Apprendre l'informatique par la programmation des robots : cas de Nao

▶ **Claver NIJIMBERE** (EDA, Université Paris Descartes)

■ **RÉSUMÉ** • Cet article vise la construction des savoirs informatiques chez les étudiants de licence 3 dans un contexte de projet de programmation de Nao, un robot humanoïde. Il s'inscrit dans le cadre d'une thèse de doctorat en cours à l'université Paris Descartes. Des observations ethnographiques des pratiques en contexte de projets et des entretiens périodiques en groupes, ont été effectués puis complétés par des entretiens en groupe conduits auprès des enseignants-encadrants de ces projets. Toutes les données ont été analysées et les résultats discutés à la lumière de la littérature disponible. Les résultats montrent beaucoup de connaissances informatiques pluridisciplinaires construites et/ou d'autres utilisées. Des connaissances mathématiques avancées, pourtant indispensables pour la performance de Nao, ont été limitées par une absence de modélisation, laquelle modélisation est adaptée à la décomposition des tâches en actions élémentaires.

■ **MOTS CLÉS** •Apprentissage, informatique, programmation, robot, Nao

■ ***ABSTRACT*** • *This article aims at studying the acquisition of computer knowledge by bachelor students in a context learning to program with Nao, an humanoid robot. It is a part of an ongoing doctoral dissertation at the Paris Descartes University. Ethnographic observations were performed in the context of realizing projects. We have led periodical group discussions with students and with instructors. Results show the mobilization of much multidisciplinary computer knowledge. Advanced mathematical knowledge, mandatory for Nao's performance, was limited by the lack of a model which would be required in order to breakdown the tasks into elementary actions.*

■ ***KEYWORDS*** • *Learning, computer science, programming, robot, Nao*

1. Contexte, éléments de problématique et hypothèses de recherche

Plus qu'hier, l'informatique connaît aujourd'hui de nombreuses applications dans tous les domaines socio-professionnels. Chez les jeunes, si les environnements informatiques sont omniprésents, ils en comprennent moins leur fonctionnement (Bers, 2008). L'informatique, devenue de plus en plus incontournable dans la vie quotidienne, une nécessité d'en offrir une culture générale à tous les citoyens est à la mode dans beaucoup de pays (Dowek *et al.*, 2011). Néanmoins, les systèmes éducatifs des divers pays ont eu de la peine à prendre la mesure et à définir des contours pour cette discipline-carrefour, entre technologie et science autonome, en évolution constante et dont les contours ont beaucoup variés au cours du temps (Baron, 2012). Au sein d'un même pays, diverses approches sont souvent mises en œuvre.

En France, après de longues années d'absence, l'informatique a revu le jour en tant que discipline de formation générale dans les lycées scientifiques à la rentrée 2012, après des ouvertures et interruptions périodiques il y a une trentaine d'années (Archambault, 2011 ; Rapport de l'académie des sciences, 2013). Elle est en voie de généralisation à toutes les terminales à la rentrée 2014. À l'université, la situation est différente du secondaire. L'informatique est enseignée depuis les années 60 (Baron, 1987). Plusieurs approches sont mises en œuvre : approches par projets, program-mation des technologies innovantes, etc.

Cet article s'intéresse à l'apprentissage de l'informatique par la programmation d'un robot Nao par les étudiants de licence 3 en informatique à l'université Paris Descartes.

1.1. 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 pouvions penser que l'informatique ferait exception, il n'en est malheureusement rien. Janiszek, Bouilc'h, Pellier, Mauclair et Baron (2011a) indiquent que le manque de motivation de l'informatique chez les jeunes a poussé à imaginer et élaborer des parcours pédagogiques plus attractifs et mieux adaptés aux étudiants pour motiver l'apprentissage de l'informa-

tique. Ce contexte d'apprentissage semble entrer dans la ligne suggérée par 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).

1.2. 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 quatre-vingt. Fortement attaché à la formation professionnelle, Vivet indique dans un document rédigé dans les années 1980 et publié en 2000 (Vivet, 2000), 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 (Baron & Denis, 1993 ; Duchâteau, 1993 ; Marchand, 1991) : 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.

Leroux est l'un des pionniers de la RP. Ses travaux de recherche s'intéressent à l'alphabétisation 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. Si son but n'est pas de former des programmeurs, une initiation à l'informatique et surtout à la programmation est faite au public peu ou pas scolarisé. 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.

Encouragée dans plusieurs domaines dont le milieu éducatif (Bonnel, 2010), l'utilisation de la robotique pour les apprentissages scolaires a pris de l'ampleur. Néanmoins, sa cherté, soulignée depuis sa naissance (Duchâteau, 1993), fait qu'elle reste jusqu'aujourd'hui, le monopole d'établissements privilégiés (Janiszek *et al.*, 2011a). Pour Duchâteau, 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, il 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 »[1]. Les robots peuvent être utilisés dans les apprentissages pour tout public et à tous les niveaux d'enseignement, du plus jeune âge jusqu'à l'université : à l'école maternelle (Komis & Misirli, 2011) ; au secondaire (collège) et pour les professionnels faiblement qualifiés (Leroux, 1995, 2005 ; Leroux & Vivet, 2000) et à l'université (Janiszek *et al.*, 2011b)

1.3. Robotique et potentialités éducatives

Depuis les années quatre-vingt-dix, des recherches se sont intéressées à des approches comparatives entre ordinateur et robot (ou l'informatique et la robotique). 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 métaréflexion 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 à 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 aussi été 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, Chou, Chen, Wang et Chang (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) : comme un ensemble mécanique et électronique, un robot 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 maitrise de la programmation chez les apprenants par rapport à l'utilisation du seul ordinateur (Pap-Szigeti *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 suivi, 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 (Duchâteau, 1993).

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 dernier 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. Le concept d'information est central en informatique (Dowek *et al.*, 2011). S'il reste le plus souvent une abstraction en programmation classique, Duchâteau trouve que les activités de la robotique pédagogique peuvent lui donner corps : la robotique pédagogique apporte une extension du champ accessible à la modélisation et un éclairage supplémentaire sur l'information et son traitement.

