissances informatiques construites, les compétences à construire un algorithme, les connaissances mathématiques mobilisées, les

formes d'interaction des élèves au travail et enfin, les difficultés vécues et les stratégies utilisées pour les surmontées.

2.1. Motivations contrastées de choix de l'option ISN : entre passion et refuge

Trois principales raisons de choix d'ISN peuvent être relevées : passion, volonté de découverte et refuge.

La première catégorie correspond aux élèves qui ont une passion de l'informatique. Au sein de cette catégorie, deux sous catégories peuvent être distinguées. La première sous-catégorie est celle des élèves à la fois bons en informatique et en mathématiques. Pour ceux-là, voulant faire l'informatique dans l'enseignement supérieur, leur choix entre leurs deux a porté sur l'informatique. La deuxième sous-catégorie est celle d'élèves qui sont passionnés seulement par l'informatique.

Leurs motivations de choix de la spécialité ISN est la même et d'enjeux communs : la poursuite des études supérieures en informatique. Animés d'une volonté d'apprendre mieux l'informatique et bien, leur motivation du choix d'ISN est fondée sur la volonté d'acquisition d'une approche théorique solide pouvant leur permettre d'aller plus loin dans leurs apprentissages de l'informatique.

Étant intéressés par l'informatique depuis le collège, certains d'entre eux possédaient déjà un certain bagage informatique avant leur entrée dans la spécialité ISN. Quelques exemples, tirés de leurs déclarations, peuvent être donnés.

- Au lycée B : un élève avait déjà des pages web alors qu'un autre dit assurer la maintenance du réseau informatique familial ;
- Au lycée C : un élève fait la maintenance informatique à la maison ; un autre sait programmer une calculatrice depuis le collège.

Malgré toutes leurs compétences déclarées en informatiques, certains sont démunis dans leurs approches méthodologiques : ils se sentent limités pour aller plus loin dans leurs apprentissages. L'ISN est donc vue comme un cadre propice d'un apprentissage formel et structuré de l'informatique. Leurs choix d'ISN est, chez les motivées et passionnés de l'informatique, une occasion d'approfondir leurs apprentissages dans un cadre formel. Ici sont présentés quelques-uns de leurs discours à propos :

- *« apprendre (informatique) dans un contexte plus formel. L'info en cours, c'est plus structuré… »* (élève du lycée B)
- *« tout le monde peut faire la licence info sans avoir fait l'ISN, moi ma motivation c'était avant tout de découvrir la programmation. »* (élève du Lycée C) ;
- *« j'avais besoin d'avoir un cours théorique formalisé. Jusque-là, jamais j'ai pas eu de cours, j'ai toujours essayé de me débrouiller tout seul. J'étais obligé de passer par d'autres fonctions… »* (un élève du lycée C)

La majorité d'entre eux ont l'ambition de poursuite de l'enseignement supérieur en informatique. Trois orientations constituent leur rêve : les écoles préparatoires, l'université, l'IUT.

Les passionnés par l'informatique en général et la programmation en particulier, semblent ne pas être satisfaits de la qualité et la quantité des savoirs acquis. Les quelques exemples révèlent qu'ils sont restés sur leur soif.

- *« Avec l'ISN, j'attendais apprendre des choses sérieuses, des fonctions utiles… »* (élève du lycée B) ;
- *« j'aurais voulu avoir l'occasion d'acquérir le C++ plus actuel que C »* (élève du lycée B qui se dit quand même satisfait d'avoir programmé en C un robot LEGO MINDSTORMS NXT) ;
- « *en programmation, on n'a pas fait beaucoup de choses : je m'attendais à plus que ça ! : approfondir le codage, les langages python, C et C++ »* (élève du lycée C) ;
- *« le cours est superficiel ! »* (un élève du lycée C)
- *« peut-être un langage de programmation plus diffusé parce que le python, il n'y a pas beaucoup qui l'utilisent. J'aurais voulu peut-être le Java et le C++, c'est plus utilisés »* (un élève du lycée C) ;

La deuxième catégorie comprend les élèves motivés par l'informatique. Certains parmi eux n'ont pas de connaissances antérieures en informatique. Cette catégorie d'élèves peut être scindée en deux sous-groupes. Le premier sous-groupe comprend des élèves qui sont intéressés par l'informatique et qui attendent la découvrir à travers l'ISN. La découverte leur permettra de « changer de routine »[113] : tourner le dos aux spécialités mathématiques et physique

113 Ceux-là parlent de routine pour dire les autres spécialités déjà existantes qui sont là depuis longtemps et dont ils sont déjà familiers telles que les mathématiques, la physique, les SVT…

considérées comme anciennes[114]. Cette sous-catégorie d'élèves se retrouvent dans les deux lycées concernés. Quelques exemples sont donnés ici :

- *« je ne savais pas que ça existe, mais une fois qu'on m'en a parlé, cette spécialité m'a intéressé plus que les autres »* (élève du lycée B) ;
- *« l'ISN, je ne l'avais jamais entendue parler, contrairement aux maths et à la physique qu'on faisait déjà : tester un truc différent, une matière différente… »* (élève du lycée C) ;
- *« je préfère les mathématiques parce que je préfère largement manipuler les chiffres que les logiciels… Mais la création d'un site tel qu'on voulait le faire ça change en peu la routine : les maths on en a fait depuis le début du lycée. C'est pour cela qu'on a changé un peu pour choisir l'ISN »* (élève du lycée B)

La deuxième sous-catégorie comprend des élèves qui trouvent que l'informatique est incontournable actuellement. Ils y sont intéressés, non pas pour en faire leur spécialité d'étude dans l'enseignement supérieur, mais pour acquérir une culture informatique. Des exemples illustratifs sont donnés :

- *« découvrir, voir ce que c'est l'ISN » ; « en fait, ce n'était pas forcément parce que j'aimais (l'informatique), c'est vraiment pour découvrir l'informatique »* (élève du lycée B)
- *« l'informatique, ce n'est pas quelque chose que je vais faire après à l'université. Je compte faire STAPS[115] »* (un élève du lycée B).
- *« l'informatique c'est l'avenir »… « elle (informatique) va bientôt nous remplacer »* (élève du lycée C) ;

La troisième catégorie est celle d'élèves moins « bons » dans les spécialités mathématiques, physique… Ces élèves, sans être motivés en informatique, préfèrent aller s'essayer dans cette nouvelle spécialité parce qu'ils n'avaient pas d'autres choix. L'enseignante d'ISN dans le lycée C qualifie leur choix d'ISN d'*un choix par défaut*. L'ISN semble avoir été vue comme leur refuge : *« ça fait moins peur le faible coefficient d'ISN (qui est de 2) au bac par rapport aux autres spécialités »*. Derrière ce faible coefficient d'ISN, beaucoup d'élèves de cette catégorie font certains calculs : ils estiment que le petit coefficient de la spécialité ISN ne va

114 La spécialité SVT n'est pas évoquée : elle est minimisée et semble réservée aux élèves plus faibles.
115 Sciences et Techniques des Activités Physiques et Sportives

pas faire beaucoup chuter la moyenne comme le feraient les autres spécialités à grand coefficient, ce dernier pouvant aller jusqu'à 6.

Le choix d'ISN n'est pas le résultat d'une motivation pour l'informatique, mais d'un certain calcul pour la réussite au baccalauréat. Pour ce groupe, quelques illustrations sont données :

- *« Je ne voulais pas la spécialité qui touche à mes coef où j'étais un peu moyens : j'avais peur de n'avoir pas de bac et je me suis dit : je prend la matière qui a un coéf extérieur : la spé informatique ne touche ni au coef de la physique ni au coef des maths ni au coef de la SVT, c'est un coef deux à part… La spé physique a un coef 6 au Bac. Si on prend la spé Physique, ça rajoute deux en plus et du coup, au bac ça sera le coeff. 8. Moi, je ne veux pas que ça touche » (élève du lycée C)*
- *« être sur les ordinateurs » ; « en informatique on est plus libre de faire ce qu'on veut (…) mais il faut être capable de le faire, alors que dans les autres disciplines plus scolaires comme les maths et la physique, on doit plus apprendre des théorèmes, des propriétés du cours… »* (un élève du lycée C).

Certains élèves de cette catégorie sont déçus par cette spécialité et/ou par son enseignement.

La déception est d'abord justifiée par trop de polycopiés donnés : *« J'ai pas trop apprécié le fait. J'ai du mal, souvent des pages de 15 copies, 45 copies, 35, 20… c'est beaucoup trop. C'est comme un nouveau cours en plus, une spécialité. Quand on voit d'autres spécialités, comme la physique, la SVT, on n'a jamais autant de choses à apprendre. De plus c'est un nouveau cours où on doit apprendre autant de choses, autant de polycopiés et qu'on doit le faire tout seul chez nous. ».* Les exigences de ce cours contrastent avec l'attente de plus de liberté chez certains d'entre eux.

La deuxième raison justifiant leur déception est beaucoup de travaux de programmation : *« je pense que cette année, l'erreur de la prof est qu'on a fait trop de programmation en fait la majeure partie de l'année. ».*

Mais, ils y reconnaissent leur responsabilité pour avoir pris à la légère la spécialité. Et du coup, ils commencent à projeter de faire autre chose que l'informatique à l'université. Cette déception est soulignée au lycée C.

Le tableau suivant donne un récapitulatif du rapport des élèves d'ISN à l'informatique dans les deux lycées avant l'entrée dans cette spécialité.

	Lycée		
Etat d'esprit des élèves	**Lycée B**	**Lycée C**	**Total**
Passionnés	3	2	**5**
Motivés	6	5	**11**
Refuge	0	4	**4**
Total	**9**	**11**	**20**

Tableau VII.1: État d'esprit des élèves vis-à-vis de l'informatique avant l'entrée en ISN

Conclusion : Les effectifs mentionnés ici sont fonction du nombre d'élèves interviewés et non des effectifs totaux des classes. Les résultats montrent que la spécialité ISN accueille en général des élèves motivés voire certains passionnés par l'informatique : 100 % (9 sur 9) au lycée B et plus de deux tiers (7 sur 11) au lycée C. Une particularité s'observe au lycée C : environ un tiers d'élèves (4 sur 11) ont choisi ISN non pas parce qu'ils étaient intéressés par cette spécialité mais parce qu'ils espéraient y trouver une spécialité moins contraignante pouvant leur permettre de rehausser la moyenne au bac. Ils ont été séduits par une sorte de campagne de mobilisation pour l'ISN. En effet, l'ancien professeur d'ISN dans ce lycée faisait une publicité pour cette spécialité en cours de mathématiques, la « facilité » de cette spécialité était plus officialisée par ses élèves d'ISN en terminale avec de bonnes notes pour tous au bac.

2.2. Organisation du travail de groupes

Comme toute activité, le contexte de projets nécessite une certaine organisation. Les différents groupes d'élèves semblent organisés de la même façon avec quelques nuances. Le lycée B compte 5 groupes et le travail en projets est strictement organisé en binôme. Le lycée C compte 7 groupes dont 5 formés de 3 élèves chacun, 1 de 2 élèves et 1 monôme : cet élève a préféré rester seul[116]. Au lycée, certains membres du groupe travaillaient sur la même tâche et le travail était partagé avant de rentrer chez eux. Mais l'essentiel du projet a été fait pendant les heures de cours.

116 Deux raisons peuvent justifier son d'être seul. La première est qu'il aurait des difficultés de s'organiser avec son groupe auquel il serait associé : il travaille souvent tard la nuit. Une autre raison semble liée au fait que, ayant déjà de l'expérience en conception de site, il a choisi un sujet en rapport avec une activité commerciale familiale (vente des pièces automobiles).

Au lycée B, chacun des membres du binôme avait une tâche du projet à faire. Une particularité pouvant être soulignée est celui d'un binôme qui a opté d'utiliser, pour un même projet, deux langages différents pour les comparer : *« pour réaliser et compiler un programme sur un microcontrôleur, nous avions au choix deux catégories d'IDE (Environnement de Développement Intégré »*. Le premier permet de concevoir un algorithme sous forme de blocs fonctionnels pour permettre à des utilisateurs non expérimentés de faire fonctionner des systèmes électroniques complexes et de les simuler. Le second choix de langage autorise l'élaboration de l'algorithme sous forme de lignes de code avec un langage particulier : *« pour notre projet, mon camarade Johann a utilisé Flowcode appartenant à la première catégorie et moi, j'ai utilisé Mplab appartenant à la deuxième afin de les comparer. »*. Leur choix de deux langages n'ont pas manqué de conséquence : chacun des membres du binôme a été obligé de faire la totalité du projet.

2.3. Des connaissances informatiques variées construites chez les élèves

Il n'est pas facile de décrire les connaissances acquises dans cette spécialité d'ISN, tellement elles sont nombreuses. En rapport avec les notions informatiques de base, certains élèves d'ISN, de part leurs cours de mathématiques de seconde et première qui ont intégré de l'algorithmique, s'y sont déjà familiarisés : variable, instruction, boucle, algorithme… Les difficultés commencent lorsque ces notions sont utilisées pour résoudre un problème en général et un problème complexe tel qu'un projet. C'est dans ce sens que la spécialité ISN constitue une rupture par rapport aux classes précédentes, en ce qui concerne l'étendue des connaissances acquises notamment les typologies relatives à une variable : *« Jusqu'en terminale, je savais qu'une variable ne peut prendre qu'un chiffre. C'est grâce à ISN que j'ai compris qu'une variable peut être tout… ».*

Les notions passent d'un statut « objet » d'étude à celui d'outil mobilisé dans un projet pour résoudre un problème.

Néanmoins une particularité semble s'observer pour chaque lycée. Dans les deux lycées, les élèves considèrent la classe de seconde comme une année de découverte de l'algorithmique. C'est à partir de la première que de petits algorithmes ont été prescrits. Dès la seconde, les élèves motivés s'y sont mis avec des tutoriels. Par exemple, un des élèves du binôme qui ont programmé le robot affirme avoir appris le langage C tout seul sur le site du zéro[117].

117 C'est un site avec des tutoriels conçus pour s'initier aux différents langages de programmation en autodidacte

Les élèves du lycée C ont profité de l'expérience et de la motivation de leur professeur de mathématiques qui était en même temps professeur d'ISN en terminale pour apprendre la construction de petits algorithmes depuis la classe de première. Mais, contrairement à l'ISN, le champ d'action était petit et les connaissances encore moins larges. En rapport avec la robotique, les élèves du lycée C n'ont eu qu'une séance de 3 heures. Ils déclarent la programmation du robot « facile ». Cette facilité est plutôt justifiée par le fait que le robot LEGO MINDSTORMS NXT programmé ne demande qu'*« un assemblage de fonctions déjà programmées et compilées »*, comme le déclare un des élèves. Il ne s'agit selon eux que d'un déplacement de morceaux à relier.

Les connaissances informatiques utilisées et construites par les élèves d'ISN varient selon les projets et les langages utilisés. En rapport avec les notions informatiques de base, les difficultés des élèves dans les classes antérieures ne sont plus de la même ampleur en terminale. Leur coût cognitif semble avoir diminué progressivement avec les années. Certaines justifications de cette diminution de ce coût peuvent être expliquées par trois raisons : les notions concernées, la fréquence d'usages et les contextes variés.

Nos résultats précédents ont déjà montré que les difficultés des élèves augment en passant de la structure « Si… Alors… Sinon » à celle de « Tant… que » en passant par « Pour ». En ISN, les mêmes notions interviennent mais, dans un champ plus complexe, rendu possible par des problèmes ouverts que les élèves n'ont jamais connus avant. Les élèves sont d'abord perturbés par la complexité de ces problèmes même s'ils sont très formateurs au niveau du raisonnement et la réflexion exigés. Les différentes notions sont fréquemment répétées et manipulées dans des contextes variés et à travers des problèmes différents. D'un contexte à un autre, l'élève vit des contraintes différentes qui occasionnent des erreurs différentes. Grâce à la correction de ses erreurs, l'élève se construit lui-même en construisant ses savoirs de telle sorte qu'il sort du circuit d'apprentissage renforcé dans ses connaissances. En fin de compte, c'est cette reprise de notions d'abord acquises puis utilisées dans des situations diverses qui fait que ces dernières sont davantage familiarisées et appropriées et difficilement oubliables parce qu'ayant fait objet de fréquentes manipulations.

2.4. Compétences des élèves à construire un algorithme

« Programmer a une fin et vise la production d'un programme qui doit être exécutable » (Rouchier, 1990). Elle nécessite, en amont, une activité de construction d'un algorithme à

programmer en vue de résoudre un problème par un ordinateur. Souvent banalisée chez les spécialistes programmeurs, cette étape de construction d'un algorithme est incontournable chez les débutants. Elle leur permet de mobiliser des notions algorithmiques de base : un élève qui n'est pas capable de construire un algorithme pour un problème donné ne peut pas le résoudre par la programmation d'une machine.

Les notions algorithmiques font partie du programme depuis la classe de seconde, dans les deux lycées. Néanmoins, peu d'élèves étaient capables de construire un algorithme avant l'entrée en ISN. Selon leurs déclarations, nous pouvons les classer en trois groupes :

- Un petit groupe d'élèves passionnés par l'informatique et dont certains étaient capables de construire un petit algorithme simple au collège. Ils se retrouvent dans les deux lycées.
- Un groupe d'élèves, motivés par l'informatique et qui ont appris à construire un algorithme avant l'entrée en ISN. Au lycée B, bon nombre d'élèves étaient capables de construire un algorithme qui fonctionne depuis la classe de première : *« c'est depuis la première parce qu'en seconde, c'étaient les premières découvertes »*, précise un élève. Ils ont profité du bon rapport à l'informatique de leur enseignant de mathématiques depuis la seconde : « ça dépend du prof… » affirme un autre élève. Ils se retrouvent aussi dans les deux lycées.
- Un autre groupe d'élèves qui en ont été capables en terminale : *« la construction d'un algorithme c'est grâce à ISN vraiment. () si je n'étais pas en ISN, je pense que je ne serai pas capable de le faire tout simplement. »* (un élève du lycée C). Ils se retrouvent aussi dans les deux lycées.

Le tableau suivant indique la répartition des élèves d'ISN selon l'évolution de leurs compétences dans la construction d'un algorithme.

Lycée	Niveau de scolarité				
	Collège	Seconde	Première	Terminale (ISN)	Total
Lycée B	**2**	**0**	**2**	**5**	**9**
Lycée C	**2**	**0**	**5**	**4**	**11**
Total	**4**	**0**	**7**	**9**	**20**

Tableau VII.2: Compétences des élèves à construire un algorithme selon les niveaux scolaires

Relativement aux acquis des élèves, une différence fondamentale s'observe en première : 2 élèves sur 9 ont appris à construire un algorithme pour le lycée B contre 5 sur 11 au lycée C. Cette différence peut être expliquée par le fait que le professeur d'ISN au lycée C[118] était aussi celui de mathématiques dans certaines autres classes (2nde et 1ère) : les élèves ont été familiarisés à la construction des algorithmes depuis la classe de première. À la fin de la terminale, les élèves de deux lycées disent être tous capables de construire des algorithmes simples.

Il est permis de supposer que les élèves en fin d'ISN sont en général capables de gérer et d'utiliser des notions intervenant dans la construction d'un algorithme. De plus, il semble que la notion de variable qui normalement pose des difficultés conceptuelles aux débutants (Samurçay, 1985), les multiples occasions de sa manipulation à travers divers autres objets informatiques (boucles, algorithmes, programme…) et à travers diverses formes d'apprentissages (cours, TD, TP, Projet…), ont contribué à ce qu'elle soit bien appropriée.
Ces résultats sont confirmés par Andrée Tiberghien et Layal Malkoun dans leur étude portant sur des pratiques d'enseignement et acquisitions des élèves du point de vue du savoir (Tiberghien & Malkoun, 2007). Selon eux, certains éléments de savoir sont d'autant mieux appropriés par les élèves quand ils sont appris dans les savoirs enseignés. Cet apprentissage est mis en relation avec la notion de continuité qui traduit la reprise d'une notion déjà abordée. La diversité de situations d'apprentissages des élèves (cours, TD, TP, mini-projet, projet) constitue des contextes de continuité au sein desquels les notions de savoirs sont réutilisées. Cette notion de continuité semble convenir par rapport à celle de répétition que nous avons précédemment utilisée étant donné que ces notions même si elles sont répétées, interviennent dans des formes et des contextes différents.

2.5. Des connaissances mathématiques élémentaires

Contrairement aux connaissances informatiques, les connaissances mathématiques, autres qu'algorithmiques, construites et utilisées par les élèves restent moins nombreuses. Elles varient selon les projets. Au lycée B, deux projets sur cinq ont fait intervenir des notions élémentaires : coordonnées d'un point, repère, échelle, vitesse, distance, quadrillage, maillage, division euclidienne. Parmi eux, le groupe qui a travaillé sur la programmation du robot Lego a aussi utilisé des notions d'angle, de sens, de rotation. Ce sont des notions relatives aux

118 Le professeur que j'ai observé n'est pas celui qui a enseigné l'ISN et les mathématiques l'année d'avant.

commandes du robot dans ses différents mouvements et actions : « avancer », « reculer », « tourner à gauche », « tourner à droite ». Ce sont des notions mathématiques déjà connues : aucune connaissance mathématique nouvelle n'a été construite.

Au lycée C, les élèves ont préféré des sujets relatifs aux jeux par rapport à ceux proposés par l'enseignant. Excepté le sujet sur la conception d'un site web, 6 sur 7 projets ont fait intervenir des notions mathématiques mais des notions qui restent élémentaires : géométrie euclidienne (coordonnées, distance…) ; arithmétique (division euclidienne, opérations). Les nouvelles notions ont aussi été construites comme les notions de matrices par deux groupes. Les élèves constatent en général que la spécialité ISN est moins exigeante que les autres spécialités en ce qui concerne les mathématiques.

2.6. Des interactions plus circonscrites au groupe ?

L'une des potentialités du projet est le travail en groupe. Au lycée C, l'essentiel du travail en projets est fait au lycée. La salle de classe est en même temps la salle informatique. Concernant les interactions des élèves au travail, pendant plus de deux mois de travail, la collaboration entre les élèves est presque strictement limitée à son groupe. Au sein d'un groupe, chaque élève travaille sur son ordinateur fixe[119]. En cas de difficulté, ils se retrouvent sur un même poste informatique pour s'entraider. Les interactions élargies au travers des différents groupes ont commencé vers la fin des projets. Non seulement, elles étaient moins étendues, mais aussi, elles n'avaient pas la même ampleur avec tous les groupes, certains groupes restant plus isolés que d'autres. Par exemple, pour la bibliothèque PyGame utilisée par presque tous les groupes, cet élève se demande comment ils l'ont tous adoptée : *« PyGame, tout le monde à part Victor l'a utilisé. Je suis allé sur Internet voulant un logiciel pour le jeu avec python : je suis tombé sur PyGame avec une librairie gratuite que j'ai installée sur ma clé USB et après je suis venu au lycée et j'ai entendu les autres dire : « voilà j'ai découvert PyGame… Moi, j'étais étonné parce qu'on a découvert au même moment PyGame sans nous avoir consulté… » »*.

De plus, au sein d'un groupe, c'est l'élève en difficulté qui approche les autres pour demander de l'aide mais, dans le cas présent, c'est au contraire les « bons » élèves qui, après avoir terminé ou suffisamment avancé, se déplacent et se dirigent vers les autres groupes. À ce mo-

119Seuls deux élèves, appartenant à deux groupes différents, apportent leurs propres ordinateurs. Les autres disent avoir au moins un ordinateur fixe chez eux.

ment, des échanges peuvent commencer ou des questions peuvent être posées. Pour certains c'est d'ailleurs à ce moment qu'ils commencent à prendre connaissance des sujets de projets des autres : « vous travaillez sur quoi ? »

Est-ce la présence permanente de l'enseignante qui a limité les échanges et les mouvements des élèves les uns vers les autres ? La présence de l'enseignant semble avoir eu un impact sur la limitation des interactions des élèves. Le fait que cette bibliothèque Pygame a été utilisée par la majorité d'élèves sans avoir été enseignée en classe, laisse supposer d'éventuels échanges en dehors de la classe qui n'ont pas eu lieu en classe en présence du professeur. Peut-être qu'un retrait de temps en temps de l'enseignant loin des élèves pourrait avoir eu un impact sur les interactions des élèves. L'autonomie de ces derniers, caractéristiques d'un travail en projet et en groupe, a manqué, ce qui semble avoir limité les interrelations complémentaires entre eux.

Même si les interactions intergroupes sont intervenues tardivement, elles se sont manifestées sous plusieurs formes : échange verbal pour expliquer ou poser des questions, écrit sur papier avec crayon pour mieux expliciter, taper dans le code de son voisin (corriger des erreurs)… La présence permanente du professeur semble avoir des effets positifs et négatifs. Comme effets positifs, cela a permis à chaque groupe d'avoir le temps suffisant de réfléchir à fond à son projet avant de s'ouvrir aux autres et de s'adresser au professeur quand ils ont des difficultés auxquelles ils ne parviennent pas à remédier. Comme inconvénients, il semble que les échanges menés avec leurs copains sont fructueux étant donné que la confrontation de leurs points de vue se fait sans complexe et d'égal à égal contrairement à ceux de l'enseignant qui peuvent être considérés par les élèves comme des vérités universelles : ceci limite la compréhension.

2.7. Contraintes, difficultés, erreurs et stratégies des élèves

Dans leur apprentissage, les élèves se sont heurtés à un certain nombre de contraintes et de difficultés. Ces dernières varient selon les élèves, selon les groupes d'élèves et selon les tâches à faire. Les contraintes vécues par les élèves sont surtout en rapport avec la compréhension du problème à résoudre et la gestion du temps.

- **Comprendre le problème**

« Comprendre ce qu'il faut faire » pour résoudre le problème donné est une étape incontournable pour les étudiants. Cette étape a pris beaucoup de temps aux élèves. Par exemple au lycée C, malgré les séances de cours consacrées aux projets, ces derniers n'ont pas tous été achevés. Beaucoup d'élèves ont eu des difficultés à comprendre les sujets proposés par la professeure et ont préféré proposer les leurs. Le démarrage n'a pas non plus été facile pour autant : ils ne voyaient pas par où et comment commencer. C'est avec beaucoup d'explications de la professeure orientées dans le sens de la décomposition du problème et de sa modélisation par des schémas que plus tard certains sont parvenus à comprendre ce qu'il fallait faire.

- **Concevoir un algorithme**

Une fois cette précédente étape franchie, une autre difficulté courante est le passage des idées à la construction d'un algorithme pour résoudre le problème. C'est ce que témoigne cet élève du lycée C : « *en algorithmique depuis la classe de seconde, première, ça a toujours été pour moi un truc impossible. Je ne pense avoir beaucoup progressé en ISN. Mais, si je relis le cours et me concentre un peu, je peux faire quelque chose. Mais en algorithmique, j'avoue que j'ai encore du mal »*. Une fois cet algorithme construit, juger de son efficacité est aussi une autre difficulté.

- **Contraintes posées par le langage**

Une autre difficulté est celle liée au langage. Dans ce sens, certains élèves témoignent : *« la difficulté ne vient pas tellement de la boucle « While » utilisée mais elle vient plutôt de Python. Je trouve des erreurs partout. (...) j'écris mon code, je vais ajouter une ligne de code au milieu de mon code là, Python trouve que non, qu'il y a une erreur. Je rajoute une ligne de code en plein milieu de mon code général et il trouve une erreur. Au fait tu enregistres, tu quittes et tu relances et là, il n'y a plus d'erreur. En fait, Python à un certain moment fait des caprices. Ça m'est arrivé dans le programme souvent : il faut enregistrer, quitter, renoncer… Quand il y a des erreurs, il faut des fois… soit c'est une erreur humaine soit c'est python qui veut pas. »*.

Le passage d'un algorithme déjà connu à son implémentation sur un langage utilisé est aussi une autre difficulté. Ce problème de traduction a été vécu surtout par un groupe du lycée C : traduction de l'algorithme de Djikstra en langage Python. Cette difficulté a bloqué la suite du travail : *« on comprenait ce que ça devait faire l'algorithme mais on n'a pas su le traduire*

en Python…. nous, on savait comment ça marchait, Dijsktra, on comprenait très bien comment ça marchait, mais il fallait le traduire en Python et ça, on n'a pas pu faire. On s'est arrêté à l'initialisation, l'algorithme n'était pas là, quoi. ». On a fait quelque chose de copier-coller sur Internet… ». À cause de cette difficulté, les élèves n'ont pas pu avancer. Ils assument leur responsabilité : *« c'est vraiment de notre faute. »*

D'autres difficultés des élèves sont en rapport avec la compilation du code. C'est le cas de l'un des groupes du lycée B : *« mais la difficulté majeure c'est lorsque ça (le code) ne marche pas pour trouver l'erreur. »*. Concernant le langage C, bien qu'utilisé toute l'année en classe, son utilisation en projet n'a pas été pour autant facile chez certains les élèves. Ces derniers lui reprochent la difficulté de son appropriation mais aussi le fait que *« il demande beaucoup de lignes de codes avec le risque de s'y perdre ».*

Face à ces difficultés et contraintes, différentes stratégies ont été mises en œuvre tout au long du processus de résolution des problèmes. La première stratégie a été de faire le travail en deux étapes. Après avoir surmonté la difficulté de compréhension : un travail de planification consistant à la construction de l'architecture globale du code « à la main » sur papier a toujours précédé celui de codage sur la machine. Dans cette construction de l'architecture, le groupe se préoccupe du « choix de fonctions efficaces et de leur succession », comme l'affirme un élève du lycée B.

Une autre difficulté est celle d'encodage, vécue par un seul monôme qui travaillait seul sur la conception d'un site. En général, cet élève affirme ne pas avoir connu de difficultés majeures avec cette spécialité qu'il *« recommande aux autres élèves »* des promotions futures.

- **Difficultés des choix des notions informatiques de base**

Si les notions informatiques de base (variables, boucles…) ont été vues depuis la seconde, des difficultés liées à leurs utilisations efficaces restent posées chez les élèves. : *« la boucle « TANT QUE » est presque la même chose que la boucle « POUR » même si je ne sais pas très bien distinguer les deux »*,

Pour ce groupe, la stratégie utilisée est d'adopter un questionnement en vu d'un choix d'une boucle adéquate : « pourquoi ? Jusqu'à quand ? Dans quel but ça se répète ?…. Selon ce groupe, la boucle « TANT QUE » pose souvent des difficultés aux débutants pour deux raisons. La première raison est liée à la langue, étant donné qu'elle est souvent utilisée en anglais au cours de la programmation. La deuxième raison est liée à la non-transparence de ce

qu'il faut faire : *« on ne voit pas les étapes à faire alors que les autres boucles, on voit vraiment où aller. POUR par exemple chaque étape qu'il faut faire est… »*. Pour un autre élève du lycée C, la difficulté d'utilisation de cette boucle augmente avec le nombre de variables sur lesquelles porte la condition. Malgré les difficultés de choix, la boucle POUR a été rarement utilisée par rapport à TANT QUE dans leur projet. Le groupe C2 du lycée C, sans avoir de difficulté avec elle, justifie la faible utilisation de la boucle POUR par le fait qu'elle ne peut pas être arrêtée en cours d'exécution : *« on ne peut pas arrêter une boucle « FOR », on est obligé d'attendre se répéter un certain nombre de fois. On ne peut pas l'interrompre pour modifier quoi que ce soit. »*.

Pour beaucoup d'élèves d'ISN, ces boucles ne sont pas des boîtes noires : il suffit de s'y familiariser :

> *« une boucle quand tu la découvres, tu ne sais pas comment elle fonctionne. Il faut faire plusieurs fois la boucle pour la découvrir. Comme ça, tu sauras que si tu fais ça, il y aura une erreur et à la fin, tu t'en as approprié…(...) c'est pareil pour tout autre chose : « si tu t'entraînes à faire des mathématiques, tu sais faire des mathématiques ».*

Face à toutes les difficultés liées au contexte de la programmation en projet, ce groupe trouve l'encadrement incontournable : « malgré l'existence des tutoriels, la place de l'enseignant est prépondérante ».

- **Erreurs commises et stratégies utilisées**

Deux grands types d'erreurs sont commis chez les élèves : erreurs de syntaxe et erreur de logiques.

Les principales et multiples erreurs commises par les élèves en programmation sont des **erreurs de syntaxe**. Pour les élèves, la programmation est exigeante : « beaucoup de fonctions à implémenter », « beaucoup de réflexions » à mobiliser… La rigueur de la programmation fait que le code implémenté est très sensible à la moindre erreur commise : Les erreurs de syntaxe sont fréquemment commises par tous les groupes : ponctuation non mise (oubli de point-virgule, ouverture ou fermeture d'une accolade et/ou parenthèse), formule mal écrite… Souvent liées au langage utilisé et par conséquent à la manière dont le code a été construit, il y a un contraste dans le discours des élèves : pour certains, les erreurs de syntaxe sont faciles à

corriger comme l'affirme cet élève B2 du lycée B alors que d'autres affirment avoir des difficultés à les localiser et mais aussi à les corriger. Une stratégie utilisée par un élève du lycée B est la recherche d'erreur par niveau dans le programme : *« normalement ce que je fais, c'est faire petit à petit. C'est d'abord faire un morceau, le tester et voir si ça marche pas. Si ça marche pas, on regarde et on corrige. Ensuite, on avance et si ça ne marche pas, ce qu'il y a d'autres erreurs, on les corrige… ». « Rester ordonné, ne pas s'embrouiller, ne pas mélanger les variables, bien choisir les variables et leurs valeurs »*, est la stratégie qu'il recommande d'utiliser pour les éviter sinon les réduire.

L'autre type d'erreur commise par les élèves est l'**erreur de logique.** Elle est liée à l'utilisation des variables non déclarées ou la mauvaise formulation d'une condition. Identifiée par le compilateur comme « name error », cette erreur intervenait lorsqu'ils utilisaient une variable dont la casse est différente de celle dans laquelle elle a été déclarée. Par exemple, si la variable déclarée est n, au lieu d'appeler n (en minuscule), c'est le N (en majuscules) qui est utilisé plus tard dans le code.

Bien que les langages utilisés dans la programmation soient différents, les méthodes de résolution de problèmes sont en général les mêmes : passer de la phase algorithmique à celle de la programmation, compilation et correction d'erreurs par morceaux de code…

Quant aux difficultés des élèves, elles sont à la fois méthodologiques, conceptuelles et syntaxiques. Si l'étape de comprendre ce qu'il faut faire n'est pas aussi aisée à franchir pour des sujets donnés, il leur était aussi difficile de mettre en œuvre leurs idées. Les stratégies choisies pour les surmonter diffèrent selon les groupes. Pour le cas des polycopiés souvent volumineux donnés en lycée C, les élèves les voient comme une contrainte à leur emploi du temps : il leur faut plus de temps pour les lire. Une analyse détaillée des apprentissages et contraintes des élèves d'ISN est donnée en annexe (Annexe 30). Le point suivant s'intéresse aux pratiques de leurs enseignants.

3. Représentations et pratiques des enseignants d'ISN

Les pratiques professionnelles des enseignants d'ISN des lycées B et C, sont présentées dans cette section de façon comparative. Elles sont structurées en trois points : représentations des enseignants sur des rapports mathématiques/informatique, ensuite leurs pratiques et enfin, une synthèse comparative.

3.1. Rapport mathématiques/informatique : des points de vue contrastés chez les enseignants selon leur discipline d'origine

La spécialité ISN est en général enseignée par des gens issus des spécialités disciplinaires différentes. Bien que sélectionnés sur dossier, ils ont des rapports différents à l'informatique : *« Ça dépend des profs parce qu'il y en a qui avaient déjà des connaissances en informatique et d'autres moins quoi ! »,* précise la professeure du lycée C.

Au lycée C, l'enseignante d'ISN a une double spécialité : sciences des matériaux et mathématiques. En plus d'un doctorat en sciences des matériaux, elle a aussi suivi un master 2 à l'université Paris 7, où elle a eu l'occasion d'approfondir l'aspect théorique de l'algorithmique, précise-t-elle. Elle est enseignante dans trois établissements. En plus du lycée C, elle est aussi professeure de mathématiques dans un collège et dans une école préparatoire aux grandes écoles. Intéressée très tôt par la programmation dont elle a acquis les bases sur le tas depuis le collège, l'informatique « pratique » lui était familière. Au cours de sa formation initiale, elle a eu des cours de programmation en Turbo Pascal.

Le professeur du lycée B est agrégé en génie électrique et informatique industrielle avec une spécialité en électronique. Dans sa formation initiale, deux langages – le C et l'assembleur –, ont été acquis. Parallèlement à l'enseignement de la spécialité ISN, il donne des TP d'informatique en langage C dans un IUT à temps partiel.

Selon ce professeur, il y a une relation de dépendance entre l'informatique et les mathématiques : les mathématiques ont plus besoin de l'informatique que l'informatique a besoin des mathématiques dans leur enseignement et apprentissage. Dans ses pratiques, il fait rarement intervenir des mathématiques dans son enseignement d'ISN, des pratiques justifiées par le fait que « *l'informatique (en ISN) n'a pas besoin de mathématiques pour être enseignée »*. Quant à l'algorithmique, elle ne fait pas partie des mathématiques mais de l'informatique, précise-t-il. Il affirme un petit apport des mathématiques en ISN justifié par son côté logique : *« Moi, j'ai quasiment pas fait des maths ! J'ai fait un tout petit peu de simples conversions en 5 minutes décimal/hexadécimal et d'algorithmie ; c'est de la logique quoi. Dans ce sens-là ce sont des maths, je n'ai pas besoin de reprendre des notions de mathématiques. Peut-être après c'est la programmation un peu plus avancée où on a besoin des opérateurs mathématiques si on veut par exemple faire de la convolution, des choses comme ça… ».*

Par contre, il reconnaît que, non seulement la nécessité de l'informatique en mathématiques est grande, mais souligne aussi les effets de l'informatique dans l'enseignement et l'apprentissage des mathématiques : l'informatique rend les mathématiques plus concrètes et pratiques. D'une part, l'informatique permet des applications concrètes des notions théoriques vues en mathématiques notamment par la programmation et, d'autre part, l'algorithmique, totalement considérée comme faisant partie de l'informatique par ce professeur, aide au développement du raisonnement logique, en tant que compétence très utilisée en mathématiques.

3.2. Pratiques enseignantes en ISN : entre contraintes et débrouillardise ?

a. Rapport des enseignants d'ISN à l'informatique

Les enseignants d'ISN ont été recrutés parmi les spécialistes d'autres domaines et sélectionnés sur dossier. Un stage de formation de 60 heures[120] devait être ensuite suivi en vue de cet enseignement. Néanmoins, cette formation est jugée lacunaire par les bénéficiaires pour deux raisons : son caractère plus théorique que pratique et le manque d'approfondissement.

Cette formation est qualifiée par le professeur du lycée B de très théorique, une orientation qu'il justifie par le statut des formateurs, issus du monde de la recherche : « elle est assurée par des chercheurs de haut niveau ». Selon lui, une telle formation adressée surtout aux spécialistes des mathématiques devrait plus se focaliser sur l'informatique pratique étant donné que c'est ce dont ces derniers ont le plus besoin.

