Systèmes Multi-Agents :

Projet : Nao Messenger

Marie Bouette,
Maxime Escamez

9 mai 2018

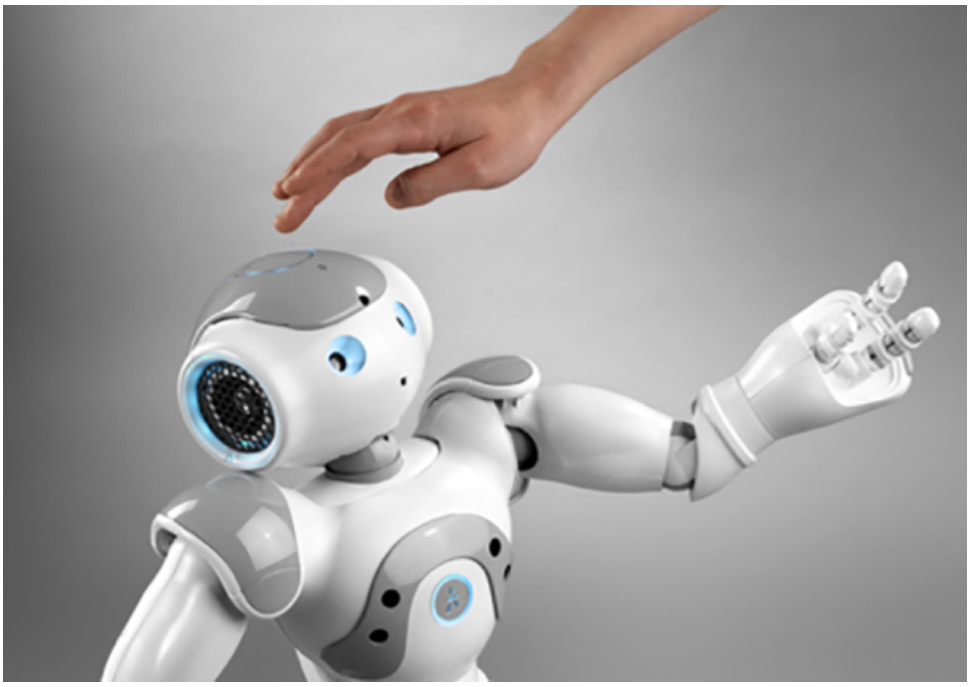

Table des matières

Introduction

Le robot Nao est un robot de la société SoftBank Robotics, anciennement appelée la société Aldebaran. Ce robot, créé en 2006, est principalement destiné à la recherche et à l'éducation. La société SoftBank Robotics a mis en place un certain nombre d'outils pour facilité sa programmation, et la découverte de son fonctionnement. Ainsi, il est possible de programmer le robot à l'aide de l'outil Choregraphe, que nous vous présenterons un peu plus bas, mais il est aussi possible d'écrire directement des scripts dans les langages C++ ou Python, en utilisant la bibliothèque NAOqi, et ces scripts s'exécutent directement sur le robot.

Nous travaillons dans le cadre d'un cours de Systèmes multi-agents. C'est donc dans cette optique que nous avons voulu intégrer à la fois le concept d'*agent intelligent* dans le robot, avec les propriétés de *réaction*, *pro-action*, *autonomie* ; mais aussi le concept de *communication* avec l'environnement, avec les autres agents.

Ce rapport a pour but de vous présenter notre projet, et comment nous avons implémenté le concept de *système multi-agents* dans le robot Nao, mais aussi d'expliquer les différents outils que nous avons utilisé pour parvenir à nos fins, dans le but de faciliter les travaux futurs qui pourraient être réalisés sur le robot.

Nous vous présenterons tout d'abord les différents outils que nous avons utilisé : Choregraphe et NAOqi, et nous décrirons brièvement leurs fonctionnements à fin de facilité une future prise en main. Nous parlerons ensuite du projet que nous avions l'idée d'implémenter, et de pouvoir ceci n'a pas pu fonctionner. Enfin nous présenterons notre idée finale, son principe, l'implémentation du concept d'agent et du concept de système multi-agents, etc.