1.4. Initier à la programmation : du langage Logo aux langages à blocs

Mendelsohn (1985) 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 80 (Papert, 1994), l'approche Logo a été caractérisée comme faisant partie de la robotique pédagogique. 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, Martin, Sargent et Silverman (1996), la prolifération des micro-ordinateurs dans les années 1970 a fait que la communauté Logo a orienté son 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'alphabétisation informatique étaient liées au manque d'autonomie (Resnick, Martin, Sargent & Silverman, 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. Ceux qui ont été conçus après Logo 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 programmes (édition, exécution, etc.), nous avons limité les problèmes que rencontrent les stagiaires dans nos formations d'alphabétisation technologique et informatique au cours de l'utili-

sation de l'environnement de programmation Logo : erreurs syntaxiques et des difficultés liées à l'édition et à l'exécution de programmes. »

Guzdial (2004) donne toute une liste de langages successeurs de Logo : Logowriter, Starlogo, Moose crossing, Boxer Smalltalk, etc. Chaque environnement de programmation a sa spécificité. L'environnement Starlogo, par exemple, en plus d'être un langage de programmation, est un environnement de conception des simulations, de construction et de test (Klopfer & Begel, 2003).

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 et conçus pour l'alphabétisation 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 quatre-vingt-dix par the MIT Media Lab (Tempel, 2013). La littérature donne une liste non exhaustive de tels langages : Turtle Art, PICO Blocks, MIT App Invetor, Alice, Scratch (Tempel, 2013), Program your robot (Kazimoglu *et al.*, 2012), PlayLogo (Paliokas *et al.*, 2011). Ils souvent orientés sous forme de jeux, notamment des jeux sérieux, à l'instar de Robocode (O'Kelly & Gibson, 2006), Prog&Play (Muratet *et al.*, 2012), Colobot (Muratet, 2010), 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 une situation de programmation de ces 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. Tempel, à partir d'un exemple de Scratch, montre comment fonctionne le travail de programmation de langages à base de blocs en simulant le jeu entre un chat et une souris, le premier poursuivant ce dernier dans un mouvement aléatoire sur l'écran. La structure de tels langages à blocs est pédagogiquement importante pour le débutant. Elle offre une meilleure représentation visuelle du programme,

lequel programme devient plus compréhensible que le code construit dans un autre langage (Tempel, 2013). 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 de ce programme.

De plus, ces types de langages à blocs permettent un apprentissage en séquence de la programmation. Baron et Voulgre (2013) ont expérimenté le langage Scratch auprès des étudiants de Master en science de l'éducation, novices en informatique à l'université Paris Descartes. Ils indiquent que, dans ce langage, la structure à base d'assemblage de blocs permet aux novices de passer directement à la programmation sans devoir passer par la construction d'un algorithme. Un autre intérêt important de cette 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 en rapport avec la syntaxe, souvent importante chez les débutants en programmation, leur permet de ne se concentrer que sur d'autres aspects de la programmation. En effet, les caractéristiques structurelles offertes par de tels types d'environnements les rendent favorables et abordables aux novices sans trop de difficultés : ils sont capables de les manipuler à leur guise, de créer des scripts, d'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 (Tempel, 2013). 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. 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 (Tempel, 2013) : chaque étudiant peut évoluer selon son rythme et acquérir des compétences lui permettant la poursuite de la carrière informatique.

Le choix de l'approche par projet robotique est justifié 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. Pour Arnaud (1999), cette approche pédagogique, vue comme stimulante pour des étudiants déjà intéressés par cette discipline, entraîne un effet positif sur l'assimilation des connaissances. Il rejoint ainsi le point de vue de Tardif (1997) qui, deux ans avant, évoquait la recherche de solution mise en œuvre au cours de l'activité. Pour Tardif, 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 mais aussi de construire un ensemble de connaissances issues de beaucoup de domaines[2] : informatique, ingénierie, mathématiques, physique… La robotique pédagogique est donc inscrite dans une approche constructiviste 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.

Parmi les potentialités offertes par la robotique pédagogique, 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 des apprentissages, non seulement aux savoirs étroitement liés à la discipline informatique, mais aussi à d'autres savoirs issus de plusieurs domaines. 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).

S'intéressant à l'apprentissage par la programmation des microrobots, 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, une solution adaptée pour résoudre le problème de formation professionnelle en informatique et en technologie des personnels

pas ou peu qualifiés initialement. Une dizaine d'années après la naissance de la robotique pédagogique, Darche (1994) applique la notion d'acteur à la robotique, où un ensemble d'acteurs robotiques interviennent en réseaux mobiles et communiquant. Il propose d'adapter ce système à l'enseignement comme un outil au service de l'apprentissage des notions informatiques issues du parallélisme et de la communication. Leroux (2005) a expé-rimenté un type particulier de logiciel : RoboTeach, un assistant logiciel pédagogique. Conçu au milieu des années quatre-vingt-dix en vue d'une alphabétisation technologique et informatique chez les enfants, RoboTeach est « un environnement informatique support des activités menées dans le cadre d'une démarche de projet en micro-robotique pédagogique » (Leroux, 1995). La découverte des notions technologiques, techniques, la programmation, etc., mais aussi le pilotage d'un micro-robot étaient visées dans son expérimentation (Leroux, 2005). Un avantage important de ce type de langage est la possibilité de limiter d'abord, chez les débutants, l'apprentissage des notions informatiques : des 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 une acquisition d'une certaine maturité chez les apprenants. En RP, l'intérêt de la pédagogie de projet est souligné (Leroux *et al.*, 1996 ; Leroux, 2005) : des groupes de 2 à 3 apprenants travaillent en coopération[3] dans la résolution des problèmes prescrits par l'enseignant. Le rôle de ce dernier est d'assurer le guide.

Les recherches sur les kits robotiques dont fait partie le robot Lego Mindstorm NXT (Nijimbere *et al.*, 2013) affirment leurs potentialités éducatives, 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.

1.5. Problématique, questions et hypothèses de recherche

Les étudiants actuellement en licence d'informatique ont en général suivi au lycée des filières générales scientifiques. Ils n'ont jamais bénéficié d'une formation scolaire théorique en informatique. De ce fait, les étudiants de licence ne sont finalement pas loin des débutants en informatique. De plus, les recherches montrent que la programmation classique pose de nombreuses difficultés aux étudiants de ce niveau d'enseignement. Si près de 30 % d'étudiants échouent dans les premières années

d'université, Guibert *et al*. (2004), se référant aux travaux de Kaasboll, indiquent 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 et 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. Ce contexte d'apprentissage est vu 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. Cet article de recherche 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 informatique (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 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 ?