La professeure du lycée C reproche la formation d'être courte et non approfondie. Pour elle, deux chapitres, non approfondis en stage, lui posent particulièrement plus de difficultés : la robotique et les réseaux. Cette insuffisance de formation s'accompagne de conséquences sur ses pratiques pédagogiques. Néanmoins, la formation reçue lui a été profitable et bénéfique pour acquérir du recul pour l'algorithmique et des paradigmes de programmation. Bien qu'elle se soit très tôt intéressée à la pratique de la programmation, elle reconnaît que ses lacunes en informatique théorique n'ont pas malheureusement été comblées par le stage de formation reçu.

Une critique commune faite au programme d'ISN par les deux enseignants est son caractère ambitieux qui exige d'aller très vite et de le « survoler ». Si le professeur d'ISN au lycée B semble n'avoir pas d'autres difficultés que les contraintes d'un programme trop long, celle du lycée C fait recours à la collaboration avec ses collègues dans la préparation des chapitres

120 La formation a lieu pendant toute une année en raison d'une demi-journée par semaine

mentionnés ci-dessus. Cette collaboration leur permet de compléter et d'améliorer leurs pratiques pédagogiques, en bénéficiant chacun des compétences de l'autre, comme elle le mentionne ici dans le cadre du chapitre sur les réseaux informatiques avec son collègue[121] :

> *« J'ai échangé beaucoup avec mon collègue de Charlemagne qui fait ISN. Lui, il est plus un prof de techno, il n'est pas de maths. C'est un ancien prof de techno au collège. (...) Lui, il a pris congé d'un an pour suivre une formation de réseau au CNAM. Donc, si tu veux, lui, il est plus carré en réseaux que moi étant donné qu'il a un an de formation en réseau. Par contre, il n'a pas de formation que j'ai eue en algo. On a échangé un petit peu sur certaines choses. En tout cas, on est tous deux dans des groupes de discussion de l'autre. Je reçois tous les méls du lycée Charlemagne et il reçoit tous les méls du lycée Decour. Nous avons surtout échangé des documents, on discute un peu sur les projets, surtout. Mais, on n'a pas vraiment un travail collaboratif en dehors de ça quoi. C'est en rapport avec les cours : quand moi, j'ai préparé un cours, je te renvoie… »*

Pour des raisons d'efficacité, la stratégie de pratiques collaboratives est de plus en plus pratiquée par les enseignants d'ISN (Breton, 2013) : partage du travail selon leurs domaines de compétences particulièrement lors de la préparation. Malgré cette collaboration, le caractère ambitieux de ce programme reste un reproche principal : la quantité des contenus à enseigner ne correspond pas à son volume horaire, ce qui oblige les enseignants à les survoler pour terminer. Face à cette contrainte qui semble commune à beaucoup d'entre eux, des propositions sont émises par les enseignants : réduire le programme ou augmenter le nombre d'heures. Ces propositions semblent soutenues par d'autres enseignants des autres lycées (Breton, 2013). Le constat qu'« *il est impossible de tout faire* » sur le programme, pousse certains enseignants à adopter d'autres stratégies : juger et séparer les chapitres fondamentaux de ceux à la carte pour que les premiers soient abordés en classe et les autres en projet pour élèves.

b. Des ressources en ligne au centre de la préparation ?

La préparation de ce cours nécessite une combinaison de ressources. La formation reçue pour l'ISN semble n'avoir pas beaucoup contribué dans la préparation. Vue comme très théorique chez l'enseignant du lycée B, elle était focalisée sur certaines parties de l'informatique et en a

121 Ancien professeur de technologie, il a pris un congé d'une année pour suivre une formation en réseau au CNAM

minimisé d'autres, selon celui du lycée C. Chez les deux enseignants, les ressources en ligne semblent privilégiées par rapport aux ressources papier dans leur préparation. Quant au manuel de Dowek et al., (2011) destiné pour cet enseignement, ils affirment ne pas beaucoup l'utiliser : il est moins adapté pour être utilisé avec les élèves, précise un des enseignants. Deux défauts, liés au déséquilibre dans sa conception, lui sont particulièrement reprochés par le professeur du lycée B : d'une part, si ce manuel contient un peu de tout sur les contenus du programme, il a des lacunes sur la partie robotique qui reste moins développée et, d'autre part, contrairement au chapitre sur la robotique, d'autres chapitres sont trop détaillés et trop séparés. Le programme étant très vaste, il est impossible d'achever le programme avec la progression proposée : *« des morceaux doivent être rassemblés et recollés pour constituer une progression pédagogique qui tient la route »*, souligne le professeur d'ISN.
Concernant la préparation, les deux professeurs semblent adopter des pratiques à la fois similaires et différentes. Si tous font recours aux ressources en ligne, le professeur du lycée B a un site privilégié : cours d'un autre professeur. Avec la programmation en langage C, il se sert en ISN des documents qu'il utilise à l'IUT. Quant au professeur du lycée C, en plus de ressources en ligne, des échanges collaboratifs avec des collègues sont privilégiés : *« il (son collègue) a pris un congé d'un an de formation pour suivre une formation réseau au CNAM. Donc, si tu veux, lui en réseau il est beaucoup plus carré que moi étant donné qu'il a un an de formation en réseau. Par contre, il n'a pas la formation que j'ai eue en algo. On a échangé en petit peu sur certaines choses… ».*

Les difficultés posées par la préparation varient selon les parties du cours concernées. S'il y a recours à la collaboration des enseignants collègues, ceci reste limité sur les parties du programme qui posent plus de difficultés : les réseaux et la robotique pour cette enseignante du lycée C alors que son collègue a des difficultés en algorithmique. Le programme étant ambitieux comme le soulignent les enseignants, des stratégies sont adoptées dans la préparation : un polycopié est donné à chaque élève pour pouvoir approfondir son cours chez lui. Cette activité reste exigeante selon l'enseignante du lycée C : *« ça me prend beaucoup de temps »*. Si cette difficulté est liée au manque de manuels, elle est accentuée par le manque de formation suffisante en informatique et surtout dans certains chapitres, ce qui ne manque pas de conséquences sur ses pratiques pédagogiques : *« en robotique moi, j'ai pas eu de formation. (…) J'ai fait de l'informatique comme programmation mais pas la robotique. (…)*

Ça m'a demandé un investissement énorme et j'étais incapable d'assurer le cours de robotique en début d'année tellement c'était la bête noire. »

c. Des pratiques enseignantes influencées par leur spécialité ?

En dressant un parallélisme comparatif entre leurs pratiques enseignantes en ISN, nous constatons qu'il y a des différences et des ressemblances.

- **Choix de langages**

Si les contenus enseignés sont les mêmes, les choix effectués concernant les langages varient selon les professeurs et, par conséquent selon les lycées. Les choix sont présentés dans le tableau suivant :

	Langages et technologies utilisés selon les lycées	
Nom du lycée	**Langage**	**Technologie**
Lycée B	C ; NXC	Robot MINDSTORMS NXC[122]
Lycée C	Python ; NXT	Robot MINDSTORMS NXT

Tableau VII.3: Langages et technologies utilisés selon le lycée

Rappelons qu'aucun sujet portant sur un robot n'a été proposé en projet au lycée C[123].

- **Distinguer les phases « algorithmique » et « programmation »**

Dans leurs pratiques, les deux enseignants semblent avoir le pari de faire distinguer à leurs élèves les deux phases « algorithmique » et « programmation » lors de la résolution des problèmes. Par exemple, elles sont abordées distinctement pour initier les élèves comment passer de la première phase à la deuxième. L'enseignant du lycée B justifie ici le pourquoi de ses pratiques : dissocier les deux phases permet de dissocier et de limiter les difficultés chez les élèves. Cette pratique est d'autant plus importante chez les élèves pour des problèmes complexes comme ceux portant sur des projets devant respecter un cahier de charges, précise-t-il : d'une part, comment mettre en œuvre un cahier de charges, qui est du ressort de la compétence algorithmique et, d'autre côté, comment l'implémenter qui est celle des compétences de la programmation. Selon lui, mettre en évidence des liens entre l'algorithmique et la programmation en tant que deux phases différentes mais liées est importante dans son enseignement : le langage de programmation utilisé est susceptible d'évoluer ou de changer alors qu'en soit, l'algorithme lui, ne change pas. D'où l'importance

122 Not Exactly C, un langage proche du C
123 La programmation du robot en cours a été faite pendant la dernière séance de l'année

de connaître l'algorithme avant l'implémentation, ce qui suppose sa construction d'abord. Une stratégie utilisée par le professeur du lycée B est de donner souvent un problème qui exige les deux compétences :

> *« Je fais les deux l'un après l'autre. En fait en ISN, on a très très peu de temps. On n'a que deux heures par semaine. Il faut aller super vite et il faut que ça plaise aux élèves. Moi, ce que je fais ce qu'en début de séance, en moins d'une demi-heure, je leur fais un petit cours, mais ils ont un polycopié, et puis à gauche ils ont un algorithme et à droite ils ont la machine pour implémenter ».*

- **Des choix différents d'activités**

Bien qu'ils aient la même ambition d'amener leurs élèves à distinguer les deux phases précédentes, les choix effectués de tâches pour y arriver diffèrent.

Au lycée C, les exercices de départ sont ceux des années antérieures : algorithmes construits puis ceux à construire après avoir révisé les notions algorithmiques (variables, boucle, algorithme…). Par contre, l'enseignant du lycée B dit ne pas passer du temps sur ces notions algorithmiques qu'il considère comme suffisamment approfondies ou pouvant être approfondies avec les professeurs de mathématiques. Lui, il les faire intervenir implicitement dans les tâches proposées aux élèves.

Malgré la centration sur la programmation dans leurs pratiques d'enseignement, les enseignants d'ISN précisent qu'il n'est pas question de former de programmeurs : il s'agit de les familiariser à l'informatique et de tenter de susciter l'envie de l'informatique par tous les moyens.

- **Plus de temps de pratique avec de légères différences**

À voir les pratiques des enseignants, l'ISN est une spécialité plus pratique que théorique. En raison d'une séance de deux heures par semaine, le professeur du lycée B dit passer environ un quart du temps à la partie théorique et le reste du temps à la pratique. Une légère différence apparaît au lycée C avec un peu plus de temps à la théorie qu'au lycée B. La variabilité des pratiques enseignantes est plus liée à leurs rapports aux contenus abordés. Deux exemples peuvent être donnés. Face au caractère ambitieux du programme, le professeur du lycée B dit survoler tous les chapitres en donnant seulement l'essentiel. Il déclare ne pas passer beaucoup de temps sur ces notions algorithmiques (variable, boucle…) considérées comme suffisamment enseignées par ses collègues de mathématiques. Il passe

directement aux problèmes à résoudre.

La professeure du lycée C commence par ces notions algorithmiques. Elle approfondit plus certains chapitres par rapport à d'autres. Le premier exemple est le chapitre sur la robotique qui semble avoir été banalisé[124]. Elle ne se voyait pas à la hauteur en début d'année. Un deuxième exemple est celui du chapitre *réseau.* Ce dernier a été plus théorique que pratique. La comparaison des tableaux du point montre des différences en programmation réseau chez les deux enseignants.

d. Des formes de séances d'enseignement diversifiées

Une des spécificités de l'enseignement d'ISN est la diversité des formes de séances : cours théorique, TD, TP, mini-projet et projet. Au lycée C où nous avons participé aux débats, les parties du programme – « aspects sociétaux », réseaux – étaient exposées par les élèves en binôme pendant une vingtaine de minutes, suivi de débats dirigés par le professeur.

L'activité portant sur la programmation de la carte Arduino a été abordée dans la formation continue. Vue comme une activité « pauvre » par l'enseignante du lycée C, celui du lycée B a fait d'Arduino un savoir de base à acquérir qui a fait l'objet d'un mini-projet de ses élèves. Beaucoup de connaissances relatives aux logiciels et matériels ont été appropriées lors de ce mini-projet :

- **Des logiciels** : Notepad++, explorateurs (Firefox, Chrome, Explorer…), HyperTerminal (comme Windows par exemple pour configurer la liaison RS232) ; logiciel Arduino (pour programmer la station météo) ; Wampserveur (pour configurer un serveur) ; Wireshark pour visualiser les trames)…
- **Des matériels** : Câbles RS232 femelle/femelle pour relier deux PC (1 pour 2 PC) ; Oscilloscope et connecteurs simples pour liaison ; Station météo (carte Arduino + carte Shield Ethernet + Capteurs de température et d'humidité) et Hub et câble RJ45.

Au Lycée C, une série de sujets d'exposés pour les élèves était organisée en binôme. : numérisation des images, numérisation des sons, compressions des données, persistance de l'information, aspects légaux, fonctions récursives, transmission point à point, les langages HTML, PHP et MySQ, réseau et enfin, sécurisation de données et cryptographie. Après ces exposés, certains thèmes étaient repris par l'enseignante pour faire l'objet de cours, de TD et/ou de TP.

124 Si le chapitre comme la robotique semblait motiver les élèves, il a fait objet de cours le dernier jour de l'année seulement. Et contrairement au lycée B, aucun projet pour les élèves ne concernait la robotique. Vu cette motivation des élèves, la professeure prévoyait de commencer par ça l'année d'après

Pour la professeure du lycée C, les liens entre l'informatique et les mathématiques et, en particulier entre l'algorithmique et les mathématiques, sont plus affirmés. Selon elle, les spécialistes des mathématiques manquent de recul en informatique et vont plus se focaliser sur l'aspect « outil » de l'algorithmique dans leur enseignement des mathématiques. Elle précise que l'algorithmique ne devrait pas seulement être vue comme un « outil » au service des mathématiques, mais aussi comme un « objet » en lui-même d'étude. Dans ses pratiques, elle aborde l'aspect « objet » de l'algorithmique par l'étude de la complexité[125] des algorithmes avec des problèmes sur la formule de Pascal, la fonction factorielle, fonction exponentielle…

Avec des algorithmes mathématiques, l'aspect objet de l'algorithmique est par la suite approfondi. L'enseignante ne s'arrête pas au fait qu'un programme traduisant un algorithme marche mais, elle amène les élèves à se poser des questions sur son efficacité, sur ce qui prouve qu'il va marcher, renvoyer la bonne réponse, sur les preuves de correction, de terminaison, sur la complexité… Bien que les situations-problèmes proposées aux élèves restent liées aux mathématiques, l'algorithmique est, dans ses pratiques, abordée à la fois en tant qu' « outil » et « objet » d'enseignement. Ceci distingue ses pratiques en algorithmique avec celles des enseignants de mathématiques. La focalisation de ses choix sur des problèmes mathématiques semble être justifiée par ses rapports avec la spécialité mathématique.

3.3. Synthèse comparative

Étant la première année d'enseignement de cette spécialité, les deux enseignants n'ont pas encore stabilisé leur pratique et cherchent à les améliorer. Mais les insatisfactions ne sont pas de même degré. Le professeur du lycée B est satisfait de sa progression pour des prestations de première année. Un petit changement de pratiques envisagé concerne l'addition bit à bit, tout en restant dans la même ligne directrice pour ses pratiques. Prévoyant de commencer avec la programmation en langage HTML, il projette d'augmenter le temps à passer sur la programmation en langage C dans le cas des projets.

Par contre, l'enseignante C ne semble pas satisfaite. Elle projette de changer de pratiques l'année d'après : par exemple commencer par la robotique pour stimuler la motivation des élèves. Comme stratégies pour surmonter ses difficultés en robotique, une collaboration sera engagée entre elle et des professeurs d'université notamment pour avoir des activités

125 Selon le Wikipdéia, la complexité est une notion mathématique et généralement de l'informatique théorique. Elle s'intéresse à la quantité de ressources (temps et espace mémoire) que nécessite la résolution d'un problème par exécution d'un algorithme.

adaptées ou des idées de projets avec le logiciel NXT. Enfin, elle projette de commencer les projets des élèves très tôt avant les vacances de fin d'année (de Noël).

La synthèse des pratiques comparatives de ces deux enseignants d'ISN respectivement dans les lycées B (spécialiste d'informatique) et C (spécialité des mathématiques) est présentée en annexe (Annexe 32).

4. Discussion et conclusion

Rappelons, avant d'entrer dans la discussion des résultats, les hypothèses initialement formulées pour ce chapitre.

Hypothèse 1 : Les choix de la spécialité ISN et des sujets de projets chez les élèves sont motivés par leurs visées professionnelles futures.

Hypothèse 2 : La diversification des situations et des formes de séances d'enseignement favorise l'appropriation des savoirs informatiques chez les élèves.

Hypothèse 3 : Les pratiques des enseignants d'ISN sont influencées par leur spécialité d'origine.

4.1. L'ISN, une spécialité pour tous avec une faible représentativité féminine

Nous allons montrer que la première hypothèse n'est pas vérifiée. Bien que notre échantillon soit trop petit pour tirer des conclusions qui s'imposent, le constat est que l'ISN est une spécialité avec une faible représentativité féminine. Ce résultat est confirmé par d'autres travaux récents (Breton, 2013 ; Drot-Delange & More, 2013). Les stéréotypes de sexe semblent expliquer cette situation comme cet élève du lycée B le croit : *« Je pense que l'informatique est plus quelque chose pour les garçons que pour les filles, quoi. C'est dur de dire ça mais… c'est plus nous les hommes qui préférons travailler dans la pratique de l'informatique, etc. ».* Notre étude n'a porté que sur des garçons, ce qui reste une limite d'avoir des points de vue de filles. Drot-Delange et More (2013) montrent que les élèves qui suivent l'option ISN n'adhèrent pas aux stéréotypes de sexe relativement à l'informatique, contrairement aux autres élèves de terminale qui ne la suit pas.

L'ISN n'est pas une spécialité réservée à un public d'élèves avec des qualités spécifiques. Choisie en terminale, elle réunit des élèves aux profils différents, issus des spécialités diffé-

rentes allant des mathématiques et physique à la SVT en passant par les sciences de l'ingénieur (SI) mais, avec une faible proportion des SVT. Ces derniers, par rapport à ceux des autres spécialités et particulièrement les SI, doivent fournir plus d'efforts : beaucoup de notions informatiques ont été déjà vues en SI dans leurs cours de spécialité. Une explication possible de la faible représentation féminine en ISN peut être liée au fait que cette spécialité ne recrute pas beaucoup en SVT, la première spécialité à avoir une forte représentation de filles par rapport aux garçons[126]. La prédominance du public masculin dans la spécialité ISN est aussi confirmée par d'autres recherches (Alayrangues et al., 2013). Contrairement au recrutement des années quatre-vingts pour l'option informatique qui était sélective (Baudé, 2010a), l'ISN procède par une approche différente : l'entrée est libre, les élèves sont encouragés à faire cette spécialité.

Ces publics ont des visées différentes qui vont d'un enjeu de spécialisation en informatique pour certains et celui de l'acquisition d'une simple culture informatique. Une partie a trouvé en cette nouvelle spécialité une opportunité pour ne pas rater leur baccalauréat, ou au mieux d'augmenter leur moyenne, une possibilité non offerte en suivant les autres spécialités mathématique et physique déjà existantes. Ceci a d'ailleurs été un argument de certains enseignants pour rassurer les élèves qui avaient des réticences et peurs de ne pas réussir (Breton, 2013) : *« personne ne peut avoir des lacunes d'une spécialité nouvelle »,* dit-il à ses élèves.

La spécialité ISN est convoitée par beaucoup d'élèves. Ces derniers ont des pratiques contrastées. Les passionnés de l'informatique sont demandeurs de plus de programmation avec une diversification de langages, alors que ceux qui ont fait l'ISN par défaut se retrouvent totalement débordés.

Chez les motivés, elle est vue comme une filière qui véhicule une nouvelle approche d'apprentissage mais aussi, semble avoir un statut unique par rapport aux autres spécialités :

> *« On ne peut pas arriver en S et comprendre les maths, la physique, la chimie, les SVT sans avoir rien fait avant. Il faut toujours les avoir suivies au collège et au lycée, alors qu'en informatique tout le monde peut rentrer : c'est une autre logique, c'est une autre science. C'est pas la même chose, quoi. On peut aller en ISN, donc en informatique à partir de n'importe quelle classe, à partir de la première, de la seconde… (L'informatique) c'est une science dont on n'a pas besoin d'en avoir appris avant, donc de*

126 https ://www2.ac-lyon.fr/enseigne/math/IMG/pdf/isn-2013.pdf, document consulté le 27 février 2015

connaître des bases alors que la physique, les maths… les sciences générales, un élève de seconde ne peut pas aller en terminale, il ne comprendra rien parce que c'est trop pointu. » (élève du lycée C).

Malgré leurs motivations contrastées pour suivre l'ISN, les élèves ne semblent pas avoir des visées professionnelles, du moins après le baccalauréat: l'enjeu est la poursuite des études en enseignement supérieur. Quant aux choix de sujets de projets, le jeu et la facilité du sujet étaient au centre des motivations mais avec une prédominance de la motivation orientée vers le jeu. La première hypothèse n'est pas vérifiée.

Les enseignants soulignent un conflit de territoire entre l'ISN et les autres spécialités scientifiques. Selon eux, l'ISN recrute au détriment des autres spécialités en général et celles des sciences dures en particulier : mathématiques et physique. Leurs propos sont confirmés par les élèves d'ISN qui, ayant la double compétence, mathématique et informatique, préfèrent faire l'ISN. Ceci peut justifier la tendance à la masculinisation de l'ISN étant donné qu'elle est plus choisie par ceux qui auraient été en sciences dures, où l'on trouve déjà plus de garçons. Certains enseignants de mathématiques s'inquiètent que la spécialité mathématique généralement suivie en vue des écoles préparatoires, risque de manquer d'élèves avec l'introduction de l'informatique dans les écoles préparatoires. Ceci semble confirmé par nos résultats : les élèves ayant de bons rapports à la fois aux mathématiques et à l'informatique préfèrent faire l'ISN.

4.2. L'ISN : un atout pour les apprentissages des élèves?

Justifions la véracité de la deuxième hypothèse. L'enseignement de la spécialité informatique et science du numérique (ISN) est un enseignement plus pratique que théorique. Si aucun de deux enseignants en question n'est spécialiste de l'informatique, celui du lycée B a une plus grande proximité avec l'informatique qu'avec les mathématiques. C'est le contraire pour l'enseignante C. Interrogé sur les difficultés que peuvent avoir les spécialistes des mathématiques dans l'enseignement d'ISN, le professeur B évoque des difficultés à programmer sur des choses concrètes telles que les robots, la visualisation des trames, la réalisation des serveurs météo… Selon lui, ils vont plus focaliser leurs pratiques sur des graphismes et, pour éviter des difficultés liées à l'informatique pratique, ils préfèrent carrément rester au niveau de l'informatique théorique.

Dans l'enseignement d'ISN, les élèves ont différentes occasions de rencontrer les notions informatiques à apprendre dans diverses formes de séances : cours théoriques, TD, TP, exposés préparés en mini-projets et des projets[127]. Si une notion conceptuellement difficile à saisir demande beaucoup de temps pour appropriation, l'ISN semble servir de cadre bien adapté pour ça avec la diversité de problèmes à résoudre prescrits. En effet, les différentes formes de séances d'apprentissages sous lesquelles les savoirs sont construits dans divers contextes tout au long de l'année, sont complétées par le travail en projet qui prescrit des problèmes plus ouverts et complexes. Ces occasions permettent de mobiliser des savoirs acquis de telle sorte que leur coût cognitif initialement élevé, va progressivement régresser avec la familiarité des notions. Il semble que les problèmes variés choisis dans ces contextes divers vont successivement faire intervenir les différents savoirs sous diverses formes de telle sorte que progressivement leur appropriation finie par être réelle.

De plus, l'approche de projets utilisée pour cet enseignement est un atout pour les apprentissages des élèves. L'une des caractéristiques de l'enseignement-apprentissage de la spécialité ISN est la centration sur l'approche par projets. Avec cette approche, les problèmes prescrits sont ouverts et nécessitent un vaste champ de raisonnement et une grande réflexion lors de la construction des savoirs. Les élèves soulignent des différences de contextes d'apprentissage de l'informatique en ISN et en mathématiques, en se basant sur des problèmes posés :

> *« pour les mathématiques, on vous donne tout ce que vous faites alors qu'en informatique, c'est un peu plus autonome. Si on vous donne un sujet, on peut passer par plusieurs moyens. () Par exemple, on a une question comme créer un algorithme pour que l'écran affiche le résultat X<5 », donc on sait qu'on va mettre TANT QUE et on va mettre X<5 alors qu'en ISN, on donne le sujet « faire en sorte que le robot tourne quand il voit un obstacle ». Par exemple là, il n'y a pas explicitement le TANT QUE. En maths, il y avait le « TANT QUE x<5, alors qu'en ISN c'est vague. C'est nous qui choisissions exactement quelle boucle mettre… »,* précise un élève du lycée B.

La qualité des problèmes posés pour les apprentissages justifie l'efficacité de cette approche : ce sont des problèmes plus ouverts et complexes. C'est cette ouverture des problèmes posés en ISN qui laisse à l'élève un large cadre de réflexion pour susciter une ouverture d'esprit. Ce cadre créé par l'approche par projets en ISN est propice aux apprentissages fructueux :

127 Ils valent pour l'épreuve du baccalauréat

face au non-succès des choix effectués, de nouveaux choix seront investis et interrogés. Le raisonnement de l'élève et ses essais successifs sont formateurs et par conséquent déterminants pour les apprentissages. Dans le cas de la programmation informatique par exemple, des erreurs mises en évidence par la compilation doivent être absolument corrigées pour la suite de la résolution du problème posé. C'est par des essais et erreurs successifs que finalement la compilation sera réalisée : les erreurs et les solutions apportées seront des repères monumentaux dans les pratiques futures de la programmation chez l'élève.

Pour expliquer les potentialités de l'approche par projets, Andrade-Barroso et al., (2013) se basent sur des échanges et des interactions qui sont occasionnés par un travail de groupes dans la résolution des problèmes donnés. Sans se restreindre à une quelconque spécificité disciplinaire, Philippe Perrenoud (1999) présente et regroupe en dix points, les potentialités de la démarche par projets dans le contexte scolaire, des points aussi identifiés avec les projets d'élèves d'ISN. C'est notamment la mobilisation des savoirs et des savoir-faire déjà acquis, la découverte de nouveaux savoirs et de nouveaux univers orientés vers la sensibilisation ou la motivation, la confrontation à des obstacles nécessitant de nouveaux apprentissages pour les surmonter. L'hypothèse 2 semble vérifiée.

4.3. Effets de la spécialité d'origine sur les pratiques enseignantes ?

Les enseignants de la spécialité ISN sont tous volontaires pour cet enseignement et, sont issus des différents domaines scientifiques. Les choix de tâches prescrites à leurs élèves semblent être largement influencés par leur spécialité d'origine : l'enseignant B d'ISN, ayant de bons rapports à l'informatique, semble minimiser des activités orientées mathématiques alors que les mêmes choix sont faits par celle du lycée C, spécialiste des mathématiques. Plus à l'aise avec cette partie algorithmique acquise en master et, reprise en stage de formation, elle consacre plus de temps à l'algorithmique et programmation qu'à d'autres chapitres d'ISN. Cette pratique semble justifiée par le fait que l'algorithmique du programme est plus proche de sa spécialité d'origine, les mathématiques (Tort et al., 2013).

Par ailleurs, les résultats montrent aussi des différences de pratiques chez les enseignants. L'absence de sujets de projets portant sur la programmation des objets concrets tels que ceux liés à la robotique peut être justifiée par le manque de familiarité de l'enseignante avec cette technologie alors que de tels projets sont abordés chez l'enseignant B, de spécialité plus proche de l'informatique. D'autre part, la prédominance des sujets plus orientés vers les ma-

thématiques chez le professeur C alors ces sujets sont pratiquement absents chez le professeur B semble témoigner de la nuance des pratiques enseignantes en fonction de la spécificité disciplinaire d'origine.

Au lycée C, les pratiques de l'enseignante sont souvent accompagnées de plus de difficultés dès la phase de préparation de la leçon à son enseignement jusqu'à sembler regretter pour avoir accepté ce cours. Si cette enseignante dit s'être intéressée très tôt, toute seule, à l'informatique, il est évident que l'informatique étant un domaine vaste, toutes les parties composites de la science informatique n'ont pas été maîtrisées au même niveau, ce qui est d'ailleurs normal : il y a une différence entre les apprentissages en autodidacte et ceux acquis à l'école. Contrairement à celui du lycée B, spécialiste de la discipline, celle-ci se heurte à des difficultés qui ne peuvent être évitées par la formation continue courte. Par exemple le manque de recul, vraisemblablement la principale difficulté vécue, peut être relié à un manque de formation initiale en informatique, malgré son intérêt affiché depuis longtemps pour l'informatique, complété par la formation continue trop courte reçue. Ceci pourrait expliquer le peu de temps consacré aux chapitres, *robotique* et *réseau*, alors que d'autres comme *algorithmique et programmation*, plus familiers ont tendance à prendre plus de temps. Ces résultats sur des pratiques différentes par des enseignants de spécialité respectivement mathématiques et informatique, sont confirmés par le rapport de la SIF à propos de la formation des enseignants de cette spécialité (Alayrangues et al., 2013).

L'hypothèse 3 semble ainsi vérifiée.

Cependant, en conclusion, sans être spécialiste de l'informatique, les difficultés de l'enseignante C diffèrent d'un domaine informatique à l'autre. Elles peuvent différent d'un enseignant à l'autre. Une formation ciblée, bien conduite et approfondie peut apporter des résultats satisfaisants sans être spécialiste du domaine.

4.4. Conclusion : nécessité des enseignants spécialisés en informatique

Les pratiques précédemment observées sont justifiées par la volonté et la motivation des enseignants. Bien que motivés, investis et engagés, les pratiques des non-spécialistes de l'informatique restent limitées. Une formation plus approfondie devrait leur être donnée pour avoir plus de recul et faire face au programme ambitieux d'ISN afin de pérenniser cette spécialité. Une formation approfondie doublement orientée à la fois technique et théorique est indispensable pour acquérir plus de compétences et de recul. Dans le contexte de manque d'ensei-

gnants spécialistes d'informatique, la formation continue plus approfondie des enseignants volontaires, est nécessaire, à long terme, pour stabiliser cette spécialité. Dans l'entre temps, il est judicieux que les enseignants déjà en enseignement d'ISN continuent chaque année à bénéficier des stages de formation sur des parties du programme de leurs choix pour avoir suffisamment de recul et des compétences nécessaires. Ceci contraste avec la façon dont des formations actuelles en informatique sont menées à l'endroit des professeurs dont leur première compétence n'est pas celle de l'informatique (Alayrangues et al., 2013) : caractérisées par de grandes disparités selon les académies, ces formations ne sont pas de nature à mettre l'équité dans la formation en informatique des élèves. La spécialité ISN trouve une grande adhésion parmi les élèves des filières scientifiques du lycée. Si les motivations de cette adhésion massive restent multiples chez eux, le fait que cette spécialité n'est pas sélective est une singulière qui fait d'ISN une spécialité à disposition des élèves de niveaux différents.

Après ce chapitre portant sur l'enseignement de l'option ISN pour débutants au lycée général, nous nous intéressons dans le chapitre IX suivant à son enseignement chez les débutants de l'université en licence informatique.

Chapitre VIII NIVEAU LICENCE : Apprendre l'informatique par la programmation des robots

Comme il a été indiqué plus haut, nous avons souhaité étudier ce qu'il advenait en début d'enseignement supérieur où les étudiants de licence (L2 et L3), comme ceux de lycée, sont aussi débutants en informatique. Bien entendu, ces étudiants ne peuvent plus être considérés comme des débutants complets, dans la mesure où ils ont déjà suivi des cours d'informatique en première année qui est commune aux mathématiques et à l'informatique. C'est cependant la première fois qu'ils réalisent des projets de grande envergure, selon un protocole bien fixé.

> *Ce chapitre reprend largement la substance de deux de nos articles déjà publiés. Le premier, sur la programmation des robots LEGO MINDSTORMS NXT, a été publié dans un ouvrage récent (Baron, Bruillard, & Drot-Delange, 2015, p. 266-276). Le deuxième, la programmation du robot NAO, est issu de notre contribution à la revue STICEF (Nijimbere, 2014). Le chapitre sera structuré en deux points développés respectivement sur les deux technologies robotiques.*

1. Contexte

La recherche s'est intéressée aux pratiques des étudiants de L2 et L3 en projets de programmation des robots à l'université Paris Descartes. Ces projets ont lieu au second semestre 2012 et sont pris en compte dans la validation de l'année scolaire en cours. Leur soutenance a lieu au mois de juin. Après soutenance, les étudiants ont pu participer, après sélection, à une compétition robotique inter-universitaire entre trois universités : Paris Descartes, Paris Diderot et Paris Nord.

2. Cas des robots LEGO MINDSTORMS NXT en licence 2

Cette section sera développée en trois points. D'abord, nous préciserons le contexte et les hypothèses de recherche, ensuite nous présenterons les résultats et enfin, terminerons par une brève conclusion.

2.1. Contexte et hypothèses de recherche.

a. Brève présentation du projet observé : un jeu de balles entre robots

Nous avons observé la conduite de projets consistant à programmer un jeu où deux robots adverses tentent d'envoyer des balles dans les buts adverses. Ces projets étaient orientés dans ce cadre d'une compétition interuniversitaire, organisée à la fin de l'année universitaire.

Le robot utilisé, que nous appellerons simplement « robot » dans la suite, est un kit MINDSTORMS NXT assemblable et programmable. Fabriqué par la société LEGO[128], il est principalement construit autour d'une « brique intelligente » NXT, dotée d'un microprocesseur 32 bits. Ce type de robot dispose d'actionneurs et de capteurs de pression, de couleur, de sons et d'ultrasons[129], de servomoteurs… le modèle ici construit dispose deux pinces permettant de ramasser des objets.

Illustration VIII.1: Robot LEGO MINDSTORMS NXT avant le ramassage de la balle

Face au problème posé, les étudiants devaient donc imaginer une solution mobilisant une pensée algorithmique. Non seulement ils devaient anticiper ce qui allait se passer (par exemple les balles seront très certainement déplacées au cours de la partie), comment le robot aller se déplacer, les problèmes éventuels qui allaient se poser mais ils devaient également prévoir les fonctions à mettre en œuvre, les nommer et les programmer. Enfin, ils devaient arriver à anticiper le fonctionnement de cet algorithme, et notamment l'ordre des fonctions lors de l'exécution dynamique du programme, aussi bien en simulation sur ordinateur que sur le robot.

128 http://mindstorms.lego.com/en-us/default.aspx
129 Les capteurs à ultrason permettent de détecter des objets et de mesurer les distances

b. Questionnement et hypothèses

Nous nous sommes intéressés à la manière dont des étudiants de licence (L2), en informatique à l'université Paris Descartes apprennent par la programmation du robot LEGO de type MINDSTORMS NXT. Nous nous interrogeons sur la manière dont ils apprennent lors de la programmation de ce robot. Nous gardons le même questionnement que pour la L3 : quelles connaissances construisent-ils au cours de ces projets ? Quelles difficultés majeures rencontrent-ils ? Quelles stratégies mobilisées pour les surmontées ?

Deux hypothèses ont orienté notre recherche.

Hypothèse 1 : La robotique pédagogique ne pose a priori pas de difficultés majeures aux étudiants de licence, déjà initiés à la programmation classique.

Hypothèse 2 : La programmation d'un robot exige beaucoup de connaissances chez les étudiants et par conséquent, permet l'acquisition de notions informatiques relativement avancées.

2.2. Présentation des résultats

Trois principaux types de résultats seront présentés : les heuristiques, les connaissances mobilisées et les interactions des étudiants en projets.

a. Quelques heuristiques explorées par les étudiants

Les étudiants avaient à résoudre des problèmes relativement ouverts consistant à programmer un robot dans un jeu de ramassage de balles ; ils ont pour cela utilisé un certain nombre d'heuristiques :

Le **balayage** pour cartographier le terrain de jeu est une stratégie qui a été mise en œuvre en deux temps. D'abord, le robot, dans sa position initiale, prend connaissance de tout ce qui se trouve sur terrain : balles, lignes, les couleurs, le robot adverse et tout autre objet pouvant constituer un obstacle. Cette stratégie est conditionnée par la connaissance qu'a le robot de sa position au moment T : il doit se géolocaliser à tout moment. Au moyen d'un balayage horizontal, il s'approprie tout ce qui s'y trouve en se géolocalisant d'abord lui-même. Ensuite, les informations recueillies sont utilisées pour se déplacer. Une fois la balle en contact avec le capteur de pression, il ferme ses pinces, initialement ouvertes, pour la ramasser. Tous les trois groupes ont utilisé cette stratégie.

La deuxième stratégie a consisté à **cibler prioritairement la balle centrale**. Cela présente un double avantage. D'abord, la configuration du terrain n'a pas encore changé et les balles sont encore dans leur position d'origine. Ensuite, elle permet de marquer le premier but, pour bénéficier d'un bonus (Règlement de la compétition, 2012) : « Un bonus de 3 est accordé à l'équipe qui dépose la première balle dans l'en-but adverse ». Pour accéder à cette balle, le déplacement basé sur le suivi des lignes nécessite beaucoup de temps. Il a donc été préférable que le robot, à partir de sa position initiale, fasse une rotation d'un certain angle, initialement déterminé (après plusieurs essais) pour se déplacer en diagonale. Cette stratégie a été utilisée par tous les groupes.

Prendre **le chemin le plus court** pour atteindre la balle cible est une autre stratégie utilisée par tous les groupes. Une fois les balles identifiées sur le terrain, les distances entre elles et le robot sont calculées puis comparées. Le robot se dirige alors vers la balle la plus proche. Cela lui permet d'atteindre la zone d'en-but rapidement et ainsi de marquer le but. Après ce premier parcours vers la zone d'en-but adverse, des calculs de la plus courte distance sont de nouveau faits. La balle la plus proche du robot, à partir de la zone d'en-but adverse est ciblée. Dans son parcours, il est toujours question de faire stratégiquement un bon choix sur la façon dont le robot tourne.

Des paramètres tels que la distance à parcourir, l'angle sous lequel il doit tourner, le côté vers lequel il doit aller, la vitesse à laquelle il va se déplacer… sont autant d'éléments très importants pris en compte dans la programmation des robots pour les rendre plus performants. Pour se diriger vers cette nouvelle balle cible, la stratégie du groupe « L2_MS_2H » est unique : pour ne pas bousculer les autres, des chemins extérieurs du terrain sont privilégiés. Pour la programmation, ce groupe prend le terrain pour une matrice où, les balles sont considérées comme ses éléments se trouvant à l'intersection des lignes et colonnes de cette matrice.