1 Choregraphe

Choregraphe est l'outil le plus simple et le plus intuitif qu'il est possible d'utiliser pour contrôler Nao.

1.1 Fonctionnement de Choregraphe

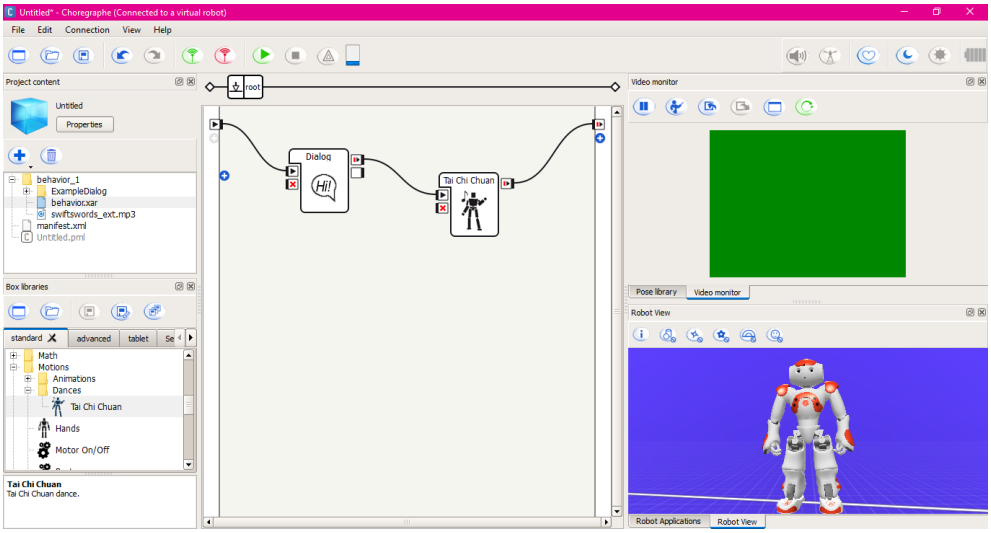

FIGURE 1 – Choregraphe

Cet outil permet de contrôler Nao avec plusieurs *boîtes* qui sont en fait des fragements de codes. Chaque boîte représente une fonctionnalité. Par exemple, sur l'image ci-dessus, l'exécution de cette ligne fera parler Nao : "Hi", puis il dansera le Tai Chin Chuang. Ces boîtes d'exécution offrent un grand nombre de possibilités, mais il est aussi possible d'en créer de nouvelles pour ajouter de nouvelles fonctionnalités.

Avec Choregraphe, il est possible de se connecter à un robot virtuel, comme c'est le cas sur l'image 1, et il est aussi possible de se connecter au véritable robot, que l'on connecte avec le numéro du port et l'identifiant du robot.

En haut, à droite, vous remarquerez sur l'image un panneau vert. Ceci correspond au Video monitor, c'est-à-dire à ce que vois le robot en temps réel. Sur la capture d'écran, Choregraphe est relié au robot virtuel, il n'est donc pas possible de savoir ce qu'il voit. Mais lorsque le logiciel est relié à un robot réel, c'est par ce moyen qu'il est possible de surveiller ce qu'il voit, les objets qu'il apprend par exemple...

1.2 Apprentissage de visages

Comme nous l'avons évoqué au paragraphe précédent, il est possible d'apprendre à Nao à reconnaître des objets ou des visages humains. Puisque notre rapport a aussi pour objet d'initier à l'utilisation de Nao, nous expliquerons brièvement ici comment apprendre au robot Nao à reconnaître votre visage.