Deux hypothèses de départ ont orienté la recherche :

Hypothèse 1 : la robotique pédagogique, intégrée dans les apprentissages depuis l'école primaire, dans certaines écoles, ne pose a priori pas de difficultés majeures aux étudiants de L3.

Hypothèse 2 : la programmation de Nao exige beaucoup de connaissances chez les étudiants, elle permet par conséquent l'acquisition des notions informatiques avancées.

Je m'intéresse donc à la programmation de Nao, un robot humanoïde, par les étudiants de licence au cours de leurs projets. Ce type de technologie n'est prescrit aux étudiants qu'en L3 pour continuer en Master. L'hypothèse 2 (H2) formulée est motivée par le fait que la prescription institutionnelle du robot Nao commence tardivement, ce qui laisse penser que sa programmation est plus exigeante. Je suppose que la difficulté de Nao peut se situer en termes de connaissances et de créativité nécessaires par rapport à d'autres technologies robotiques, telles celles prescrites dans les classes antérieures : les étudiants de L2 utilisent le robot Lego Mindstorm qui est morphologiquement et structurellement différent de Nao.

2. Méthodologie

2.1. Population

La recherche s'intéresse donc aux étudiants de L3 en informatique à l'université Paris Descartes. Elle s'inscrit dans le cadre d'une thèse de doctorat en cours qui s'intéresse à l'apprentissage de l'informatique chez les débutants. Les étudiants de L3, par rapport à ceux de licence 1 et licence 2 ne sont pas novices en informatique mais peuvent servir de comparaison avec les débutants. En effet, si leur deuxième année de licence (L2) est totalement concentrée sur l'informatique, ils ont aussi suivi une année commune en première année de licence (L1) de mathématiques et informatique, avec une moitié de modules mathématiques et une autre informatique.

La recherche 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 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. Après soutenance de leurs projets, les étudiants pouvaient participer, après sélection, à une compétition inter-universitaire entre trois universités : université Paris Descartes, Université Diderot et Université de Paris Nord.

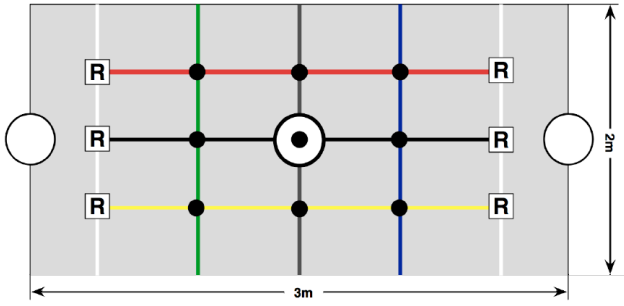

Figure 1 • Schéma de principe du terrain de jeu (RoboUni Cup, 2012)

2.1. Brève présentation du terrain de jeu et du robot utilisés

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 claire 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)[4]. La figure 1

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).

Figure 2.1 • Robot Nao

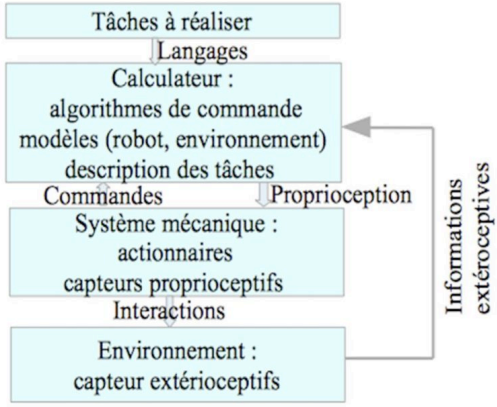

Figure 2.2 • Schéma général de fonctionnement d'un robot

2.2. Brève présentation du terrain de jeu et du robot utilisés

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 claire 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)[5]. La figure 1 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).

Roby-Brami et Laffont (2002), dans leur article « Gestes et technologie », définissent un robot comme un système programmable et multi-

fonctionnel, comportant un système de commande et un système mécanique (articulant plusieurs éléments réunis par des articulations) doté de moteurs, de récepteurs permettant de coder l'état des articulations – propriocepteurs – et d'informer sur l'état de l'environnement – extérocepteurs. Il est constitué de deux parties principales (Neboit et Poyer, 1991) : l'une composée d'une structure mécanique polyarticulée, regroupant des fonctions de déplacement et de manipulation ; l'autre regroupant des fonctions de traitement des informations et de commande. Dans la résolution d'un problème, le fonctionnement d'un robot suit un processus complexe, tenant compte de son système mécanique et des paramètres issus de son environnement (Roby-Brami et Laffont, 2002) comme elles le montrent sur le schéma de la figure 2.2.

La technologie utilisée dans ce projet robotique est Nao. Fabriqué par la société Aldebaran-robotics[6], Nao[7] 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 ultrason 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, 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 (Robot UniCup, 2012).

2.3. Une approche qualitative pour la collecte de données

Une méthode qualitative de type ethnographique a été privilégiée pour répondre aux questions de recherche. Marcel (2002) justifie la pertinence de ce type d'approche par quatre raisons principales : 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 ; 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.

Dans cette approche qualitative, deux outils de collecte de données ont été utilisés : des observations en situation d'apprentissage et des entretiens semi-directifs. Les observations de pratiques ont été faites pendant tout le second semestre de l'année scolaire 2011/2012. Avec cet outil, deux moments ont été particulièrement privilégiés : 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. Au total, 29 séances pratiques ont été observées. Parmi elles, 11 séances par groupe et sept séances communes. 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. Les observations ont été complétées par des entretiens de groupes[8] semi-directifs[9]. Initialement utilisée en sociologie américaine, la pratique de l'entretien de groupe a vite acquis ses lettres de noblesse en recherche (Morrissette, 2011). De plus en plus utilisée en recherche universitaire et surtout dans le domaine des sciences appliquées (santé publique, études féministes, sociologie, éducation…), l'entretien de groupe est une méthode d'enquête qui a, selon Morrissette, augmenté sa popularité en raison de la complexité des problématiques qui interpellent la recherche contemporaine et qui requièrent l'éclairage des différents acteurs impliqués. Les potentialités de cette méthode reposent globalement sur la fécondité des interactions qu'ils engagent entre les participants. Morrissette, se référant à Laplantine, montre l'intérêt de l'entretien de groupe : chercher à faire advenir avec les autres ce qu'on ne pense pas, plutôt que de vérifier sur les autres ce qu'on pense. Pour Morissette qui se réfère aussi à Poupart et Demazière, les entretiens de groupe sont conçus comme l'un des meilleurs moyens pour 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.