On observe aussi la technique qui consiste à tenter de **déstabiliser l'adversaire** en appréhendant ses stratégies pour mieux les neutraliser. Elle consiste à empêcher le robot adverse, par tous les moyens possibles, de marquer rapidement des points : « *s'introduire directement dans le camp adverse pour voler les balles qu'il voudrait prendre* », « *sécuriser chez nous en dégageant notre terrain* »… Elle a été utilisée par un seul groupe, L2_MS_2H, qui affirme avoir eu des difficultés dans sa mise en œuvre : elle nécessite de « *se mettre dans la peau de l'ennemi* » et « *un certain travail, un effort supplémentaire* ». Tentée par le même groupe, la

stratégie de boussole a été utilisée pour simuler une sorte de géolocalisation. Cependant, ils ne l'ont pas mise en œuvre car ils l'ont estimée trop risquée : risque de désynchronisation au moindre choc sur le terrain, sans moyen de remettre à jour.

On a aussi relevé un **déplacement en zigzag.** Cette stratégie permettait de retrouver les balles bousculées lors des mouvements des robots sur le terrain, qui se retrouveraient décalées de leur position initiale, ainsi qu'un **déplacement en spirale.** Mise en œuvre par le seul groupe « L2_DJ_5H », ce choix a été motivé par l'idée du groupe selon laquelle, dans son mouvement vers la zone d'en-but, un robot peut, de façon involontaire, bousculer certaines balles. Ces derniers peuvent par conséquent, être écartés de leurs positions initiales. Par un déplacement en spirale, le groupe imaginait permettre au robot d'accéder aux balles qui se seraient décalées de leur position initiale. Cette stratégie n'est conçue que pour être mise en œuvre après avoir fait le premier mouvement vers la zone d'en-but adverse.

Après avoir ramassé la balle, les robots devaient se déplacer jusqu'à la zone d'en-but adverse afin de déposer la balle. Le robot était programmé pour suivre les couleurs des lignes sur terrain. La zone d'en-but, de couleur blanche, était le seul indicateur de cette zone qu'il fallait à tout prix identifier. La stratégie utilisée était alors la détection de la couleur blanche : une fois cette couleur détectée, le robot était sûr de se retrouver dans la zone favorable et ainsi pouvait déposer sa balle par simple ouverture de ses pinces.

Pour une seule tâche, plusieurs stratégies ont été utilisées. C'est notamment le cas de celles visant à localiser les balles sur le terrain mais aussi celles consistant à se déplacer vers une balle cible.

b. Des connaissances utilisées

Les étudiants ont principalement mis en œuvre, dans un contexte nouveau, des connaissances qu'ils avaient déjà acquises avant : variables, instructions, affectation, algorithme, condition, alternatives, boucles, itérations, fonctions, structures, pointeurs. Le langage C avait été enseigné en L2 au premier semestre de la même année, il était familier aux étudiants. Ils ont aussi dû s'approprier des connaissances nouvelles pour eux, notamment celles qui régissent le fonctionnement en temps réel du robot. En pratique, deux langages ont été utilisés : NXC et BrickCC, proches du langage C.

Les différentes tâches à effectuer ont été implémentées sous forme de fonctions : task ajuster_couleur() ; task balayage() ; task avancer() ; task attraper()…… Dans chacune des

fonctions, les notions informatiques sont mobilisées avec de possibles imbrications de deux à trois boucles à la fois, dont une utilisation très fréquente de la boucle « while ». Au total, quelques centaines de lignes de codes ont été construites par chaque groupe d'étudiants. Comparativement aux projets de programmation « classiques », cette taille de code reste modeste. L'écart semble essentiellement dû aux difficultés et contraintes techniques relatives au robot.

En dehors de l'informatique, d'autres connaissances ont été utilisées. Nous nous sommes penchés sur les mathématiques mobilisées pour faire se mouvoir les robots. Notre observation nous a montré que seules des connaissances élémentaires, vues au cours de la scolarité obligatoire, sont intervenues : repère, vecteur, coordonnée, rotation, distance, trigonométrie, angle, équations, fonctions circulaires : sinus, cosinus… Il est à noter qu'un groupe (L2_MS_2H) a aussi fait intervenir des notions de matrices à deux dimensions et d'homographie. La réponse suivante est typique de ce que nous avons observé : « *Pas de connaissances (mathématiques) mais des souvenirs du collège et lycée…* ». L'enseignant-encadrant E1 justifie cette absence de construction de nouvelles connaissances mathématiques par le fait que, dans ce contexte de programmation, « *les mathématiques sont utilisées comme outil, mais pas comme un objet d'apprentissage, donc pas comme un objet d'étude en tant que tel.* ».

c. *Un travail en commun avec de nombreuses interactions*

Ce projet pédagogique basé sur la programmation des robots a occasionné de nombreuses interactions. Nous avons surtout observé ce qui s'est passé au cours des tests de code sur le robot, dans la salle spécialisée mise à la disposition des étudiants. Celle-ci était assez exiguë et partagée selon un planning d'accès strict (seuls 2 robots MINDSTORMS étaient opérationnels). Chaque groupe y avait accès une fois par semaine seulement. À ce moment, des échanges avaient lieu entre deux ou trois étudiants, ou entre tous les membres du groupe. À l'issue de ces interactions, certaines productions ont été mises de côté pour ne garder que celles jugées meilleures. D'autres productions qualifiées d'insuffisantes ont alors été complétées ensemble ou améliorées.

Si les échanges ont été fréquents au sein des membres d'un groupe, ils ont continué aussi entre des groupes différents mais ayant un même projet : les échanges consistaient en des explications de ce qu'ils n'avaient pas compris ou n'avaient pas pu faire correctement, de stratégies à utiliser, des explications sur des logiciels à utiliser mais non encore maîtrisés, etc.

Bien que ces échanges soient valorisés pour la plupart des groupes, ils n'ont pas été toujours possibles au sein de tous les groupes de projet qui travaillaient sur le même sujet. Par exemple, le groupe L2_DJ_5H qui affirme être « content de réussir ensemble avec les autres », déplore un refus de collaboration de certains groupes ciblés qui travaillaient sur un même projet qu'eux comme il l'affirme ici : « certains sont réfractaires à l'échange, surtout un groupe particulier ! ». Ce groupe, tout en critiquant certains étudiants d'un groupe de sa classe qui semblaient être en compétition avec d'autres, rendait hommage à ceux de L3 dans leur façon de travailler et de collaborer avec les autres : « Les L3 nous aident, nous, alors qu'on était sur des trucs différents. (...). On parlait avec les L3 mais, avec les L2, non ! ».

Les interactions ne limitent pas seulement aux étudiants en projets. Le réseau d'interactions se construit et s'élargit jusqu'à atteindre les promotions précédentes encore en master au sein de l'Université. Les enseignants-encadrants parlent d'une « communication intergénérationnelle » qui se développe de plus en plus chez les étudiants, révélant ainsi une sorte de culture de projet qui se met de plus en plus en place. C'est ce que l'enseignant E2 explique ici : *« J'ai le sentiment qu'il y a une culture de tout ça qui s'adapte auprès des étudiants. Les étudiants de L3 parlent aux étudiants de L2 et quand les étudiants de L2 ont vécu un projet, ils font pareil en L3. Les étudiants qui sont en masters qui sont encore chez nous, vont parler à ceux qui sont en L3 comment ça s'est passé. Donc, il y a une communication transgénérationnelle qui fait que globalement la qualité (du travail) s'améliore et progresse… ».*

Quant aux questions adressées par les étudiants à leurs enseignants-encadrants, ce sont, selon E2, des questions avec enjeux et, donc moins naïves et bien construites. Ce sont des questions qui montrent qu'ils ont suffisamment réfléchi au problème mais qu'ils ont été bloqués et cherchent donc comment débloquer la situation pour aller plus loin dans leurs apprentissages. Quant aux enseignants, leurs interventions sont restées essentiellement orales, suscitant plutôt l'initiative et la recherche des étudiants. Comme exemple d'illustration, l'enseignant-encadrant E1 affirme donner rarement une réponse directe à la question posée : « *il est évident qu'on ne peut pas connaître tous les détails, toutes les astuces de toutes les technologies. Par contre, on peut les pousser à adopter justement une démarche de recherche de l'information pour les amener à s'interroger. Par exemple moi, je donne jamais ou très rarement la réponse à la question »*. Le fait de ne pas donner des réponses directes semble avoir comme objectif d'amener les étudiants à réfléchir pour chercher eux-mêmes des solutions à leurs problèmes.

2.3. Conclusion

L'approche par projets de programmation des robots est une approche prometteuse. Ses potentialités formatrices semblent justifiées par le fait qu'elle est souvent associée à l'idée de pluridisciplinarité. Duchâteau (1993) évoque un éclatement des frontières disciplinaires comme caractéristique fondamentale commune à l'informatique et à la robotique pédagogique. Ce que nous avons vu est typique d'une pluridisciplinarité asymétrique, où les mathématiques interviennent au service d'un projet clairement perçu comme informatique.

Deux points peuvent être soulignés. D'abord, ce sont les stratégies discutées en commun qui ont été les plus réussies à mettre en œuvre. Ensuite, l'accent a plutôt porté sur la collaboration. Un groupe qui ne voulait pas communiquer avec les autres a été critiqué et considéré comme marginal. Ceci témoigne de l'importance accordée à la collaboration lors du travail en groupe chez les étudiants. Ces derniers finissaient généralement par surmonter la difficulté, même si cela pouvait quelquefois prendre beaucoup de temps. Les recours à l'enseignant, finalement peu fréquents, leur permettaient d'obtenir des pistes d'aide, mais ils étaient incités à trouver seul la solution.

Contrairement aux sujets portant sur la programmation des robots, beaucoup de choix de projets chez les autres étudiants concernaient des sujets orientés vers la programmation classique des ordinateurs ou des smartphones. Sur 43 groupes que comptent, il y a 2 groupes sur NAO et 5 groupes sur LEGO MINDSTORMS NXT. Donc, 36 groupes ont choisi la programmation classique. Ces différences semblent, dans cet ordre, justifiées par la complexité et la difficulté des projets choisis : NAO est plus difficile à programmer que LEGO MINDSTORMS et ce dernier plus difficile que la programmation classique. Comme on l'a vu avec NAO, certains facteurs peuvent expliquer la difficulté de programmation LEGO. Le premier facteur est le fait que les robots en général sont, dans leurs mouvements, influencés par des facteurs externes (température, mouvement, couleurs, son…) qui sont difficiles à être prise en compte par les étudiants. Un autre facteur qui justifie la difficulté de la programmation de LEGO MINDSTORMS et qui, par conséquent fait la différence avec des robots actuels programmés par les enfants, est le fait que ces robots pour les enfants sont programmés en langages à base de blocs alors que pour LEGO MINDSTORMS, il faut un codage pointu et où des difficultés algorithmiques et syntaxiques sont toutes en présence. Donc, la programmation des robots avec des langages à blocs et la programmation avec le langage NXT sont deux contextes dif-

férents avec des langages différents et, ce dernier nécessité une certaine maturité. Par conséquent, les exigences de la programmation de LEGO MINDSTORMS avec le langage NXT nécessitent beaucoup de connaissances. L'hypothèse 2 semble vérifier.

Contrairement à la programmation classique, beaucoup de facteurs externes, comme on l'a vu avec NAO, difficiles à prendre en compte pour des débutants, peuvent justifier les difficultés vécues par les étudiants. Nous supposons que le peu d'étudiants qui préfèrent programmer les robots peut être associé aux difficultés vécues dans la robotique plus que la programmation classique.

De plus, il semble que les apprentissages dans les deux contextes (la programmation classique et la programmation des robots) sont différents. Une recherche sur les acquisitions des étudiants dans la conduite de projets de programmation classiques permettrait une comparaison des apprentissages dans les deux contextes.

3. Cas des robots NAO en licence 3

Nous développerons cette section en trois points. Nous présenterons brièvement d'abord l'environnement NAO et le rappel des hypothèses, ensuite nous présenterons les résultats et enfin, nous terminerons par une brève discussion.

3.1. Contexte et rappel des hypothèses

Cette section concerne deux groupes de sept étudiants, les seuls à avoir travaillé sur le robot NAO sur un total de 43 groupes de L3 en raison de 3 à 4 étudiants par groupe. Les autres (41) groupes d'étudiants restants de la classe ont préféré d'autres types de projets et ne sont pas pris en compte dans cet article. Dans la suite, pour les distinguer, les deux groupes seront respectivement notés « groupe A » et « groupe B ». Ils étaient sur un même sujet de projet : la programmation du robot NAO qui « joue au basket ». Si dans tous les groupes, aucun étudiant n'a déjà programmé un tel type de robot, la composition des groupes est telle qu'il se trouve dans chacun d'eux, un étudiant « pilote », plus en avancé par rapport aux autres en informatique. Dans le groupe A se trouve un étudiant qui a fait une spécialité informatique au lycée à l'étranger. Dans le groupe B, un étudiant a déjà participé au projet de programmation à l'Institut Supérieur de Technologie (IUT), d'un robot qui se déplace dans un labyrinthe. De plus, tous les étudiants ont déjà participé à au moins un projet, non nécessairement lié à la robotique, notamment en L2.

a. *Brève présentation du terrain de jeu et du robot utilisés en projet*

Entouré d'une bordure rigide de 15 cm, le terrain de jeu mesure trois mètres sur deux. Les zones de terrain sont délimitées par des lignes de couleurs différentes : gris clair pour le fond du terrain, blanc pour les zones d'en-but, vert et bleu pour les lignes Est et Ouest du terrain, rouge et jaune pour le Nord et le Sud et, noir pour les lignes partageant le terrain en son milieu de l'Est à l'Ouest et du Nord au Sud. Deux robots qui s'affrontent commencent chacun à l'extrémité du terrain, où trois positions sont possibles. Sur le terrain de jeu, neuf balles ont été positionnées aux intersections des lignes. Parmi elles, cinq balles et une balle centrale commune, se trouvent devant chaque robot. Ce dernier a le droit de prendre uniquement des balles situées devant lui, dans sa moitié de terrain ainsi que la balle centrale et, la dépose dans le panier du camp adverse (Règlement de la compétition RoboUni Cup, 2012)[130]. La figure suivante illustre, au départ, le positionnement des balles et des robots sur le terrain de jeu (les gros points noirs représentent les balles, les lettres R représentent les positions possibles des robots et les ronds blancs, aux extrêmes gauche et droite, représentent les paniers) :

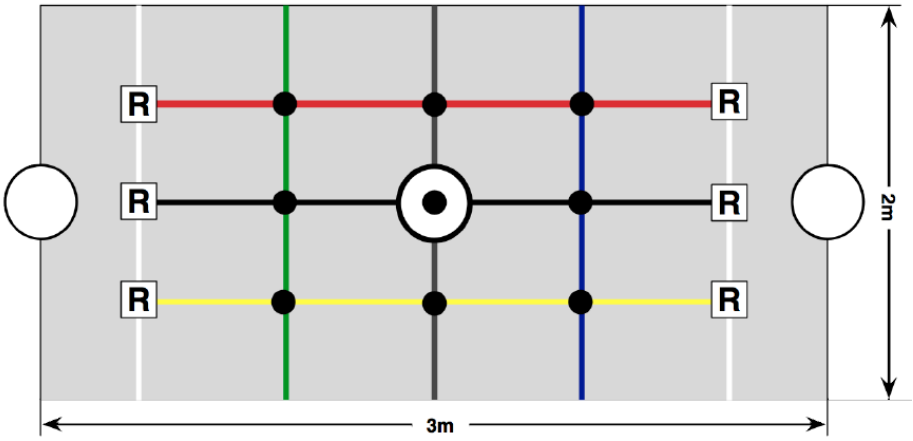


Illustration VIII.2: Schéma de principe du terrain de jeu (Règlement RoboUniCup, 2012)

Pour résoudre un problème, un robot fonctionne en suivant un processus complexe, tenant compte de son système mécanique et des paramètres issus de son environnement (Roby-Brami & Laffont, 2002) comme le montre l'illustration VIII.3 suivante.

130 http://www.univ-paris-diderot.fr/sc/site.php ? bc=vivre_universite&np=pageActu&g=m&ref=4234, document consulté le 14 octobre 2012

Illustration VIII.4: Robot NAO

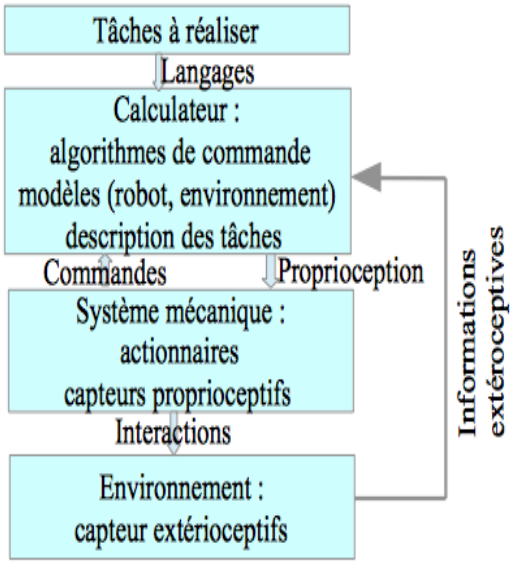


Illustration VIII.3: Schéma général de fonctionnement d'un robot (Roby-Brami & Laffont, 2002)

La technologie utilisée dans ce projet robotique est NAO. Fabriqué par la société Aldebaran-robotics1, NAO est un robot de type humanoïde, avec des jambes, des bras et une tête. Il est équipé d'une centrale inertielle de cinq axes, de capteurs d'interaction, des capteurs à ultra-son allant dans deux directions différentes et de capteurs de pression sous les pieds. Pour la synthèse vocale, la localisation des sons dans l'espace ou encore la reconnaissance d'obstacle (humain ou objet), il est doté d'un système multimédia composé de quatre microphones, de deux caméras sur sa tête, orientées respectivement devant lui et en direction des pieds pour voir de près et de loin, et enfin, de deux haut-parleurs. Grâce à ses 25 degrés de liberté (ddl), il peut exécuter les principaux mouvements d'un humain (se baisser, prendre un objet donc ici une balle) et peut bien s'orienter sur un terrain lisse, grâce à son système de fonctionnement lui permettant de s'approprier des informations de son environnement. Le règlement de la compétition est à cette adresse[131].

b. *Rappel des hypothèses*

La difficulté à proposer des contenus de formation en lien direct avec les applications du monde courant serait à l'origine de l'absence de motivation des étudiants face à la discipline informatique (Muratet, 2010). NAO, est l'une des innovations technologiques et

131 http://www.sorbonne-paris-cite.fr/index.php/en/component/simpledownload/?task=download&fileid=cmVnbGVtZW50LWNvbXBldGl0aW9uX3JvYm90aXF1ZS5wZGY%3D. : site consulté le 20 mai 2013

pédagogiques, proposé pour soutenir cette motivation. Mais, des questions se posent. Si la programmation classique fait autant de problèmes (échecs et abandons des étudiants en début d'études supérieures) (Kaasboll, 2002), la programmation de NAO arrive-t-elle à soutenir leur motivation vis-à-vis de l'informatique ? Qu'apprennent – ils et comment les étudiants de licence lors de la programmation des robots ?

Ces questions générales visent à identifier les connaissances construites par les étudiants au cours de leurs projets de programmation des robots. Les questions de recherche suivantes se posent : quelles sont les motivations des choix de sujet de projet chez les étudiants ? Quelles sont les connaissances construites au cours des projets de programmation de NAO ? Quelles sont les stratégies utilisées et comment sont-elles mises en œuvre pour gagner la compétition ? Quelles sont les difficultés rencontrées au cours de ces activités ?

Pour rappel, deux hypothèses de départ ont orienté la recherche. La première est que la RP ne pose a priori pas de difficultés majeures aux étudiants de licence, déjà initiés à la programmation classique. La seconde est que la programmation d'un robot exige beaucoup de connaissances chez les étudiants et par conséquent, permet l'acquisition de notions informatiques relativement avancées.

3.2. Présentation des résultats

Quatre types de résultats seront présentés : les motivations qui soutiennent les choix des sujets chez les étudiants, les stratégies conçues, les connaissances utilisées et les difficultés vécues.

a. Des motivations diverses pour le choix des sujets

Dès le début de l'année universitaire, les étudiants ont été informés de leur participation aux projets de programmation au second semestre. Une séance réunissant les étudiants et leurs futurs encadrants (notés E1, E2…) a donné lieu à l'explication du contexte dans lequel les projets allaient avoir lieu. Ils ont été présentés par l'enseignant encadrant principal, entourés de ses collègues enseignants et/ou encadrants. Les encadrants sont issus du département informatique de l'université Paris Descartes. Insuffisants face au grand nombre d'étudiants, ils sont aidés par des encadrants venant de l'extérieur, essentiellement du monde de l'entreprise. Après cet exposé, les étudiants intéressés par l'un ou l'autre des sujets proposés1, étaient invités à entrer en contact avec l'encadrant, promoteur du sujet qui les intéresse pour plus de

clarifications sur ses attentes. En groupe de quatre, les étudiants devaient faire quatre choix de sujets et les classer par ordre de préférence. Après, en fonction de l'ordre de choix de ce sujet et du nombre d'étudiants qui l'ont choisi, une répartition des sujets était faite entre les groupes.

Dans le domaine de la robotique, à l'université Paris Descartes, la programmation de NAO est un domaine nouveau d'apprentissage pour les étudiants de licence. Les deux sujets de projets proposés sur NAO, « NAO joue au basket » et « NAO lit nos pensées », n'ont pas trouvé la même préférence chez les étudiants. Différentes motivations ont orienté leurs choix. Si aucun des groupes n'a placé un projet robotique en première position, un des groupes, noté A, était à l'unanimité motivé pour en faire. Le caractère concret de la technologie et l'intérêt pour leur avenir professionnel semblent être les raisons de ce choix. Dans le groupe noté B par contre, un seul étudiant, qui semble « leader », était particulièrement motivé par le projet « NAO lit nos pensées ». Faute de matériel, ce sujet n'a pas été continué et a été remplacé par un autre qui était moins préféré pour eux : « NAO joue au basket ».

Six principales variables ressortent comme des raisons qui sont intervenues pour influencer les choix : le thème, la familiarité avec technologie, la forme humanoïde de NAO, l'intérêt professionnel, l'occasion unique d'acquérir de nouvelles connaissances et la facilité présupposée du sujet.

La première motivation a été la thématique proposée dans le sujet. Chez les étudiants, pour une même technologie, un sujet peut être plus intéressant qu'un autre. La non-diversité des sujets sur les robots a été une limite pour laisser voir plusieurs choix possibles même si la variable « thématique » sur laquelle porte le sujet, a été une source de motivations de choix des sujets. Par exemple, pour tous les groupes de L3, le projet sur le sujet « NAO lit dans les pensées » a été placé en avant sur le sujet « NAO joue au basket ». Cette variable « thématique » est mise en relation chez les étudiants avec la notion de facilité. En l'absence d'expériences dans le domaine, l'anticipation des facilités ou des difficultés susceptibles d'être vécues en contexte de projets intervenaient et influençaient les choix des sujets : un sujet qui leur semble difficile n'était pas choisi. La mise en avant du projet « NAO lit dans nos pensées » était due au fait que les étudiants l'imaginaient moins compliqué que le sujet « NAO joue au basket » mais, a été, par la suite, supprimé faute d'une ressource nécessaire qui obligeait une licence d'accès. Le groupe B a anticipé des difficultés liées à la précision dans la mesure des distances, ce qui a constitué la non-priorité de son choix. Ils s'expriment ainsi pour justifier

leur choix : *« pour NAO qui manipule des balles, ce qui est compliqué c'est d'estimer la position dans l'espace et pour la puissance 4, il faut de l'Intelligence Artificielle. Donc, ça me paraissait vraiment difficile…* ».

La deuxième motivation est la familiarité envers la technologie proposée : un sujet faisant intervenir une technologie familière était plus préféré. Malgré l'intérêt porté aux thématiques en rapport avec NAO, notamment motivé par son caractère concret, aucun groupe ne l'avait placé en première position : ils étaient respectivement placés en deuxième position pour le groupe B avec le sujet « NAO lit nos pensées » et en troisième position pour le sujet « NAO joue au basket » pour le groupe A. Leurs premiers choix portaient plus sur des projets sans rapport avec la robotique. Le projet sur la « reconnaissance faciale » a été le premier choix du groupe B : au-delà de la thématique intéressante, ce choix était justifié par la familiarité avec la technologie « androïde » utilisée et bien maîtrisée.

La troisième motivation est la forme de NAO. Par sa forme humanoïde, il a été trouvé sympathique par les étudiants. Ils espéraient le faire jouer : *« On a regardé les images, il avait l'air sympa ! […] on s'imaginait un truc… qu'il marque un point et qu'il danse… on voulait faire pleins de choses mais au final on voit que c'est très compliqué, voire impossible ! »*. Contrairement aux attentes, la compétition des robots n'a pas suscité de grande motivation : elle avait un caractère stressant. Les étudiants affirment n'avoir pas pris en compte ce critère de compétition dans leur choix : « on n'avait pas lu les petites lignes en bas qui disaient qu'il y aurait une compétition ». En l'absence de familiarité avec NAO (dont certains disent ne l'avoir jamais vu que par des images et des vidéos sur YouTube), la sympathie envers ces robots a donc primé sur la compétition dans les choix des sujets.

Une quatrième motivation est l'intérêt professionnel des sujets proposés. Si le contexte de projet les oblige à un consensus sur un choix de sujet, des centres d'intérêt sont souvent divergents, ce qui explique leurs préférences différentes. C'est le cas du groupe B par exemple : l'un d'entre eux aurait voulu faire un projet concernant le « planning » d'un calendrier de rugby, un autre préférerait travailler sur le « Web ». Ce dernier qui semble peu motivé par le projet NAO, reste sceptique sur le choix : il dit attendre de voir ce que ça va donner. Il apparaît qu'un leader, ayant déjà programmé un robot à l'IUT, a influencé son groupe vers ses préférences. Ce groupe justifie le choix du projet par le prestige actuel de cette technologie : « on en parle à la télé ». La vision professionnelle semble avoir influencé leurs motivations : faire un projet sur NAO leur apportera un grand avantage notamment dans leur CV.

Selon eux, travailler sur la robotique augmente leurs acquis dans le domaine informatique et les opportunités professionnelles, comme ils l'expriment ici : « quand on va postuler pour un stage, dire « j'ai déjà fait de la robotique », c'est bien c'est une facette de l'informatique ».

La cinquième motivation est l'acquisition de nouvelles connaissances. En effet, si la programmation des technologies robotiques est connue comme difficile, leur choix est motivé par l'opportunité jugée unique de construire des connaissances non acquises ailleurs dans d'autres modules. Le groupe B justifie le choix de leur sujet parce qu'il fait intervenir un nouveau langage : *« Normalement quand on fait un master à l'université Paris Descartes, on n'est pas censé apprendre du Python : nous, c'est un plus… »*. Les étudiants du groupe A considèrent le projet sur la programmation de NAO comme pouvant leur permettre d'apprendre à travailler en groupe et à gérer un projet ensemble. De plus, les projets leur offrent des contextes de construire différemment leurs savoirs même s'ils sont construits : ces derniers varient en fonction des sujets de projets faits au cours de l'année ou de leurs cursus.

La dernière motivation des étudiants est donnée par les enseignants. Elle concerne la facilité supposée du sujet. Comme il n'est pas aisé de préjuger la facilité d'un projet sans l'avoir fait, le recours à un vocabulaire familier est souvent le critère utilisé par les étudiants, comme le montre cet enseignant : *« s'il y a très peu de mots qu'ils connaissent dans le sujet proposé, ils vont le considérer comme difficile, alors qu'ils ne connaissent juste pas le nom de la technologie, ils ne connaissent pas le nom de l'algorithme ! Alors que c'est pas forcément difficile ! Mais parce qu'ils ne connaissent pas le nom, c'est difficile ! »*. L'enjeu de la recherche d'un sujet facile est l'obtention d'un meilleur résultat, d'une bonne note.

b. Les connaissances informatiques utilisées et celles construites

La programmation des robots fait intervenir beaucoup de connaissances informatiques. Parmi elles, on distingue celles de base et celles avancées. Celles déjà acquises précédemment ne posent pas de difficultés majeures comme les étudiants l'expriment ici : « pour l'algo, les choses en rapport avec les boucles, variables, franchement ça n'était pas plus grave ». Elles étaient utilisées dans les connaissances avancées telles que les structures, les fonctions… Comme exemples illustratifs, les balles sont déterminées dans le plan du terrain par ses coordonnées dans un repère ou comme éléments d'une matrice. Le robot, quant à lui, est considéré comme une structure pour bien l'implémenter. Le tableau 3.2 suivant synthétise les connaissances informatiques utilisées.

Connaissances informatiques	Robot NAO	Groupe concerné
Notions de base	Variables, instructions, affectation, algorithme, condition, alternatives, boucles, itérations…	Tous les groupes
Notions avancées	Fonctions, structures	Tous les groupes

Tableau VIII.1: Caractérisation de notions informatiques utilisées dans la programmation du robot NAO

Comme toute programmation informatique, la programmation des robots nécessite un langage. Python a été utilisé pour programmer NAO et il s'agissait d'un nouveau langage pour les étudiants. Si Python était recommandé, aucun langage ou technologie n'était obligatoire et leurs choix étaient libres comme le témoigne l'enseignant E1 : *« au niveau des langages, pas une limitation et on ne cadre pas sur les langages utilisés ou les technologies utilisées. (...) il y en a qui manipulent des langages qu'ils n'avaient jamais manipulés avant »*. Le contexte de programmation leur semblait complexe de telle sorte que tout semblait à reconstruire chez les étudiants : *« il y a des variables qu'on connaissait pas en fait. Il y avait déjà des boîtes programmables dans chorégraphe. Il fallait tout apprendre en fait, mais ça, c'était plus compliqué… »*. Pour une tâche donnée, les étudiants mobilisaient diverses compétences, notamment des métaphores qui les aident à construire des représentations des actions de programmation au niveau de la conception, la construction et l'implémentation d'un algorithme, comme ils l'expriment ici :

> *« Pour écrire un algorithme, il y avait un mur de la salle, des petits carreaux de briques, en fait on disait que c'était des pixels. C'est de là qu'il est venu comment on a fait pour écrire l'algo et après, au fur et à mesure on a optimisé. Ça nous a donné un truc pratique acceptable. Et puis le robot ne prend plus une heure à chercher la bonne position ! Rires. »*

D'autres connaissances construites sont plus techniques telles que des fonctions régissant le fonctionnement de NAO notamment dans ses mouvements : AlMotion : Walk To et AlMotion : SetWalkTargetVelocity. Le Tableau VIII.2 suivant présente quelques connaissances acquises, classées selon leur type (langage, logiciels…).

Types de connaissances informatiques	Robot NAO	Groupe concerné
Langages	Python	A et B
Librairies et logiciels	Chorégraphe, NAOsim, librairie OpenCV, Simulateur, Timiline, Connectify	A et B
	Highgui, PIL	A
Fonctions traitant des mouvements	WalkTraker, WalkTo, AlMotion : Walk To, AlMotion : SetWalkTargetVelocity	A et B
Imagerie	Analyse d'image, pixels, filtre d'image, conversion des couleurs suivant les angles et la luminosité	A et B
Recherche informationnelle sur Internet	Recherche documentaire : usage des moteurs de recherche et des mots clés	A et B

Tableau VIII.2: Notions informatiques intervenues dans la programmation de NAO selon le groupe

Il apparaît que les connaissances informatiques mobilisées par les groupes restent les mêmes.

c. *Le déficit des connaissances mathématiques*

Outre les connaissances informatiques que nous venons de voir, la programmation des robots nécessite des connaissances mathématiques. Néanmoins, ces dernières sont restées modestes. Tous les groupes affirment n'avoir pas été très loin dans les notions mathématiques utilisées comme l'affirme le groupe A : *« Au début, on s'est dit : il y aura beaucoup de maths en fait. Mais c'est quoi ? C'est plus… Au niveau des connaissances mathématiques, on se serait arrêté au niveau de Pythagore, trigonométrie avec les histoires des angles. C'était pas non plus quelque chose de très compliqué ! ».* Quant à la question de savoir si ça ne serait pas plutôt un défaut de connaissances mathématiques qui leur posait plus de difficultés avec leur robot, le groupe B répond par la négative. Ils affirment qu'il n'y a pas de lien entre les difficultés vécues dans la programmation et les limites de connaissances mathématiques utilisées parce que, selon eux, au niveau des connaissances mathématiques, ils ont essayé « beaucoup de choses ». Ceux du groupe A reconnaissent des lacunes dans les connaissances mathématiques utilisées, connaissances qui pourraient leur permettre d'aller plus loin pour améliorer

leur projet : « on aurait utilisé plus de maths si on veut que le programme fonctionne plus correctement. ».

Les limites des étudiants de licence à construire et utiliser des mathématiques sont aussi connues de leurs encadrants. Ces derniers affirment même que, dans le contexte de la programmation des robots de type NAO, les étudiants de niveau master sont, eux aussi, à la limite de leurs connaissances en mathématiques. Différentes raisons sont données pour expliquer ces limites. La première concerne l'approche utilisée dans la résolution des problèmes. L'absence de modélisation, pourtant convenable selon les enseignants, est la cause d'un défaut de construction des connaissances mathématiques. L'enseignant E2 reconnaît que la modélisation est une approche qui, pour NAO, nécessite plus de connaissances mathématiques et est donc difficile à mettre en œuvre par les étudiants : *« s'il s'agissait de la modélisation du problème, donc connaître la position du robot, la position des moteurs, modéliser la position du bras en fonction des mouvements des moteurs, ça serait plus simple pour résoudre le problème, sauf que mathématiquement ça serait plus compliqué à réaliser pour eux. (...) les étudiants rentrent rarement dans une démarche expérimentale (...) ».*

La deuxième raison réside dans le fait que les étudiants ont des difficultés à mettre en relation ce qu'ils ont à faire et les connaissances à mobiliser. À la question de savoir s'il y a des connaissances mathématiques qui leur permettraient d'aller plus loin dans leur travail mais qui n'ont pas été utilisées, ils répondent par l'affirmative que : « Oui, c'est évident ! ».

La troisième raison est donnée par l'encadrant E1. Il tente de justifier cette absence de construction de nouvelles connaissances mathématiques par le fait que, dans le contexte de la programmation des robots par les étudiants de licence, « les mathématiques sont utilisées comme « outil », que comme objet d'apprentissage, donc pas comme un objet d'étude en tant que tel-. ». Le tableau 3.3 suivant présente la synthèse des notions mathématiques utilisées dans ces projets.

Domaine mathématique	NAO	Groupe concerné
Géométrie	Repère à deux ou trois dimensions, coordonnées, cercle, diamètre, rayon, circonférence, centre d'un cercle, rotation, distance, théorèmes de Thalès et Pythagore	A et B
Trigonométrie	Angle, radian, fonctions circulaires, fonctions circulaires inverses (asinus, acosinus, atan…)	A et B

Tableau VIII.3: Domaines et notions mathématiques intervenant dans la programmation de NAO

Si des connaissances mathématiques nouvelles sont peu élaborées, ce n'est pas le cas dans d'autres disciplines, où des connaissances sont construites au cours de projets : recherche d'informations sur Internet, acquisition et amélioration de la langue anglaise grâce à leur confrontation avec une documentation en anglais, gestion des relations humaines, utilisation des logiciels pour communiquer à distance au moment du travail en synchrone, notions de physique, de mécanique, etc. C'est le cas par exemple des notions de vitesse, d'accélération, de vitesse angulaire utilisées dans la programmation de NAO comme de LEGO MINDSTORMS NXT (Nijimbere et al., 2015).

Dans la programmation de NAO, les étudiants étaient confrontés aux problèmes compliqués d'algorithmique, plus visibles lorsqu'ils essaient de mettre en œuvre leurs idées et leurs stratégies en termes d'algorithmes – la conception – (des) actions que le robot doit exécuter et leurs formalisations en termes de codes. Cette complication est, selon l'encadrant E2, en partie liée à la technologie utilisée : la conception particulière des robots fait que leur programmation pose des difficultés spécifiques notamment dues à beaucoup de problèmes de mise au point, par rapport à la programmation classique. Cette particularité du robot engendre de nombreuses répétitions obligatoires des mêmes tâches au cours de la réflexion, notamment sur comment contourner les difficultés rencontrées. Selon l'enseignant-encadrant E1, ceci est en contradiction avec les exigences de la programmation classique qui, malgré les nombreuses réflexions nécessaires sur comment réaliser le projet ou les algorithmes à utiliser, est beaucoup moins longue dans sa réalisation. Les difficultés qui se posent aux étudiants dans ce genre de programmation restent davantage des problèmes algorithmiques qui s'ajoutent aux problèmes techniques.

Néanmoins, procédant par une comparaison tenant compte de la métrique « nombre de lignes de code », il apparaît que la taille du code implémenté reste très courte dans le cas de la robotique alors qu'elle peut facilement aller, sur des applications plus classiques, à plusieurs milliers de lignes.

d. Autres difficultés rencontrées

- **Python, un langage pas encore maîtrisé**

Les étudiants en programmation de NAO se sentaient plus en difficulté par rapport à ceux des autres projets. Ils déclaraient le besoin d'être *« plus aiguillés »* pour s'en sortir. Le langage utilisé, Python, était pour eux nouveau et ils devaient l'apprendre sur le tas. Selon les étudiants, si la découverte d'un nouveau langage était une bonne chose, ils déplorent le fait qu'ils n'ont pas été mis au courant des contraintes de ce nouveau langage. Le groupe B qui semble justifier son retard par un suivi insuffisant, dit avoir passé beaucoup de temps à apprendre ce nouveau langage alors qu'il était possible d'utiliser le C++, déjà connu et plus adapté à ce sujet : *« On n'a pas été informé, et du coup, on s'est lancé en Python, mais il y avait plus de documentation que le C++. C'est un peu dommage ! (...). Par exemple pour le langage Python, on aurait dû être régulier dès le départ puisque le langage C++ aurait dû être plus approprié pour le projet »*. Ils déplorent aussi le manque d'informations qui a concerné l'analyse d'image, étape qui a posé pas mal de difficultés dans l'avancement des projets pour tous les groupes sur NAO, alors qu'il y avait un autre groupe de master qui y travaillait.

De même, le groupe A affirme, lui aussi, que le langage C++ aurait été plus adapté pour ce genre de projet et, s'il y avait à le refaire, c'est ce langage C++ qu'il utiliserait, ajoute-t-il. Contrairement au groupe B qui regrette la perte du temps occasionnée par ce langage, le groupe A assume ses choix même s'il reconnaît des différences en termes de performance de ces deux langages : *« si on avait utilisé le C++, ça serait bien parce que le C++ est un peu plus rapide quand on l'exécute que Python. […] Le prof nous a dit : prenez ce que vous voulez [comme langage] et nous, on a vu que le logiciel qu'on utilise, chorégraphe, était codé en Python. Donc, on s'est dit : pourquoi pas continuer ! On va continuer avec Python, peut être on va être trop dépaysé, on connaîtra un petit peu et on est parti sur Python… »*. Les étudiants demandent que des conseils ou des orientations leur soient donnés assez tôt pour avoir le temps de les mettre en œuvre.