Pour apprendre un visage humain au robot, nous devons gérer sa base de données. Nous utilisons donc deux *boîtes* de l'outil Choregraphe :

— **Unlearn all faces** : pour vider la base de données de tous les visages après : il suffit d'exécuter d'exécuter deux fois la *boîte* **Unlearn All Faces** que vous trouverez dans le dossier **Vision** de Choregraphe ;

— **Learn Face** : pour ajouter votre visage à la base de données : il suffit de lancer l'exécution, puis de double double-cliquer sur la flèche d'entrée de la boîte **Learn Face**, que vous trouverez dans le dossier **Vision**, de rentrer un identifiant unique, puis de mettre votre visage devant celui du robot Nao, et de le fixer jusqu'à ce que ses yeux deviennent verts.

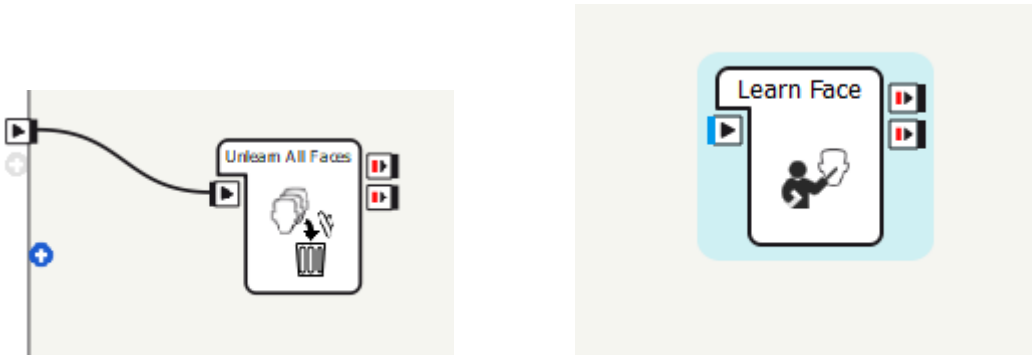

FIGURE 2 – Apprentissage de visages

2 NAOqi

Le framework NAOqi est le framework qu'utilise la société SoftBank Robotics pour ses robots. Il permet de lier les différents modules, c'est-à-dire les différentes fonctionnalités du robot, et de programmer en les deux langages C++ et Python. Il est composé d'une très grande bibliothèque de fonctionnalités, NAOqi API, que l'on peut composer à l'infini, avec une documentation très détaillée que vous trouverez à l'adresse suivante : `http://doc.aldebaran.com/2-4/naoqi/`

`index.html`.

Pour utiliser le framework, il est essentiel de comprendre certains concepts fondamentaux [1].