Les entretiens ont concerné les étudiants puis les enseignants-encadrants. Pour chaque groupe d'étudiants, 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 débutant 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 juin, auprès des deux enseignants d'informatique et encadrants desdits projets, après leur soutenance. Ces enseignants seront notés E1 et E2 dans la suite.

En résumé, des observations ethnographiques, trois entretiens semi-directifs et périodiques ont été menés avec les étudiants en groupes ont été réalisés à des fins d'analyse. Les données recueillies ont été complétées par un entretien semi directif en groupe avec les enseignants-encadrants de ces projets.

3. Présentation des résultats

3.1. 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 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 proposés[10], 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 ».

Quatre principales variables ressortent comme des raisons qui sont intervenues pour influencer les choix : le thème, la technologie, l'intérêt professionnel 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 lequel 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 à 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 ». 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 liée à leur rapport de familiarité envers la technologie proposée : un sujet faisant intervenir une technologie familière était 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.

Par sa forme humanoïde, Nao 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 plein 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'intérêts 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* ».

Une autre motivation est l'acquisition de nouvelles connaissances. 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. Le choix du sujet est motivé par les apprentissages permis par cette technologie, des connaissances non acquises ailleurs dans leurs cours. S'ils poursuivent les mêmes modules en classe, les projets leur permettent de faire des différences dans leurs acquisitions : elles varient en fonction des sujets de projets faits au cours de l'année ou de leurs cursus.

La dernière motivation 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 sou-

vent 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 ce n'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.

3.2. Quelques stratégies utilisées

Beaucoup de stratégies ont été essayées. Selon son efficacité, celle jugée plus performante que d'autres était retenue, ce qui suppose qu'il fallait d'abord tout essayer avant la sélection. Le groupe A explique ainsi ses choix : « *on essayait de penser comment le robot peut gagner un point facilement. Il y avait plein de stratégies, et on cherchait la plus efficace. Lorsqu'on fait une compétition, on cherche à gagner d'abord !* ».

Le balayage pour cartographier le terrain de jeu, est la première stratégie utilisée par tous les groupes. Elle a été mise en œuvre en deux temps. D'abord, le robot, dans sa position initiale, elle lui permettait de prendre connaissance de tout ce qui se trouve : balles, lignes et leurs couleurs, robot adverse et tout autre objet éventuel pouvant constituer un obstacle. La connaissance de la position du robot conditionnait celle des différentes positions des balles. Au moyen d'un balayage horizontal, le robot s'appropriait tout ce qui s'y trouvait en se géolocalisant lui-même. Ensuite, cette stratégie de balayage était utilisée pendant le déplacement du robot.

La deuxième stratégie utilisée a consisté à cibler prioritairement la balle située au milieu du terrain. Cette stratégie avait un double avantage. D'abord, au début, selon les étudiants, la configuration du terrain n'était pas encore changée et les balles sont dans leur position d'origine. En effet, si le robot circule sur le terrain, certaines balles sont bousculées et peuvent perdre leur place. Ensuite, cette stratégie permettait de marquer le premier panier, en premier avant le robot adverse, pour bénéficier d'un bonus (Règlement de la compétition, 2012) : « Un bonus de 3 (points) est accordé à l'équipe qui dépose la première balle dans l'en but adverse ».

Prendre le chemin le plus court pour atteindre la balle cible a été la troisième stratégie. Une fois les balles identifiées, des distances entre elles et le robot étaient calculées puis comparées au moyen d'un algorithme construit. C'est la plus courte distance qui était parcourue par le robot pour se diriger vers la balle la plus proche. Elle permettait au robot d'atteindre rapidement la zone d'en-but 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 nouveaux faits. La balle la plus proche du robot, à partir de la zone d'en but adverse est ciblée en premier. La synthèse des précédentes stratégies est présentée dans le tableau 1.

Tableau 1 • Variations des stratégies utilisées selon les tâches et les groupes

Tâche	Stratégie utilisée	Groupe concerné
Cartographie du terrain de jeu	Balayage avec la tête (caméra)	A et B
Géolocalisation des balles	Prise de la photographie du terrain + analyse de cette photo	A et B
Gain le bonus	Priorité à la balle centrale	A et B
Rechercher un plus court chemin à suivre	Calcul de la distance minimale entre la position actuelle de Nao et les balles	A et B
Déplacement vers la balle cible	Jonglage sur la vitesse équilibrée permettant un déplacement rapide de Nao en direction de la balle cible	A et B
Ramassage de la balle	Prise de position à la distance (Nao, Balle) préalablement mesurée	A et B
	Ajustements des articulations du bras en mouvement jusqu'au-dessus de la balle + Mouvements des articulations de la main pour saisir la balle	A
Localisation du panier	Détection des Naomarks[11] par balayage horizontal	A et B
Bien se positionner face au panier	Conception et utilisation d' un algorithme « SearchMark »	A
Marquer un but : mettre la balle dans le panier	Prise de position à la distance (Nao, Panier) préalablement mesurée	A et B
	Nao ajuste ses articulations pour monter le bras jusqu'au-dessus du panier	A et B
	Ajustement de la main pour lâcher la balle dans le panier	A et B

3.3. Des connaissances utilisées et celles construites

3.3.1. Connaissances informatiques nécessaires

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 2 page suivante synthétise les connaissances informatiques utilisées.

Tableau 2 • Notions informatiques intervenues dans la programmation des robot

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

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 ne 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 représentation 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. »

Tableau 3 • Notions informatiques intervenues dans la programmation des robots

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
Recherches informationnelles sur Internet	Recherche documentaire : usage des moteurs de recherche et des mots clés	A et B

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 3 présente quelques connaissances acquises, classées selon leur type (langage, logiciels…).