- **Des choix méthodologiques souvent peu efficaces**

Une autre difficulté vécue par les étudiants est le choix de la méthode à utiliser pour mettre en œuvre les stratégies choisies. Comme l'indique un des enseignants, dans une recherche d'information avec un moteur de recherche, un mauvais choix de mots-clés, peut conduire à des résultats erronés. Un exemple donné est celui des groupes qui peuvent passer un week-end à chercher une information sur Google et ne pas trouver alors qu'une bonne recherche, donc utilisant les bons mots-clés, permet de trouver instantanément. L'utilisation d'une méthodologie non adaptée a plusieurs conséquences : disproportionnalité du temps utilisé par rapport aux résultats atteints ou même inadéquation entre l'effort fourni et la réalité du travail nécessaire pour arriver à résoudre un problème donné.

Des choix de méthodologie peu efficaces ont été systématiquement observés chez les étudiants. Par exemple, ils faisaient *« apprendre par cœur des positions favorables »* au robot et ce, à la main, des positions jugées stratégiques pour faire l'un ou l'autre geste. Au moment du test le robot se bloquait et perdait l'équilibre avec des risques de se casser : il était rattrapé avant qu'il ne tombe par terre. Cette façon de faire se trouvait inefficace surtout qu'il devient aussi impossible pour le robot de combiner des séquences élémentaires consécutives pour exécuter une tâche complexe. À cet effet, Paulin *et al*., (2006) montrent que certaines marches des robots humanoïdes se réduisent à dérouler automatiquement une séquence d'actions élémentaires, calculées à l'avance et limitées à des plages de fonctionnement parfaitement sécurisées afin de limiter les risques de chute dont le coût financier est parfois important.

La difficulté est d'abord située au niveau du choix de l'approche à utiliser conduisant à la conception d'un algorithme comme méthode de résolution d'un problème. L'absence d'une approche de modélisation est évoquée comme une lacune majeure. Les étudiants ont privilégié une approche qui nécessite des algorithmes plus simples, mais comme le disent les enseignants, ces algorithmes simples nécessitent aussi beaucoup de mises au point et, ainsi rallongent le travail à faire.

Si une approche choisie convoque un type de raisonnement conséquent, l'encadrant E2 trouve cette approche plus adaptée à l'apprentissage du fait qu'il intègre des mises en place des boucles de rétroaction, de la compréhension, des calculs de position des bras, des positions des moteurs, une approche plus cinématique. Cette efficacité de l'approche choisie est

aussi soulignée par la recherche. Le contexte de la programmation d'un robot est adapté pour les apprentissages parce qu'il met en interaction divers niveaux et types de difficultés d'une tâche (Neboit & Poyet, 1991) : des caractéristiques techniques qui ont une incidence directe sur les niveaux de complexité de la tâche, des représentations mentales des opérateurs sur le dispositif et son action sur le réel, fortement dépendant du degré de transparence du système quant à son fonctionnement, des situations d'utilisation des dispositifs particulièrement contraignantes et instables et enfin, des compétences utilisées dans la programmation en partie liées au domaine de la représentation de l'espace et du mouvement, domaine souvent mal maîtrisé par les opérateurs.

- **Des difficultés plus techniques : une distance de l'imagination à la réalité**

Dans la programmation des robots, au-delà des difficultés conceptuelles liées aux problèmes algorithmiques (Nijimbere *et al.*, 2015), les étudiants rencontrent de fortes difficultés techniques. Le plus souvent difficiles à anticiper, elles créent des surprises aux étudiants.

L'enseignant E1 justifie les difficultés des étudiants à la distance qui existe entre ce qu'ils imaginent pouvoir faire faire à ce robot et ce qu'ils sont réellement capables de faire avec lui. Pour cet enseignant, ce sont des sujets qui ont l'air simples mais qui font un véritable choc aux étudiants surtout quand ils se heurtent à certaines difficultés qu'ils n'avaient pas anticipées. En effet, cette distance est très large dans le cas de NAO : la forme anthropologique de cette technologie donne l'impression d'une facilité de programmation ce qui n'est pas le cas. Pour cet enseignant, *« c'est difficile parce qu'il y a toujours une différence entre ce que l'étudiant s'imagine pouvoir faire, surtout sur les projets NAO, où c'est très anthropomorphique et la réalité de la combinatoire et la machine de Turing. (Rires). Et donc ça crée là… c'est un véritable verrou technique. »*.

Dans la programmation de NAO, les étudiants confirment les propos des enseignants concernant l'origine de leurs difficultés : le côté technique. Pour eux, *« c'est la précision de NAO »* qui leur a posé le plus de difficultés. Leurs conséquences se caractérisent par un écart qui se dessine souvent entre ce qu'ils prétendent programmer et ce à quoi ils sont réellement capables dans leur programmation : « il ne peut pas reproduire les gestes qu'on veut », « on s'imaginait qu'il marque un point et qu'il danse. On voulait faire beaucoup de choses, mais au final, on voit que c'était très compliqué voire impossible ! », précise un étudiant de L3.

En effet, les codes, après leurs implémentations et compilations sur l'ordinateur, sont transférés sur le robot pour être testés. C'est à ce moment que des difficultés techniques se faisaient remarquer. D'une part, ce n'est pas parce qu'un code compile bien qu'il fonctionne sur le robot. D'autre part, les codes sont exécutés sur des robots en mouvement, ce qui pose des problèmes d'équilibre (le robot manquait d'équilibre en voulant faire telle ou telle autre action ou geste) et de précision (il perd son chemin en suivant une cible donnée). Si certains défauts peuvent provenir des codes, d'autres proviennent de l'imperfection des robots utilisés. En effet, le robot, après avoir dû être conduit chez un spécialiste technique pour réparation, a été retourné avec les défauts finalement corrigés. Les étudiants s'exprimaient ainsi : *« Ce n'est plus le même NAO qui est retourné »*. Les difficultés et contraintes posées par des projets robotiques sont connues des enseignants qui en tiennent compte dans leurs évaluations.

- **Une mise en œuvre limitée de la modélisation**

Dans la programmation de NAO, certaines difficultés sont différemment vécues chez les étudiants. Pour le groupe B par exemple, leurs difficultés se situaient sur deux plans : « détecter la balle » et « attraper la balle ». Pour la deuxième tâche notamment, *« il [le robot] arrive à détecter une balle, se diriger vers elle, se mettre dans certaines positions, donc il se baisse mais n'arrive pas à récupérer la balle à coup sûr (...) en utilisant une analyse spécifique. (...). Pour nous, c'était le point le plus compliqué, le plus difficile ! »*. Affirmant ne pas avoir de difficultés au niveau de la programmation, ils situent le problème au niveau de la réflexion correcte à mener : « On avait plein d'idées, mais ça marchait pas en fait. Du coup, on n'avait pas la bonne en fait ! ».

Si NAO est plus évolué par rapport à d'autres technologies robotiques telles que LEGO MINDSTORMS NXT, son imperfection est vue comme une limite pour mettre en œuvre les stratégies implémentées. Par rapport aux vidéos vues sur Internet et qui constituent leur référence, ce groupe trouve moins perfectionné et moins puissant le robot utilisé : *« les NAO qui sont à l'extérieur qui ont été modifiés au niveau des articulations des moteurs sont, on peut dire, plus fiables, plus rapides et plus puissants. C'est pourquoi, dans les vidéos, quand ils prenaient la balle, ils ajustaient et pouvaient marquer. NAO qu'on a là, ce n'est pas possible. La balle ne fait même pas un centimètre devant lui, un centimètre c'est tout (Rires) ! »*. Même si ce groupe a finalement réussi à programmer ces tâches, il affirme y avoir investi beaucoup de temps : *« On se rappelle qu'on avait réussi à le faire baisser, ramasser la balle… On a passé tout un mois sur cela. On avait fait plusieurs façons de comment le bais-*

ser mais on n'arrivait toujours pas à le faire relever… Quand, on a réussi à le faire relever, je crois que tout le monde était content ! ».

3.3. Discussion

Avant d'entrer dans la discussion, rappelons les deux hypothèses qui ont été formulées. La première hypothèse stipule que les étudiants qui sont initiés à l'informatique depuis la première année de licence, n'ont pas de difficultés majeures avec la programmation des robots. Ceci est d'autant vrai pour NAO qui a l'essentiel des caractéristiques articulaires humaines. La deuxième hypothèse qui lui est complémentaire, stipule que la programmation de robots exige un certain niveau en informatique et permet donc d'aller plus loin dans l'apprentissage de la programmation.

a. NAO : une simplicité trompeuse

Programmer un robot de type humanoïde n'est pas chose aisée. Nous sommes parti de l'hypothèse que la programmation des robots ne pose pratiquement pas de difficultés aux étudiants de licence en informatique qui sont initiés à la programmation depuis la première année de licence. Mais, la réalité est toute autre. La forme humanoïde de NAO donne l'impression de simplicité et de facilité pour sa programmation. Mais, la programmation de NAO semble poser aux étudiants de L3 d'énormes difficultés auxquelles ils ne s'attendaient pas et cela malgré leur expérience en programmation informatique.

Cette situation est complexifiée par l'apprentissage d'un nouveau langage de programmation : le langage Python. En plus des difficultés algorithmiques, syntaxiques, techniques, etc., auxquelles les étudiants étaient confrontés, avec NAO, le contexte de la programmation embarquée est aussi une source de difficultés supplémentaire : un code qui marche sur l'ordinateur ne marche pas nécessairement sur le robot.

La complexification de la situation dans ce contexte de programmation des robots a été déjà relevée dans les recherches. L'ordinateur devient un « robot » intermédiaire entre l'apprenant et le robot qui doit exécuter le programme : le code est transféré sur le robot par l'apprenant via son ordinateur, pour être exécuté, une situation qui augmente la complexité de la programmation (Niboit et Poyer, 1991 ; De Dormale, 2003). Cette complexité est peut-être l'une des raisons qui expliquent le fait qu'il n'est prescrit aux étudiants que plus tard, depuis la L3.

Néanmoins, certains parmi eux ont produit un travail remarquablement intéressant. Certaines conditions ont contribué à ce qu'il en soit ainsi : une motivation engendrée par les rapports de sympathie des étudiants envers cette technologie, ce qui fait qu'ils restent accrocher et, des bouts de codes déjà construits et des vidéos obtenus grâce aux recherches en ligne.

Les résultats semblent infirmer la première hypothèse : les étudiants de L3 ont plus d'une année d'expérience en informatique mais les difficultés qu'ils ont vécues en programmation de NAO laissent penser que, sa programmation exige un certain niveau en informatique : sa programmation ne semble pas être pour des débutants complets mais nécessite une certaine maturité en informatique. Certaines variables semblent être prises en compte pour prescrire certains types de technologies robotiques aux apprenants : niveau d'étude, âge, expérience, etc. NAO semble plus complexe que les autres robots antérieurement programmés tels LEGO MINDSTORMS NXT. Ceci peut justifier finalement pourquoi il occasionne de difficultés non connues par les étudiants dans les classes précédentes et par conséquent pourquoi il est prescrit un peu plus tard dans la scolarité.

b. De l'affection envers la technologie à l'apprentissage avec la technologie

Les états affectifs des apprenants sont déterminants dans leur motivation pour l'apprentissage. La sympathie envers la technologie, plus prononcée dans le cas de NAO, semble être due à sa forme humanoïde. Cette apparence humaine leur a permis d'aller plus loin dans leurs apprentissages : leur affection envers lui les poussait à chercher à lui faire faire ce qu'ils pensent que tout humain peut faire. Ces résultats sont confirmés par d'autres recherches faites dans ce domaine. Se référant aux travaux antérieurs d'Izard (1993), Janiszek *et al*., (2011b) montrent que le rôle des émotions, ce qu'on peut mettre ici en relation avec la sympathie, n'est pas à négliger dans les apprentissages : des liens forts entre émotions et cognition ont été établis, soulignant que les émotions permettent de construire et d'organiser notre système cognitif et d'adapter nos procédures et notre comportement aux besoins de la situation. Selon ces études, les capacités d'attention, de mémorisation, de planification, d'apprentissage et d'interaction seraient davantage mobilisées si la tâche à accomplir est plaisante ou si on se sent satisfait et optimiste (Janiszek *et al.*, 2011b) : le plaisir et l'émulation qu'engendrent le travail et la communication avec NAO semble avoir eu un effet positif sur les capacités cognitives des étudiants et leur implication physique et intellectuelle dans la tâche à accomplir.

Après deux ans de formation à l'informatique, les étudiants de licence ont déjà acquis un bon niveau dans la programmation classique. Mais, les recherches montrent une distance entre la programmation classique des ordinateurs et celle des robots. Ce dernier contexte de programmation complexifie la situation. Dans la présente recherche, la complexité a été accentuée pour les étudiants par l'apprentissage d'un nouveau langage qui a ajouté des difficultés liées à la syntaxe, les bogues, etc. origine de la lenteur du projet. Le langage était encore une découverte à faire et la motivation initiale pour NAO s'en est trouvée réduite, les autres connaissances susceptibles d'être construites, limitées. Les difficultés dans l'apprentissage de nouveaux langages sont connues dans les recherches antérieures. Selon Leroux, des difficultés rencontrées en programmation des robots sont inhérentes à l'apprentissage de tout langage de programmation (Leroux, 1996). Il trouve intéressant de découvrir la technologie au travers des constructions de situations d'apprentissage favorisant une relation dialectique entre la théorie et la pratique. Quoi qu'il en soit, les étudiants étaient dans une situation beaucoup plus complexe que ça : si l'acquisition du langage n'était pas l'objectif premier de l'apprentissage, ils étaient contraints de l'acquérir pour l'utiliser dans la construction d'autres connaissances.

c. Robotique pédagogique (RP) : vers une recomposition des disciplines ?

Les résultats de cette étude montrent les potentialités de la RP pour une ouverture à l'acquisition, par les étudiants, de nombreux savoirs issus de disciplines ou domaines variés. Ce caractère transversal lui confère la qualité d'être « une branche de passerelle » ou « une discipline de surface» pour reprendre les termes de Daniel Marchand (1991), permettant « une navigation » à travers les disciplines et les domaines différents pour des apprentissages multiples. De plus, comme l'ordinateur, la robotique est souvent abordée sous deux facettes, comme outil mais aussi comme objet d'apprentissage. Si un ordinateur peut aussi permettre cette ouverture pour l'apprentissage de divers savoirs pluridisciplinaires, celle permise par la RP est plus large. D'une part, elle offre des contextes divers d'application des savoirs déjà acquis notamment dans diverses disciplines mais aussi, elle permet aux étudiants de construire d'autres connaissances permises par les interactions du robot avec l'environnement.

Les disciplines sont généralement closes et leur « intérieur » est généralement laissé aux seuls spécialistes du domaine. Leur décloisonnement est une des potentialités de la RP, même si cette qualité n'est pas son unique propriété. Duchâteau (1993) parle d'un éclatement

des frontières disciplinaires comme caractéristique fondamentale commune à l'informatique et à la robotique pédagogique. L'intérêt de cette ouverture des portes des disciplines, permise par l'approche de la programmation des robots, est d'offrir aux apprenants les possibilités d'accéder à l'interrelation des disciplines. Par là même, cet accès à la jonction des disciplines permet une acquisition de compétences transversales dont la responsabilité de leur enseignement n'est pas toujours clairement attribuée dans les curriculums enseignés (Lebeaume *et al*., 2011). La réflexion suscitée par la programmation des robots permet aux apprenants d'aller au-delà des limites disciplinaires pour interpréter les comportements de l'exécutant – robotique. Les potentialités de l'approche par projet de la RP dans les apprentissages sont multiples et ont été largement développées (Nijimbere, 2014) : problèmes complexes conduisant à des apprentissages complexes, motivation des apprenants et collaboration dans leurs apprentissages, utilisation et construction des connaissances multidisciplinaires, méta-réflexion permettant un développement de l'autonomie…

d. Robots humanoïdes et modélisation

Contrairement à la programmation classique, beaucoup de paramètres environnementaux (température, luminosité, pression, son…) entrent en jeu dans la programmation des robots. Les résultats de l'exécution du programme sont influencés par ces paramètres, indépendamment du code produit. Cela, semble-t-il, a influé sur les difficultés vécues par les étudiants : il était nécessaire d'anticiper pour comprendre et prévoir le comportement du robot face aux possibles fluctuations de ses comportements changeant au cours du temps en fonction de ces paramètres. Comme il s'agissait d'un contexte nouveau de programmation, il était très difficile d'anticiper tout ce qui pouvait se passer et influencer les comportements, d'autant plus que le robot reste une boîte noire pour eux.

Les difficultés liées à la complexité des tâches de programmation des robots ont été évoquées dans des recherches précédentes. Neboit & Poyet (1991) soulignent deux conditions qui rendent difficiles les représentations des mouvements du robot. La première difficulté est liée au nombre d'axes et de degré de liberté (ddl) sur les axes du robot. Ils évoquent la quasi-impossibilité de la décomposition d'un mouvement en déplacement d'axes élémentaires d'un robot dont tous les axes peuvent bouger simultanément, à des vitesses différentes, dans des orientations et des sens différents. La deuxième difficulté est liée aux mouvements très différents d'un robot selon une commande effectuée. Cette difficulté est, selon eux, justifiée par le fait que la commande des déplacements d'un même axe et pour une même valeur de déplace-

ment se fait dans plusieurs systèmes de coordonnées spatiales différents : référentiels polaires, articulaires, cartésien, repère de base, d'outil et de tâches.

Ces résultats révèlent des préoccupations des recherches actuelles en robotique éducative. Selon. Espiau & Oudeyer (2008), si les robots doivent être autonomes, la complexité du monde et, donc de l'environnement dans lequel ils évoluent, reste l'obstacle majeur à leur autonomie. Ils justifient ainsi les enjeux des recherches actuelles en robotique : concevoir des méthodes permettant aux robots de se mouvoir, de prendre des décisions et d'interagir habilement avec leur environnement, naturel et humain. Au-delà des connaissances mathématiques élémentaires utilisées, les paragraphes suivants montrent que, dans la programmation de NAO, des connaissances mathématiques avancées sont aussi indispensables pour garantir sa performance et son autonomie. En effet, les étudiants avaient du mal à corriger les déséquilibres fréquents auxquels le robot faisait face, lesquels témoignaient de l'existence des imperfections techniques du robot. Le gros problème pour les étudiants était de ne pas arriver à identifier ce qui manquait pour corriger ou améliorer la situation. Des questions que l'on peut se poser sont les suivantes : l'absence des connaissances mathématiques construites est-elle due au fait qu'elles n'étaient pas nécessaires dans ces projets ? N'y a-t-il pas des occasions où elles devraient intervenir ? Qu'apporterait l'utilisation de mathématiques un peu plus avancées dans ces projets ?

Programmer un robot évoluant dans un tel contexte complexe suppose une modélisation et une planification des différentes tâches en actions élémentaires. Si la programmation de NAO nécessite la construction des connaissances mathématiques un peu plus avancées, leur absence a fortement limité sa performance. Le manque d'équilibre de NAO et ses imprécisions dans les calculs des distances, les prises de positions, les mouvements de ses membres (pour ramasser des balles, pour marquer un but…)… sont quelques-unes des raisons qui, selon moi, sont justifiées par une absence des connaissances mathématiques, pourtant nécessaires, qui devraient être construites et utilisées. Par exemple, les changements d'angle sous lequel le robot, en mouvement, voit la balle, en position fixe, pour pouvoir se mettre en position « favorable » de son ramassage, renseignent sur une probable implication des variations d'angle en fonction du temps. Ceci laisse supposer l'intervention des équations différentielles. La nécessité de ces dernières a d'ailleurs été soulignée par les enseignants qui affirment leur contribution fondamentale dans la programmation de NAO. Mais elles n'ont pas été utilisées. Les essais des étudiants pour faire prendre ou bloquer NAO, « à la main », dans

une position « favorable » pour pouvoir poser tel ou tel geste souhaité, tel que le ramassage de balle, sont des approches qui se sont révélées non appropriées. Les pratiques des étudiants consistaient à faire prendre manuellement des positions qui, « informatiquement » n'ont pas de sens pour résoudre ce problème. Ces pratiques justifient à mon sens un aspect important de leurs représentations mentales de son fonctionnement : face à NAO, et donc inspirés par sa forme humanoïde, les étudiants raisonnent plus en termes de gestes humains qu'en termes de codes informatiques à implémenter. Cette façon de voir et de raisonner a influencé leur façon de faire et a masqué, chez eux, le recours à la modélisation des tâches complexes en petits éléments à implémenter et qu'il faudrait par la suite combiner. Cette approche des étudiants contraste avec le point de vue des chercheurs (Nonnon, 2002) pour qui la RP permet la modélisation mathématique du phénomène physique par une expression algébrique. Un peu après Nonnon, Paulin, et al., (2006) ont montré les limites d'un robot à « combiner ces différentes actions élémentaires pour qu'il (le robot) réalise des tâches de plus haut niveau ». Selon eux, la modélisation est, actuellement, devenue une préoccupation pour les roboticiens qui cherchent à modéliser les actions élémentaires sous forme de systèmes d'équations mathématiques complexes.

A côté des équations différentielles, évoquées comme indispensables, d'autres connaissances mathématiques avancées ont été soulignées dans les recherches : les équations dynamiques. En effet, Espiau & Oudeyer (2008) évoquent un premier défi à relever dans la programmation de NAO, système mécanique muni d'actionneurs : sa locomotion. Cette dernière appelle des équations dynamiques associées à un système rigide articulé et obtenues à partir des équations de Lagrange. Ces équations auraient pour intérêt de gérer des commandes engendrant une marche réaliste : supporter les variations de terrain, prévenir les chutes… Selon eux, à la différence des robots à base fixe, les robots humanoïdes, ont des exigences particulières. Comme Paulin et al., (2006), Espiau & Oudeyer justifient ces exigences par leur configuration spatiale répartie en deux parties : l'une, classique, regroupant les coordonnées articulaires commandées par des moteurs et définissant la posture du robot hors toute influence extérieure (pesanteur…) ; l'autre regroupant ses coordonnées par rapport à un repère de référentiel de son environnement.

Si les déplacements de ces robots sont effectués au moyen des jambes, les difficultés posées par leur programmation ne se limitent pas, dans le cas des robots humanoïdes, à la modélisation des seules jambes. Pour améliorer les déplacements et l'agir du robot, il importe de tra-

vailler sur le corps complet (Espiau & Oudeyer, 2008) : jambes, pieds, bras, mains, système de perception…, un travail qui nécessite chaque fois des modélisations. Il est notamment question de modéliser des contacts des pieds avec le sol pour faire face à l'influence de forces de réaction avec le sol, modéliser des frottements pour optimiser les mouvements afin de pouvoir faire face aux contraintes éventuelles de glissement et de synchroniser les déplacements et les mouvements des bras et la vision… Dans cette dynamique de la modélisation, une présence des mathématiques est de plus en plus évoquée en robotique. En tant que créature artificielle, NAO fonde son existence sur la puissance de l'algorithmique et des mathématiques comme outils de représentation du réel (Espiau & Oudeyer, 2008). Ces chercheurs jugent indispensables des modèles mathématiques dans la programmation des robots humanoïdes pour assurer l'autonomie de leurs décisions et faire des actions réfléchies. Ces modèles devraient s'appuyer sur son environnement et permettre des représentations géométriques et topologiques mais aussi des modèles perceptifs. Des modèles mathématiques de raisonnement interviendraient en utilisant des techniques d'optimisation combinatoire (recherche de la meilleure solution parmi un nombre fini de solutions) ou de la programmation dynamique pouvant permettre une prise de décision.

NAO, à l'instar de beaucoup d'autres d'environnements orientés compétition pour la RoboCup (Muratet, 2010) comme RoboCupSoccer, RoboCupRescue et RoboCupJunior, est un robot physique qui pose plus de difficultés aux apprenants : il nécessite beaucoup de concepts informatiques. Même si les étudiants de L3 disposent des connaissances de base qu'il faut, la diversité des paramètres intervenant rend la programmation de NAO complexe et contraignante. L'acquisition du nouveau langage de programmation Python y a aussi contribué.

Il me semble que le peu d'étudiants qui choisissent les sujets de programmation de NAO est dû à leur mauvais souvenir de L2 lié à la complexité des activités de programmation des robots, notamment vécue dans le cadre des projets de programmation des robots de type LEGO MINDSTORMS. La deuxième hypothèse stipulant que NAO, dont la programmation est très exigeante en connaissances, ne serait pas destiné aux débutants en informatique mais, servirait à aller plus loin en programmation, se voit confirmée par ce qui précède. La modélisation exigée par NAO nécessite un bon niveau de réflexion et un bagage de connaissances de base. La programmation des tâches modélisées appelle souvent des connaissances un peu plus avancées que ce soit en informatique ou en mathématique. La programmation de NAO, par sa morphologie et sa structure, devient plus complexe. Il semble pour cela que programmer

NAO n'est pas une activité adaptée aux tous novices en informatique mais, permettrait d'aller plus loin dans l'acquisition des compétences en programmation à ceux qui ont déjà des connaissances de base.

4. Conclusion du chapitre

Dans ce chapitre, nous nous sommes intéressés à la construction des connaissances par la programmation des robots chez les étudiants de licence, respectivement le robot NAO en L3 et LEGO MINDSTORMS NXT en L2. Moyennant certaines conditions favorables, préalables pour sa meilleure efficacité (Laborde et al., 1985) telle que des connaissances de base, l'approche de programmation des robots semble fructueuse et porteuse d'un intérêt éducatif important. Au-delà des apprentissages en informatique, des connaissances pluridisciplinaires et non disciplinaires sont construites grâce à la potentialité de la RP à décloisonner les frontières imposées par des spécificités disciplinaires.

L'absence d'une approche de modélisation a sensiblement limité la mise à profit de certaines des stratégies imaginées et a réduit, par conséquent, la construction de certains apprentissages mathématiques, indispensables pour la performance de NAO. En effet, en licence, les étudiants ont un certain nombre de cours orientés vers l'apprentissage de la programmation (Delozanne et al., 2011). Contrairement à la programmation robotique, les recherches montrent que les mêmes étudiants vivent moins de difficultés en programmation classique (Nijimbere et al., 2013). Nous pouvons supposons qu'après une à deux ans d'apprentissage de la programmation, ils avaient déjà acquis un bon niveau de pratique de cette activité. Comparativement à LEGO MINDSTORMS NXT, le contexte de la programmation de NAO est complexe et occasionne de difficultés plus importantes, orientées techniques. Elles semblent accentuées par les multiples ddl de NAO et les conditions environnementales changeantes dans lesquelles NAO était appelé à évoluer et qui influent sur ses mouvements et ses actions.

Ces difficultés auraient nécessité des connaissances mathématiques un peu plus avancées, construites à l'aide de l'approche par modélisation, malheureusement absente. Les limites des étudiants sont lieu au fait que leurs représentations mentales qu'ont de NAO les poussent à raisonner en termes de mouvements humains et par conséquent à lui faire prendre des positions jugées « favorables » « à la main » plutôt qu'en termes de codes informatiques à construire. Ceci a contribué à ne pas penser à l'usage de la modélisation pouvant leur per-

mettre la décomposition en actions élémentaires des tâches à faire faire par NAO et ainsi arriver au bout de ces difficultés techniques par des mathématiques un peu plus avancées.

Leurs pratiques laissent transparaître des représentations mentales de NAO lacunaires : il reste une boîte noire. La maîtrise d'une technologie nouvelle telle qu'un robot exige plus qu'une simple connaissance des composantes du hardware, de la syntaxe et de la sémantique de programmation. Une connaissance préalable des principes de son fonctionnement est nécessaire (Gaudiello & Zibetti, 2013). Pour le cas de NAO, ces derniers manquent chez les étudiants de licence malgré leurs compétences reconnues dans la programmation classique.

Mon hypothèse est que des séances d'initiation et de familiarisation à sa structure et son fonctionnement contribueraient à leur faire acquérir des images mentales nécessaires et suffisantes de NAO, robot très complexe. Elles leur permettraient aussi d'anticiper certains de ses comportements sous l'influence des divers paramètres de son environnement et ainsi pouvoir aller plus loin dans sa programmation. Programmer NAO à l'aide d'un langage déjà connu serait encore, me semble-t-il, un moyen de réduire les contraintes pour aller plus loin dans la modélisation afin d'augmenter la performance de ce robot.

Il semble, par rapport à la programmation classique et des kits robotiques, que la programmation des robots humanoïdes, dont NAO fait partie, est très exigeante et nécessite un certain niveau en programmation. L'activité de programmation des robots humanoïdes semble tellement complexe qu'elle serait, par conséquent, un peu plus haut placée par rapport au niveau des débutants en informatique : elle constituerait, selon nous, non pas un début mais une étape pour un perfectionnement en programmation, ce qui justifie finalement sa prescription institutionnelle tardive en licence.

Le chapitre IX suivant clôture la thèse. Nous discuterons les résultats obtenus, conclurons et proposerons des perspectives de recherche.

Chapitre IX DISCUSSION GÉNÉRALE, CONCLUSION ET PERSPECTIVES

Cinq et trois ans après la mise en place respectivement des nouveaux programmes de mathématiques avec une composante informatique au lycée et la spécialité ISN en terminale S, cette recherche d'orientation didactique a eu pour objectif d'étudier comment les enseignants et leurs élèves se sont approprié l'enseignement et l'apprentissage des notions informatiques prescrites et, en particulier les notions de variables et de boucles, incontournables pour une initiation en informatique. L'étude a procédé par une approche comparative des pratiques des lycéens avec celles des étudiants de licence en informatique. Les étudiants de licence sont eux aussi débutants en informatique.

Les deux publics d'apprenants construisent en général leurs connaissances dans un contexte de projets. Nous nous situons dans un cadre conceptuel de la théorie de l'activité et, une approche qualitative combinant plusieurs outils (analyse du prescrit, observation, questionnaire, entretien semi-directif et groupé) ont été utilisés.

Avant d'entrer dans la discussion, rappelons brièvement deux questions générales de recherche et les hypothèses respectives qui ont guidé notre réflexion. Comment les élèves et les étudiants débutants en informatique s'approprient-ils les savoirs informatiques ? Comment les enseignants, spécialistes des mathématiques, prennent en compte les savoirs informatiques dans leurs pratiques professionnelles ?

Au regard de ce questionnement, nous avons formulé deux hypothèses générales suivantes. La première hypothèse est qu'en apprentissage collaboratif notamment les apprentissages en contexte de projet, les élèves ou étudiants plus ouverts, plus communicants et plus collaboratifs s'approprient mieux les savoirs informatiques construits. La deuxième hypothèse est que les enseignants spécialistes de mathématiques dont les représentations de l'informatique sont proches des mathématiques adhèrent mieux à l'enseignement de ces nouveaux programmes de lycées.

1. Limites de la thèse

Avant de discuter nos résultats et conclure notre recherche, nous avons choisi de présenter les limites de la thèse parce qu'elles nous semblent importantes à prendre en compte pour

mieux apprécier des discussions et conclusions qui vont suivre. Ces limites tiennent à cinq éléments principaux :

1. un accès difficile et perturbé aux terrains de recherche et à la population expérimentale,
2. une population réduite ne permettant pas une généralisation des résultats obtenus,
3. une méthodologie qualitative de type ethnographique utilisée qui a été rendue difficile par une longue période de collecte de données et par une population expérimentale dispersée,
4. des pratiques contrastées qui ont rendu difficile la conception d'une grille d'analyse unique
5. une orientation comparative rendue difficile par des contextes variés de pratiques et d'apprentissages des populations concernées.

Initialement, l'idée de la recherche était l'étude des pratiques des élèves et leur évolution au cours du temps dans la mise en œuvre des nouveaux programmes du lycée. La recherche devait avoir lieu dans un seul établissement et se focaliser sur des classes bien spécifiées : une classe de seconde, une classe de première, une classe de terminale et une classe de spécialité ISN. Cela aurait permis d'accéder de plus près aux pratiques des élèves et de leurs enseignants, de suivre le processus de construction des connaissances des élèves, leur évolution mais aussi les difficultés vécues au cours de l'enseignement et d'apprentissage. Un seul terrain nous aurait permis d'entrer en profondeur dans nos questions.

Malheureusement, il a fallu quatre établissements pour avoir toutes les données nécessaires. Cette multiplicité d'établissements a rendu difficile l'utilisation plus fine de la méthodologie escomptée : la méthodologie qualitative de type ethnographique. Elle s'est accompagnée de la complexité de situations (horaires différents, rendez-vous dispersés des enseignants, contraintes administratives pour accéder dans un établissement puis dans une classe, entretiens dans l'établissement et observations dans un autre. Bref, ce contexte semble nous avoir fait perdre le caractère idéal de la méthodologie ethnographique initialement envisagée.

Une autre limite dans l'utilisation de cette méthode est liée au nombre limité de séances des enseignants de mathématiques faisant intervenir l'algorithmique. Généralement abordée en demi-classe, au lycée A, notre terrain principal, l'algorithmique apparaît au plus une fois le mois. Si les occasions de faire intervenir l'algorithmique dans les séances d'enseignement

étaient limitées chez les enseignants, les chances d'y être convié étaient encore plus réduites pour deux raisons. D'abord, il fallait régulièrement demander aux enseignants quand ils allaient faire l'algorithmique pour négocier l'accès à la séance de cours. Ensuite, ils ne semblaient pas toujours très favorables à un œil observateur extérieur, ce qui a aussi limité l'accès au cours. Toutes ces circonstances semblent avoir rendu difficile l'utilisation de la méthodologie préconisée, bien qu'elle soit adaptée à cette étude.

D'autres limites de la thèse sont liées aux pratiques enseignantes. Après une séance d'enseignement, nous aurions bien voulu avoir des occasions d'observer les élèves seuls, en train de travailler ensemble et d'échanger entre eux sur des problèmes à résoudre, sans l'influence de l'enseignant comme c'était pour le cas des étudiants à l'université. Cette situation pose le problème de l'autonomie. Si, à l'université, les étudiants sont plus autonomes et ont plus de temps libres sans cours, au lycée, la situation est différente : des créneaux d'un travail en groupe en dehors des cours sont généralement hors établissement.

La situation est restée la même au lycée C avec l'ISN. La quasi-totalité du travail en projet s'est faite en salle de classe qui servait aussi de salle informatique. L'enseignante était toujours avec eux et des séances de cours ont été dédiées aux projets. La salle était fermée par le même enseignant juste après le cours : les élèves n'avaient pas droit d'y être seuls en l'absence de leur professeur. Cette situation semble limitative des occasions de travail des élèves, des occasions libres où les pratiques interactives des élèves devraient avoir lieu.

Une autre variable qui semble avoir limité les résultats de la thèse est l' « ancienneté » de la catégorie d'enseignants dont les pratiques ont été observées. Certains professeurs sont de la génération des programmes Bourbaki, époque où les mathématiques enseignées étaient extrêmement théoriques, ce qui fait que, comme le dit cette enseignante, l'algorithmique est perçue comme totalement décalée par rapport à leur formation initiale. Si des jeunes enseignants ont été interrogés, leur absence parmi la population observée, a limité la confirmation des effets de cette variable « ancienneté ». L'enquête par questionnaire révèle une adhésion des jeunes enseignants de mathématiques à l'enseignement de l'informatique en mathématique avant même des stages de formation : des observations de classes auraient apporté un complément sur leurs déclarations. Seule une enseignante parmi ceux qui ont suivi un stage de formation a été observée en contexte de pratique de classe. De plus, cette enseignante était volontaire et motivée pour cet enseignement, ce qui n'est sans doute pas le cas pour une partie de ceux qui ont été formés.

Si l'enquête par questionnaire révèle une adhésion des jeunes enseignants de mathématiques à l'enseignement de l'informatique avant même des stages de formation, des occasions d'observation de leurs pratiques de classe, n'ont pas été possibles. En nous contentant de leurs déclarations, il apparaît que la variable « jeunesse dans le métier » occupe une grande place dans l'adhésion des enseignants à l'enseignement de l'informatique, les moins anciens étant les plus favorables à cet enseignement. Néanmoins, une population de jeunes professeurs aurait été nécessaire pour confirmer ou infirmer leurs déclarations. Appliquée à cette population jeune d'enseignants, une approche ethnographique aurait contribué à la vérification par une analyse comparative avec les pratiques précédentes.

Une autre limite de cette thèse est son orientation comparative des pratiques des apprenants qui ne sont pas dans les mêmes contextes d'apprentissages. Par exemple, les élèves d'ISN et les étudiants sont tous en projets de programmation, mais leurs contextes d'apprentissages étaient différents et avec des exigences différentes. Ceux d'ISN étaient en programmation classique alors que les étudiants étaient en programmation des robots.

Enfin, les résultats obtenus ne sont pas généralisables parce que la population expérimentale sur laquelle a porté l'étude est très petite pour être représentative de la population enseignante de l'informatique en mathématiques et de l'option ISN au lycée en France.

La discussion générale suivante concerne les résultats principaux relatifs aux questions et hypothèses générales de recherche.

2. Discussion générale

Nous allons focaliser notre discussion sur trois points : les tensions entre le prescrit et les pratiques en situation d'enseignement/apprentissage, l'impact de l'informatique sur le changement des pratiques dans l'enseignement et apprentissages des mathématiques et enfin, les facteurs qui soutiennent l'évolution des pratiques chez les enseignants et des apprentissages chez les apprenants.

2.1. Des tensions entre le prescrit et les pratiques

L'enseignement de l'informatique en mathématiques au lycée (classes de seconde, de première et de terminale), en dehors de la spécialité ISN, semble se heurter à des difficultés dues aux prescriptions institutionnelles. Les tensions entretenues dans les prescrits (programmes et

manuels) à propos de l'algorithmique ne laissent pas neutres les pratiques des enseignants et de leurs élèves.

Le premier obstacle qui est aussi une source de tension est lié à la contrainte institutionnelle d'enseigner l'algorithmique en mathématiques, de façon transversale. Cette injonction de nature systémique, semble ajouter une contrainte supplémentaire aux enseignants de mathématiques sans bagage informatique suffisant. Une telle situation complexifie leur exercice didactique (par exemple dans leur choix d'activités) et déstabilise leurs pratiques professionnelles.

Si les manuels sont souvent « imités » qu'utilisés chez les enseignants (Bruillard, 2005), ces derniers se heurtent à l'interdiction institutionnelle de faire un cours théorique d'algorithmique alors que ce cours est déjà préparé en début des manuels scolaires utilisés.

Les contraintes auxquelles les enseignants semblent confronter dans l'exercice de leur mission sont liées à la contrainte d'enseigner directement l'informatique à l'intérieur des chapitres de mathématiques. Qualifiée d'« *enseigner l'algorithmique sans l'enseigner* » par les formateurs, cette pratique complexifie et complique la pratique enseignante. Au regard des résultats, les vraies pratiques des enseignants dans la de mise en œuvre de ces programmes semblent celles du passage de l'enseignement *de* l'algorithmique en mathématiques à l'enseignement des mathématiques *avec* l'algorithmique : l'aspect logique de l'algorithmique est visiblement enseigné chez certains enseignants, et l'algorithmique devient un « outil » au service des mathématiques. Cela n'est pas l'esprit du nouveau programme de mathématiques au lycée.