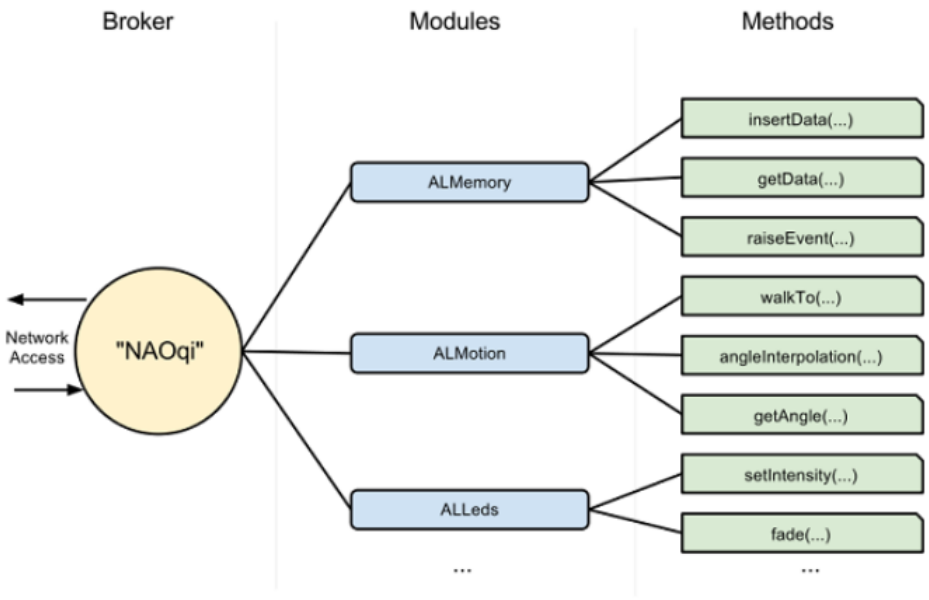

FIGURE 3 – NAOqi

2.1 Broker

Pour utiliser l'API de NAOqi, il faut tout d'abord faire appel au **Broker**. Cet outil permet de connecter le programme au robot, mais aussi de faire appel aux différents modules, c'est-à-dire les différentes fonctionnalités du robot Nao.

2.2 Modules abordés

Nous avons utilisé plusieurs modules de la bibliothèque NAOqi : ALMemory, pour l'accès à la mémoire des visages mémorisés, ALTextToSpeech pour faire parler le robot, ALFaceDetection pour la détection des visages, ALTracker pour traquer les visages que nous détections, ALMotion pour faire avancer le robot ... Nous allons en détailler le fonctionnement de ceux qui nous ont servi le plus.

2.2.1 Reconnaissance d'objet

Nao peut reconnaître des objets préalablement appris. L'apprentissage d'un objet se fait a partir du Video Monitor de Chorégraphe et la boite *learn object*. Les données liées à cet objet (id, forme etc..) sont stockés dans la base de données locale de Nao.
En utilisant la boîte Chorégraphe *object recognition* ou le module ALVisionRecognition NAOqi, Nao peut reconnaître un objet appris. Néanmoins, le distance entre Nao et l'objet doit être environ 30 cm.
ALVisionRecognition est un module de la famille ALVision qui interagit avec les caméras de Nao situées dans ses yeux.

2.2.2 Reconnaissance et détection de visage

Nao peut aussi reconnaître des visages préalablement appris mais aussi en détecter sans avoir appris. Ici aussi, l'apprentissage se fait à partir du Vidéo Monitor de Chorégraphe et la boîte *learn face*. Les données liés aux visages appris, notamment l'id, sont stocké dans la base de données locale de Nao.
Les boîtes Chorégraphe *face detection* ou la fonction de base du module NAOqi ALFaceDetection permettent de détecter n'importe quel visage et de le suivre. La boîte Chorégraphe *face recognition* permet de reconnaître des visages en particulier. Alternativement, sous NAOqi on peut vérifier si le visage détecté est situé dans la base de données de Nao.
La détection de visage est plus performante que la reconnaissance d'objet. Nao peut détecter et même reconnaître un visage à au plus deux mètres.

2.2.3 Tracker

Le tracker de Nao permet de tracer différentes cibles en utilisant une ou plusieurs parties de son corps (tête, bras, tout le corps..).
Le principal but du tracker est d'établir un lien entre la détection de cible et le mouvement de façon à avoir la position spatiale de la cible pour que Nao puisse se mettre en mouvement.
Sous NAOqi, c'est le module ALTracker qui emploie le tracker. Lorsque que ce module est appelé, on a accès à un grand nombre de fonctions telles que *getTargetPosition()* ou *getTargetCoordinates()*. Ces fonctions, une fois la détection d'objet ou de visage effectuée, permettent de récupérer la position spatiale de l'objet ou du visage dans le champ vision de Nao depuis un calcul basé sur les dimensions

de la cible. De cette facon, Nao obtient des coordonnées qui pourront être utilisées dans la phase de mouvement.