3.3.2. Un déficit de connaissances mathématiques

Dans la programmation des robots, les connaissances mathématiques utilisées sont restées rares. Les étudiants qui s'attendaient à se confronter à d'énormes besoins en rapport avec des mathématiques plus compliquées, se disent étonnés de constater le contraire. Ils affirment n'avoir pas connu des difficultés relatives à ce sujet. 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 (…)* ».

Tableau 4 • Notions et domaines mathématiques intervenant dans la programmation de Nao

Domaine mathématique	Nao	Groupe concerné
Géométrie	Repère à deux ou trois dimensions, coordonnées, cercle, diamètre, rayon, circonférence, centre d'un cercle, rotation, distance	A et B
Trigonométrie	Angle, radian, fonctions circulaires, fonctions circulaires inverses (asinus, acosinus, atan…)	A et B
Algèbre	Théorèmes de Thalès et Pythagore	A et B

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 4 présente la synthèse des notions mathématiques utilisées dans ces projets.

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.*, 2013).

3.4. Difficultés rencontrées

3.4.1. 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.

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 préférés chez les étudiants. Par exemple, ils faisaient « *apprendre par cœur des positions favorables* » au robot et ce, à la main, des positions qu'ils trouvaient stratégiques pour faire l'un ou l'autre geste. Au moment du test le robot se bloquait et risquait de tomber avec des possibilités de se casser. Face à cette situation très risquée pour le robot, les étudiants devaient être prêts pour le rattraper 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, laquelle donne lieu 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 et 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.

3.4.2. 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 préférés chez les étudiants. Par exemple, ils faisaient « *apprendre par cœur des positions favorables* » au robot et ce, à la main, des positions qu'ils trouvaient stratégiques pour faire l'un ou l'autre geste. Au moment du test le robot se bloquait et risquait de tomber avec des possibilités de se casser. Face à cette situation très risquée pour le robot, les étudiants devaient être prêts pour le rattraper 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, laquelle donne lieu 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 et 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.

3.4.3. 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.*, 2013), les étudiants rencontrent de fortes difficultés techniques. Si ces dernières se font remarquer dans le cas des projets robotiques en général, elles sont plus particulièrement marquées pour le cas présent. Plus souvent difficiles à anticiper, elles créent des surprises aux étudiants. La forme humanoïde de Nao donne l'impression trompeuse d'une certaine simplicité de la programmation de cette technologie alors qu'il n'en est rien.

L'enseignant E1 attribue 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.* ».

À la question de savoir où se situent les plus grandes difficultés dans la programmation de Nao, les étudiants confirment les propos des enseignants et pointent le côté technique. Pour eux, « *c'est la précision de Nao* » qui leur a posé le plus de difficultés. Les conséquences de ces difficultés techniques sont identifiables. Elles se caractérisent par un écart qui se dessine souvent entre ce qu'ils prétendent programmer et le résultat de cette programmation. Si avec cette forme humanoïde de Nao, ils espéraient lui faire faire autant de mouvements qu'un humain est capable de faire, la distance entre les souhaits et les possibles reste énorme : « *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 !* »

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 donné). 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.

3.4.4. Difficultés algorithmiques

À côté des choix méthodologiques s'identifient des difficultés concernant les types d'algorithmes utilisés. Les étudiants se retrouvent confrontés dans la programmation de Nao, 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 façon dont la technologie est construite : la conception particulière des robots fait que leur programmation pose des difficultés particulières 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 de licence dans la programmation des robots 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.

3.4.5. 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. (..) c'est rattraper la balle à coup sûr, voilà 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 vu comme le robot le plus évolué par rapport à d'autres technologies robots 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 cm devant lui, un cm 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 baisser 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 !* ».

4. Discussion

Avant d'entrer dans la discussion, rappelons les deux hypothèses qui ont été formulées. La première hypothèse stipule que les étudiants de L3 en programmation de Nao n'ont pas de difficultés fondamentales en robotique étant donné que même les enfants de très bas âges programment des robots. La deuxième hypothèse est contraire à la première. Elle stipule que la programmation de Nao exige un certain niveau en informatique et permet donc d'aller plus loin dans l'apprentissage de la programmation. Elle n'est en conséquence pas adaptée aux tous débutants en informatique.

4.1. Nao : une simplicité trompeuse

Programmer un robot de type humanoïde n'est pas chose aisée. Si l'hypothèse postule que la programmation des robots ne devraient pas poser pratiquement de difficultés aux étudiants de licence en informatique,

la réalité est toute autre. Nao semble poser d'énormes difficultés aux étudiants. Cette situation est complexifiée par l'apprentissage d'un nouveau langage de programmation. Inspirés de Logo, les langages destinés aux enfants et, plus généralement aux novices en algorithmique et programmation, sont des langages simples ou simplifiés pour présenter certaines souplesse et transparence. Orientés vers le jeu, jeux sérieux, vidéos, etc., ces langages sont conçus de telle sorte que les apprenants sont dispensés de certaines difficultés : syntaxiques ou conception de la méthode à suivre, qui sont souvent source de difficultés majeures (Duchâteau, 2002). L'analyse de la littérature révèle une centration généralisée sur des langages à blocs ou à briques pour une initiation à la programmation. Ce contexte diffère de celui des L3 qui programment Nao : le langage Python, nouveau pour eux, ne relève pas de cette catégorie. 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.

Si la forme humanoïde de Nao donne l'impression de simplicité et de facilité pour sa programmation, cette dernière pose beaucoup de difficultés aux étudiants de licence malgré leur expérience en programmation informatique. Un code produit sur la machine est envoyé pour être exécuté sur le robot, un autre exécutant séparé de l'ordinateur mais formant un système avec lui, ce qui engendre des difficultés supplémentaires : un même code peut ne pas fonctionner sur ce dernier alors qu'il n'a pas de problèmes sur ce premier. L'ajout de la complexité dans ce contexte de la programmation des robots a été déjà relevé dans les recherches. Dans ce type de programmation, 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 (De Dormale, 2003 ; Duchâteau, 1993 ; Niboit & Poyer, 1991). Cette complexité est peut-être l'une des raisons qui explique qu'il n'est pas prescrit aux étudiants avant la L3. Si certains parmi eux ont produit un travail intéressant, certaines conditions ont contribué à ce qu'il en soit ainsi : la motivation engendrée par les rapports de sympathie des étudiants envers cette technologie et, des bouts de codes et des vidéos déjà construits obtenus par leurs 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 dans la programmation de Nao, laissent penser que, si les enfants de l'école primaire peuvent programmer les robots, ces derniers sont généralement adaptés à leur niveau et leur âge. Certaines variables semblent être prises en compte pour prescrire certaines technologies robotiques aux apprenants. Leur complexité évolue en fonction du niveau d'étude, de l'âge, de l'expérience, etc. Nao semble plus complexe que les autres robots prescrits antérieurement. Ceci peut justifier finalement pourquoi il occasionne de difficultés non connues par les étudiants dans les classes précédentes.