Contrairement à la confusion caractérisant le prescrit relatif à l'informatique en mathématiques, l'option ISN s'en démarque : son programme est conçu pour aire acquérir des bases d'une culture informatique générale large, construite sur les concepts informatiques de base, et acquise au moyen d'un seul langage au choix, à travers diverses formes de séances de cours. Les aspects algorithmique et programmation y sont abordés de façon complémentaire.

Néanmoins, si son curriculum ne suscite pas de confusion, sa mise en œuvre met en évidence deux tensions liées d'une part aux niveaux différents des élèves et d'autre part au programme d'enseignement ambitieux. Deux tensions peuvent aussi être soulignées. La première est vécue comme un paradoxe chez les enseignants. Alors qu'ils sont obligés à courir après le pro-

gramme très long pour le terminer, ils doivent en même temps tenir compte des hétérogénéités des élèves en informatique.

La deuxième tension, liée à la première, a lieu entre le professeur et les élèves passionnés de l'informatique. Ayant choisi l'ISN pour aller plus loin dans leurs apprentissages, ils ne sont pas toujours satisfaits des pratiques de leur professeur qu'ils trouvent limitant. La première raison est que l'enseignant doit aussi s'intéresser aux tous débutants en informatique en commençant par présenter les notions jugées de base. Si ces notions sont bien connues par certains élèves, ces derniers considèrent ce passage comme une perte de temps, les empêchants d'apprendre plus. Alors que l'enseignant utilise un seul langage au choix, les élèves voudraient s'approprient plusieurs. Par exemple, cette passion d'apprendre de langages s'est observée lorsqu'un binôme du lycée B s'est proposé de faire chacun tout le projet mais avec des langages différents pour comparer leurs performances.

Une autre raison semble liée à la formation non suffisante des professeurs en informatique, ce qui les empêche d'avoir suffisamment de recul pour certains chapitres qui, en conséquence semblent banalisés et limitent les apprentissages dont les élèves devraient bénéficier.

Nous développons cette sous-partie en deux points : tensions dans les pratiques des enseignants d'une part et, les tensions entre les pratiques des enseignants et celles de leurs élèves d'autre part.

a. Des pratiques enseignantes contrastées : entre adhésion et résistance ?

L'analyse du prescrit révèle une confusion présente dans les textes officiels (programmes et guide d'accompagnement) et reprise par les manuels scolaires : une pratique de l'algorithmique et une écriture de programmes sont indistinctement mêlés. Régulièrement ponctué d'aspects technologiques, le prescrit au lycée, relativement à l'informatique en mathématiques, laisse penser à un enseignement de l'algorithmique qui ne peut pas être envisagé sans logiciel et donc en dehors de la programmation informatique en mathématiques.

Les résultats mettent en évidence que cette « pollution technologique » n'aide pas les enseignants de mathématiques à clarifier ce qui relève des mathématiques de ce qui relève de la programmation informatique et par conséquent, limite leur adhésion à cet enseignement.

Les représentations des enseignants et leurs rapports vis-à-vis de l'informatique influent sur leur adhésion à son enseignement. Le premier est celui des enseignants motivés par ces

programmes parce que, selon eux, ils apportent un renouveau pour l'enseignement des mathématiques. En creusant un peu plus, le constat est que ces enseignants ont un passé particulier par rapport à l'informatique et ces programmes sont pour eux une occasion de mettre en pratique leurs connaissances. Ce groupe peut être réparti en trois catégories.

La première est celle des jeunes enseignants de mathématiques qui ont eu des modules informatiques dans leur formation initiale à l'université. La deuxième catégorie est formée de ceux qui ont suivi des formations d'informatique en privé ou en autodidacte. La troisième catégorie est formée d'enseignants qui sont des formateurs ou ex-formateurs d'autres enseignants.

Dans cette catégorie, certains enseignants ont d'abord adopté l'approche de la programmation informatique comme le centre d'intérêt de leurs pratiques avant que le stage de formation ne recommande des pratiques focalisées sur l'approche papier-crayon. Leur difficulté, qui reste d'ailleurs générale, est la transversalité de l'enseignement de l'algorithmique sans avoir préalablement fait un cours théorique. Face à cette difficulté, les stagiaires ont été amenés à faire petit cours théorique afin de retravailler dans la suite les notions au sein des chapitres.

Pour ce groupe d'enseignants, il semble que ce n'est pas seulement parce qu'ils rapprochent l'informatique aux mathématiques qu'ils adhèrent à cet enseignement mais, parce qu'ils trouvent les deux sciences sont complémentaires : il y a un apport pressenti de l'informatique dans l'enseignement des mathématiques. C'est comme s'il y a des limites de l'enseignement des mathématiques qui peuvent être comblées par l'informatique. Éric Bruillard et Maurice Nivat semblent le confirmer. Selon Bruillard, malgré un lien complexe entre mathématiques et informatique, cette dernière ne se réduit pas aux mathématiques (Bruillard, 2009). Se référant à Maurice Nivat (2007),

Bruillard précise que ce sont des algorithmes qui assurent ce lien et font de l'informatique une discipline qui instrumente les mathématiques et contribue à son évolution en associant le calcul et le raisonnement. C'est donc plus l'apport présumé de l'informatique à améliorer et faire évoluer l'enseignement des mathématiques qui semble motiver leur adhésion à cet enseignement. L'hypothèse II est validée.

Un deuxième groupe est celui des enseignants non motivés par cette composante informatique du programme de mathématiques pour laquelle où ils se sentent illégitimes

d'enseigner. Faut-il parler de formes de résistance ou d'illégitimité pour certains enseignants de mathématiques en ce qui concerne la mise en œuvre de ce programme ? Des effets de « réticence » voire de « résistance » semblent se sont observés suivant que les conceptions des enseignants de l'algorithmique sont plus ou moins proches de l'informatique que des mathématiques. Ce qui est clair, il semble y avoir eu au lycée A, une forme de solidarité négative quant à la non disponibilité de tous les enseignants de mathématiques pour participer au stage de formation à l'algorithmique. Faut-il parler de contrainte institutionnelle d'horaire comme certains le disent ? Il semble que non parce qu'ils ont participé à d'autres stages dans d'autres domaines mathématiques (probabilités et statistiques, usages de quelques logiciels de géométrie dynamique en maths…). Il semble que les formations longues réclamées pour l'algorithmique, autres que celles organisées les mercredis, sont des prétextes. Néanmoins, sa présence au baccalauréat semble avoir amené les enseignants à fournir un effort : ils se disent contraints de l'enseigner.

Les enseignants de mathématiques ont en effet des difficultés à intégrer les outils informatiques. Ces difficultés ont été analysées en didactique des mathématiques (Artigue, 2002) comme provenant de la non-neutralité des outils sur les mathématiques à enseigner (techniques, objets et symbolisations modifiées, apparition de nouveaux objets et symboles, de nouvelles techniques…).

Dans le cas du tableur par exemple, Haspekian (2005) montre que les résistances des enseignants seraient dues à une trop grande distance, dite instrumentale (générée par l'instrument), entre les mathématiques telles qu'ils les conçoivent et celles que font vivre l'instrument. Des aspects sociaux, culturels et épistémologiques contribuent à créer de la distance.

L'algorithmique et les savoirs informatiques en général prescrits en mathématiques ne sont pas un « nouvel outil », mais un nouveau domaine à enseigner (Haspekian & Nijimbere, 2012). En ce sens, ils ne génèrent pas eux-mêmes une distance comme le fait l'outil tableur, mais comme dans les représentations des enseignants, ils sont plus proches de l'informatique que des mathématiques, des phénomènes de résistance analogues à ceux créés par la distance instrumentale s'observent. Ces résistances semblent être justifiées par le sentiment d'illégitimité vécu par certains enseignants de mathématiques face à cet enseignement.

Par contre, une adhésion à ces programmes s'observe chez certains enseignants qui semblent avoir un passé particulier. Certains se sont intéressés à l'informatique avant même ces nouveaux programmes et ont déjà eu des formations en dehors des contextes professionnels ou en autodidacte. D'autres sont ou ont été des formateurs d'enseignants en formation continue et enfin, d'autres sont des jeunes enseignants dans le métier, ce qui laisse penser pour eux qu'ils ont bénéficié de modules informatiques dans l'enseignement supérieur. Les pratiques de certains, initialement orientées programmation informatique, ont été par après le stage formation, focalisées sur l'approche papier-crayon de l'algorithmique.

b. Tensions enseignant-élèves : papier-crayon ou clavier ?

Une autre tension s'observe entre les deux principaux partenaires de l'action didactique dans la mise en œuvre de ces programmes : les élèves et leur enseignant. Un enseignant non favorable à la programmation (et donc un travail sur machine) se trouve en tension avec des élèves en difficulté avec la rigueur algorithmique. Si la volonté des élèves est de n'utiliser qu'une machine pour « fuir » les difficultés vécues par le travail « à la main », elle est rarement honorée par les enseignants pour deux raisons. La première est que cette pratique n'est pas formatrice : exécuter un programme déjà prêt sans travail préalable de recherche sur papier-crayon de sa construction est pédagogiquement inefficace. La deuxième raison est que certains enseignants passent beaucoup moins de temps à l'activité de programmation. Rarement, des algorithmes déjà construits à tester sur machine sont donnés pour adoucir la tension, sans toutefois aller plus loin pour semble-t-il éviter le discrédit par leurs élèves : ne pas être capable d'apporter une aide dont un élève peut avoir besoin en programmation. Cette tension correspond à celle de quatrième niveau de la TA d'Engeström.

Cette tension semble due à une recherche de facilité chez les deux partenaires de l'action didactique : élèves et enseignants. Si les premiers se heurtent, dans cette approche orientée logique, à des difficultés conceptuelles difficilement franchissables, cette approche est préférée chez les enseignants par sa proximité aux mathématiques. De même, si les élèves veulent seulement faire directement la programmation sans passer par la phase algorithmique au papier-crayon, c'est parce que cette approche leur permet le bidouillage, sanctionné par un feedback direct de leurs essai-erreurs alors que l'approche papier-crayon ne leur permet pas de savoir si ce qu'ils ont fait est vrai ou pas avant l'intervention de l'enseignant. Certains élèves ont du mal à affronter et se casser la tête avec le travail « à la main » en algorithmique, une

situation qui peut être expliquée par le phénomène de « dévolution » de Guy Brousseau (1990).

Dans la mise en œuvre du programme d'ISN, deux tensions peuvent aussi être soulignées. La première vécue comme un paradoxe chez les enseignants, a lieu entre l'enseignant et le programme du cours. Alors qu'ils sont obligés de courir après le programme très long pour le terminer, ils doivent en même temps tenir compte des niveaux hétérogènes des élèves en informatique.

La deuxième tension, liée à la première, a lieu entre le professeur et les élèves passionnés de l'informatique. En effet, certains élèves ont choisi l'ISN pour découvrir davantage l'informatique et même dirons-nous, pour aller un peu plus loin en informatique en général et en programmation en particulier parce qu'eux ont déjà des bases en programmation. Ils ne sont malheureusement pas toujours satisfaits des pratiques de leur professeur qu'ils trouvent limitant pour deux raisons essentielles.

La première raison est que l'enseignant doit aussi s'intéresser aux élèves totalement débutants en informatique en commençant par présenter les notions jugées de base. Si ces notions sont bien connues par certains élèves, ces derniers semblent considérer ce passage comme une perte de temps qui les empêche d'apprendre des notions avancées notamment liées aux différents langages de programmation. Il y a contradiction entre les pratiques de l'enseignant et de celles de certains élèves. L'enseignant choisit de focaliser ses pratiques sur un langage ciblé alors que certains de ses élèves voudraient s'en approprient plusieurs. C'est le cas par exemple d'un binôme du lycée B, passionné de programmation, qui s'est proposé de faire chacun d'entre eux tout le projet mais avec des langages différents pour comparer leurs performances.

Une autre raison semble liée à la formation non suffisante en informatique des professeurs d'ISN. Cela ne manque pas de conséquences. Les enseignants manquent du recul suffisant pour certains chapitres, ce qui pousse à banaliser certains ou leurs parties. Un exemple qui peut être donné est celui de la robotique qui, tout en motivant les élèves d'ISN au lycée C, n'a pas été suffisamment exploité en cours ou n'a pas constitué un sujet de projets. Dans ce lycée, les élèves n'ont pas bénéficié des contenus qui pourtant étaient prescrits pour enseignement. Dans le lycée B, ce chapitre concernant la robotique et donc la programmation des

robots a été abordé en cours et un binôme a réussi à programmer un robot LEGO MINDSTORMS NXC[132] ; les élèves de ce binôme déclarent être fiers des compétences acquises.

Si une telle situation perdure, elle risque d'occasionner des « fractures scolaires » : les élèves fréquentant des établissements dotés d'enseignants motivés ou spécialisés pour l'informatique deviennent privilégiés que d'autres dans leurs formations.

2.2. L'informatique en mathématiques : des changements de pratiques ?

L'enseignement de l'informatique en mathématique et, par conséquent, par des spécialistes des mathématiques, a été différemment accueilli chez les enseignants spécialistes des mathématiques. Leurs pratiques restent contrastées. Certains se sont montrés enthousiastes par ces nouveaux programmes avec une composante informatique qui apporte un renouveau dans leurs pratiques d'enseignement des mathématiques. D'autres, moins motivés, ont affiché des réticences voire des résistances à cette « intrusion » de l'informatique en mathématiques, les obligeant ainsi à être « à la fois prof de maths et d'informatique ». Le refus de certains à participer à des stages de formation organisés semble donner un signal éloquent de résistances Dans tous les cas, que les enseignants de mathématiques aient suivi ou pas un stage de formation à l'algorithmique, l'inscription de l'algorithmique aux épreuves de baccalauréat semble les avoir contraints à cet enseignement.

a. Transversalité vs chapitre à part de l'algorithmique?

Les pratiques des enseignants de mathématiques en informatiques sont contrastées. Certains ont vite adhéré à cet enseignement : ils étaient motivés. D'autres ont affiché des réticences voir des résistances avant de s'accrocher petit à petit. D'année en année, ils s'y familiarisent et « augmentent la dose ». Le grand problème des enseignants semble être la transversalité de l'informatique à travers les autres chapitres. Les choix de certains d'entre eux à faire un petit cours théorique avant visent à contourner cette contrainte : ils semblent plus à l'aise que les autres dans cet enseignement.

Les enseignants de mathématiques peuvent enseigner ces notions algorithmiques à condition qu'elles soient d'abord enseignées comme chapitre à part avant de servir d'instruments pour

132 C'était le robot LEGO MINDSTORMS NXC qui circuler dans le labyrinthe à la recherche d'une sortie. Précisons qu'un robot similaire, avec le langage NXT a été programmé par les étudiants de licence 2 à l'université.

les autres chapitres. Pour cela, il serait vu au début de l'année et avant les autres chapitres pour permettre son instrumentalisation dans la suite du programme.

b. Des pratiques contrastées aux risques de fractures ?

Malgré la collaboration entre les enseignants, il y a des différences entre les pratiques entre les enseignants motivés et formés et d'autres.

Au lycée, contrairement aux attentes des programmes selon lesquelles l'informatique en mathématiques peut relever le niveau des élèves faibles en mathématiques, les résultats montrent qu'à part quelques exceptions, les « très bons » élèves en informatique en général et, en algorithmique en particulier, sont ceux qui sont normalement « bons » en mathématiques. Deux raisons peuvent expliquer cette situation. La première est liée à l'informatique en mathématiques. Si l'algorithmique est logiquement rigoureuse, cette rigueur est aussi incarnée par les mathématiques.

Ceci peut expliquer en partie les forts liens entre les mathématiques et l'algorithmique (Modeste, 2012b). Néanmoins, le fait que les élèves ne trouvent en algorithmique ni des mathématiques ni de l'informatique, mais un domaine scientifique nouveau à découvrir, peut servir d'une porte d'entrée pour aborder les mathématiques autrement si bien qu'elles sont souvent vues comme très théoriques et donc difficiles. La seule condition pour cet enseignement semble d'être assuré par un enseignant motivé associant l'enseignement de l'informatique en mathématiques avec de la programmation informatique : cette dernière, non excessive, semble soutenir la motivation des élèves habitués à manipuler des technologies numériques.

La deuxième raison est liée à l'option ISN. Les élèves de cette option ont des profils différents et des rapports différents aux mathématiques. Les résultats révèlent que s'ils réussissent quel que soit son profil d'origine à condition d'être motivé pour la spécialité, ceux qui l'ont choisie comme refuge par peur des spécialités mathématiques ou physique-chimie, avaient plus de difficultés : ils doivent fournir plus d'efforts que les autres et leur réussite demande un grand coût. Par contre, si certains élèves « bons » en mathématiques ont préféré faire l'option ISN à laquelle ils réussissent d'ailleurs sans peine, ils se disent capables de faire n'importe quelle spécialité : ils n'ont donc pas de contrainte dans leur choix.

Enfin, une autre raison montrant l'influence des rapports aux mathématiques sur les pratiques en informatique chez les élèves peut être tirée dans les pratiques des étudiants de licence. Le manque d'équilibre et de précision des robots programmés semble être justifié par des no-

tions mathématiques modestes utilisées par les étudiants. Des modèles mathématiques avancés (équations différentielles, équations de Lagrange, optimisation combinatoire, programmation dynamique…) associées à la modélisation, pourtant limitée dans les pratiques des étudiants, auraient amélioré les mouvements des robots, leur précision et leurs équilibres, et par conséquent, leurs performances surtout dans le cas de NAO (Nijimbere, 2014).

c. Une informatique quasi « débranchée » en mathématiques?

Les mathématiques sont souvent considérées comme une science théorique. L'informatique en mathématiques a été vue et accueillie comme une occasion de faire de la programmation chez les enseignants qui ont de bons rapports avec l'informatique. Pour d'autres, cet enseignement n'a pas été bien accueilli. Pour susciter l'adhésion de tous les enseignants en difficultés avec la programmation, le stage de formation a suscité la focalisation sur des pratiques « à la main ». Contrairement à ce que laissent voir les textes officiels et les manuels scolaires, le stage de formation a stipulé que chaque notion informatique ne va pas systématiquement être illustrée sur machine ou calculatrice, même si rarement cela permet de vérifier la véracité de certains algorithmes construits ou donnés. Les enseignants motivés trouvent une nouveauté dans l'introduction de l'algorithmique en mathématiques : *« ce programme me plaît parce qu'il introduit la nouveauté dans mon métier, je n'ai aucune raison de le rejeter parce que j'ai horreur de la routine ; donc, j'aime la nouveauté, voilà ! »,* déclare F_D, enseignante bientôt en retraite. Elle est la seule, dans son lycée, à apprécier positivement ce programme.

En effet, l'exécution d'un algorithme sur une calculatrice permet aux élèves de s'assurer de la véracité de leur programme construit. Cette étape de vérification, offerte par l'algorithmique dans la pratique enseignante des spécialistes de mathématiques parce qu'elle permet d'aller au-delà de la théorie mathématique, est vue comme un apport de cet enseignement dans l'acquisition de la rigueur mathématique, comme le souligne l'enseignant A_N :

> *« Un des intérêts par rapport au travail sur un algorithme et ensuite le transport du programme sur la calculatrice et là pour eux, ça peut devenir une façon de faire venir quelque chose de concret et voilà ça fonctionne. C'est pas que la théorie, c'est pas que quelque chose d'abstrait, c'est pas que de l'intellectuel. Donc, sur une calculatrice je veux faire ça, je l'ai bien écrit, et ça fonctionne : j'ai le résultat et je vois, ça donne un côté visuel, un côté concret, voilà. »*

Le foisonnement technologique dans les prescriptions donne l'impression que l'algorithmique ne reste pas une activité papier-crayon au service des mathématiques. Il est aussi une affirmation d'un enseignement de l'algorithmique et la programmation. Ce foisonnement de logiciels semble avoir différentes interprétations chez les enseignants.

Certains ont vu en ces programmes une impulsion à l'enseignement de l'informatique et, ont affiché les premiers signes de résistances notamment par le refus à participer aux stages de formation. Face à l'« obligation » institutionnelle de l'enseignement de l'algorithmique notamment par son inscription aux épreuves de baccalauréat, l'approche privilégiée semble celle de l'informatique[133] débranchée (Drot-Delange, 2013). Généralement utilisée en marge des disciplines scolaires, cette approche de l'enseignement de l'informatique, sans ordinateur, permet de ne pas aborder la programmation informatique qui, elle, appelle un apprentissage d'un langage.

Si cette approche permet de limiter la complexité de la tâche enseignante, elle est aussi réductrice du champ des savoirs à apprendre : certains concepts (langage, programme, syntaxe…) et méthodes offertes par la science informatique ne peuvent pas être abordés. Néanmoins, même l'enseignement des activités de ce genre d'informatique exige des connaissances suffisantes chez l'enseignant pour arriver à apporter des réponses aux questions et remarques des élèves, précise Drot-Delange.

Pour d'autres enseignants motivés par l'informatique, les logiciels sont pour eux des instruments de vérification et de confirmation du travail initialement fait en papier-crayon. Après le stage de formation, la tendance dans la pratique des enseignants est de ne pas faire systématiquement ce travail de programmation pour éviter le paroxysme de la rigueur de l'informatique qui risque de rebuter les élèves débutants.

133 Les enseignants en question n'acceptent pas qu'ils enseignent l'informatique, mais de l'algorithmique qu'ils qualifient de logique mathématique. Ceci contraste avec ceux qui ont de bons rapports à l'informatique et qui sont motivés pour son enseignement. Pour eux, en algorithmique, ils affirment et sont fiers d'enseigner l'informatique.

2.3. Principaux facteurs en faveur de l'évolution des pratiques

Les résultats de cette recherche permettent de relever certains facteurs qui sont à la base de la variabilité des pratiques enseignantes et partant, celles de leurs élèves et étudiants : la formation des enseignants, la motivation, la jeunesse dans le métier et l'approche pédagogique mise en œuvre.

a. Formation

Le stage de formation semble avoir légitimé, orienté et décomplexé les enseignants de mathématiques à l'égard de l'enseignement de l'informatique en mathématiques ou en ISN. Chez les enseignants de mathématiques, enseigner transversalement des notions algorithmiques au travers des mathématiques semble avoir posé un problème de faisabilité. Initier les élèves aux notions algorithmiques par cours théorique n'est pas prévu par les textes officiels. Une solution préconisée par le stage de formation a été de passer par un petit cours théorique, ce qui semble avoir permis de remédier aux déstabilisations des enseignants occasionnées par le besoin d'une notion comme outil pour résoudre un problème donné alors qu'elle n'a pas encore été vue. Un bon nombre parmi les enseignants ayant suivi ce stage semblent décomplexés et outillés pour cet enseignement.

Même quand les élèves étaient motivés par l'informatique, leurs pratiques se sont construites en fonction de celles des enseignants. La formation de ces derniers a donc un impact positif sur la motivation et les pratiques des élèves.

Le fait que l'algorithmique est perçue chez les élèves comme un nouveau domaine d'apprentissage est important. Il peut servir de porte d'entrée des enseignants pour améliorer les rapports des élèves aux mathématiques, des rapports qui ne sont pas toujours au beau fixe. Néanmoins, le manque de formation des enseignants en informatique entraîne des limites dans leurs pratiques en algorithmique. Comme conséquence de ce manque de formation, l'absence répétitive d'aide aux élèves risque d'être une cause potentielle de la démotivation de ces derniers en informatique et peut être même à la longue, un risque de perte progressive de l'envie d'apprendre. Le contraste entre, d'une part les réticences observées chez les enseignants non formés et, d'autre part, l'adhésion de certains d'entre eux ayant bénéficié de stage de formation souligne la part importante de la formation pour le succès de cet enseignement. Celle-ci devrait être longue et approfondie pour faire acquérir aux enseignants des capacités

conceptuelles, méthodologiques et pédagogiques en vue d'une prise de recul nécessaire vis-à-vis de l'informatique.

b. Motivation

Si la formation est indispensable pour la mise en œuvre de ces programmes et surtout la pérennité de l'enseignement de l'informatique au lycée, la motivation semble être le premier facteur sur lequel se fonde l'évolution des pratiques des enseignants et partant de leurs élèves[134]. Les enseignants ayant de bons rapports à l'informatique, malgré les différences constatées chez leurs élèves vis-à-vis de l'algorithmique, exercent des effets positifs chez les élèves même les plus faibles en mathématiques. Cette motivation semble liée aux représentations qu'ont les enseignants de l'algorithmique et de l'informatique en général. La non-obligation de suivre des stages de formation fait que ce sont les enseignants volontaires qui y participent. Cette volonté est liée à une idée de motivation dont certains facteurs suivants semblent être à la base :

- un lien perçu entre évolution technologique et renouveau dans l'enseignement des mathématiques : certains enseignants trouvent en informatique, un atout pour renouveler leurs pratiques dans l'enseignement des mathématiques. Ces résultats sont confirmés par Modeste (2012b) et Vagost (2010). Selon eux, l'évolution des mathématiques, l'omniprésence de l'informatique et de ses applications et le développement technologique renforcent les liens mathématique-informatique et poussent à requestionner l'enseignement des mathématiques. L'une des orientations à lui donner est le caractère algorithmique. Cette idée de nouveauté est aussi soulignée par Tort et al., (2013) pour justifier la motivation des enseignants de mathématiques à s'engager dans l'encadrement de leurs élèves dans les initiatives privées de l'apprentissage de l'informatique comme le concours castor.
- Un certain bagage de connaissances informatiques : Pour d'autres enseignants, leur motivation pour l'enseignement de l'informatique est due au fait qu'ils avaient certaines connaissances de base pour l'aborder, des connaissances acquises soit lors de

134 Les enseignants ayant de bons rapports à l'informatique, malgré les différences constatées chez les élèves en algorithmique qui sont influencées par leurs rapports en mathématiques, relèvent des apports de l'algorithmique chez les élèves moins bons en mathématiques : ça demande d'abord que le prof soit d'abord convaincu de leur nécessité dans l'enseignement/apprentissage des mathématiques pour les élèves le réalisent ensuite.

la formation initiale soit en autodidacte. Selon eux, ces programmes leur donnent une occasion pour mettre en œuvre et valoriser leurs connaissances en informatique.

La motivation pour l'informatique est davantage confirmée dans le recrutement des enseignants d'ISN, où ce sont les volontaires qui postulent pour cet enseignement.

Comme chez les enseignants, la motivation des élèves semble jouer un grand rôle pour leurs pratiques et la réussite : si les « bons » en mathématiques réussissent mieux en informatique, les résultats montrent aussi que les moins « bons » en mathématiques réussissent aussi bien en ISN à condition d'être motivés. Deux raisons peuvent aussi le justifier. La première est que les élèves poursuivant ISN, ne sont pas tous forts en mathématiques. Il y en a même qui sont faibles étant donné qu'ils proviennent de tous les profils organisés au lycée. La deuxième raison est donnée par Jacques (Baudé, 2010a) : *« l'informatique, ce n'est pas des mathématiques »*.

La différence de motivation se fait remarquer dans la fréquence de pratiques algorithmiques chez les enseignants. Les enseignants observés ayant suivi le stage font intervenir fréquemment l'algorithmique (au moins une fois par mois) dans leurs pratiques même en tant qu'outil contrairement à ceux sans stage qui le font environ une fois le trimestre. Cette fréquence d'utilisation de l'algorithmique semble expliquer l'appréciation qu'en ont leurs élèves. Si les enseignants motivés soulignent ses effets positifs, ceux sans stage et donc non motivés trouvent que les élèves sont perturbés par l'algorithmique, qui ajoute une difficulté supplémentaire aux mathématiques : elle demande davantage de maturité conceptuelle d'après les enseignants sans stage.

Chez les élèves en apprentissage de l'informatique en cours de mathématiques, la motivation semble liée à l'approche de groupe acquise en stage des professeurs de mathématiques et non directement à l'algorithmique, ce qui leur donne une forme d'apprentissage collaborative. Par contre, les difficultés vécues par les élèves semblent aussi liées à celles de leurs enseignants sans stage. Le fait de trouver en algorithmique un nouveau domaine à apprendre[135], déconnecté des mathématiques et de l'informatique, peut servir d'entrée pour installer chez les élèves de nouvelles représentations liées à leurs nouveaux rapports aux mathématiques et à l'informatique. Cela nécessite de leur montrer des liens entre les mathématiques, l'algorithmique et l'informatique, surtout qu'en ce moment de découverte de l'algorithmique, sans encore de préjugés négatifs.

135 Selon eux, l'informatique c'est l'utilisation des machines avec des logiciels

Quant aux élèves d'ISN, certains ne sont pas seulement motivés mais aussi passionnés de l'informatique. Ces derniers ne se retrouvent pas seulement parmi ceux qui ont commencé l'informatique très tôt mais, d'autres, avec le début de ces programmes, se passionnent pour cette spécialité ISN, avec un souhait de commencer la spécialité ISN depuis le début du lycée.

c. Jeunesse dans le métier

La variable « jeunesse dans le métier » est un autre facteur important qui favorise l'engagement des enseignants de mathématiques envers l'enseignement de l'informatique. Par rapport aux enseignants plus anciens, les résultats montrent qu'en général, les jeunes enseignants sont plus motivés par cet enseignement. Plus de 70 % (20 sur 28) des enseignants ayant participé au stage de formation en algorithmique ont au plus 10 ans d'expérience dans le métier. Aussi, plus d'un 1/3 (10 sur 28) parmi eux ont entre 5 et 10 ans d'expérience. Bien que leurs pratiques n'aient pas été suivies en classe, les témoignages révèlent une adhésion pour cet enseignement et déclarent n'avoir pas beaucoup de difficultés pour cet enseignement.

Par contre, les résultats révèlent que des enseignants un peu plus anciens dans le métier, n'ont pas seulement des difficultés avec cet enseignement de l'informatique en mathématiques mais semblent y avoir résisté. Un exemple donné est celui du lycée A où les enseignants interviewés et dont leurs pratiques de classes observées ont une expérience variant de 20 à 38 ans : aucun d'entre eux n'a voulu participer à un quelconque stage en algorithmique. Peut-on peut-être penser à des formes de résistances ? Cela n'est pas exclu étant donné qu'ils participent à d'autres stages de formation organisés sur d'autres domaines des mathématiques.

Deux raisons peuvent justifier cette adhésion plus motivée de jeunes enseignants par rapport aux autres. La première est que les jeunes ont peut-être bénéficié de modules de formation en informatique à l'université, contrairement aux anciens. Cette hypothèse peut-être justifiée par leur adhésion à cet enseignement dans 75 % (21 sur 28 enseignants) avant ce stage de formation.

La deuxième raison est que, contrairement aux plus anciens qui sont en fin de carrière, les jeunes enseignants ont encore beaucoup d'années devant eux à prester. Ceci peut motiver leur adhésion, leur participation au stage de formation étant motivée par la volonté de s'approprier ces programmes d'enseignement en vigueur.

Ces résultats infirment notre hypothèse selon laquelle la variable « ancienneté » dans le métier influence le rapport enseignant/algorithmique (Haspekian & Nijimbere, 2012) : cette hypothèse a été motivée par le fait que cette étude a porté sur un groupe d'enseignants anciens dans le métier, particulièrement motivés et volontaires à participer aux entretiens. Certains d'entre eux (1/3) étaient spécialistes en informatique.

d. Approche pédagogique utilisée

Souvent l'approche par projets est associée avec une approche de groupe. Ces deux approches changent la façon habituelle de faire en donnant plus de responsabilité aux apprenants. Pour l'apprentissage de l'informatique en mathématiques, les enseignants ayant reçu un stage en algorithmique procèdent par l'approche de groupe : les élèves en groupe de quatre construisent leurs connaissances. Motivés pour ce travail entre pairs, ils interagissent en toute autonomie et ne font appel à l'enseignant qu'en dernier recours.

En ISN et en licence, c'est la pédagogie de projets qui était utilisée dans les apprentissages. Elle comprend l'approche de groupe précédente, à la seule différence que les projets demandent plusieurs jours pour être réalisés. Dans les deux approches, le travail est de la responsabilité des apprenants. Ils sont appelés à mutualiser leurs compétences au profit de leurs apprentissages dans un contexte de complémentarité plutôt que de compétition.

L'efficacité de cette approche dans les pratiques des élèves est liée aux problèmes complexes qu'elle permet d'aborder chez les apprenants. Sa productivité est associée au travail de groupe, ce qui conduit à mettre en œuvre des stratégies diverses qui émanent des compétences de chacun (Rogalski, 1988) : *« l'existence de connaissances et de points de vue différents permet au groupe d'entrer dans un problème complexe moins difficilement qu'un élève (étudiant) seul ».*

Le point suivant montre une forte adhésion à l'ISN d'un public qui reste majoritairement masculin.

2.4. L'ISN, une option valorisée ?

L'option ISN est convoitée au lycée. Elle est suivie par des élèves de tous les profils de la filière S mais avec des motivations différentes. Ce résultat était prévisible. Maurice Nivat (2007) l'une des défenseurs de cet enseignement au lycée, témoignait avant son institutionnalisation que

« tout le monde peut faire avec succès l'informatique, ceux qui aiment l'abstraction et la réflexion, et aussi qui préfèrent l'action et le concret, ceux qui ont du goût pour les sciences exactes comme ceux qu'attirent les sciences du vivant, ceux qu'attirent les lettres et humanités comme ceux qui se destinent au droit, au management et à l'économie ».

Nos résultats révèlent une absence de filles dans les deux lycées[136]. Cependant, on ne peut généraliser étant donné la non-représentativité de l'échantillon. Si notre étude n'est pas quantitative, les recherches disponibles révèlent certains changements. Le premier changement est une croissance progressive des établissements qui organisent cette option et des effectifs d'élèves qui la fréquentent chaque année[137]. Le deuxième changement est liée à l'hypothèse de Dowek et al., selon laquelle le nouveau programme d'ISN va casser l'image d'une science informatique dominée par des garçons en attirant aussi des filles (Dowek et al., 2011, qui est confirmée par des travaux déjà effectués relativement à cette option. En effet, les recherches récentes montrent une évolution de la représentativité féminine en ISN bien que encore inférieure à celle des garçons (Breton, 2013 ; Drot-Delange & More, 2013).

Au-delà de cette représentativité féminine en croissance, Drot-Delange et More montrent, dans leur étude sur les attitudes des élèves envers l'informatique, que les élèves qui ont choisi cette option au lycée « n'adhèrent pas aux stéréotypes de sexe concernant l'informatique », contrairement à leurs collègues de classe. Ce changement semble justifié par l'augmentation sensible des domaines dans lesquels l'informatique intervient. L'intérêt des filles pour ces domaines pourrait être la cause de leur motivation pour cette option, précisent Drot-Delange et More.

Ces résultats actuels dans le cadre d'ISN, diffèrent de ceux jusqu'ici publiés. Dans leur étude concernant l'engagement et la motivation des enseignants de la scolarité secondaire pour le concours castor, Tort, Kummer_Hannoun et Beauné soulignent une baisse avec le niveau de scolarité du nombre de filles motivées par l'informatique et, un contraste entre les niveaux collège et lycée. La raison serait la distance perçue par les élèves entre l'informatique vue au collège et celle du lycée : l'informatique vue au collège est considérée comme une informatique moins dure et accessible par tous (y compris donc les filles) alors qu'elle est perçue comme devenant de plus en plus dure et donc une « informatique réservée à des spécialistes »

136 Une seule fille inscrite en ISN au lycée C mais, elle n'a pas terminé l'année.
137 https ://www2.ac-lyon.fr/enseigne/math/IMG/pdf/isn-2013.pdf

au fur et à mesure que le niveau de scolarité croît. Cette perception semble accentuée par une systématisation de la sélection des élèves des filières S du lycée avec une réduction du nombre de filles participant au concours castor (Tort et al., 2013) :

> *« la participation massive au collège, le choix de tous les élèves de seconde dans certains lycées, et des arguments d'accessibilité et de valorisation de certains élèves, vont dans le sens d'un enseignement de l'informatique pour tous. D'un autre côté, la sélection systématique des élèves de séries scientifique au lycée, qui d'ailleurs réduit la participation des filles, correspond plutôt à une image de l'informatique réservée à des spécialistes. Ces choix s'expliquent peut-être plus simplement par le format concours et aussi le nombre élevé d'enseignants de mathématiques tentés de faire entre leurs classes une sélection déjà présente dans l'organisation en séries du lycée. »*

Cette situation ne se limite pas à la France et, est pareille au niveau international. Comme le montrent Montagnier & Van Welsum (2006) dans le cadre de l'OCDE, la faible adhésion des filles dans la spécialité informatique est justifiée par la mauvaise image qu'elles ont des métiers de l'informatique qui semblent dédier aux hommes.

Dans le point suivant, nous montrons les potentialités de la pédagogie par projets utilisée.

2.5. Approche par projet, un outil pédagogique efficace pour apprendre ?

La démarche de projet est une pédagogie efficace dans la construction des savoirs. Elle permet d'aborder des savoirs variés *(*Perrenoud, 1999*) :* disciplinaires, pluridisciplinaires ou non disciplines de l'ordre des compétences. Elle est appropriée pour faire expérimenter le phénomène de la ZPD de Vygotski dans un système d'activités. Là où une seule personne ne peut pas arriver ou ce qu'elle n'est pas capable de faire tout seul, avec l'appui du groupe, permis par l'approche de projet, il y arrive. Chacun des membres bénéficie de l'apport de ses coéquipiers pour construire ses propres connaissances avec un objectif commun. La mise en commun de leurs connaissances et compétences individuelles se fait par des interactions. L'importance des connaissances construites est fonction de l'étendue de leurs champs d'action. Pour le cas de notre thèse, ce champ d'interaction semble évoluer avec le niveau scolaire d'apprenants impliqués dans les apprentissages.

Nous faisons un parallélisme des motivations des apprenants dans leurs choix des sujets de projets et les formes de leurs interactions au travail.

a. *Des motivations différentes en projets chez les élèves et les étudiants ?*

Notre recherche s'est focalisée sur la construction de connaissances en contexte de projets chez les élèves et les étudiants scientifiques. Les résultats révèlent des motivations différentes pour les deux catégories d'apprenants.

Les attentes des étudiants en projets sont la préparation au travail en contexte professionnel qui, selon eux, diffère du contexte académique. Ils semblent particulièrement motivés par la découverte des exigences et de contraintes de travail, de connaissances professionnelles nécessaires… Le choix des projets supposés difficiles comme ceux portant sur la programmation des technologies émergentes telles que les technologies robotiques sont motivées par la valorisation de leur CV, le travail en projet fréquent dans les entreprises, mais aussi parce qu'ils peuvent être appelés à programmer de tels objets dans leur future vie professionnelle en entreprise en tant que spécialiste de l'informatique. Malgré les exigences de ce genre de projets, ils sont, selon eux, un avant-goût de leur vie professionnelle : si la facilité du projet avait sa place dans leur motivation de choix, elle occupe une petite place pour influencer leurs choix.