2.2.4 Mouvement

Pour mettre Nao en mouvement, on peut utiliser Chorégraphe et ses nombreuses boîtes pré-définies. On peut aussi utiliser le module ALMotion de NAOqi. Ce module fournit les différentes fonctions nécessaires au mouvement du robot. Des fonctions comme *moveTo()* ou *moveTowards()* prennent en argument des coordonnées spatiales. De cette facon, on récupérant les coordonnées de la cible avec le tracker, on peut faire bouger Nao en direction de la cible qu'il a précédemment détecté.

3 Nao Fetcher

3.1 Principe

L'objectif initial du projet était de mettre en place un programme où Nao aurait reconnu des objets de différentes formes et couleurs. Ces objets aurait été préalablement appris et hiérarchises. De cette façon, Nao aurait pour tâche de récupérer tous les objets qu'il a appris mais dans un ordre précis défini par la hiérarchisation.

3.2 Limites

Tout d'abord, cet idée ne peut être testé que sur un vrai Robot ou un robot virtuel dans un environnement virtuel. En effet, l'apprentissage, la reconnaissance et la récupération d'objet ne peut pas se faire avec un robot virtuel sans monde virtuel. Nous avons donc d'abord crée un monde virtuel avec le logiciel Webots et utilisé un robot virtuel avec Chorégraphe mais le résultat n'était pas satisfaisant.
Après avoir récupéré le vrai robot pour la suite du projet, nous avons commencé par lui faire apprendre des objets. C'est à ce moment que nous avons réalisé que la reconnaissance d'objet ne marchait pas à plus de 30 cm des yeux de Nao. Cette limite fut la principale raison pour laquelle nous avons décidé de rénover les objectifs du projet.
De plus, la saisie d'objet se serait certainement révélée difficile à mettre en place car il aurait fallu implémenter un programme gérant le multi-tâche entre la détection d'objet, le tracker, le mouvement et enfin la saisie.

4 Nao Messenger

4.1 Principe

Comme nous l'avons vu précédemment, notre proposition initiale de projet Nao Fetcher nous a confronté à un certain nombre de difficultés qu'il était difficile de contourner. Notre projet a donc évolué au fur et à mesure de sa réalisation pour devenir le Nao Messenger. Son principe est le suivant : le robot Nao se promène dans un couloir où les gens circulent. Il a pour mission de porter un, ou des messages à ses cibles. Il connaît évidemment les visages de ses cibles, mais aussi d'autres visages, et connaît également la liste des connaissances de chacune de ses cibles.

Le Nao robot est positionné dans le couloir, à la recherche de sa cible ou d'une de ses connaissances. Il sonde le visage des gens qui évoluent sur son chemin. Lorsqu'il reconnaît une de ses cibles ou une connaissance de ses cibles, il l'interpelle, puis se déplace jusqu'à elle pour lui délivrer son message, ou lui demander de transmettre le message à la cible.

Le robot Nao mesure un peu moins de 60 cm, et les caméras qui représentent la vue sont disposées sur son visage. Pour cette raison, il lui est difficile de détecter les visages des personnes qui seraient trop proches de lui, puisqu'il ne peut monter la tête que jusqu'à une hauteur limite. Pour ces raison, si nous devions améliorer encore notre projet, il serait plus intéressant de travailler sur le robot Pepper dans la mesure où celui-ci est plus grand.

4.2 Modèle BDI

Afin d'utiliser le robot en tant qu'agent appartenant à un système multi-agents nous avons décidé d'implémenter un modèle BDI au sein même de notre programme *Messenger*. Le modèle BDI correspond à Beliefs, Desires, Intentions et est utilisé pour programmer des agents intelligents. Les concepts de croyances, désirs et intentions sont utilisés par l'agent pour résoudre un problème particulier. Dans notre cas, le problème à résoudre est la transmission d'un ou plusieurs messages. Ici, nous allons définir en détail les concepts et leur correspondance dans notre programme.