4.2. De l'affection envers la technologie à l'apprentissage avec la technologie

Les états affectifs des apprenants sont déterminants dans leur motivation. La sympathie envers la technologie, plus prononcée dans le cas de Nao, est 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.

4.3. Robotique pédagogique : 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 surfacique » pour reprendre les termes de 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 programma-

tion 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 *et al.*, 2013).

4.4. 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, surtout 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 et Poyer (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 déplacement 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 et 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 con-

naissances 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.

À côté des équations différentielles, évoquées comme indispensables, d'autres connaissances mathématiques avancées ont été soulignées dans les recherches : les équations dynamiques. En effet, Espiau et 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 et Oudeyer (2008) 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 travailler sur le corps complet (Espiau et 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 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. Il me semble que 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 Mindstorm. La deuxième hypothèse stipulant que Nao, très exigeant 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 tout 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.

5. Conclusion et perspectives

Dans cet article d'orientation didactique, j'ai étudié les connaissances acquises par les étudiants de licence 3 d'informatique en projets de programmation de Nao, un robot humanoïde. Moyennant certaines conditions favorables, préalables à 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 sont construites grâce à sa potentialité à décloisonner les frontières imposées par des spécificités disciplinaires.

L'absence de mise en œuvre de modélisation a sensiblement limité la mise à profit de certaines des stratégies imaginées. C'est ce qui a limité aussi 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) et les recherches récentes montrent qu'ils n'ont pas de difficultés particulières dans la programmation classique (Nijimbere *et al.*, 2013). Les étudiants peu initiés à la programmation rencontrent de difficultés avec le robot de type Lego MINDSTORM NXT, difficultés liées à la complexité du contexte non vécues dans la programmation classique. Le contexte de la programmation de Nao est complexe et occasionne de difficultés plus importantes, orientées techniques. Elles étaient 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 représentations mentales qu'ont les étudiants 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 permettre la décomposition en actions élémentaires des tâches à faire faire par Nao et ainsi arriver au bout de ces difficultés techniques. Ces pratiques laissent transparaître des représentations mentales de Nao lacunaires chez les étudiants : 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 de 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 de 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 que, contrairement à la programmation classique et même celle des kits robotiques, celle des robots humanoïdes, dont Nao fait partie, nécessite un certain niveau en programmation. Les multiples paramètres environnementaux qui interviennent contribuent à complexifier le travail à faire et exigent des représentations mentales suffisantes et des compétences d'anticipation un peu plus poussées. 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, à mon sens, non pas un début mais une étape pour leur perfectionnement en programmation, ce qui justifie finalement sa prescription institutionnelle tardive en licence.

En perspective, une étude qui s'oriente vers l'analyse des pratiques d'élèves plus jeunes permettra de rendre compte de l'appropriation de savoirs informatiques pour d'autres débutants. Une approche comparative est en cours au moyen de données issues de la découverte de l'informatique par des élèves de lycées, en filières scientifiques.

Remerciements

Au terme de cet article, je me réjouis d'adresser ma reconnaissance aux personnes suivantes qui ont contribué à son aboutissement. Mes remerciements sont d'abord adressés à Georges Louis Baron, professeur en sciences de l'éducation à l'Université Paris Descartes pour avoir pris son temps pour la correction et la relecture de cet article malgré ses multiples obligations. Mes remerciements s'adressent aussi à Messieurs David Janiszek et Damien Pellier, enseignants au département d'Informatique à l'Université Paris Descartes. Après m'avoir permis d'accéder à l'observation des pratiques et des échanges avec leurs étudiants en projets, ils m'ont donné un entretien pour plus de compléments. Enfin, mes remerciements s'adressent aux étudiants de licence en projet en informatique pour l'année scolaire 2011/2012. Ils ont, non seulement accepté d'être suivis dans leurs pratiques, mais aussi, ont contribué à les expliciter dans différents entretiens donnés.

1 http://pagestec.org/web2001/article.php?sid=513, site consulté le 24 mai 2014

2 http://www.aldebaran-robotics.com/For-Education/introduction.html, consulté le 20 septembre 2012

3 Leroux, Vivet, et Brézillon distinguent la coopération de la collaboration : Dans la collaboration, le travail à faire, les tâches et les responsabilités sont les mêmes alors qu'en travail coopératif, même si le travail est le même, les tâches et les responsabilités sont séparées, moyennant une mise en commun du travail final (Leroux *et al*., 1996)

4 http://www.sorbonne-paris-cite.fr/index.php/en/component/simpledownload/?task= download & fileid=cmVnbGVtZW50LWNvbXBldGl0aW9uX3JvYm90aXF1ZS5wZGY%3D, document consulté le 10 février 2014

5 http://www.aldebaran-robotics.com/fr/, site consulté le 17 février 2014

6 Fabriqué en 2005 par la société française aldebaran-robotics, le robot Nao est commercialisé pour des fins éducatives : recherches dans des laboratoires et universités

7 Si cette expression a plusieurs appellations (focus group, groupe de discussion, interview de groupe, groupe focalisé, entretien collectif). Colette Baribeau, Jason Luckerhoff et François Guillemette trouvent qu'il est préférable que les chercheurs en recherches qualitatives utilisent un seul terme : entretien de groupe.