Par contre, en ISN, le choix de la spécialité semble n'avoir pas de rapport direct avec leur vie professionnelle. Certains élèves n'ont pas choisi l'ISN en rêvant de devenir spécialiste d'informatique dans le futur mais, la volonté de découvrir et se familiariser avec l'informatique : bien que grands utilisateurs des technologies informatiques, ils sont conscients de leurs lacunes en informatique, d'où leur motivation à acquérir l'informatique dans un cadre formel. Néanmoins, certains commencent à voir que l'informatique prend de plus en plus d'ampleur dans la vie et devenant même incontournable, comme des expressions de certains d'entre eux le montrent : *« l'informatique va bientôt nous remplacer »*, *« l'informatique c'est l'avenir »*. Quant à leur choix de sujet de projet, la motivation est plus orientée vers l'apprentissage de l'informatique en contexte ludique, ce qui explique leur choix des sujets portant sur du jeu, sans mise en perspective direct avec une quelconque profession future.

b. *Des interactions évoluant avec les niveaux d'étude des apprenants ?*

Nous sommes partis du constat que les élèves des lycées scientifiques et les étudiants de licence sont tous débutants en informatique. Il fallait comparer leurs stratégies de construction des connaissances en contexte de projets de programmation. Les résultats ont révélé des différences de pratiques entre eux, liées aux stratégies mises en œuvre et notamment leurs inter-

actions en contexte de travail. Ces dernières semblent évoluer selon leurs niveaux scolaires. Trois niveaux d'interactions peuvent être distingués : celui des lycéens en dehors d'ISN, celui des élèves d'ISN et enfin, celui des étudiants de licence.

- **Premier niveau : des interactions timides et restreintes**

Ce genre d'interactions s'observe chez les lycéens lors de leurs apprentissages de l'informatique en mathématiques. Ce sont des interactions qui semblent être dictées par la difficulté du problème posé sans que l'enseignant l'a prévu. En effet, il n'y a pas de travaux prescrits en groupe ou en projet chez les enseignants. Leur travail est général individuel. Pour s'entraider, les élèves peuvent adopter une approche de groupe autant pour un travail papier-crayon que sur machine. Certains élèves préfèrent quelquefois laisser de côté leur poste pour travailler ensemble en binôme sur une même machine. Cela semble se passe entre deux élèves dont l'un serait faible et l'autre un pivot. D'autres élèves interagissent avec leurs voisins en vue d'une éventuelle aide face à une quelconque difficulté rencontrée tout en restant sur leurs postes respectifs. Dans les deux cas, le recours à l'enseignant en vue d'une aide n'est pas prioritaire : l'élève cherche d'abord une aide chez son voisin, à gauche ou à droite, avant de faire appel au professeur. Cette pratique des élèves est confirmée par Robert Pléty. Selon lui,

> *« l'apprentissage qui a lieu dans la confrontation directe avec un camarade est plus autonome que celui qu'engendre la confrontation avec un adulte présentant un modèle de réponse (...). Ce n'est pas tant l'exactitude des propositions avec lesquelles le sujet entre en contact qui joue un rôle que le fait même d'y être confronté dans la mesure où les tâches qu'il doit résoudre l'obligent à dépasser les contradictions qu'il rencontre : ce serait la nécessité de résoudre des conflits socio-cognitifs qui semble être à l'origine des progrès cognitifs. »* (Pléty, 1996, p.119).

Non prescrite par l'enseignant(e), cette approche de groupe est utilisée par les élèves par leur propre initiative : certaines de ses potentialités sont exploitées en coconstruisant certains savoirs. Néanmoins, utilisée à volonté et de façon restreinte chez quelques élèves, toutes ses potentialités semblent ne pas être exploitées étant donné qu'elle est mobilisée pour résoudre des problèmes conçus pour être faits individuellement. Nous supposons qu'une fois prescrite part l'enseignant, cette approche de groupe contribuerait à résoudre des problèmes plus ouverts appropriés à un travail de groupe et ainsi coconstruire plus de connaissances.

- **Deuxième niveau : des interactions plus circonscrites à un groupe de travail**

À côté des contenus informatiques construits qui diffèrent selon que l'informatique est acquise en mathématiques ou en ISN, une différence fondamentale entre les élèves de lycée est l'approche de projets utilisée en ISN : elle fait d'elle une singularité. Les projets sont faits en binôme. Dans leurs pratiques, les interactions étaient plus limitées au sein d'un groupe avec de rares occasions d'ouverture aux autres groupes pour échanger. Les projets ont été presque totalement faits en classe. Comme les autres élèves lors de l'apprentissages de l'informatique en mathématiques, leurs collaborations restent presque localiser au sein du binôme. Cette situation laisse voir qu'ils ne sont pas habitués à travailler en groupe. Mais, contrairement aux précédents, ceux d'ISN résolvent des problèmes qui nécessitent plus d'interactions qui ne sont malheureusement pas mobilisées. Nous faisons l'hypothèse que la présence régulière de l'enseignante, à l'image d'une séance de TP, aurait contribué à limiter le développement des interactions.

Si la libération de certaines séances de cours au profit des projets était une bonne chose, il semble que la professeure a beaucoup pris par la main les élèves, ce qui semble avoir limité leur autonomie et liberté. Peut-on penser que les élèves se sont sentis surveillés ? Non, il semble plutôt qu'elle les considère comme incapables de mener à bout un projet et est, restée prête pour les aider. Nous sommes d'accord avec Aguirre et Raucent pour qui, *« le tuteur doit faire confiance au groupe dans sa dynamique et sa capacité à résoudre ses problèmes »* (Aguirre & Raucent, 2002).

D'un autre côté, la pratique de l'enseignante semble justifiée par la peine et les difficultés vécues par les élèves confrontés pour la première fois à un travail aussi complexe et exigeant que celui de projets qu'ils n'ont l'habitude de faire. La contradiction de niveau 5 de la l'approche de la TA permet de justifier cette situation (Herviou & Taurisson, 2012). Même si les projets ont été quasiment faits en classe, nous supposons que plus d'autonomie des élèves aurait conduit à plus d'initiatives et donc d'interactions entre les élèves.

- **Troisième niveau : des interactions nombreuses et élargies**

Les étudiants affichent des pratiques plus avancées et autonomes par rapport à celles des lycéens. Contrairement aux interactions en général localisées au sein d'un groupe chez les lycéens, celles des étudiants sont plus étendues. Si on peut associer l'étendue des interactions à la complexité du travail à faire, il semble que leur étendue est fonction du niveau scolaire des apprenants. Pour le cas des étudiants en projets de programmation des robots, leurs interac-

tions vont au-delà jusque même à impliquer des groupes non directement concernés par leurs projets (autres groupes de licences, étudiants de master…). Les connaissances construites sont en conséquence étendues et diversifiées à la lumière des interactions qui ont servi à les produire.

Les étudiants semblent mettre à profit toutes les potentialités offertes par l'approche par projets : tous les moyens sont mis en œuvre pour résoudre leurs problèmes et surmonter les contraintes auxquelles ils sont confrontés. Même au sein des étudiants, les interactions sont plus ouvertes chez les L3 que les L2.

En général, une aide est cherchée partout où ils espèrent la trouver. Si les échanges et la collaboration permettent à chaque groupe de résoudre surmonter certaines difficultés, d'autres les amènent à aller chercher un secours dans la communauté : le projet leur confère un véritable cadre d'apprentissage socioconstruictiviste. Contrairement aux élèves, ils ont une totale autonomie : ils sont responsables et garants de leur travail, de vrais acteurs de leurs apprentissages. Des rapports hebdomadaires à fournir sur l'état d'avancement du travail de groupes exigent cette responsabilité et un travail régulier, selon un agenda préalablement fixé par chaque groupe d'une part et un comité d'organisation de projets d'autre part.

Cette organisation et les conditions de travail des étudiants diffèrent de celles des élèves d'ISN : un encadrement sollicitant et serré, transformant le travail de projet en celui d'un travail dirigé n'a pas beaucoup favorisé les apprentissages des élèves. Si le guidage en projet est incontournable, nous faisons l'hypothèse que son efficacité en projet nécessite une autonomie des acteurs et des interactions plus ouvertes, larges et croisées. Néanmoins, il semble que ce n'est pas seulement cette organisation et ces stratégies mises en œuvre par les étudiants qui ont contribué à leurs performances en programmation par rapport aux élèves. D'autres facteurs tels que l'entrée en projet avec de bases solides en programmation chez les étudiants, ne sont pas à minimiser.

Comme le précise Antoine Meyer, enseignant d'informatique à l'université et formateur en algorithmique, l'organisation de l'enseignement de l'informatique à l'université est différente du lycée (en mathématiques). À l'université, l'enseignement de l'informatique procède par des modules à la fois différents et complémentaires : *algorithmique* et *programmation*. Il explique la facilité de l'appropriation de la programmation classique en licence par deux raisons. La première raison est que ces modules sont enseignés en dehors de tout autre contexte

disciplinaire, contrairement à ce qui se fait au lycée. La deuxième raison est qu'ils sont dispensés par des enseignants spécialistes et ce, de façon complémentaire.

En définitif, des stratégies organisationnelles et un encadrement autonomisant les apprenants conduisent à de nombreuses interactions qui engagent toute la communauté dans laquelle les apprenants œuvrent. De telles interactions nombreuses sont des atouts pour des apprentissages fructueux en projets. Mais, elles ne suffisent pas pour justifier les pratiques et connaissances avancées des étudiants en projets de programmation des robots, encore moins des robots humanoïdes connus pour la complexité de leur programmation telle que NA0. Si ces stratégies interactionnelles sont nécessaires, leur efficacité semble due au fait qu'elles sont accompagnées par certains savoirs, savoir-faire et compétences déjà acquis chez les étudiants. Dans notre première hypothèse nous avons supposé qu'une meilleure appropriation des savoirs informatiques chez les apprenants est liée à leurs multiples interactions avec les autres membres de la communauté. Cette condition semble nécessaire mais pas suffisante. Notre première hypothèse n'est pas vérifiée.

3. Conclusion

L'objectif de la thèse était de comprendre comment les enseignants non-spécialistes de l'informatique s'approprient-ils l'enseignement des savoirs informatiques, mais aussi comment les élèves des filières scientifiques au lycée et les étudiants de licence informatique à l'université en France, débutants en informatiques construisent les savoirs informatiques. Une approche comparative a été adoptée pour comparer les pratiques des apprenants – lycéens et étudiants – d'un côté, et celles des enseignants de lycées, de l'autre côté.

Nous sommes partis des deux questions générales de recherche suivantes :

- Comment les élèves et les étudiants des filières scientifiques s'approprient l'enseignement des savoirs informatiques ?
- Comment les enseignants, spécialistes des mathématiques, prennent en compte les savoirs informatiques dans leurs pratiques professionnelles ?

Relativement à ceux questions, deux hypothèses générales ont été formulées. Dans la première hypothèse, nous avons supposé qu'en contexte de projets, les élèves ou les étudiants qui communiquent plus et qui sont ouverts à la collaboration avec les autres, s'approprient facilement les savoirs informatiques à construire. La deuxième hypothèse est relative aux en-

seignants de lycée. Nous avons supposé que les professeurs spécialistes des mathématiques qui ont des représentations de l'informatique proches des mathématiques adhèrent plus facilement à l'enseignement de l'informatique en mathématiques que leurs collègues.

Dans cette recherche, nous nous sommes focalisé sur leurs apprentissages à propos de variable et de boucle, deux savoirs informatiques jugés incontournables pour une initiation informatique chez les lycéens débutants : notion de variable et de boucle. Un cadre théorique de la Théorie de l'Activité d'Engeström et une méthodologie qualitative de type ethnographique ont été privilégiés.

La recherche a eu lieu dans cinq établissements dont quatre lycées et une université. Au secondaire, elle a porté sur 88 élèves et 42 enseignants. À l'université, elle a porté sur 20 étudiants de licence et deux enseignants. Les résultats ont montré des pratiques contrastées mais en évolution chez les enseignants. Quant aux élèves, leurs apprentissages sont influencés par les pratiques de leurs enseignants. Ces élèves semblent ne pas appréhender un lien qui existerait entre l'algorithmique enseignée en mathématique et les mathématiques elles-mêmes ou l'informatique : c'est un nouveau domaine à part qui leur complique la vie. Selon eux, l'informatique commence en terminale avec l'option ISN.

Les apprentissages des étudiants sont avancés par rapport à ceux des élèves. Les programmes construits sont sophistiqués mais affichent des limites dans la construction des connaissances mathématiques pourtant indispensable pour la performance des robots.

Relativement à l'informatique en mathématiques, ces pratiques se caractérisent par un décalage de l'esprit des programmes et même des manuels scolaires. Chez les enseignants, leurs pratiques varient d'un enseignant à un autre et, selon les rapports de chacun à l'informatique. Avec le temps, leurs pratiques semblent s'harmoniser pour des enseignants d'un même lycée tout en restant variables d'un établissement à un autre. Trois facteurs influent sur les pratiques des enseignants en informatique. Par ordre d'importance, ils peuvent être présentés comme suit : la motivation (associée aux représentations), la formation continue et la jeunesse dans le métier. Le quatrième facteur concerne plus pratiques des élèves d'ISN et des étudiants. Il s'agit de l'approche pédagogique utilisée.

La difficulté principale connue par les enseignants de mathématiques dans la mise en œuvre de ces programmes est liée à la contrainte institutionnelle d'initier les élèves aux notions algorithmiques à l'intérieur du cours de mathématiques, sans faire de cours théorique d'algo-

rithmique. Les pratiques enseignantes peuvent être classées en deux catégories. La première catégorie est celle des pratiques des enseignants qui ne font pas de cours théorique d'algorithmique mais qui l'introduit sur le tas dans le cours de mathématiques. La deuxième catégorie est celle des enseignants qui en font un avant que l'algorithmique ne soit utilisée pour résoudre des problèmes dans les chapitres du cours de mathématiques.

Les enseignants qui n'ont participé à aucun stage de formation, n'ont pas de cours théorique d'algorithmique. Se représentant l'algorithmique comme faisant partie intégrante de l'informatique, ce refus de participer à des stages de formation semble être une forme de manifestation de leur désapprobation de cet enseignement : ils ont résisté. Dans leurs pratiques, l'enseignement de l'algorithmique est orienté vers la logique mathématique. Cette approche logique de l'algorithmique est l'une des approches possibles en tension permises par le large champ d'interprétations des programmes d'enseignement et des manuels scolaires à leur disposition. Si leurs choix sont motivés par des raisons de facilité, un enseignant oriente ses pratiques en fonction de ses motivations (qui sont liées à ses représentations) et surtout de ses rapports à l'informatique.

La deuxième catégorie de pratiques est celle des enseignants qui ont suivi des stages de formation. Dans leurs pratiques, ils commencent par un petit cours théorique avant d'instrumentaliser l'algorithmique pour la résolution des problèmes mathématiques à travers les chapitres. À la différence de la première catégorie, certains enseignants de cette deuxième catégorie avaient tendance au départ à focaliser leurs pratiques sur la programmation informatique, une pratique qui a été réduite par le stage de formation, au profit d'un travail plus centré sur l'approche papier-crayon.

Leur motivation envers l'enseignement de l'informatique semble liée à leur passé particulier qui peut justifier leurs rapports et pratiques particuliers à l'informatique. La majorité d'entre eux enseignait l'algorithmique avant même le stage de formation. Avec le stage, une activité de programmation informatique, non systématique, a rarement lieu pour des besoins d'illustration. En algorithmique, l'utilisation d'un ordinateur étant plutôt rare, la programmation se fait plus avec une calculatrice. Et si une occasion se présente pour utiliser un ordinateur, c'est avec le langage Algobox.

Les enseignants de mathématiques, donc non experts de l'informatique et dont leurs représentations de l'algorithmique se situent du côté totalement informatique, adhèrent difficile-

ment à l'enseignement de l'algorithmique en mathématiques. Cette situation est amplifiée par les programmes qui, au lieu de constituer une aide pour le professeur, contribuent à entretenir la confusion.

Certains enseignants commencent par un petit cours théorique avant que l'algorithmique ne serve d'outils pour résoudre d'autres problèmes au sein des chapitres. Leurs pratiques qui, pour des motivés, associaient le travail papier-crayon et le travail sur machine, s'intéressent plus après au travail « à la main ».

Relativement à l'informatique en mathématiques, les pratiques des élèves restent influencées par celles de leurs enseignants. Les élèves ne voient pas de liens qui existent entre mathématiques, algorithmique et informatique : dans leurs représentations, l'algorithmique est un autre domaine introduit en mathématiques et qui ne fait que les complexifier davantage. En général, leurs pratiques en algorithmique sont focalisées sur un travail au papier-crayon, avec des différences de modalités de travail.

Les élèves des enseignants qui ont un stage en algorithmique ont plus une approche de groupes avec des activités plus ouvertes et nécessitant des interactions. Au contraire, les enseignants sans stage ont des prescriptions d'activités plus ou moins fermées avec une approche plutôt individuelle pour leurs élèves.

Par rapport à l'informatique dans les programmes de mathématiques, le programme d'ISN en terminale S est particulier. La diversité de ces contenus informatiques rend possible la construction d'une culture numérique large chez les élèves, à travers diverses formes d'organisation de cours et d'apprentissage. L'une d'elles, l'approche de projets, lui est d'un atout particulièrement intéressant : elle permet des apprentissages dans un cadre socioconstructiviste. Objet de convoitise, cette option accueille des élèves de tous les profils en filières scientifiques et avec des motivations différentes particulièrement la découverte de l'informatique. Issus des spécialités différentes, les enseignants sont très motivés par cet enseignement.

Leurs pratiques restent influencées par leur spécialité d'origine, avec des difficultés liées au manque de recul chez les non-spécialistes d'informatique. Diverses formes de tensions s'observent entre les élèves et leurs enseignants. Les élèves rebutés par des difficultés algorithmiques en mathématiques, où les pratiques sont généralement centrées sur le travail « à la main », veulent ne rester qu'au travail sur machine. Quant aux élèves d'ISN passionnés par

l'informatique, ils ne veulent rester qu'en programmation et naviguer entre différents langages.

Si les élèves d'ISN et les étudiants de licence sont tous des débutants en programmation, des différences de pratiques s'observent entre les deux publics d'apprenants. Contrairement aux étudiants, certains élèves d'ISN ne visent pas une spécialisation en informatique à l'enseignement supérieur. Quant à leurs choix, des sujets de projets orientés jeu constituent leur centre d'intérêt plus que les connaissances à construire. Ceci contraste avec les motivations des étudiants. Ces derniers visant la spécialisation en informatique. Leurs choix de sujets de projets sont plus motivés par des connaissances et des compétences jugées nécessaires dans leur vie professionnelle en entreprise.

C'est d'ailleurs ce qui justifie les choix des technologies robotiques jugées d'actualités malgré les contraintes qu'elles posent pour leur programmation. L'avenir professionnel et la valorisation de leur CV constituent des préoccupations des étudiants dans l'approche de projet. Les savoirs informatiques construits par les étudiants restent plus avancés que ceux des lycéens. Mieux organisés, les étudiants mobilisent de stratégies plus efficaces dans leurs tâches de programmation : leurs interactions au cours des projets sont plus nombreuses et plus larges que celles des lycéens. Les contextes complexes, les exigences des problèmes à résoudre et le vaste champ des connaissances à mobiliser pour l'efficacité des codes construits… semblent être des facteurs favorisant le développement des performances des étudiants. Les projets constituent pour eux un véritable cadre socioconstructiviste d'apprentissage.

De même que le rapport à l'informatique de l'enseignant influe sur son adhésion et ses pratiques par rapport aux nouveaux programmes du lycée scientifique, de même les résultats montrent qu'en général, les élèves bons en algorithmique sont bons en mathématiques. Les mêmes résultats se retrouvent chez les étudiants en licence informatique : la non-performance des robots programmés, surtout en ce qui concerne leurs mouvements et gestes est dans la plupart de cas liée aux notions mathématiques utilisées qui étaient élémentaires alors qu'il en faut de plus avancées.

4. Perspectives : Enseignement de l'informatique au Burundi. Quelles leçons tirer des choix français ?

L'éducation est l'un des piliers importants d'un pays sur lesquels se fonde l'intégralité de son développement. Le niveau de développement dépend de la qualité de l'éducation dispensée et des choix effectués. Au terme de notre travail, avant des perspectives pour le Burundi, il est important d'élucider brièvement sa situation éducative en général et celle de l'enseignement et apprentissage de l'informatique en particulier.

4.1. Le contexte éducatif burundais

Le Burundi est l'un des pays les plus pauvres et des plus peuplés au monde. Par sa situation géographique, il appartient à plusieurs ensembles sous-régionaux et socio-économiques d'Afrique : Communauté Économique des Pays des Grands Lacs (CEPGL)[138], « East African Community » (EAC)[139], « Common Market for Eastern and Southern Africa » (COMESA)[140]. Comme les autres pays, son appartenance aux différentes communautés l'engage à œuvrer en synergie (PASEC/CONFEMEN, 2010) notamment du point de vue de son organisation du système éducatif. À l'intérieur des deux grandes catégories connues d'écoles publiques et privées, les écoles burundaises revêtent quatre statuts selon leur système de gestion : les écoles publiques gérées par l'État ; les écoles publiques sous conventions[141], les écoles privées et les écoles consulaires.

En quelques chiffres, le Burundi se présente comme suit[142,143] :

138 Cette communauté comprend le Burundi, la République Démocratique du Congo (RDC) et le Rwanda
139 La Communauté de l'Afrique de l'Est est composée de cinq pays : le Burundi, le Kenya, le Rwanda, l'Ouganda et la Tanzanie
140 Cette communauté du marché commun de l'Afrique orientale et australe est composée de 19 pays
141 Ces écoles sont souvent gérées par les confessions religieuses
142 donnees.banquemondiale.org/pays/burundi, site consulté le 17 décembre 2014
143 http://www.cnidh.bi/%C3 %A9tude-sur-%C2 %AB-les-r%C3 %A9formes-du-syst%C3 %A8me-%C3 %A9ducatif-burundais-et-le-droit-%C3 %A0-l%E2 %80 %99 %C3 %A9ducation-%C2 %BB, consulté le 17 décembre 2014

Variable	Caractéristique
Population	10 millions d'habitants (2014)
Superficie	27 834 km^2
Densité	391 habitants/km^2 (2014)
IDH[144]	180e/187 (2014)
PIB	2,7 milliards $ (2013)
Taux du budget de l'État pour l'éducation	25,30%
Taux Brut de Scolarisation (primaire)	135 % (2013)[145]
Ratio élèves/enseignant (primaire)	49
Taux Brut de Scolarisation (secondaire)	31,9 % (2012)[146]
Ratio étudiants/enseignants Docteur (UB)[147]	58
Taux d'écoles raccordées à l'électricité	3 %
Utilisation d'Internet pour 1000 habitants	7 (2008)

Tableau IX.1: Le Burundi en quelques chiffres

Comme beaucoup de pays en voie de développement, le pays a longtemps adopté un système éducatif calqué sur celui de son ancienne puissance coloniale (la Belgique) dans sa structure organisationnelle et ses contenus d'enseignement. Si beaucoup de ces pays ont évolué en opérant des réformes récurrentes, des calques des modèles du nord et reproductions de leurs expériences continuent à être observés (Nijimbere et al., 2014). Dans un contexte de mondialisation, une école doit être pensée, réfléchie et adaptée à son environnement en vue de servir d'instrument de développement. Pour y arriver, des *« stratégies pour une refondation réussie des systèmes éducatifs »*, ont été proposées à Dakar en 2001[148].

144 Indice Humain de Développement. Créé par le PNUD en 1990, l'indice statistique IDH permet d'évaluer le niveau de développement humain d'un pays, calculé à base de trois critères : niveau de vie, espérance de vie à la naissance et niveau d'éducation.

145 http://knoema.fr/atlas/Burundi/topics/%C3 %89ducation/%C3 %89ducation-primaire/Taux-brut-de-scolarisation, site consulté le 29 décembre 2014

146 L'accès à l'enseignement secondaire a été longtemps limité et conditionné par la réussite au concours national : la note d'admission a été toujours fixée en fonction du nombre d'élèves voulus. Ce taux est passé de 3, 27 à 31,9 % pendant une vingtaine d'années (1981 à 2012)

147 Université du Burundi

148 http://www.confemen.org/wp-content/uploads/2012/08/DocRef. Strategies.pdf

- **Défis et réformes éducatives en cours**

Dans un système éducatif en place, une réforme des curriculas n'est pas improvisée. Elle est conséquente d'un constat des dysfonctionnements jugés importants qu'il faut absolument corriger (Delorme, 2010). Sept principaux facteurs de dysfonctionnements caractérisent le système éducatif burundais (PASEC/CONFEMEN, 2010, p. 12) :

> *« (1) l'insuffisance du personnel enseignant ; (2) la forte pression sur les infrastructures scolaires ; (3) l'insuffisance de supports pédagogiques ; (4) les programmes d'enseignement inadaptés ; (5) les disparités géographiques dans l'allocation des infrastructures scolaires ; (6) la faible qualification des enseignants des écoles communales ; (7) le faible rendement pédagogique suite à la faiblesse de la qualification d'un bon nombre d'enseignants, à la surpopulation des classes et à l'insuffisance des supports pédagogiques… »*

Pour y faire face, beaucoup de réformes éducatives, dont certaines datent des années soixante-dix, ont été opérées à tous les niveaux de la scolarité (Ntibashirakandi, 2011 ; CNIDH, 2014) : « Kirundisation », « Ruralisation » et écoles communautaires (1973)[149] et passage de l'école primaire au secondaire après réussite à un concours national (depuis 1973) ; double vacation des locaux et des maîtres (1982) ; mise en place de l'Institut Pédagogique (1980), devenu l'Institut de Pédagogie Appliquée (1993) ; Renaissance de l'ENS (1999) ; gratuité des frais de scolarité primaire (2005) et passage à quatre langues dans l'enseignement primaire (anglais et Kiswahili) en plus du français et du Kirundi (2007)…

Récemment, deux autres réformes éducatives de grande envergure, qui ont touché en profondeur à la fois l'organisation dudit système et les curriculas, se sont rajoutées aux précédentes (CNIDH, 2014) : enseignement fondamental à l'école primaire[150] (2013) et le système LMD[151] (ou BMD : Bachelor, Master, Doctorat selon l'appellation adoptée) à l'enseignement

149 Cette réforme véhiculait la nationalisation, la rationalisation et la rentabilisation Ce sont des moyens pour réussir de rompre avec le système éducatif colonial. Par la Kirundisation, il fallait *nationaliser* l'éducation en prescrivant des curricula et les apprenant dans sa propre langue. La Ruralisation consiste à adopter le système éducatif conformément aux réalités socio-économiques du pays, et donc la rationalisation. Pour y arriver, des écoles communautaires et donc rurales devaient être créées pour les villageois

150 Par cette réforme, l'enseignement primaire passe de 6 à 9 ans. Elle a comme conséquence la suppression du niveau collège à l'enseignement secondaire

151 Le système BMD (Bachelor, Master, Doctorat). Pour éviter des confusions éventuelles entre l'ancienne Licence et la nouvelle dans le nouveau système, le terme de « Bachelor » a été choisi. Si ce dernier est différent du Baccalauréat francophone qui correspond à un diplôme de fin du lycée, lui, correspond au niveau Licence dans le monde anglophone. Il a été institué par la Loi n° 1/22 du 30 décembre 211 portant réorganisation de l'enseignement supérieur au Burundi

supérieur. Débuté à la rentrée 2011-2012 avec les licences 1, le système BMD sera totalement adopté en 2015.

Quant aux universités privées, certaines, orientées vers le monde anglo-saxon, l'ont adopté dès leur création alors que d'autres y entrent progressivement[152]. Ces réformes font face à bon nombre de défis du système éducatif. Ces derniers ont mobilisé la tenue des « États Généraux de l'Éducation » qui étaient impatiemment attendues (Ntibashirakandi, 2011) et qui, récemment en date du 2 décembre 2014[153], ont réuni différents acteurs de l'éducation au niveau national, afin d'« *améliorer ses performances et sa compétitivité* », précise le ministre de l'Enseignement Supérieur et de la recherche scientifique dans son discours.

Soulignons enfin, une récente harmonisation des textes réglementaires qui a rendu équitable l'accès à l'enseignement supérieur tant public que privé. Initialement moins exigeantes pour entrer à l'enseignement supérieur privé, avec une note autour de 30 % à l'examen d'État (Nijimbere, 2012), depuis 2012, les conditions d'accès sont aujourd'hui les mêmes[154] : l'obtention d'un diplôme d'État. Cette réforme a contribué à sa valorisation officielle.

- **Un enseignement privé en pleine croissance**

Loin d'être des rivaux, les enseignements public et privé sont complémentaires. Par exemple, non organisé par l'État, l'enseignement pré-primaire est totalement pris en charge par le secteur privé. Ce niveau reste moins développé. Comme le pré-primaire, la place du secteur privé dans le primaire reste inférieure à 2 % (2012)[155]. La place du privé grandit depuis le secondaire et s'amplifie à l'enseignement supérieur. Les deux secteurs, privé et public, sont sous le suivi de la même inspection de l'État.

Si le passage limité de l'école primaire au secondaire public[156] semble être l'une des raisons de la multiplication des écoles secondaires privées, la majorité de ces dernières se trouvent dans les centres urbains pour deux raisons. La première raison est qu'une école ne peut être construite que par ceux qui ont des moyens et qui, en général, habitent les villes ou les

152 www.diplomatie.gouv.fr/fr/IMG/pdf/Burundi_fiche_curie_janvier_2014_cle0e918d.pdf, document consulté le 26 décembre 2014

153 http://fr.africatime.com/burundi/articles/etats-generaux-de-leducation-les-participants-ont-du-pain-sur-la-planche, site visité le 25 décembre 2014

154 http://www.diplomatie.gouv.fr/fr/IMG/pdf/Burundi_fiche_curie_janvier_2014_cle0e918d.pdf, document consulté le 26 décembre 2014

155 http://knoema.fr/atlas/Burundi/topics/%C3 %89ducation/%C3 %89ducation-primaire/Inscription-dans-des-%C3 %A9tablissements-priv%C3 %A9 s, site consulté le 29 décembre 2014

156 Environs 30 % d'élèves seulement passent de l'école primaire au secondaire. Ce pourcentage était de 3 % dans les années 80

centres urbains. Ils les construisent alors tout près de chez eux. La deuxième raison est la demande sociale de formation qui augmente avec les regroupements des gens, lesquels sont aussi dans les centres urbains.

L'enseignement supérieur affiche une forte croissance. L'augmentation rapide du nombre d'établissements et d'effectifs d'étudiants est très frappante. Alors que depuis 1964, date de naissance de l'unique université nationale du Burundi jusque récemment en 2014, il n'existait que cinq institutions d'enseignement supérieur public[157], l'enseignement supérieur privé commencé vers les années 2000, comptait jusqu'en 2012, dix-sept (17) établissements, dont dix (10) universités et sept (7) instituts. Leurs effectifs d'étudiants sont passés d'un tiers (1/3) de ceux de l'enseignement supérieur public en 2004 (Tonye, 2008), pour dépasser ceux du public après dix ans : en 2014[158], les effectifs d'étudiants sont passés à environs 16. 000 (privé) contre moins de 15. 000 (public).

Malgré le coût élevé de l'enseignement privé au Burundi, son succès peut être justifié par la diversité des spécialités offertes, mais aussi par les conditions d'études souvent meilleures par rapport au public.

Qu'en est-il de la situation de l'enseignement de l'informatique au Burundi ? C'est l'objet du point suivant.

4.2. L'informatique en éducation au Burundi

Dans les pays du nord, l'informatique et les sciences des TIC en éducation sont des priorités des politiques éducatives et on note une tendance à rendre universel l'enseignement de l'informatique à tous les niveaux de la scolarité. Quant à l'informatique pour tous, l'Afrique a, en général[159], encore du chemin à faire étant donné que même l'universalité de la scolarisation des enfants n'est pas encore atteinte. Concernant l'informatique, Karsenti & Tchameni Ngamo (2009) soulignent une situation paradoxale et frappante : une absence généralisée de l'informatique et des TIC dans les écoles en Afrique alors qu'elles sont fréquentes dans la vie sociale.

157 http://www.diplomatie.gouv.fr/fr/IMG/pdf/Burundi_fiche_curie_janvier_2014_cle0e918d.pdf : l'université du Burundi (UB), l'École Normale Supérieure (ENS), l'Institut Nationale de Santé Publique (INSP), l'Institut Supérieur des Cadres Militaires (ISCAM) et l'Institut Supérieur de Police (ISP), document consulté le 26 décembre 2014

158 www.diplomatie.gouv.fr/fr/IMG/pdf/Burundi_fiche_curie_janvier_2014_cle0e918d.pdf, document consulté le 26 décembre 2014

159 Ici des nuances doivent être faites, certains pays africains sont plus avancés que d'autres : les pays de l'Afrique du Nord, l'Afrique du Sud, certains pays de l'Afrique de l'Ouest.

Dans le cas spécifique du Burundi, la situation de l'informatique éducative a été décrite par Claver Nijimbere. Si l'informatique en général et les TIC en particulier occupent de plus en plus une place importante dans la vie sociale (Nijimbere, 2012) et administrative (Butoyi et al., 2012), cette place est limitée dans la formation scolaire. Malgré son absence à l'école, l'informatique est une nécessité. Notons l'absence de politique éducative pour la promotion de l'informatique.

a. *Niveau primaire : non-existence d'un enseignement de l'informatique*

La rencontre de l'informatique avec l'école primaire burundaise est récente : elle date de l'année 2008. Certains enseignants et leurs responsables hiérarchiques ont formellement connu pour la première fois une initiation à l'informatique et aux TIC à travers le projet Initiative francophone pour la Formation à Distance ces Maîtres (IFADEM)[160]. Cette initiation ne s'est pas seulement limitée aux seuls instituteurs mais, a été étendue aux directeurs d'écoles, aux conseillers pédagogiques, aux inspecteurs même si tout le monde n'a pas été formé. Avec ce projet, une approche hybride, associant les formations en présentiel et à distance, a été privilégiée. Le contexte de ressources matérielles et numériques limitées a obligé de recourir à la radio scolaire pour la formation à distance. Si les enseignants ont été contents de la formation reçue, sa mise en œuvre dans leurs pratiques professionnelles reste quasi inexistante dans ce contexte d'enclavement numérique qui rend totalement inaccessibles des ressources numériques pour les différents acteurs (Voulgre & Netto, 2014).

Sauf pour des fins d'initiatives privées et individuelles ou une politique qui interviendrait prochainement à la suite de ce projet de formation pour institutionnaliser l'enseignement ou l'instrumentalisation de l'informatique dans les pratiques enseignantes à l'école primaire ou pour faire suite aux recommandations qui sont de plus en plus orientées vers la scolarisation des technologies (Karsenti & Collin, 2010), à notre connaissance, « *jusqu'à maintenant, il n'y a pas encore de formation à l'informatique et aux TIC en général pour les élèves de l'école primaire au Burundi, aussi bien comme outil que comme objet d'enseignement, mais elle pourrait bien commencer dans un proche avenir, après la formation de tous les enseignants.* » (Nijimbere, ibidem.).

Si l'un des objectifs principaux de l'introduction de l'enseignement fondamental est de garder le plus longtemps possible les enfants à l'école jusqu'à l'âge de 15-16 ans, les principaux

160 http://www.ifadem.org

nouveaux cours supposés leur apporter des compétences nécessaires à une auto-prise en charge sont ceux d'entrepreneuriat et d'arts. Il n'y a aucune acquisition des compétences informatiques. Si les proportions des écoles élémentaires privées restent numériquement beaucoup moins nombreuses que celles publiques, certaines d'entre elles, essentiellement privées, organisent une initiation à l'informatique orientée à l'usage des technologies.

b. Niveau secondaire : informatique comme objet d'enseignement technique

Contrairement au primaire public, le niveau secondaire connaît un enseignement de l'informatique, limité à quelques lycées, essentiellement techniques. Organisée en filières de spécialité informatique, la formation dispensée aux élèves leur confère des profils de techniciens, obtenus après trois ans de formation post collège, soit en informatique de maintenance, soit en informatique des opérateurs, soit en informatique de gestion ou de secrétaires de bureau.

Comme d'ailleurs dans beaucoup de pays africains (Karsenti & Tchemeni Ngamo, 2007), l'enseignement de l'informatique au Burundi est fondé sur l'approche objet. Il s'agit d'un enseignement de l'informatique théorique et donc rarement orientée programmation informatique. L'utilisation de l'informatique comme outil d'enseignement pour instrumenter d'autres disciplines est aussi rare. Les situations contrastées relativement aux approches objet/outil de l'informatique éducative révèlent de nouvelles formes de fracture numérique en Afrique (Nijimbere, 2012). Si ce choix de l'approche objet de l'informatique est aujourd'hui privilégié dans les pays du nord, les motivations ne sont pas les mêmes avec les pays du sud. Dans les pays développés, des technologies informatiques à la fois abondantes et « guidant leurs utilisateurs »[161], ont occulté la nécessité d'apprendre les bases conceptuelles de l'informatique, pourtant incontournables pour un apprentissage raisonné et fructueux de l'informatique. Plutôt qu'apprendre à les programmer et à comprendre leur fonctionnement, la priorité dans la formation a été donnée à leurs utilisations. La nécessité d'une approche objet a été motivée par des limites constatées avec cette approche outil de l'informatique dans la formation scientifique informatique d'un élève.

Dans les pays du sud, la problématique est davantage celle de manque de ressources matérielles (ordinateurs, électricité, Internet…). Au Burundi par exemple, le manque d'ordinateurs suffisants pour des classes souvent pléthoriques fait que la discipline informatique est

161 C'est un discours souvent marchand lié au marketing

dispensée théoriquement, la pratique occupant un temps relativement court. Malgré les conditions matérielles souvent difficiles dans lesquelles cette spécialité informatique est enseignée, la facilité d'insertion professionnelle des diplômés ou même de possibilités de création de leurs propres emplois, contribue à la valorisation sociale des spécialités informatiques.

c. *Enseignement supérieur : d'une informatique de spécialité à celle d'une discipline d'enseignement supérieur*

Si beaucoup de réformes sont en cours d'exécution dans l'enseignement supérieur public, le secteur de l'informatique semble ne pas encore une priorité du Gouvernement du Burundi. À l'UB par exemple, les principaux objectifs à atteindre d'ici 2017 sont consignés dans un Plan Stratégique adopté pour dynamiser cette institution en 2012. Parmi eux figurent *« la généralisation de l'utilisation des TIC et la modernisation de la gestion des ressources documentaires »*[162] et, de façon générale, *« la diversification des filières et leur professionnalisation dans le secteur public et privé, à la fois dans les domaines classiques et ceux plus nouveaux tels que la communication, l'informatique, les droits de l'homme et la gestion des conflits. »*[163].