Beliefs : Les croyances sont les informations que l'agent possède sur l'environnement et les autres agents existant de cet environnement. Les croyances ne

sont pas nécessairement des connaissances car elles peuvent s'avérer incorrectes, incertaines ou incomplètes. Dans notre cas, les croyances sont les visages que Nao a appris ainsi que leurs connaissances. De cette façon, lorsqu'il reconnaîtra un visage il pourra vérifier si c'est une connaissance de la cible ou la cible elle même. Les croyances sont sous forme de dictionnaire Python. Voici un exemple de croyances :

{'Marie' : ['Johnny'], 'Max' : ['Anna', 'Jake']}

Les cibles sont Marie et Max. Johnny est la connaissance de Marie tandis qu'Anna et Jake sont les connaissances de Max.

Desires : Les désirs d'un agent représentent les états que l'agent aimerait voir réalisés. Simplement, c'est la tâche que l'agent veut accomplir. Pour Nao Messenger,cette tâche est transmettre les différents messages aux cibles. Les désirs de l'agent Nao seront aussi stockés sous forme de dictionnaire Python. On y retrouvera un message direct ou indirect pour chaque cible. Le message direct est à délivrer dans le cas où Nao rencontre la cible elle-même alors que le message indirect doit être transmis s'il rencontre une de ses connaissances. Voici un exemple de désirs :

{'Anna' : ['L'équipe t'attend en salle de réunion', 'Dis à Anna que l'équipe l'attend en salle de réunion']}

Ici, il n'y a qu'une seule cible : Anna. Le premier message est le message direct et le deuxième est le message indirect.

Intentions : Les intentions d'un agent sont les actions que l'agent a décidé d'accomplir afin de combler un de ses désirs. Dans notre cas, l'intention est la mise en mouvement de Nao vers une cible ou une de ses connaissances afin de transmettre le message. La mise en mouvement est lancé lorsque Nao reconnaît une cible ou une de ses connaissances et utilise les données du tracker afin de se diriger vers la personne. Nao commencera la phase d'intention en interpellant la cible ou une de ses connaissances en appelant son nom et déclarant qu'il a un message à transmettre.

4.3 Système multi-agents

L'objectif de notre projet était, non seulement d'intégrer le modèle d'agent au sein du framework Nao, mais aussi d'y mêler les concepts de multi-agents, c'est-à-dire d'interaction avec l'environnement, et en particulier les interactions avec les autres agents, la *communication* et la *coopération* ou *compétition* avec ces autres agents.

Pour la réalisation de notre projet, nous n'avions à disposition qu'un seul robot : le Nao robot. Pour intégrer les notions de *multi-agents*, nous avons donc choisi de faire communiquer et coopérer le robot avec nous autres êtres humains. Ceci suit parfaitement les valeurs de l'entreprise SoftBank Robotics, responsable de la création du Nao robot, qui pense que les robots humanoïdes se doivent d'être bienveillants et coopérants avec les êtres humains. Le robot est donc en coopération, par la communication avec les êtres humains, et en aucun cas en compétition. Il sonde et cherche à reconnaître les visages des êtres humains qu'il croise en temps réel et agit de façon autonome lorsqu'il reconnaît une cible ou une connaissance.

4.4 Améliorations possibles

Dans un premier temps, l'on pourrait penser à améliorer encore la communication entre le Nao robot et les être humains qui l'entourent. Pour ce faire, il serait intéressant que le Nao robot puisse écouter et réagir aux messages que les êtres humains lui diraient. Ainsi, il serait possible de donner soi-même un message à Nao pour qu'il aille lui le porter à la cible que l'on aura désigné, ou à une de ses connaissances. Il serait aussi très intéressant de pouvoir modifier l'arbre de connaissances que le Nao robot possède sur chacun des êtres humains contenus dans sa base de données. Pour l'ajout de ces deux fonctionnalités, il faudrait abonner le Nao robot au module ALListener, l'abonner à un nouvel événement qui serait entendre un certain message déclencheur tel "Nao, porte le message suivant :". Dès que cet événement serait détecté, le Nao mettrait à jour ses bases de données et irait porter les messages demandés.