8 Contrairement aux questionnaires, l'entretien semi-directif a l'avantage de pouvoir relancer une question pour éclairer davantage un point de vue ou approfondir une réponse donnée

9 Les étudiants étaient déjà au courant des sujets qui se trouvent sur le site du département de l'informatique

10 C'est un dispositif circulaire avec un numéro monté sur le panier de but pour le caractériser

BIBLIOGRAPHIE

ARCHAMBAULT J.P. (2011). Un enseignement de la discipline informatique en Terminale scientifique. Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques. *Actes du quatrième colloque international DIDAPRO 4-Dida&Stic,* Université de Patras. p. 205–212

ARNAUD P. (1999) *Des moutons et des robots : Architectures de contrôle réactive et déplacements collectifs de robots*. Presses Polytechniques et universitaires romaines.

BARON G.-L. (2012). L'informatique en éducation : quel(s) objets d'enseignement ? In *Le "e-Dossiers de l'audiovisuel : l'Education aux cultures de l'information. Translittératies : enjeux de citoyenneté et de créativité* p. 81-88. ENS-Cachan et Université Sorbonne nouvelle.

BARON, G.-L. (1987). *La constitution de l'informatique comme discipline scolaire; le cas des lycées*. Thèse de doctorat. Université Paris IV.

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*.

BARON G.-L., VOULGRE, E. (2013). Initier à la programmation des étudiants de master de sciences de l'éducation ? Un compte rendu d'expérience. *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. http://edutice.archives-ouvertes.fr/edutice-00875549

BERS M.-U. (2008). *Blocks to robots : Learning with technology in the early childhood classroom*. Teachers College Press New York.

BONNEL B. (2010). *Vive la robolution*. Paris : Éditions Jclattès.

BOUDREAULT Y., PREGENT R. (2005). Projet intégrateur pluridisciplinaire exploitant la robotique pour les étudiants de première année en génie informatique et en génie logiciel. *Actes du 8e colloque francophone de Robotique Pédagogique* p. 81-88 . La Ferté-Bernard, France.

BOURGUIBA R. (2000). *Conception d'une architecture matérielle reconfigurable dynamiquement dédiée au traitement d'images en temps réel*. Thèse de doctorat, Université de Cergy-Pontoise, Cergy-Pontoise. France.
http://cat.inist.fr/?aModele=afficheN&cpsidt=14196035

BRUILLARD É., DELOZANNE É., LEROUX P., DELANNOY P., DUBOURG X., JACOBONI P., LEHUEN J., LUZZATI D., TEUTSCH P. (2000). Quinze ans de recherche informatique sur les sciences et techniques éducatives au LIUM. *Revue Sciences et Techniques Educatives*, Vol. *7* n° 1, 87-145.

DARCHE P. (1994). *Le Paradigme Acteur appliqué aux Systèmes Embarqués Communicants. ActNet, un Réseau d'Acteurs Robotiques*. Thèse de doctorat en Informatique. Paris 6, Paris

DE DORMALE R.-M. (2003). Apprendre à programmer Pourquoi? Comment? *CAR : Computers Are Robots*. http://www.paixactive.org/platform/memoire.pdf

DELOZANNE E., JARRAUD P., MURATET M. (2011). Un projet Jeux sérieux pour approfondir l'apprentissage de la programmation en première année à l'université. *DIDAPRO 4-dida &STIC : Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif* p. 240-249.

DOWEK G., ARCHAMBAULT J.-P, BACCELLI E, CIMELLI C, COHEN À, EISENBEIS C, VIEVILLE T, WACK B, (2011). *Informatique et sciences du numérique: Spécialité ISN en terminale S*. Paris : Éditions Eyrolles.

DUCHÂTEAU C. (2002). *Images pour programmer*. Première partie : Apprendre les concepts de base. Édition revue et augmentée. Disponible sur Internet: https://pure.fundp.ac.be/ws/files/252136/images1-5-79.pdf
(consulté le 10 décembre 2013)

DUCHÂTEAU C., (1993). Robotique-Informatique : mêmes ébats, mêmes débats, mêmes combats : Regards sur la robotique pédagogique. *Actes du quatrième colloque de Robotique Pédagogique*, Liège.

ESPIAU B., OUDEYER P.-Y. (2008). Robotique : de l'automate à l'humanoïde. Un robot très curieux, entretien avec Pierre-Yves Oudeyer, propos recueillis par Dominique Chouchan. *La Recherche Les Cahiers de l'Inria*, Vol. 424.
http://hal.inria.fr/inria-00536681/

GUIBERT N., GUITTTET 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.

GAUDIELLO I., ZIBETTI E. (2013). La robotique éducationnelle : état des lieux et perspectives. *Psychologie Française*, Vol. 58 n° 1, 17-40.
doi :10.1016/j.psfr.2012.09.006

GUZDIAL M. (2004). Programming environments for novices. *Computer science education research*, p. 127–154.

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. DIGI '07. The First IEEE International Workshop on* p. 41-48. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4148830

JANISZEK D., PELLIER D., MAUCLAIR J., BOULC'H L., BARON G.-L., PARCHEMAL Y. (2011a). Utilisation de la robotique pédagogique pour enseigner l'intelligence artificielle: une expérience d'approche par projet auprès d'étudiants en informatique. *STICEF*, Vol. 18, p. 75-92. http://sticef.univ-lemans.fr/num/vol 2011/07r-janiszek/sticef_2011_janiszek_07rp.html

JANISZEK D., BOULC'H L., PELLIER D., MAUCLAIR J., BARON G.-L. (2011b). De l'usage de Nao (robot humanoïde) dans l'apprentissage de l'informatique. In Baron, GL, Bruillard Eric et Komis Vassilis (Eds.), *Didapro4-Did&STIC: Sciences et technologies de l'information et de la communication en milieu éducatif. Actes du Colloque International* p. 231-239.

KAZIMOGLU C., KIERMAN M., BACON L., MACKINNON L. (2012). À serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences,* n° 47, 1991-1999.

KLOPPFER E., BEGEL À. (2003). StarLogo under the hood and in the classroom. Kybernetes: *The International Journal of Systems & Cybernetics*, Vol. 32 n° 1-2, 1-2.

KYNIGOS C. (2008). Black and white perspectives to distributed control and constructionism in learning with robotics. *Workshop Proceedings of SIMPAR*.

KOMIS V, MISIRLI À. (2011). Robotique pédagogique et concepts préliminaire de la programmation à l'école maternelle : une étude de cas basée sur le jouet programmable Bee-Bot. In Baron, GL, Bruillard Eric et Komis Vassilis (Eds.), *Didapro4-Did&STIC : Sciences et technologies de l'information et de la communication en milieu éducatif. Actes du Colloque International,* p. 271-281.