Contrairement à l'enseignement secondaire où l'informatique est enseignée comme discipline de spécialité essentiellement dans les lycées techniques, dans les institutions d'enseignements supérieurs publics, *« l'informatique est enseignée aussi bien comme discipline et considérée comme outil d'enseignement au sein des autres disciplines. Néanmoins, l'informatique outil occupe une petite place, seulement chez quelques enseignants volontaires et novateurs » (Nijimbere, 2012, p. 4)*. Récemment un département des TIC, comprenant trois filières (génie informatique, génie logiciel et informatique mathématique) a été créé en 2009 à l'UB, l'unique université nationale. À l'exception de ce département, toutes les autres institutions d'enseignement supérieur public ne disposent d'aucun parcours de spécialité informatique.

L'enseignement supérieur privé est venu combler les lacunes du secteur public. Le dynamisme et le succès de ce secteur ont été fondés sur les offres de formation bien ciblées et caractérisées, absentes ou difficilement attribuées à l'enseignement public alors qu'elles font

162 www.diplomatie.gouv.fr/fr/IMG/pdf/Burundi_fiche_curie_janvier_2014_cle0e918d.pdf, document consulté le 26 décembre 2014

163 Ibidem.

objet de fortes demandes chez les étudiants. L'informatique comme d'autres filières socialement valorisées, souvent manquantes dans le secteur public, se retrouvent dans leur quasi-totalité, dans des institutions d'enseignement supérieur privé et ont fait l'objet de premières offres. Dans ces dernières institutions, non seulement l'informatique est enseignée à la fois comme un parcours de spécialité, mais aussi comme une discipline supérieure dans les autres spécialités autres que l'informatique. Comme dans le secteur public, l'approche outil de l'informatique reste moins développée que celle d'objet.

La formation à distance est encore rare au Burundi. Totalement absente dans les institutions d'enseignement public actuellement, une seule université privée, l'Université Lumière de Bujumbura (ULB), la pratique déjà, en partenariat avec l'Université Laval du Canada (ULC). Pour des formations en informatique, le programme suivi est totalement canadien. Adoptant une approche hybride, les cours à distance sont suivis par visioconférence à partir de l'ULB. D'autres étudiants, à titre individuel, suivent des formations, en général de master en informatique ou dans d'autres domaines, dans des universités étrangères. La contribution de l'AUF de Bujumbura est grande : elle met à leur disposition une connexion Internet (Tonye, 2008). Néanmoins, le coût cher de cette formation et l'Internet non encore généralisé au Burundi limitent largement le nombre d'étudiants qui suivent ce type de formation.

4.3. Analyse des tendances et suggestions

a. Tendances actuelles : pas de prise en compte prioritaire d'un enseignement général d'informatique

Au Burundi, l'informatique est enseignée du lycée technique à l'enseignement supérieur. L'absence de ressources matérielles laisse entrevoir un choix pour une approche objet de l'informatique orientée plus théorique que pratique. L'enseignement de l'informatique seulement dans les lycées techniques révèle une vision de cette spécialité tournée vers la technique. Cette représentation de l'informatique semble contraster avec le choix de son approche objet plus orienté *théorique.* Malgré ses potentialités dans les apprentissages, l'approche par projet, utilisée dans la scolarité française, est encore absente au Burundi.

Si la formation des spécialistes de l'informatique ne semble être prise en compte par des institutions publiques d'enseignement supérieur, l'enseignement de l'informatique est l'une des caractéristiques fondamentales du secteur privé : elle est généralisée dans les universités pri-

vées qui sont récentes. Malgré tout, dans le privé, la formation reste en général limitée au niveau de la licence.

La politique éducative du Gouvernement du Burundi est inscrite dans la politique générale : Cadre Stratégique de Lutte contre la Pauvreté I et II (CSLP I : 2005-2010 et CSLP II : 2011-2015)[164]. Les réalisations des dix prochaines années ne visent pas la création d'une nouvelle discipline informatique comme sa projection dans la « Vision Burundi 2025 »[165] qui fait suite aux deux cadres stratégiques précédents le montre :

> *« la promotion des technologies nouvelles, une de ses priorités. Cette promotion se fera au niveau des réformes de l'enseignement à tous les niveaux. Il sera fait une place de choix à la science et à la technologie dans les programmes pédagogiques, en mettant notamment l'accent sur la recherche et sur les TIC qui servent généralement de levier important pour acquérir et adopter de nouvelles technologies de pointe. » (p. 53).*

Selon les niveaux de formation, cette politique ambitionne d'atteindre les objectifs suivants dans le système éducatif :

1. dans l'enseignement fondamental, c'est réussir la scolarisation universelle avec un enseignement de qualité par la mise à disposition des ressources nécessaires. Relativement à l'enseignement de l'informatique, rien n'a été prévu dans le CSLPII concernant l'enseignement fondamental malgré les réformes en cours.
2. dans le secondaire et la formation professionnelle, l'enseignement technique et le secteur des métiers professionnalisants devraient bénéficier d'un effort particulier, notamment par la création de nouvelles filières dont celles liées aux TIC.
3. quant à l'enseignement supérieur, la continuité de la réforme du système BMD, le recours aux TIC dans l'enseignement, la promotion de l'assurance qualité, le développement des sciences, de la technologie et de la recherche ont été affirmés.

Il n'y a pas de politique prioritaire du Gouvernement concernant l'enseignement de l'informatique au Burundi : cela est vraiment déplorable au XXI^{ème} siècle. La volonté de recourir aux

164 http://planipolis.iiep.unesco.org/upload/Burundi/Burundi_Document_CSLPII.pdf, document consulté le 25 décembre 2014

165 http://www.burundiconference.gov.bi/IMG/pdf/Vision_Burundi_2025_synthese_FR.pdf, document consulté le 25 décembre 2014

TIC dans les enseignements semble être en phase avec des recommandations sur les TIC issues du forum de Tunis en 2013 (Rapport de l'Éducation continentale, 2014).

b. Suggestions

Le Burundi est de loin incomparable à la France en ce qui concerne l'enseignement en général et l'enseignement et l'apprentissage de l'informatique en particulier. Beaucoup de leçons peuvent être tirées de l'expérience française. La scolarisation universelle qui se cherche encore au Burundi est presque une réalité en France[166] (99 %). Au Burundi, le passage d'un cycle de formation à un autre est sélectif. Moins d'un tiers d'élèves passent du primaire au secondaire[167]. L'un des principaux objectifs de l'école fondamentale est de garder à l'école le plus longtemps possible les élèves qui ne vont pas continuer le secondaire pour atteindre ou approcher l'âge de la majorité. La gratuité des frais de scolarité à l'école primaire en 2005 a été un grand défi pour le pays : elle a fait augmenter le taux bruit de scolarisation de 83 % en 2004-2005 à 135 % en 2013-2014[168]. Si la réforme conduisant à l'école fondamentale consiste en la prolongation de l'ancienne école primaire (de 6 ans) jusqu'à trois premières années du collège (7e, 8e et 9e)[169], même à ce niveau, l'autonomie de l'élève n'est pas encore atteinte pour se prendre en charge. Des cours comme *entrepreneuriat* et *arts* (musique, dessins, sports…) ont été introduits dans sa formation pour, comme l'État le pense, leur permettre d'acquérir des compétences professionnalisantes nécessaires. Au niveau de l'enseignement secondaire et supérieur, la vision du Burundi est de généraliser l'approche « outil » de l'informatique à travers les disciplines déjà existantes.

Ces choix diffèrent de ceux de la France. En France, l'approche *outil* de l'informatique a déjà montré ses limites, une fois non associée à l'approche *objet* : les nouvelles orientations valorisent une hybridation de ces deux approches. Si l'informatique n'est pas encore enseignée au primaire en France, les débats en cours sur son enseignement semblent impensables au Burundi.

166 http://www.statistiques-mondiales.com/burundi.htm, site consulté le 29 décembre 2014

167 La politique de l'école fondamentale ne semble pas venir changer cette situation parce qu'avec cette politique il semble être question de préparer les affronter la vie professionnelle : la poursuite des études après ce niveau sera conditionné par la réussite à un concours.

168 http://burundi-agnews.org/sports-and-games/? p=11517. Document consulté le 9 janvier 2015. Ce taux s'accompagne d'abandons à cause des conditions difficiles (classes pléthoriques…) dans lesquelles les enfants se trouvent

169 Au Burundi, la première année du collège commence en 7e et non en 6e comme en France. Les 8e et 9e années correspondent respectivement à la 6e et la 5e année dans le système français.

L'initiative IFADEM a permis d'initier à l'informatique les enseignants déjà en poste, recrutés précédemment avec un niveau de scolarité faible. La prise de conscience d'un recrutement d'enseignants mieux formés est désormais mondialement reconnue mais compte tenu des besoins, la formation en court d'emploi ne peut pas être un moyen à exclure. Au Burundi, même si quelques acteurs sont conscients de l'importance de familiariser les enseignants aux usages des TICE, les infrastructures ne sont pas suffisantes pour penser une généralisation immédiate de formations en TICE sur le territoire.

L'approche IFADEM privilégie la construction de ressources disponibles sur papier avec des accompagnements tutorés et médiatisés à l'aide des téléphones mobiles bien plus présents que les ordinateurs. Le manque d'une vision de l'instauration d'enseignement de l'informatique au primaire est à remettre en perspective avec les moyens du pays et notamment un manque actuel de production d'électricité. Néanmoins, ne serait-il pas prudent pour le développement du Pays, d'anticiper davantage les besoins futurs en termes de créativités informatiques et numériques dont les bases de programmation et de raisonnement algorithmique sont primordiales, et dont un certain nombre de notions pourraient s'enseigner dès l'école fondamentale ?

Et si le Burundi a souscrit aux objectifs de l'Éducation pour tous (EPT) dans le cadre d'action de Dakar en l'an 2000, la recommandation portant scolarisation des technologies est encore absente malgré leurs potentialités escomptées : *« les TIC doivent être exploitées pour soutenir les objectifs de l'EPT à moindres frais. Ces technologies offrent des possibilités importantes pour diffuser les connaissances, améliorer l'apprentissage et développer des services éducatifs plus efficaces. »*[170]

Relativement à l'informatique dont l'enseignement fait une recommandation de l'UNESCO[171] avec une proposition de programme depuis plus de 20 ans, nous faisons des suggestions suivantes pour le cas du Burundi.

- **Une initiation à l'informatique et aux TIC à l'école fondamentale**

À la fin de l'école fondamentale, certains élèves entreront dans la vie courante pour se prendre en charge. Il est important de leur fournir des compétences informatiques en plus de celles acquises dans les cours prévus. Les compétences informatiques acquises en une initiation à l'informatique et l'utilisation des TIC contribueraient pour s'approprier les contenus

170 http://unesdoc.unesco.org/images/0012/001211/121147f.pdf, document consulté le 9 janvier 2015
171 http://www.epi.asso.fr/revue/76/b76p059.htm, site consulté le 9 janvier 2015

des cours prescrits notamment celui d'arts, de la musique pour ne citer que de ceux-là. Cette question d'initiation de l'informatique à l'école fondamentale fera objet de nos perspectives de recherches pour voir comment enseigner l'informatique aux élèves dans le contexte burundais.

- **Une spécialité informatique dans les lycées généraux**

Le choix d'un enseignement de spécialité dans les écoles techniques nous semble fondé. Il permet d'avoir des jeunes professionnels à la fin de la scolarité secondaire. Néanmoins, ce choix n'est pas suffisant. Un enseignement d'une discipline informatique dans tous les lycées généraux semble porteur de beaucoup d'intérêts : la familiarisation à la culture informatique. Étant donné que tous les élèves, à la fin du lycée, ne vont pas faire l'enseignement supérieur, ils auraient certaines connaissances informatiques qui leur permettraient de s'adapter plus facilement au monde social de plus en plus technologique notamment en termes de téléphonie mobile. Ceux, parmi eux, qui continueraient l'enseignement supérieur pour embrasser les spécialités relatives à l'informatique auraient alors une base de culture nécessaire à affronter le monde professionnel qui est de plus en plus demandeur de ces compétences.

- **Une spécialité informatique à l'ENS et à l'IPA**

Si un département des TIC a été récemment créé à l'UB pour former des spécialistes en informatiques, il ne faudrait pas oublier le secteur des futurs formateurs orienté vers l'enseignement. Avec la création d'un cours d'initiation à l'informatique et aux TIC à l'école fondamentale, leurs futurs enseignants pour qui n'est prévue aucune formation en informatique, devraient suivi un cours d'informatique dans leur parcours de formation universitaire[172]. Quant à ceux se destinant aux métiers de formation à l'enseignement secondaire, une spécialisation à l'enseignement de l'informatique devrait être donnée.

Deux institutions d'enseignement supérieur de formations de futurs enseignants existent au Burundi : Ecole Normale Supérieure (ENS) et Institut de Pédagogie Appliquée (IPA)[173]. Les différentes spécialisations (maths, physique, français…) commencent avec le début de l'enseignement supérieur. Si l'informatique est apprise comme discipline supérieure, la création d'une discipline informatique au lycée nécessite au préalable la formation des enseignants

172 Les enseignants de l'école fondamentale sont choisis parmi ceux qui terminent le premier cycle de l'ENS et de l'IPA et donc les futurs licenciés dans le système BMD

173 Si l'ENS est indépendante de l'université du Burundi (UB), l'IPA est l'une des structures qui composent l'UB l'UB. L'ENS et l'IPA sont donc des structures différentes quant à leur organisation mais qui, toutes, forment des futurs enseignants.

par les institutions universitaires habilités. Cette spécialisation en informatique des futurs enseignants viendrait avant la prescription de la discipline informatique à l'école fondamentale et au lycée : ce serait les enseignants formés qui tiendraient cet enseignement.

- **Construction et équipement des écoles et universités.**

Si la vision de l'État d'ici l'an 2025 est ambitieuse, la question des ressources matérielles et humaines est l'un des défis auxquels le pays fait face jusqu'aujourd'hui : manque d'enseignants, manque de locaux dans un contexte où les effectifs d'élèves et d'étudiants continuent à grimper. Non seulement de nouvelles écoles et universités doivent être construites, mais aussi elles doivent être équipées. Selon cette vision, les équipements ne doivent pas se limiter au niveau des bancs. Il s'agit de trouver les moyens pour des ressources évoluées telles que l'électrification des salles de cours, les équipements en ordinateurs, Internet…

- **Pérennisation des espaces numériques d'IFADEM**

Cette suggestion fait référence à la précédente. L'initiative IFADEM a mis en place beaucoup de ressources tant matérielles que numériques. Des accords territoriaux seraient à mettre en place pour permettre l'entretien et la pérennisation des espaces, de leurs équipements, des personnels et des besoins de ces personnels pour la formation de la population au-delà des seuls enseignants du secteur. Une fois bien ces conditions conventionnées et financées, le dispositif IFADEM pourrait prendre de l'ampleur et relayer l'ouverture à l'informatique dans la formation continue des enseignants.

Il serait profitable que le Burundi soutienne la longue période que nécessite l'appropriation des équipements informatiques au service de la pédagogie. L'initiation à l'informatique et aux technologies numériques mais aussi aux méthodes nouvelles d'enseignement contribue au renouveau éducatif. En effet, dans sa compilation des recherches portant sur les réformes de l'enseignement obligatoire, McKinsey & Company (2007) mettent en évidence l'efficacité de l'entrée par les réformes des méthodes de travail pour réussir des réformes éducatives plutôt que d'entrer par les réformes des structures et des ressources, même si, selon lui, cette dernière entrée est la tendance actuelle.

4.4. Perspectives de recherches

- **Comment enseigner l'informatique dès l'école fondamentale au Burundi ?**

Environs 1700 instituteurs[174] ont reçu une petite initiation permettant d'utiliser quelques fonctionnalités d'un ordinateur à travers le projet IFADEM. Ces enseignants font malheureusement face à un manque de ressources matérielles et humaines pour prolonger l'initiation et s'approprier des savoir-faire. La formation a essentiellement porté sur les méthodes d'enseignement du français dans des disciplines non linguistiques.

La visée qualitative que quantitative permet d'interroger les effets de la formation reçue chez les acteurs dans leurs pratiques. Comment cette formation a contribué au renouvellement des pratiques professionnelles chez les bénéficiaires qui la mettent en application ? Comment cette initiation à l'usage des technologies informatiques peut impulser une transmission d'une culture numérique dans la société burundaise ? Il serait intéressant de comprendre comment d'une part, les espaces numériques mis en place dans le cadre d'IFADEM pourraient être investis par les partenaires locaux au service du Pays et, d'autre part, comment dans un contexte de ressources informatiques précaires au Burundi, des notions informatiques peuvent être enseignées dès l'école fondamentale notamment par la programmation des robots tels que BEE-BOT (Komis & Misirli, 2011) et PRO-BOT moins exigeants en électricité.

- **Formations tutorées via la téléphonie mobile**

La formation ouverte et à distance est encore extrêmement rare au Burundi (Tonye, 2008). Le manque d'Internet et de culture informatique semble être les causes principales. Comment dans ce contexte d'enclavement numérique, une formation à distance pourrait-elle être envisagée ?

La téléphonie mobile au Burundi connaît des progrès extraordinaires de pénétration sociale par rapport aux autres technologies numériques (Nijimbere et al., 2013), mais, elle reste encore trop déconnectée du système éducatif. L'expérimentation IFADEM à Madagascar s'appuyant sur la téléphonie mobile pour l'accompagnement des formés est sans doute une piste nécessaire à explorer dans le cas du Burundi, dans l'attente de la connexion au réseau des fibres optiques généralisées aux différentes provinces du pays[175]. Enfin, il serait intéressant

174 Cet effectif ne représente qu'un dixième au plus de tous les enseignants qui ont besoin de cette formation
175 http://www.isanganiro.org/spip.php ? article4239, site consulté le 18 décembre 2014

d'étudier à partir d'expérimentations[176], comment le téléphone mobile pourrait avoir une place en classe dans des situations d'apprentissages pour et par les élèves.

176 http://www.ifadem.org/sites/default/files/divers/Focus_Mobile_Madagascar.pdf, document consulté le 28 décembre 2014

Bibliographie

1 Aguirre, E., & Raucent, B. (2002). L'apprentissage par projet… Vous avez dit projet ? Non, par projet. In 19e colloque de l'Association Internationale de Pédagogie Universitaire (AIPU), Louvain-la-Neuve-Belgique (Vol. 29). Consulté à l'adresse http://tecfa.unige.ch/proj/cvs/semin/doc_semin2/ColloqueAIPU/projets.pdf

2 Alayrangues, S., Baudé, J., Debled-Rennesson, I., De la Higuera, C., & Marquet, P. (2013). Rapport de la SIF sur la Formation des Enseignants d'ISN (p. 16). Consulté à l'adresse http://www.societe-informatique-de-france.fr/actualite/2013/Rapport_SIF_sur_ISN.pdf

3 Andrade-Barroso, G., Guillo, L., & Simonin, M. (2013). Terminale S, option ISN – Inria. http://www.inria.fr/centre/rennes/actualites/terminale-s-option-isn

4 Aoudé, P. (2011). Les futurs enseignants du primaire face aux TIC : Questions de compétences et de formation. Le cas du tableur. Thèse de doctorat, Université Paris Descartes – Paris V. Consulté à l'adresse https ://tel.archives-ouvertes.fr/tel-00767440

5 Archambault, J. P. (2009). Les travaux du groupe Informatique et TIC de l'ASTI : un programme informatique pour le lycée en France. In G.-L. Baron, É. Bruillard & V. Pochon (dir.), Informatique et progiciels en éducation et en formation : continuités et perspectives (p. 39-50). Lyon. France.

6 Archambault, J.-P. (2010). L'informatique discipline scolaire : un long cheminement. EpiNet-La revue électronique de l'association Enseignement Public & Informatique, 129. Consulté à l'adresse http://edutice.archives-ouvertes.fr/docs/00/56/42/50/HTML/

7 Archambault, J. P. (2011). Un enseignement de la discipline informatique en Terminale scientifique. In G.-L. Baron, É. Bruillard, & V. Komis (dir..), Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques. Athènes, Université de Patras, Grèce.

8 Archambault, J. P., Berry, G., & Nivat, M. (2012). L'informatique à l'école : il ne suffit pas de savoir cliquer sur une souris. http://www.rue89.com/2012/06/28/linformatique-lecole-il-ne-suffit-pas-de-savoir-cliquer-sur-une-souris-233389

9 Arnaud, P. (1999). Des moutons et des robots : Architectures de contrôle réactive et déplacements collectifs de robots. (Presses Polytechniques et universitaires romaines).

10 Arsac, J. (1989). La didactique de l'informatique : un problème ouvert ? In Colloque francophone sur la didactique de l'informatique Université René Descartes Paris les 1, 2 et 3 septembre 1988 (p. 9–18). http://edutice.archives-ouvertes.fr/edutice-00359090/

11 Arsac, J. (1992). Itération et récursivité. Actes de la troisième rencontre francophone de didactique de l'informatique Sion, du 6 au 11 juillet 1992, 81–92.

12 Artigue, M. (2002). Learning mathematics in a CAS environment : The genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. International Journal of Computers for Mathematical Learning, 7(3), 245–274.

13 Artigue, M. (2010). Le numérique dans l'enseignement en France et à l'étranger : le cas des mathémùatiques. http://afae.fr/IMG/pdf/Actes_APIGEN_2010.pdf

14 Balliot, A., Darrigrand, É., Gaude, L., Meunier, M., Millet, M., & Pinsard, D. (2011). Introduction de l'algorithmique au lycée. Institut de recherche sur l'enseignement des mathématiques de Rennes. http://publimath.irem.univ-mrs.fr/biblio/IRN11001.htm

15 Bardram, J.. (1997). Plans as Situated Action : An Interaction Approch to Workflow Systems. In Proceedings of the Fifth European conference on Computer Supported Cooperative Work, kluwer Academic Publishers. (p. 17-32).

16 Barma, S. (2008). Un contexte de renouvellement des pratiques en éducation aux sciences et aux technologies : une étude de cas réalisé sous l'angle de la théorie de l'activité. Thèse de doctorat, Université Laval, Québec.

17 Barma, S. (2010). Analyse d'une démarche de tranfromation des pratiques en sciences dans le cadre du nouveau programme de formation pour le secondaire, à la lumière de la théorie de l'activité. Revue canadienne de l'éducation, 33(4), pp. 677-710.

18 Baron, G., Bounay, M., Dautrey, P., Guelfucci, J., Hebert, D., Muller, P.,… Tourtelier, P. (1981). Dix ans d'informatique dans l'enseignement secondaire. Consulté à l'adresse http://lara.inist.fr/handle/2332/1250

19 Baron, G.-L. (1987). La constitution de l'informatique comme discipline scolaire : le cas des lycées. Thèse de doctorat. Université Paris Descartes.

20 Baron, G.-L. (1989). L'informatique discipline scolaire ? le cas des lycées. Presses Universitaires de France, 230p.

21 Baron, G.-L. (1990). L'informatique en éducation. Revue française de pédagogie, 57–77.

22 Baron, G.-L. (1994). L'informatique et ses usagers dans l'éducation. Note de synthèse pour l'habilitation à diriger des recherches, Université Paris, 154p.

23 Baron, G.-L. (2009). Pratiques, compétences et savoirs dans l'enseignement supérieur. In G.-L. Baron, É. Bruillard & V. Pochon (dir.), Informatique et progiciels en éducation et en formation : continuités et perspectives (p. 77-80). Lyon. France.

24 Baron, G.-L. (2011). L'informatique et son enseignement dans l'enseignement scolaire

général français : enjeux de pouvoir et de savoirs. In J. Lebeaume ; A. Hasni et I. Harlé (dir.), *Recherches et expertises pour l'enseignement scientifique : Technologies, Sciences, Mathématiques* (p. 79-90). Montréal. Québec. De boeck,

25 Baron, G.-L. (2012). L'informatique en éducation : quel(s) objets d'enseignement ? In Le « e-Dossiers de l'audiovisuel : l'Éducation aux cultures de l'information (p. 81-88). ENS Cachan.

26 Baron, G.-L., & Bruillard, É. (2001). Une didactique de l'informatique ? Revue française de Pédagogie, 163–172.

27 Baron, G.-L., & Bruillard, É. (2008). Technologies de l'information et de la communication et indigènes numériques : quelle situation ? Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF), 15. Consulté à l'adresse http://telearn.archives-ouvertes.fr/hal-00696420/

28 Baron, G.-L, Bruillard, É., & Bruillard. (1996). L'informatique et ses usagers dans l'éducation. PUF.

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

30 Baron, G.-L., Bruillard, É., & Komis, V. (2011). In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif ; analyse des pratiques en enjeux didactiques. Athènes, Université de Patras.

31 Baron, G.-L., Bruillard, É., & Pochon, L.-O. (2009). Enjeux didactiques de l'informatique et de ses outils : vingt ans après : regards sur un cheminement. In G.-L. Baron, É. Bruillard & V. Pochon (dir.), Informatique et progiciels en éducation et en formation : perspectives et continuités (p. 9-17). Lyon, France.

32 Baron, G.-L., & Denis, B. (1993). Regards sur la robotique pédagogique. Actes du 4e colloque international sur la robotique pédagogique. Paris : INRP, technologies nouvelles et éducation. Paris.

33 Baron, G.-L., & Voulgre, E. (2013). Initier à la programmation des étudiants de master de sciences de l'éducation ? Un compte rendu d'expérience. In G.-L. Baron, É. Bruillard & B. Drot-Delange (dir.), Sciences et technologies de l'information et de la communication en milieu éducatif : Objets et méthodes d'enseignement et d'apprentissage, de la maternelle à l'université. Clermont-Ferrand, Université Blaise Pascal.

34 Baroux, D., & Prouteau, C. (2013). A propos des exercices d'algorithmique du bac S 2012.

Bulletin de l'APMEP, (503), 132-138.

35 Batard, É. (1996). Entre informatique-outil et informatique-science : la question de l'usage : le cas de l'enseignement de la programmation. In Les actes de la 5e rencontre francophone sur la didactique de l'informatique 10-11 et 12 avril 1996 (p. 117-132). Monastir – Tunisie.

36 Baudé, J. (2010a). L'option informartique des lycées dans les années quatre-vingt et 90. Deuxième partie : le développement de l'option. Vers une généralisation. EPI, enseignement, informatique, TICE, option informatique, lycée, ITIC, historique. Consulté à l'adresse http://www.epi.asso.fr/revue/histo/h10oi_jb2.htm

37 Baudé, J. (2010b). L'option informatique des lycées dans les années quatre-vingt et 90. EPI, enseignement, informatique, TICE, option informatique, lycée, ITIC, historique, (128-129), 18.

38 Baudé, J. (2010c). L'option informatique des lycées dans les années quatre-vingt et 90. Première partie : la naissance d'une option. EPI, enseignement, informatique, TICE, option informatique, lycée, ITIC, historique, (128-129). Consulté à l'adresse http://edutice.archives-ouvertes.fr/docs/00/56/45/59/HTML/h10oi_jb1.htm

39 Baudé, J. (2010d). L'option informatique des lycées dans les années quatre-vingt et 90. Troisième partie : suppression, rétablissement et nouvelle suppression de l'option. Une politique en dents de scie. EPI, enseignement, informatique, TICE, option informatique, lycée, ITIC, historique. http://www.epi.asso.fr/revue/histo/h10oi_jb3.htm

40 Baudé, J. (2011). Quelques points de repères dans une histoire de 40 ans : l'association Enseignement Public et Informatique (EPI), de février 1971 à février 2011. Consulté à l'adresse http://www.epi.asso.fr/revue/histo/h11epi_jb.htm

41 Beaudouin-Lafon, M. (2010). Informatique : information, interaction et automatisation… Consulté 25 janvier 2014, à l'adresse http://www.epi.asso.fr/revue/articles/a1005c.htm

42 Beauné, A. (2010). Théorie de l'Activité : applications au domaine des TICE. http://www.adjectif.net/spip/spip.php ? article77&lang=fr

43 Bérard, J.-M. (1992). Peut-on définir un ensemble minimal de connaissances et de compétences en informatique permettant une utilisation rationnelle de l'ordinateur ? In G.-L. Baron & J. Baude (dir.), Intégration de l'enseignement et la formation des enseignants (p. 215-221). Actes du colloque des 28 au 29 janvier 1992 au CREPS de Châtenay-Malabry, coédition INRP-EPI.

44 Bernard, S., Clément, P., & Carvalho, G. S. (2007). Méthodologie pour une analyse didactique des manuels scolaires, et sa mise en øeuvre sur un exemple. Consulté à l'adresse

http://repositorium.sdum.uminho.pt/handle/1822/7082

45 Berry, G. (2005). La révolution numérique. L'enseignement de l'informatique de la maternelle à l'université. Conférence-débat, 4-6.

46 Berry, G. (2008). Comment donner une culture générale informatique à tous les élèves ?. EPI : Salon éducatice. http://www.epi.asso.fr/revue/docu/d0901a.htm

47 Béziat, J. (2005). Distance et B2i. Distances et savoirs, Vol. 3(3), 357-376. http://doi.org/10.3166/ds.3.357-376

48 Béziat, J. (2012). Informatique, outil ou objet ? Permanence d'une question. – [Adjectif]. http://www.adjectif.net/spip/spip.php ? article177&lang=fr

49 Bibang-Assoumou, H. (2013). L'intégration du XO dans les environnements d'apprentissages : cas à l'école ENS/B au Gabon. Thèse de doctorat, Université Laval.

50 Bloch, L. (2013). Révolution dans l'enseignement de l'informatique ? L'École 42. Consulté à l'adresse http://www.epi.asso.fr/revue/articles/a1305c.htm

51 Blondel, F.-M., & Bruillard, E. (2007). Comment se construisent les usages des TIC au cours de la scolarité ? Le cas du tableur. l'usage en travaux, Les dossiers de l'ingénierie éducative. CNDP, 139–147.

52 Blondel, F.-M., & Tort, F. (2007). Comment évaluer les compétences des lycéens en matière de tableur ? In Actes de la conférence EIAH 2007. Consulté à l'adresse http://halshs.archives-ouvertes.fr/hal-00161396/

53 Boudreault, Y., & Pregent, R. (2005). Projet intégrateur pluridisciplinaire exploitant la robotique pour les étudiants de première année en génie informatique et en génie logiciel. In Actes du 8e colloque francophone de Robotique Pédagogique (p. 81-88). La Ferté-Bernard, France.

54 Bourguiba, R. (2000). Conception d'une architecture matérielle recofigurable dynamiquement dédiée au traitement d'images en temps réel [Text]. Consulté 11 février 2014, à l'adresse http://cat.inist.fr/? aModele=afficheN&cpsidt=14196035

55 Bourguin, G., & Derycke, A. (2005). Systèmes Interactifs en Co-évolution Réflexions sur les apports de la Théorie de l'Activité au support des Pratiques Collectives Distribuées. Revue d'Interaction Homme-Machine, 6(1). Consulté à l'adresse http://www.europia.org/RIHM/V6N1/2-RIHM-Article%20Bourguin-Derycke%20PDF.pdf

56 Bracewell, R. J., Sicilia, C., Park, J., & Tung, I. (2007). The problem of wide-scale implementation of effective use of information and communication technologies for instruction : Activity theory perspectives. Tracking adoption and non-adoption of ICT

activities by teachers. In Annual Meeting of the American Educational Research Association (AERA), Chicago, IL. http://www.tact.fse.ulaval.ca/papers/Bracewell_aera2007.pdf

57 Brangier, É., & Bobillier-Chaumon, M.-E. (1996). Réflexions sur l'intervention en ergonomie de la programmation. Intervenir par l'ergonomie, SELF, Bruxelles, 2, 176–183.

58 Bréda, I. (2010). Corrélyse: Les enseignants témoignent. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00537136/

59 Breton, F. (2013). Première année de spécialité ISN au lycée : quel bilan ? – Inria. http://www.inria.fr/recherches/mediation-scientifique/informatique-et-sciences-du-numerique/bilan-isn

60 Brousseau, G. (1990). Le contrat didactique : le milieu. Recherches en didactique des mathématiques, 9(9.3), 309–336.

61 Bruillard, É. (1997). Des micro-mondes aux environnements d'apprentissage ouverts. In Les machiner à enseigner (p. 130-176). Consulté à l'adresse http://www.stef.ens-cachan.fr/annur/bruillard/MaE_ch_4.pdf

62 Bruillard, É. (1997). L'ordinateur à l'école : de l'outil à l'instrument. L'ordinateur à l'école : de l'introduction à l'intégration, 99–118.

63 Bruillard, É. (2005). Les manuels scolaires questionnés par la recherche In Bruillard Éric (dir.) Manuels scolaires, regards croisés. Manuels scolaires, regards croisés. CRDP de Basse-Normandie, Documents, actes et rapports sur l'éducation, Caen, 1(1), p, 13-36.

64 Bruillard, É. (2006). Informatique en contexte scolaire, enseignement, diffusion : quelles recherches. Séminaire de didactique des sciences expérimentales et des disciplines technologiques, 115–128.

65 Bruillard, É. (2009). Place de l'informatique dans l'enseignement secondaire : réflexions introductives. In G.-L. Baron, É. Bruillard & V. Pochon (dir.), Informatique et progiciels en éducation et en formation. Continuités et perspectives (p. 21-38). INRP.

66 Bruillard, É. (2012). Lire, écrire, computer : émanciper les humains et contrôler les machines. Consulté à l'adresse http://www.epi.asso.fr/revue/articles/a1209d.htm

67 Bruillard, É., Delozanne, É., Leroux, P., Delannoy, P., Dubourg, X., Jacoboni, P.,… Teutsch, P. (2000). Quinze ans de recherche informatique sur les sciences et techniques éducatives au LIUM. Revue Sciences et Techniques Educatives, 7(1), 87–145.

68 Brun, J. (1994). Evolution des rapports entre la psychologie du développement congitif et la didactique des mathématiques, In M. Artigue, R. Gras, C. Laborde, et P. Tavignon (dir), Vingt ans de didactique des mathématiques en France. Hommage à Guy Brusseau et Gérard

Vergnaud (p. 67-83). La pensée sauvage.

69 Brygoo, A. (1989). Option informatique : l'épreuve du Bac 1989 vue sous l'angle « d'informatique et Société ». Le Bulletin de l'EPI, (58), 163-169.

70 Burton, B. A. (2008). Informatics olympiads : challenges in programming and algorithm design. In Proceedings of the thirty-first Australasian conference on Computer science-Volume 74 (p. 9–13). Consulté à l'adresse http://dl.acm.org/citation.cfm ? Id=1378284

71 Burton, B. A., & Hiron, M. (2008). Creating informatics Olympiad tasks : Exploring the black art. Olympiads in Informatics, 2, 16–36.

72 Butoyi, A., Ndiongue, C. T., Zambo, B. N., Ngabonziza, F., Ngendakumana, A., & Viguier, L. (2012). Du programme national de réforme de l'administration publique, 85.

73 Cabane, R. (2012). Les enseignements du numérique en classe de Seconde Bilan des expérimentations pédagogiques menées en 2009-2010. *Revue Sticef. Org*, 22, 36.

74 Cartelli, A., Dagiene, V., & Futschek, G. (2010). Bebras contest and digital competence assessment : Analysis of frameworks. International Journal of Digital Literacy and Digital Competence (IJDLDC), 1(1), 24–39.

75 Causer, J.-Y. (2012). Le titre, le poste et la compétence. Questions Vives : Des usages des TIC à la certification des compétences numériques : quels processus de formation et de validation ?, 7(17).

76 Chaachoua, H., & Comiti, C. (2007). L'analyse du rôle des manuels dans l'approche anthropologique.

77 Chaguiboff, J. (1985). Informatique et apprentissages. Enfance, 38(1), 31–42.

78 Chaumon, M.-E. B., & Brangier, É. (1997). Etude des impacts du changement d'environnement de programmation chez les informaticiens. Consulté à l'adresse http://editoo.free.fr/perso/draft/Etude%20des%20impacts%20du%20changement%20d%92environnement%20de%20programmation%20chez%20les%20informaticiens.pdf

79 Chevallard, Y. (1991). La transposition didactique – Du savoir savant au savoir enseigné – RDM – Recherches en Didactique des Mathématiques -. http://rdm.penseesauvage.com/La-transposition-didactique.html

80 CNIDH. (2014). Les réformes du systèmes éducatif burundais et le droit à l'éducation. Rapport définitif (p. 269). Bujumbura. http://www.cnidh.bi/sites/default/files/LES%20REFORMES%20DU%20SYSTEME%20EDUCATIF%20BURUNDAIS%20ET%20LE%20DROIT%20A%20L%27EDUCATION.pdf

81 CSTA. (2005). The new educational imperative : improving high school computer education.

Final report of the CSTA curriculum improvment task force. Consulté à l'adresse http://www.eecs.umich.edu/cse/CS4HS/csta_pubs/CSTA_imperative.pdf

82 Dagdilelis, V., Balacheff, N., & Capponi, B. (1990). L'apprentissage de l'itération dans deux environnements informatiques. ASTER, (11), 45-66.

83 Dagiene, V. (2005). Competition in Information technology : an Informal learning. EuroLogo 2005, 228–234.

84 Dagienė, V., & Futschek, G. (2008). Bebras international contest on informatics and computer literacy : Criteria for good tasks. In Informatics Education-Supporting Computational Thinking (p. 19–30). Springer. Consulté à l'adresse http://link.springer.com/chapter/10.1007/978-3-540-69924-8_2

85 Dahl, O.-J., Dijkstra, E. W., & Hoare, C. A. R. (1972). Structured programming. Academic Press Ltd. Consulté à l'adresse http://dl.acm.org/citation.cfm ? Id=1243380

86 Darricarrère, J., & Bruillard, E. (2010). Utilisation des TIC par les professeurs de mathématiques de collège : discours et représentations. Bulletin de la société des enseignants neuchâtelois de sciences, 39. http://www.sens-neuchatel.ch/bulletin/no39/art4-39-juliana.pdf

87 Dauphin, F. (2012b). Culture et pratiques numériques juvéniles : Quels usages pour quelles compétences ? Questions Vives. Recherches en éducation, 7(17), 37-52. http://doi.org/10.4000/questionsvives.988

88 Delamotte, E. (2009). Enseigner l'informatique, enseigner la culture informationnelle. In G.-L. Baron, É. Bruillard, & V. Pochon (dir.), Informatique et progiciels en éducation et en formation : continuités et perspectives.

89 Delorme, C. (2010). Assises sur les réformes curriculaires – Brazzaville. Consulté 8 novembre 2014, à l'adresse http://www.cepec-international.org/ressources-documentaires/en-telechargement/799-assises-sur-les-reformes-curriculaires-brazzaville

90 Delozanne, É., Jarraud, P., & Muratet, M. (2011). Un projet Jeux sérieux pour approfondir l'apprentissage de la programmation en première année de l'université. In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'information et de la communication en milieu éducatif (p. 240-249). Athènes, Université de Patras.

91 Denning, P. (1997, révisé en 1999). Computer science : the discipline. Consulté à l'adresse http://denninginstitute.com/pjd/PUBS/ENC/cs99.pdf

92 Denoël, E., & Gérard, B. (2013). Enseignement obligatoire en communauté française de Belgique : comment s'inspirer des systèmes étrangers ?