Dans un second temps, pour améliorer encore l'interaction de Nao avec son environnement, il serait possible de garder le robot tout le temps actif. En effet, pour les besoins de notre projet, le robot se désactive une fois qu'il a délivré le message à la personne. Nous avons jugé que cela était plus sûr, plutôt que de laisser le robot évoluer par lui même dans la salle de classe. Mais il serait possible de demander au robot de bouger et d'avancer aléatoirement dans le couloir

dans lequel il serait disposé, pour qu'il puisse sonder plus facilement l'ensemble des visages des personnes présentes, et de ne pas se désactiver après une seule exécution.

La dernière amélioration qu'il faudrait apporter à notre projet, la dernière mais pas la moindre, serait de lier le Nao à un ensemble de Nao robot interconnectés dans l'ensemble d'une entreprise. Chaque robot aurait une certaine base de données des personnes qui sont dans son entourage. Lorsque l'un d'entre eux devra délivrer un message, tous les robots communiqueront entre eux pour déterminer qui connaît le destinataire du message, et donc qui distribuera le message, si l'un des robots connaît l'une ou l'autre de ses connaissances etc ...

Conclusion

Difficultés rencontrées

Tout au long de la réalisation de notre projet, nous nous sommes confrontés à diverses difficultés. La documentation de l'API que nous utilisions était présente, bien écrite, mais il manquait parfois certaines explications pour faciliter notre compréhension des modules. Comme cette API est très peu utilisée, nous ne trouvions pas les réponses à nos questions dans les forums ni dans la documentation. De plus, nous avions un accès au robot assez limité : quelques heures par semaines, le lundi après-midi. Comme nous faisions des tests sur la reconnaissance de visage et la parole du Nao, nous ne pouvions pas tester notre travail sur le robot virtuel fourni par Choregraphe. Le temps que nous avions a disposition a donc largement limité notre avancée. Finalement, nous avons du changer en cours de réalisation d'objectif. Nous avons expliqué plus haut pourquoi il ne nous été pas possible de réaliser le projet Nao Fetcher, et pourquoi nous nous sommes tournés vers le projet Nao Messenger. Ce changement nous a pris un temps non négligeable. Cependant, si ce que nous avons appris lorsque notre objectif était le Nao Fetcher ne nous a pas servi, un de nos objectif était d'apprendre et de comprendre le framework NAOqi, et lorsque l'on considère cet objectif, l'on peut en conclure que le temps passé sur ce premier projet n'était en aucun cas du temps perdu.

Bilan du travail fourni

Tout au long de ce projet, nous nous sommes familiarisés avec le Nao robot et avons appris à utiliser le framework NAOqi, de même que nous avons compris le fonctionnement des différents outils que nous avions à disposition tel que Choregraphe, ou encore l'outil Webots que nous avons utilisé en début de projet pour l'abandonner ensuite au profit du véritable robot. Nous avons démontré, grâce à notre implémentation, qu'il était possible d'intégrer le patron multi-agents à l'intérieur du Nao robot, en intégrant le modèle d'agent avec la représentation mentale BDI, et en intégrant la communication du robot avec les autres agents de l'environnement : les êtres humains avec lesquels il doit interagir.

Références

[1] Softbank Robotics anciennement ALDEBARAN : Key concepts. http ://doc.aldebaran.com/2-1/dev/naoqi/index.html.

[2] Softbank Robotics anciennement ALDEBARAN : NAOQI documentation, FaceDetection. http ://doc.aldebaran.com/1-14/naoqi/vision/alfacedetection.html.

[3] Softbank Robotics anciennement ALDEBARAN : NAOQI documentation, exemple pour FaceDetection. http ://doc.aldebaran.com/2-1/dev/python/examples/vision/face_detection.html.