LABORDE C., MEJIAS B, BALACHEFF N. (1985). Genèse du concept d'itération : une approche expérimentale. *Enfance*, Vol. *38* n° 2-3, 223-239.

LAUZIER I., NONNON P. (2007). Une expérience d'approche par projet pour favoriser l'intégration des apprentissages. *9ème Colloque Francophone de Robotique Pédagogique*, p. 14-21, La Ferté-Bernard, France.

LEBEAUME J., HASNI À., HARLE I. (2011). *Recherches et expertises pour l'enseignement scientifique :Technologies - Sciences - Mathématiques*. De Boeck, 193 p.

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.

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.

LEROUX P. (2005a). 20 ans de Robotique Pédagogique à l'Université du Maine, Le Mans. p.7-18. *8e colloque francophone de Robotique Pédagogique,* La Ferté-Bernard. France.

LEROUX P. (2005b). Réalisation de micro-robots au collège mise au point d'une démarche pédagogique et d'un environnement informatique support des activités. *Aster*, n° 41, Produire, Agir, Comprendre, 49-78.

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 Pierre Vérillon, Jacques Ginestié, Joël Lebeaume et Pascal Leroux. *Produire en technologie à l'école et au collège*, p. 119-168. Institut national de recherche pédagogique.

LEROUX P., VIVET M. (2000). Micro-Robots Based Learning Environments for Continued Education in Small and Medium Enterprises (SMEs). *Journal of Interactive Learning Research*, Vol. 11 n° 3, 435-463.

LEROUX P., VIVET M., BRÉZILLON, P. (1996). Cooperation between humans and a pedagogical assistant in a learning environment. In *Proceedings of European Conference in AI in Education (EuroAIED)*, Lisbon, Portugal, p. 379-385.

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.

MARCHAND D. (1991). La robotique pédagogique! Ça existe ? *Bulletin de l'EPI,* n° 65, 119-124.

MARCEL J.-F. (2002). Approche ethnographique des pratiques enseignantes durant les temps interstitiels. *Revue Spirale*, n° 30, 103-120.

MENDELSON P. (1985). L'enfant et les activités de programmation. *Grand N,* n° 35, 47-60.

MORRISSETTE J. (2011). Ouvrir la boîte noire de l'entretien de groupe. *Recherches qualitatives*, Vol. 29 n° 3, 7-32.

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.

MURATET M., TORGUET P., VIALLET F., JESSEL J.-P. (2011). Experimental feedback on Prog&Play: a serious game for programming practice. *Computer Graphics Forum,* n° 30, 61-73. Wiley Online Library.

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*, Vol. 7 n° 4, 419-433.

NEBOIT M., POYET C. (1991). La communication homme-machine en robotique : analyse comparée de différentes interfaces de programmation. *Technologies de l'Informatique et Société*, Vol. 3 n° 2-3, 231-249.

NIJIMBERE C., BOULC'H L., HASPEKIAN M, BARON G.-L. (2013). Apprentissage de l'informatique par la programmation des robots. Cas de Lego MINDSTORM NXT. In Drot-Delange B, G.-L, Baron et É. Bruillard (dirs.). *Actes du colloque Didapro5-Dida&STIC*. Clermont Ferrand.

NONNON P. (2002). Robotique pédagogique et formation de base en science et technologie. *Aster,* n° 34 « Sciences, techniques et pratiques professionnelles ».

O'KELLY J., GIBSON J.-P. (2006). RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. *ACM SIGCSE Bulletin*, Vol. *38* n° 3, 217-221.

PALIOKAS I., ARAPIDIS C., & MPIMPITSOS M. (2011). Playlogo 3d: À 3d interactive video game for early programming education: Let logo be a game. In *Games and Virtual Worlds for Serious Applications (VS-GAMES), Third International Conference,* p. 24-31. IEEE.

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.

PAP-SZIGETI R., PASZTO À., LAKATOS TOEROEK E. (2010). Effects of Using Model Robots in the Education of Programming. *Informatics in Education-An International Journal*, Vol 9_1, 133-140.

PAULIN M., BOUREAU E., DARTNELL C., KRUT S. (2006). Modélisation et planification d'actions élémentaires robotiques par apprentissage de réseaux de contraintes. *Deuxième Journées Francophones de Programmation par Contraintes (JFPC06)*. Disponible sur Internet : http://hal.archives-ouvertes.fr/inria-00085797/ (consulté le 15 novembre 2013)

RAPPORT DE L'ACADÉMIE DES SCIENCES (2013). *L'enseignement de l'informatique en France. Il est urgent de ne plus attendre*. Disponible sur Internet : http://www.epi.asso.fr/revue/docu/d1305a.htm (consulté le 2 octobre 2013

RÉGLEMENT DE LA COMPÉTITION ROBOT PRES CUP (2012). Edition 2012. Disponible sur Internet : http://www.google.fr/search?hl=fr&source=hp&q=R%C3%A8glement+de+la+comp%C3%A9tition+Robot+PRES+Cup%2C+%282012%29.+Edition+2012.&btnG=Recherche+Google&gbv=1 (consulté le 12 décembre 2013)

RESNICK M., MARTIN F., SARGENT R., SILVERMAN B. (1996). Programmable bricks : Toys to think with. *IBM Systems journal*, Vol. 35 n° 3.4, 443-452.

ROBY-BRAMI À., LAFFONT I. (2002). Gestes et technologie : la compensation des incapacités motrices. In Blandine Bril et Valentine Roux (dir.), *Le geste technique : Réflexions méthodologiques et anthropologiques,* p. 95-112. Ramonville Saint-Agne : Érès.

TARDIF J. (1997). Pour un enseignement stratégique : l'apport de la psychologie cognitive : *Les Editions Logiques*.

TEMPEL M. (2013). Blocks Programming. *CSTA Voice, 9(1). Disponible sur Internet :* http://el.media.mit.edu/logo-foundation/pubs/papers/blocks_programming.pdf. *(consulté le 12 janvier 2014).*

VIVET M. (2000). Des robots pour apprendre. *Revue des Sciences et Techniques Éducatives*, Vol. 7 n° 1, 17-60.