93 Devauchelle, B. (2004). Le Brevet Informatique et Internet (B2i) : d'un geste institutionnel

aux réalités pédagogiques. Thèse de doctorat. Université Paris 8. 345p.

94 Dowek, G. (2005). Quelle informatique enseigner au lycée ? http://www-roc.inria.fr/who/Gilles. Dowek/lycee.html

95 Dowek, G. (2010). L'informatique : meilleure alliée des mathématiques ? http://www.apmep.asso.fr/IMG/pdf/Dowek_Info_math.pdf

96 Dowek, G., Archambault, J.-P., Bacceli, E., Cimelli, C., Cohen, A., Eisenbeis, C.,… Wack, B. (2011). Informatique et sciences du numérique : Spécialité ISN en terminale S. Editions Eyrolles.

97 Dowek, G., Archambault, J.-P., Baccelli, E., Boldo, S., Bouhineau, D., Cegielski, P.,… Mounier, L. (2011). Une introduction à la science informatique pour les enseignants de la discipline en lycée. Consulté à l'adresse http://hal.archives-ouvertes.fr/hal-00765226/

98 Drot-Delange, B. (2012). Enseignement de l'informatique, éducation aux technologies de l'information et de la communication en France, dans l'enseignement général du second degré. Spirale, 50, 25-37.

99 Drot-Delange, B. (2013). Enseigner l'informatique débranchée : analyse didactique d'activités. In Actes du congrès de l'Actualité de la Recherche et Éducation et en Formation (AREF-AECSE). Université de Montpellier.

100 Drot-Delange, B., & More, M. (2013). Attitudes envers l'informatique des élèves de terminale scientifique. Quelques résultats exploratoires. In G.-L. Baron, É. Bruillard & B. Drot-Delange (dir.), Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif. https ://edutice.archives-ouvertes.fr/edutice-00877150/

101 Dubois, D. (1975). Théories linguistiques, modèles informatiques, expérimentation psycholinguistique. Langages, (40), 30–40.

102 Duchâteau, C. (1989). Quand le savoir faire ne suffit plus. Rapport interne, 5. Consulté à l'adresse http://webapps.fundp.ac.be/cefis/publications/charles/savoir-faire-5-21.pdf

103 Duchâteau, C. (1992). From« DOING IT… » to« HAVING IT DONE BY… » : The Heart of Programming. Some Didactical Thoughts. In NATO Advanced Research Workshop » Cognitive Models and Intelligent Environments for Learning Programming. Consulté à l'adresse http://ipv6.fundp.ac.be/pdf/publications/54237.pdf

104 Duchâteau, C. (1993). Robotique-Informatique: mêmes ébats, mêmes débats, mêmes combats. In *Regards sur la robotique pédagogique-Actes du 4ème colloque sur la robotique pédagogique. Paris/Liège: INRP/Université de Liège* (p. 10–33).

105 Duchâteau, C. (1994). Faut-il enseigner l'informatique à ses utilisateurs. Actes de la

quatrième rencontre francophone de didactique de l'informatique. Consulté à l'adresse https ://pure.fundp.ac.be/ws/files/988521/54322.pdf

106 Duchâteau, C. (2002). Images pour programmer. Première partie : Apprendre les concepts de base. Édition revue et augmentée. https ://pure.fundp.ac.be/ws/files/252136/images1-5-79.pdf

107 Duchâteau, C., & Sass, F. (1993). Où en est-on ? Où va-t-on ? En Belgique. Actes de la troisième rencontre francophone de didactique de l'informatique, 13–28. Sion

108 Dupuis, C., Egret, M.-A., & Guin, D. (1988). Présentation et analyse d'activités de programmation en LOGO. Petit x, 18, 47–69.

109 Dupuis, C., Egret, M.-A., Guin, D., & IREM, de S. (1988). Présentation et analyse d'activités de programmation en LOGO. Petit x n, 18, 47–69.

110 Dupuis, C., & Guin, D. (1988). Découverte de la récursivité en LOGO dans une classe. In Actes du premier colloque franco-allemand de didactique des Mathématiques et de l'Informatique. Grenoble. La pensée Sauvage.

111 Élie, F., Guerry, B., Lacroix, D., Lucaud, P., Nestel, C., Picard-Limpens, C., & Viéville, T. (2010). Est-il besoin de savoir programmer pour comprendre les fondements de l'informatique ou utiliser les logiciels ? EPINet-La revue électronique de l'EPI. Consulté à l'adresse http://hal.inria.fr/inria-00546150/

112 Engeström, Y. (1997). Activity theory and individual and social transformation. In Perspectives on activity theory (p.19-38). Cambridge England : Cambridge University Press : Engeström, Y.; Miettinen, R. et L, P.R.

113 Engeström, Y. (1999). Activity theory and individual and social transformation. Perspectives on activity theory, 19–38.

114 Engeström, Y. (2001). Expansive learning at work : toward an activity theorical reconceptualisation. Journal of Education and Work, 14(1), 133-156.

115 Espiau, B., & Oudeyer, P.-Y. (2008). Robotique : de l'automate à l'humanoïde par Bernard Espiau. Un robot très curieux, entretien avec Pierre-Yves Oudeyer, propos recueillis par Dominique Chouchan. Les Cahiers de l'Inria. http://hal.inria.fr/inria-00536681/

116 Février, L., Hiron, M., & Charguéraud, A. (2011). France-IOI : l'apprentissage de l'algorithmique pout tous. In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques (p. 2013-220). Athènes, Université de Patras.

117 Fluckiger, C. (2007). L'appropriation des TIC par les collégiens dans les sphères familières et scolaires. Thèse de doctorat. École normale supérieure de Cachan.

118 Fluckiger, C. (2008). L'école à l'épreuve de la culture numérique des élèves – Cairn.info. http ://www.cairn.info/resume.php ? ID_ARTICLE=RFPED_163_0051

119 Fluckiger, C., & Bruillard, E. (2008). TIC : analyse de certains obstacles à la mobilisation des compétences issues des pratiques personnelles dans les activités scolaires. Présenté à Conférences aux 2èmes journées de l'Orme, CRDP de l'académie d'Aix-Marseille.

120 Forisek, M. (2008). The international Olympiad in Informatics Syllabus. Consulté à l'adresse http://people.ksp.sk/~misof/ioi-syllabus/ioi-syllabus.pdf

121 Fournier, J.-P., & Wirz, J. (1993). ALLOGENE : Un environnement d'apprentissage de l'algorithmique. Actes de la troisième rencontre francophone de didactique de l'informatique Sion, du 6 au 11 juillet 1992, 101–113.

122 Furber, S. F. (2012). Shut down or restart ? The way forwar for computing in UK schools. http://royalsociety.org/uploadedFiles/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

123 Galand, B., & Frenay, M. (2005). Impact de l'approche par projet sur les connaissances des étudiants. In L'approche par problèmes et par projets dans l'enseignement supérieur : impact, enjeux et défis (p. 137-161). UCL Belgique.

124 Gander, W., Petit, A., & Berry, G. (2013). Informatics education : Europe cannot afford to miss the boat. report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education. http://germany.acm.org/upload/pdf/ACMandIEreport.pdf

125 Gaudiello, I., & Zibetti, E. (2013). La robotique éducationnelle : état des lieux et perspectives. Psychologie Française, 58(1), 17-40. http://doi.org/10.1016/j.psfr.2012.09.006

126 Giannoula, E., & Baron, G.-L. (2002). Pratiques familiales de l'informatique versus pratiques scolaires : Représentations de l'informatique chez les élèves d'une classe de CM2. Sciences et Techniques éducatives, 9(3-4), 437-456.

127 Grandbastien, M. (2012). La discipline informatique et l'éducation à la maitrise de l'information. In Le « e-Dossiers de l'audiovisuel : l'Éducation aux cultures de l'information (p. 89-99). ENS Cachan.

128 Guéraud, V., & Peyrin, J.-P. (1989). Un jeu de rôles pour l'enseignement de la programmation. In Colloque francophone sur la didactique de l'informatique Université René Descartes Paris les 1, 2 et 3 septembre 1988 (p. 47–59). http://halshs.archives-ouvertes.fr/edutice-00359376/

129 Guibert, N., Guittet, L., & Girard, P. (2004). Apprendre la programmation par l'exemple : méthode et système. In Technologies de l'Information et de la Connaissance dans

l'Enseignement Supérieur et de l'Industrie (p. 345–352). Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00000716/

130 Guibert, N., Guittet, L., & Girard, P. (2005). Initiation à la Programmation « par l'exemple » : concepts, environnement, et étude d'utilité. Consulté à l'adresse http://telearn.archives-ouvertes.fr/hal-00005762/

131 Guzdial, M. (2004). Programming environments for novices. Computer science education research, 2004, 127–154.

132 Haspekian, M. (2005). Intégration d'outils informatiques dans l'enseignement des mathématiques, Étude du cas des tableurs. Thèse de doctorat. Université Paris-Diderot.

133 Haspekian, M., & Nijimbere, C. (2012). Les enseignants face à l'entrée de l'algorithmique dans l'enseignement des mathématiques au lycée scientifique en France. In M., Gandit & B., Grugeon-Allys. CORFEM : Actes des 17e et 18e colloques (p. 265-285). Université et IUFM de Franche-Comté.

134 Hebenstreit, J. (1973). Apport spécifique de l'informatique et de l'ordinateur à l'enseignement secondaire. http://hal.archives-ouvertes.fr/docs/00/28/40/90/HTML/

135 Herviou, C., & Taurisson, A. (2012). La pédagogie de l'activité. Conférence du CASNAV. Consulté à l'adresse http://www.ac-paris.fr/portail/jcms/p1_479027/de-la-pedagogie-de-l-activite-a-la-perspective-actionnelle

136 Hoc, J.-M. (1977). Méthode d'analyse psychologique d'un travail de programmation. Le Travail humain, 15–27.

137 Hoc, J.-M. (1979). Le problème de la planification dans la construction d'un programme informatique. Le Travail humain, 245–260.

138 Hoc, J. M. (1981). Les habiletés mentales dans le travail. La planification dans la résolution de problème : l'apprentissage de la programmation informatique. Le travail humain, 44(2), 261-267.

139 Hoc, J.-M. (1988). L'aide aux activités de planification dans la conception des programmes. *Le Travail Humain*, 321–333.

140 Hsu, S.-H., Chou, C.-Y., Chen, F.-C., Wang, Y.-K., & Chan, T.-W. (2007). An investigation of the differences between robot and virtual learning companions' influences on students' engagement. In Digital Game and Intelligent Toy Enhanced Learning, 2007. DIGITY '07. The First IEEE International Workshop on (p. 41–48). Consulté à l'adresse http://ieeexplore.ieee.org/xpls/abs_all.jsp ? Arnumber=4148830

141 Janiszek, D., Boulc'h, L., Pellier, D., Mauclair, J., & Baron, G. L. (2011). De l'usage de Nao

dans l'apprentissage de l'informatique. In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'information et de la communication en milieu éducatif (p. 231-239). Athènes, Université de Patras.

142 Janiszek, D., Pellier, D., Mauclair, J., Boulc'H, L., Baron, G.-L., & Parchemal, Y. (2011). Utilisation de la robotique pédagogique pour enseigner l'intelligence artificielle : une expérience d'approche par projet auprès d'étudiants en informatique. Consulté à l'adresse http://sticef.univ-lemans.fr/num/vol2011/07r-janiszek/sticef_2011_janiszek_07rp.html

143 Jonnaert, P. (2001). La thèse socioconstructiviste dans les nouveaux programmes d'études au Québec : un trompe l'œil épistémologique ? Consulté à l'adresse http://www.unige.ch/fapse/life/textes/Jonnaert_A2000_01.html

144 Kaasboll, J. (2002). Learning Programming. University of Oslo. University of Oslo.

145 Karsenti, T., & Collin, S. (2010). Quelle place pour les TIC en formation initiale d'enseignants de français ? Le cas de l'Afrique. Revue internationale des technologies en pédagogie universitaire/International Journal of Technologies in Higher Education, 7(3), 32–47.

146 Karsenti, T., & Tchameni Ngamo, S. (2007). Qualité de l'éducation en Afrique : le rôle potentiel des TIC. International review of education, 53(5-6), 665–686.

147 Karsenti, T., & Tchameni Ngamo, S. (2009). Qu'est-ce que l'intégration pédagogique des TIC ?, 57-75.

148 Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. Procedia-Social and Behavioral Sciences, 47, 1991–1999.

149 Khaneboubi, M. (2009). Description de quelques caractéristiques communes aux opérations de dotations massives en ordinateurs portables en France. Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF), 16. Consulté à l'adresse http://telearn.archives-ouvertes.fr/hal-00696410/

150 Komis, V., & Misirli, A. (2011). Robotique pédagogique et concepts préliminaire de la programmmation à l'école maternelle : une étude de cas basée sur le jouet programmable Bee-Bot. In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'information et de la communication en milieu éducatif (p. 271-281). Athènes, Université de Patras.

151 Komis, V., & Misirli, A. (2012). L'usage des jouets programmables à l'école maternelle: concevoir et utiliser des scenarios éducatifs de robotiques pédagogiques. *Revue Skholê*, *17*,

143-154.

152 Komis, V., & Misirli, A. (2015). Apprendre à programmer à l'école maternelle à l'aide des jouets programmables. In G.-L. Baron, É. Bruillard, & B. Drot-Delange (dir.), *Informatique en éducation : perspectives curriculaires et didactiques.* Presses universitaires Blaise Pascal. Clermont-Ferrand, France.

153 Kuutti, K. (1996). Activity theory as a potential framework for human – computer interaction research. In Context and Consciousness : Activity Theory and Human Computer Interaction. (p.17-44). Cambridge, MA : The MIT Press.

154 Kynigos. (2008). Black and white perspectives to distributed control and constructionism in learning with robotics. Workshop Proceedings of SIMPAR.

155 Laborde, C., Mejias, B., & Balacheff, N. (1985). Problematique et genese du concept d'iteration : une approche experimentale.

156 Laborde, C., & Vergnaud, G. (1994). L'apprentissage et enseignement des mathématiques. In G. Vergnaud (dir.), L'apprentissage et enseignement des mathématiques (p. 63-98). Paris : hachette.

157 Lagrange, J.-B. (1992). Les objets et les types dans un enseignement de la programmation s' adressant à des débutants. Actes de la troisième rencontre francophone de didactique de l'informatique, 115–120.

158 Laisne, M. (2004). Le B2I en collège. Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF), 11. Consulté à l'adresse http://hal.archives-ouvertes.fr/hal-00696431.

159 Lauzier, I., & Nonnon, P. (2007). Une expérience d'approche par projet pour favoriser l'intégration des apprentissages (p. 14-21). Présenté au 9e Colloque Francophone de Robotique Pédagogique, La Ferté-Bernard, France.

160 Lebeaume, J., Martinand, J. -L., & Reuter, Y. (2007). Contenus, didactiques, disciplines, formation. Entretien. Recherche et formation, (55), 107–117.

161 Lebeaume, J., & Meignié, F. (2003). Technologie de l'information au collège. In Premières journées francophones de didactique des progiciels. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00145907/

162 Leong, E. -B. (2012). Les outils de la pensée. PUQ.

163 Leontiev, A. -N. (1975). Activité, conscience, personnalité. Trad. Française, Moscou, Editions du progrès, 1984.

164 Leontiev, A. -N. (1978). Activity, Consciousness, and Personalty, Hillsdale : Prentice-Hall.

165 Leroux, P. (2005a). 20 ans de Robotique Pédagogique à l'Université de du Maine, le Mans (p. 7-18). Présenté à 8e colloque francophone de Robotique Pédagogique, La Ferté-Bernard.

166 Leroux, P. (2005b). Réalisation de micro-robots au collège : mise au point d'une démarche pédagogique et d'un environnement informatique support des activités. Revue ASTER, (41), 49-78.

167 Leroux, P. (1995). Conception et réalisation d'un système coopératif d'apprentissage – Étude d'une double coopération : maître/ordinateur et ordinateur/groupe d'apprenants. Thèse de doctorat en Informatique. Université Paris 6, Paris.

168 Leroux, P. (1996). Intégration du contrôle d'objets réels dans un hypermédia. Un exemple d'implantation dans le système ROBOTEACH. In Troisième colloque Hypermédias et Apprentissages (p. 237–244). http://edutice.archives-ouvertes.fr/edutice-00000526/

169 Leroux, P., Monflier, J.-L., Guyon, S., Jambu, M., Després, C., & George, S. (2005). Démarche de projet, micro-robots modulaires et logiciel d'apprentissage dans le cadre de l'enseignement des systèmes automatisés au collège. In P. Vérillon, J. Ginestié, J. Lebeaume, et P. Leroux (dir.), Produire en technologie à l'école et au collège (p. 119-168). INRP.

170 Leroux, P., & Vivet, M. (2000). Micro-Robots Based Learning Environments for Continued Education in Small and Medium Enterprises (SMEs). Journal of Interactive Learning Research, 11(3), 435-463.

171 Leroux, P., Vivet, M., & Brézillon, P. (1996). Cooperation between humans and a pedagogical assistant in a learning environment. In Proceedings of European Conference AI in Education (EuroAIED), Lisbon, Portugal (p. 379–385). Consulté à l'adresse http://www-sysdef.lip6.fr/~brezil/Pages2/Publications/COOP-96.pdf

172 Lévy, P. (1987). La machine univers. Création, cognition et cultures informatique. *La découverte.*

173 Lewis, R. (1998). Apprendre conjointement : une analyse, quelques expériences et un cadre de travail. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00000486

174 Magakian, J. -L. (2009). Une perspective constructionniste des conversations stratégiques dans le processus d'idéation du dirigeant. Thèse de doctorat. Université Jean Louis Lyon 3, Lyon.

175 Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch : a sneak preview [education]. In Creating, Connecting and Collaborating through Computing, 2004. Proceedings. Second International Conference on (p. 104–109). IEEE. Consulté à l'adresse http://ieeexplore.ieee.org/xpls/abs_all.jsp ? Arnumber=1314376

176 Maloney, J. -H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice : urban youth learning programming with scratch. In ACM SIGCSE Bulletin. Vol. 40, (p. 367–371). ACM. Consulté à l'adresse: http://dl.acm.org/citation.cfm ? Id=1352260

177 Marcel, J.-F. (2002). Approche ethnographique des pratiques enseignantes durant les temps interstitiels. Revue Spirale, 30, 103–120.

178 Marchand, D. (1991). La robotique pédagogique ! Ça existe. Bulletin de l'EPI, (65), 119–124.

179 Mardirossian, C. (1989). Un autre problème de compatibilité : enseigner l'informatique et les lettres. Bulletin de l'EPI, (54), 107-109.

180 Martinand, J.-L. (2000). Production, circulation et reproblématisation des savoirs. Colloque Les pratiques dans l'enseignement supérieur. Toulouse, France. Consulté 16 janvier 2013, à l'adresse http://www.fcomte.iufm.fr/ejrieps/ejournal2/musard.htm

181 Mejias, B. (1985). Difficultés conceptuelles dans l'écriture d'algorithmes itératifs chez des élèves de collège. Thèse de doctorat. Université Scientifique et Médicale de Grenoble. Grenoble I.

182 Mendelsohn, P. (1985). L'enfant et les activités de programmation. Grand N, 35, 47–60.

183 Mendelsohn, P. (1986a). L'analyse psychologique des activités de programmation chez l'enfant de CMI et CM2. Enfance, 38(2-3), 213–221.

184 Mendelsohn, P. (1986b). L'enfant et les activités de programmation. Consulté à l'adresse http://www-irem.ujf-grenoble.fr/revues/revue_n/fic/35/35n5.pdf

185 Mercouroff, W. (1983). Dix ans d'informatique dans l'enseignement secondaire : 1970-1980. Revue française de pédagogie, 63(1), 90-91.

186 Miller, G. E. (1990). The assessment of clinical skills/competence/performance. Academic medicine, 65(9), S63–7.

187 Modeste, S. (2012a). Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ? Thèse de doctorat. Université de Grenoble.

188 Modeste, S. (2012b). Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ?. Université de Grenoble. Consulté à l'adresse https ://tel.archives-ouvertes.fr/tel-00783294/

189 Modeste, S. (2012c). La pensée algorithmique : apports d'un point de vue extérieur aux mathématiques. http://www-fourier.ujf-grenoble.fr/~modeste/docs/Modeste-EMF.pdf

190 Montagnier, P., & Van Welsum, D. (2006). ICTs and Gender–Evidence from OECD and

Non-OECD Countries. In UNCTAD Expert Meeting on Using ICT to Achieve Growth and Development (p. 4–5). http://unctad.org/sections/wcmu/docs/c3em29p025_en.pdf

191 Morrissette, J. (2011). Ouvrir la boîte noire de l'entretien de groupe. Recherches qualitatives, 29(3), 7–32.

192 Muratet, M. (2010). Conception, réalisation et évaluation d'un jeu sérieux de stratégie temps réel pour l'apprentissage des fondamentaux de la programmation. Université Paul Sabatier-Toulouse III. Consulté à l'adresse http://tel.archives-ouvertes.fr/tel-00554287/

193 Muratet, M., Delozanne, E., Torguet, P., & Viallet, F. (2012). Addressing teachers' concerns about the Prog&Play serious game with context adaptation. International Journal of Learning Technology, 7(4), 419–433.

194 Muratet, M., Torguet, P., Jessel, J.-P., & Viallet, F. (2008). Apprentissage de la programmation à l'aide d'un jeu sérieux. Ludovia, page (en lign e), http ://www. ludovia. org, a oût. ftp://ieut1.irit.fr/pub/pub/IRIT/VORTEX/Muratet_Ludovia2008.pdf

195 Muratet, M., Torguet, P., Viallet, F., & Jessel, J.-P. (2011). Experimental feedback on Prog&Play : a serious game for programming practice. In Computer Graphics Forum (Vol. 30, p. 61–73). Wiley Online Library. http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2010.01829.x/full

196 Muratet, M., Viallet, F., Torguet, P., & Jessel, J.-P. (2011). Utilisation de jeux sérieux pour enseigner les fondamentaux de la programmation. In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques. Démonstrations et posters du quatrième colloque international DIDAPRO 4-Dida&Stic, 24-26 octobre 2011, Université de Patras. (p. 57–60). http://halshs.archives-ouvertes.fr/edutice-00690110/

197 Neboit, M., & Poyet, C. (1991). La communication homme-machine en robotique : analyse comparée de différentes interfaces de programmation. Technologies de l'Informatique et Société, 3(2-3), 231–249.

198 Nguyen, C. T. (2005). Etude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice. Thèse de doctorat. Université Joseph-Fourier-Grenoble I. http://tel.archives-ouvertes.fr/tel-00011500/

199 Nguyen, C. T., & Bessot, A. (2003). La prise en compte des notions de boucle et de variable informatique dans l'enseignement des mathématiques au Lycée. Petit x, (62), 7–32.

200 Nicolle, A. (2002). L'informatique en éducation entre science et technique. In G.-L. Baron,

É. Bruillard (éds.), Les technologies en éducation. Perspectives de recherche et questions vives. Actes du symposium international francophone, (p. 177-190). Paris.

201 Nijimbere, C. (2012). Informatique et enseignement au Burundi, quelles réalités ? – [Adjectif]. Consulté 19 septembre 2014, à l'adresse http://www.adjectif.net/spip/spip.php ? article105&lang=fr

202 Nijimbere, C. (2013). Approche instrumentale et didactiques : apports de Pierre Rabardel. Consulté à l'adresse http://www.adjectif.net/spip/spip.php ? Breve132

203 Nijimbere, C. (2014). Apprendre l'informatique par la programmation des robots : cas de Nao. http://sticef.univ-lemans.fr/num/vol2014/09-nijimbere/sticef_2014_nijimbere_09.htm

204 Nijimbere, C., Boulc'H, L., & Baron, G.-L. (2015). Apprendre l'informatique par la programmation des robots. Cas de Lego Mindstorms NXT. In G.-L. Baron, É. Bruillard & B. Drot-Delange (dir.), Informatique en éducation : perspectives curriculaires et didactiques (p. 266-277). Clermont-Ferrand, France : Presses Universitaires Blaise-Pascal.

205 Nijimbere, C., Mbwayiba, C., & Ndayishimiye, N. (2013). La téléphonie mobile au Burundi. Consulté à l'adresse http://www.adjectif.net/spip/spip.php ? Article267

206 Nijimbere, C., Voulgre, E., & Baron, G.-L. (2014). Entre reproduction et adaptation des expériences du nord : le cas du Burundi et du Cameroun. Table ronde plénière In colloque FORSE de l'université de Rouen, le 09 octobre 2014. Rouen.

207 Nivat, M. (1985). Sur l'enseignement de l'informatique lié à des applications. Options informatiques, 2, 51-54.

208 Nivat, M. (2007). Lettre au Président de la République Française. Consulté 9 mars 2015, à l'adresse http://www.epi.asso.fr/revue/docu/d0802a.htm

209 Nivat, M. (2009). Vaches et informatique. In G.-L, Baron, É, Bruillard & V., Pochon (dir.), Informatique et progiciels en éducation et en formation : continuités et perspectives (p. 28-38). Lyon. France

210 Nonnon, P. (2002). Robotique pédagogique et formation de base en science et technologie. Aster, 2002, 34« Sciences, techniques et pratiques professionnelles ». http://documents.irevues.inist.fr/handle/2042/8787

211 Ntibashirakandi, L. (2011). L'école primaire passe de six à neuf ans au Burundi. Une fausse solution à un vrai problème. http://burundi.news.free.fr/actualites/educationprimaire.pdf

212 O'Kelly, J., & Gibson, J. P. (2006). RoboCode & problem-based learning : a non-prescriptive approach to teaching programming. ACM SIGCSE Bulletin, 38(3), 217–221.

213 Owen, C. (2008). Analyser le travail conjoint entre différents systèmes d'activités. Revue

Activités, 5(2), 70-89.

214 Pair, C. (1987). Informatique et enseignement= hier, aujourd'hui et demain. Bulletin de l'EPI (Enseignement Public et Informatique), (47), 85–97.

215 Pair, C. (1988a). Je ne sais (toujours) pas enseigner la programmation – Informatique 2, 5-14.

216 Pair, C. (1988b). Programmation, langages et méthodes de programmation. *Le Travail Humain*, *51*(4), 297-307.

217 Pair, C. (1989). L'apprentissage de la programmation. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00359385

218 Paliokas, I., Arapidis, C., & Mpimpitsos, M. (2011). Playlogo 3d : A 3d interactive video game for early programming education : Let logo be a game. In Games and Virtual Worlds for Serious Applications (VS-GAMES), 2011 Third International Conference on (p. 24–31). IEEE. Consulté à l'adresse http://ieeexplore.ieee.org/xpls/abs_all.jsp ? Arnumber=5962112

219 Paoletti, F. (1993). Épistémologie et technologie de l'informatique. Bulletin de l'EPI, (71), 175-182.

220 Papert, S. (1980). Jaillissement de l'esprit. Ordinateurs et apprentissage. New York, NY, USA : Flammarion.

221 Papert, S. (1981). Mindstorms. New York, USA : Basic Books.

222 Papert, S. (1994). L'enfant et la machine à connaître. Repenser l'école à l'ère de l'ordinateur. Traduit par Étienne Cazin. Dunod. Paris.

223 Pap-Szigeti, R., Pasztor, A., & Lakatos Toeroek, E. (2010). Effects of Using Model Robots in the Education of Programming. Informatics in Education-An International Journal, 9 (1), 133–140.

224 Parks, S. (2000). Same task, different activities : Issues of investment, identity and use of strategy. TESL Canada Journal, 17(2), 64-88.

225 PASEC/CONFEMEN. (2010). Enseignement primaire : Quels défis pour une éducation de qualité en 2015 ? Évaluation diagnostique Burundi (2008-2009). Dakar.

226 Pélisset, É. (1985). Pour une histoire de l'informatique dans l'enseignement français. Consulté 12 décembre 2013, à l'adresse http://www.epi.asso.fr/revue/histo/h85ep.htm

227 Perrenoud, P. (1995). Enseigner des savoirs ou développer des compétences : l'école entre deux paradigmes. Savoirs et savoir-faire, Paris, Nathan, 73–88.

228 Perrenoud, P. (1998). La transposition didactique à partir de pratiques : des savoirs aux compétences. Revue des sciences de l'éducation, 24(3), p.487-514.

229 Perrenoud, P. (1999). Apprendre à l'école à travers des projets : Pourquoi ? Comment ? Retrieved February, 12, 2003.

230 Pléty, R. (1996). L'apprentissage coopérant. Presses Universitaires de Lyon. Lyon.

231 Poisseroux, J., Lassaux, E., & Vandeput, É. (2009). TacTIC pour une intégration réussie des technologies en Haute École. In G.-L. Baron, É. Bruillard & V. Pochon (dir.), Informatique et progiciels en éducation et en formation : continuités et perspectives (p. 108-125). Lyon. France.

232 Prensky, M. (2001a). Digital natives, Digital immigrants Part 1. On the Horizon, 9(5), 1-6.

233 Prensky, M. (2001b). Digital natives, digital immigrants. Do They Really Think Differently? On the Horizon, NCB, Universty press. *On the Horizon, NCB, Univerty press*, *9*(6).

234 Presseau, A., & Frenay, M. (2004). Transfert des apprentissages. Comprendre pour mieux intervenir. Presses de l'université Laval.

235 Rabardel, P. (1995). Les hommes et les technologies, une approche cognitives des instruments contemporains. Armand Colin, Paris. Consulté à l'adresse ergoserv.psy.univ-paris8.fr/Site/…/ART372105503765426783. PDF

236 Rapport de l'académie des sciences. (2013). L'enseignement de l'informatique en France. Il est temps de ne plus attendre.

237 Rapport de l'Éducation continentale, U. (2014). Projet de Perspectives de l'Education en Afrique.http://www.adeanet.org/portalv2/sites/default/files/au_outlook_report_amu_french_2014_web.pdf

238 Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K.,… Silverman, B. (2009). Scratch : programming for all. Communications of the ACM, 52(11), 60–67.

239 Resnick, M., Martin, F., Sargent, R., & Silverman, B. (1996). Programmable bricks : Toys to think with. IBM Systems journal, 35(3.4), 443–452.

240 Reuter, Y., Cohen-Azria, C., Daunay, B., Delcambre, I., & Lahanier-Reuter, D. (2007). Dictionnaire des concepts fondamentaux des didactiques. De boeck Université. Bruxelles.

241 Rivière, S., Zamierie, K., & Amerirein-Soltner, B. (2011). AlgoML : un format XML pour l'apprentissage de l'algorithmique et la programmation. In G.-L. Baron, É. Bruillard & V. Komis (dir.), Sciences et technologies de l'infortmation et de la communication (STIC) en milieu éducatif : analyse de pratique et enjeux didactiques (p. 22-30). Athènes. Université de Patras, Grèce.

242 Robert, A., & Rogalski, J. (2002). Le système complexe et cohérent des pratiques des

enseignants de mathématiques : une double approche. Canadian Journal of Math, Science & Technology Education, 2(4), 505–528.

243 Robitaille, M. (2007). Quand un dispositif de développement professionnel sur l'apprentissage coopératif devient un lieu de régulations entre pairs. In Régulation des apprentissages en situation scolaire et en formation (p. 171-189). De boeck.

244 Roby-Brami, A., & Laffont, I. (2002). Gestes et technologie : la compensation des incapacités motrices. In B. Bril & V. Roux (dir.), Le geste technique : Réflexions méthodologiques et anthropologiques (p. 95-112). Ramonville Saint-Agne. Ères.

245 Rogalski, J. (1988). Enseignement de méthodes de programmation dans l'initiation à l'informatique. In Colloque francophone sur la didactique de l'informatique, Université René Descartes Paris les 1, 2 et 3 septembre 1988 (p. 61–72). Consulté à l'adresse http://hal.archives-ouvertes.fr/edutice-00359382/

246 Rogalski, J. (1992). Utiliser l'informatique pour enseigner y a-t-il matière à savoir, besoin de formation en informatique ? In G.-L. Baron & J. Baudé (dir..), L'intégration de l'informatique dans l'enseignement et la formation des enseignants (p. 238-243). Châtenay-Malabry, France : EPI – INRP. https ://edutice.archives-ouvertes.fr/edutice-00278533

247 Rogalski, J. (2012). Théorie de l'activité et didactique, pour analyse conjointe des activités de l'enseignant et de l'élève. Jornal Internacional de Estudos em Educação Matemática, 5(1). http://periodicos.uniban.br/index.php ? journal=JIEEM&page=article&op=view&path%5B%5D=285

248 Rogalski, J. (2015). Psychologie de la programmation, didactique de l'informatique. Déjà une histoire… In G.-L. Baron, É. Bruillard, & B. Drot-Delange (dir.), *Informatique en éducation : perspectives curriculaires et didactiques* (p. 280-305). Clermont-Ferrand, France: Presses Universitaires Blaise-Pascal.

249 Rogalski, J., & 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.

250 Rogalski, J., Samurçay, R., & Hoc, J.-M. (1988). L'apprentissage des méthodes de programmation comme méthodes de résolution de problème. Le travail humain, 309–320.

251 Ropé, F., & Tanguy, L. (1994). Savoirs et compétences : de l'usage de ces notions dans l'école et l'entreprise. École polytechnique : L'Harmattan.

252 Rouchier, A. (1990). Objets de savoir de nature informatique dans l'enseignement secondaire. Consulté à l'adresse http://documents.irevues.inist.fr/handle/2042/9118

253 Rouchier, A., & Samurcay, R. (1985). De « faire à » « faire faire » : planification d'actions dans la situation de programmation. Enfance, 38(2), 241-254. http://doi.org/10.3406/enfan.1985.2883

254 Rouquette, C. (1999). L'informatique : une technique assimilée par les jeunes générations, INSEE Première (643), 4.

255 Samurçay, R. (1985). Signification et fonctionnement du concept de variable informatique chez des élèves débutants. Educational Studies in Mathematics, 16(2), 143–161.

256 Saussez, F., & Yvon, F. (2010). Analyser l'activité enseignante. Des outils méthodologiques et théoriques pour l'intervention et la formation. Québec : Presses de l'Université Laval.

257 Schwartz, O. (1989). Le monde privé des ouvriers. Paris : PUF.

258 SIF, (2013a). Communiqué de la Société Informatique de France conernant une éventuelle mention mathématiques-Informatique. Consulté à l'adresse http://www.societe-informatique-de-france.fr/actualite/2013/Communique_SIF_Masters_Decembre_2013.pdf

259 SIF, (2013b). Grèce : l'éducation informatique ne doit pas être victime des coupes budgétaires. Consulté à l'adresse http://www.societe-informatique-de-france.fr/actualite/2013/Communiqu%C3 %A9_sur_la_Gr%C3 %A8ce_2013-08-28.pdf

260 Simon, J.-C. (1980). L'Éducation et l'Informatisation de la société, Rapport au Président de la République. La Documentation française. Consulté à l'adresse http://halshs.archives-ouvertes.fr/docs/00/27/77/97/HTML/h80simon2.htm

261 Tardif, J. (1997). Pour un enseignement stratégique : l'apport de la psychologie cognitive Les Éd. Logiques.

262 Taurisson, A. (2005). La pédagogie de l'activité, un nouveau paradigme ?. Thèse de doctorat, Université Lyon II, Lyon.

263 Tempel, M. (2013). Blocks Programming. CSTA Voice, 9(1). Consulté à l'adresse http://el.media.mit.edu/logo-foundation/pubs/papers/blocks_programming.pdf

264 Tiberghien, A., & Malkoun, L. (2007). Différenciation des pratiques d'enseignement et acquisitions des élèves du point de vue du savoir. Éducation et didactique, 1(1), 29-54. http://doi.org/10.4000/educationdidactique.69

265 Tonye, E. (2008). Formation continue et à distance (FOCAD) en Afrique centrale : étude de la faisabilité contextualisée, Rapport final. Yaoundé.

266 Tort, F. (2009). L'enseignement du tableur en France. In G.-L. Baron, É. Bruillard & V. Pochon (dir.), Informatique et progiciels en éducation et en formation : continuités et perspectives (p. 200-217). Lyon. France.

267 Tort, F. (2011). Le concours Castor : Un outil de promotion de l'enseignement d'informatique. In G.-L. Baron, É. Bruillard & et V. Komis (dir.), In Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques. (p. 271-284). Athènes, Université de Patras.

268 Tort, F. (2013). Enseignement d'informatique au second degré. Approche comparative entre pays. Nancy.

269 Tort, F., & Dagiène, V. (2012). Concours Castor : découvrir l'informatique autrement. In Le « e-Dossiers de l'audiovisuel : l'Éducation aux cultures de l'information (p. 101-112).

270 Tort, F., Kummer-Hannoun, P., & Beauné, A. (2013). Engagement et motivations des enseignants du secondaire pour la passation d'un concours d'informatique. In Sciences et technologies de l'information et de la communication en milieu éducatif : Objets et méthodes d'enseignement et d'apprentissage, de la maternelle à l'université. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00875655

271 Vagost, D. (2010). L'algorithmique en seconde : un exemple de mise en ouevre dans la classe. Bulletin de l'APMEP, (486).

272 Vergnaud, G. (1990). La théorie des champs conceptuels. 10. La pensée Sauvage, Grenoble.

273 Verhoeff, T., Horváth, G., Diks, K., & Cormack, G. (2006). A proposal for an IOI Syllabus. Teaching Mathematics and Computer Science, 4(1), 193–216.

274 Viallet, F., & Venturini, P. (2010). Didactique comparée et enseignement de l'informatique. In Actes du congrès de l'Actualité de la recherche et éducation et en formation (AREF). Genève.

275 Villemonteix, F., & Baron, G.-L. (2011). L'informatique à l'école : le modèle du « pair-expert » en mutation ? Questions Vives. Recherches en éducation, 6(16). Consulté à l'adresse http://www.doaj.org/doaj ? func=fulltext&aId=1061279

276 Vivet, M. (2000). Des robots pour apprendre. Revue des Sciences et Techniques Éducatives, 7(1), 17-60.

277 Voulgre, E., & Netto, S. (2014). Formation à distance au Burundi : Quelles représentations pour quels usages ? Points de vue de superviseurs pédagogiques du Primaire. http://eda.recherche.parisdescartes.fr/wp-content/uploads/sites/6/2014/06/Resume%CC%81-Voulgre-Netto-Jocair-2014.pdf

278 Vygotski, L. (1934). Pensée et langage. 3e édition. Paris : La Dispute.

279 Wittorski, R. (1998). De la fabrication des compétences. Education permanente, 135. Consulté à l'adresse http://halshs.archives-ouvertes.fr/hal-00172696/

280 Wozniak, F., Bosch, M., & Artaud, M. (2008). La théorie anthropologique du didactique. Consulté 9 décembre 2012, à l'adresse http://www.ardm.eu/contenu/yves-chevallard

Index des auteurs cités

Index des tableaux

Index des illustrations