



ArCo : une Architecture informatique pour un Compagnon Artificiel en interaction avec un utilisateur

Céline Jost

► To cite this version:

Céline Jost. ArCo : une Architecture informatique pour un Compagnon Artificiel en interaction avec un utilisateur. Robotique [cs.RO]. Université de Bretagne Sud, 2013. Français. NNT : . tel-00861311v1

HAL Id: tel-00861311

<https://theses.hal.science/tel-00861311v1>

Submitted on 12 Sep 2013 (v1), last revised 16 Sep 2013 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THESE / UNIVERSITE DE BRETAGNE SUD

sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITE DE BRETAGNE SUD

Mention : Science et Technologies de l'information et de la Communication

Ecole doctorale SICMA

présentée par

Céline Jost

Laboratoire des Sciences Techniques de l'Information, de la
Communication et de la Connaissance UMR3192

ArCo : une Architecture informatique pour un Compagnon Artificiel en interaction avec un utilisateur

Thèse soutenue le 8 janvier 2013

devant le jury composé de :

Jacques Tisseau

Pr Lab-STICC – ENI de Brest / *président*

Sylvie Pesty

Pr LIG – Université Pierre Mendès France / *rapporteur*

Jean-Claude Martin

Pr LIMSI – Université de Paris Sud / *rapporteur*

Brigitte Le Pévédic

McF Lab-STICC – Université de Bretagne Sud / *encadrante de thèse*

Dominique Duhaut

Pr Lab-STICC - Université de Bretagne Sud / *directeur de thèse*

Ce document PDF contient :

- La thèse, de la page 2 à 152 du document PDF
- Les annexes de la thèse, de la page 153 à 279 du document PDF

Ce document ne doit pas être déstructuré. Les annexes ne doivent pas être séparées de la thèse.



THESE / UNIVERSITE DE BRETAGNE SUD
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITE DE BRETAGNE SUD

Mention : Science et Technologies de l'information et de la Communication
Ecole doctorale SICMA

présentée par

Céline Jost

Laboratoire des Sciences Techniques de l'Information, de la
Communication et de la Connaissance UMR3192

ArCo : une Architecture informatique pour un **Compagnon Artificiel** en interaction avec un utilisateur

Thèse soutenue le 8 janvier 2013
devant le jury composé de :

Jacques Tisseau

Pr Lab-STICC – ENI de Brest / *président*

Sylvie Pesty

Pr LIG – Université Pierre Mendès France / *rapporteur*

Jean-Claude Martin

Pr LIMSI – Université de Paris Sud / *rapporteur*

Brigitte Le Pévédic

McF Lab-STICC – Université de Bretagne Sud / *encadrante de thèse*

Dominique Duhaut

Pr Lab-STICC - Université de Bretagne Sud / *directeur de thèse*

N° d'ordre 285

Remerciements

Merci à Dominique Duhaut, Professeur des Universités et Brigitte Le Pévédic, Maître de Conférences de m'avoir encadré et d'avoir dirigé mes recherches. Je les remercie de m'avoir permis d'effectuer ce doctorat dans les meilleures conditions possibles. Je les remercie également pour leur grande disponibilité et leur gentillesse. Leurs conseils m'ont permis d'avancer au mieux et de m'éclairer sur un chemin parfois sombre. Et je leur remercie pour leur patience légendaire qui m'a été fort utile.

Merci à mes deux rapporteurs : Madame Sylvie Pesty, Professeur des Universités à Grenoble et Monsieur Jean-Claude Martin, Professeur des Universités à Paris pour leurs remarques avisées.

Merci à Monsieur Jacques Tisseau, Professeur des Universités à Brest pour avoir accepté de participer à mon jury de thèse.

Merci à toutes les personnes qui ont accepté de participer aux expérimentations : directeurs d'écoles, instituteurs, parents et élèves des écoles de Rennes et de Vannes qui nous ont reçus.

Merci à tous mes anciens collègues du Valoria et de l'IUT pour leur bonne humeur et merci à tous mes nouveaux collègues de Brest pour leur joie de vivre, leurs blagues et les bonnes conditions de travail qu'ils m'ont fournies pour finir ma thèse.

Merci à mes parents qui ont su me soutenir et me motiver à aller toujours plus loin. Sans leur soutien, je n'écrirais pas ces lignes aujourd'hui.

Merci à Marine qui a été mon maître Jedi depuis ce magnifique mois d'octobre 2010 où elle a débarqué dans mon bureau. Je lui dois beaucoup et aucun mot ne saurait l'exprimer.

Dans un même registre, merci à Sébastien qui m'a beaucoup conseillé dès mes débuts et qui m'a appris à regarder dans la bonne direction. Je lui dois le bon esprit dans lequel j'ai évolué chaque jour. Et merci à Germain pour sa gentillesse, sa bonne volonté et l'aide qu'il m'a apportée.

Merci à Thibaut et Djamel pour avoir partagé mes coups de gueules et mes joies. Les pauses café ne sont plus les mêmes sans vous.

Merci à Christine, « la dame de la BU » qui a été d'une aide précieuse, toujours rapide, toujours souriante. Je souhaite des Christine à tous les thésards.

Merci à mes amis de m'avoir épaulé autant que possible et de ne jamais m'avoir laissé tomber.

Merci à toutes ces personnes que j'ai croisées ne serait-ce qu'une minute, une heure, un jour, un mois ou plus. Chaque rencontre, chaque sourire, chaque mot m'a apporté quelque chose.

Merci à toi qui me supporte chaque jour, avec un grand courage, parce que ce n'est pas facile de vivre avec quelqu'un qui fait une thèse.

Merci à Bina qui n'hésite jamais à venir me câliner même quand je n'ai pas le temps. Merci de venir t'allonger sur mon clavier et m'empêcher de travailler. Merci de me griffer quand j'ose te décaler un peu. Merci de m'avoir fait perdre beaucoup de temps mais toujours avec des ronrons alors au final c'était bien agréable quand même.

Table des matières

TABLE DES MATIERES	5
TABLE DES FIGURES.....	6
TABLE DES TABLEAUX.....	8
INTRODUCTION.....	9
CHAPITRE I. LE COMPAGNON ARTIFICIEL.....	13
A. DEFINITIONS	15
1. <i>Les architectures informatiques</i>	15
2. <i>L'interaction homme-machine</i>	15
3. <i>Le Compagnon Artificiel</i>	16
B. CONTEXTE : LE PROJET ROBADOM	17
1. <i>Importance de l'autonomie à domicile</i>	17
2. <i>Importance de la technologie pour l'autonomie à domicile</i>	17
3. <i>Objectifs du projet</i>	17
4. <i>Axes étudiés</i>	18
5. <i>Participation des partenaires</i>	19
C. INTERACTIONS DU COMPAGNON ARTIFICIEL	20
1. <i>Rôle de l'homme dans le Compagnon Artificiel</i>	20
2. <i>Environnement ambiant du Compagnon Artificiel</i>	21
CHAPITRE II. ETAT DE L'ART	23
A. COMPAGNONS LOGICIELS	25
B. COMPAGNONS ROBOTIQUES	31
1. <i>Introduction</i>	31
2. <i>Architectures</i>	37
C. COMPAGNONS VIRTUELS	47
D. INTELLIGENCE AMBIANTE	54
1. <i>Origines de l'intelligence ambiante</i>	54
2. <i>Définitions de l'intelligence ambiante</i>	56
3. <i>Problématiques pertinentes</i>	59
4. <i>Applications</i>	60
5. <i>Perspectives de recherche en intelligence ambiante</i>	63
E. DISCUSSION	64
CHAPITRE III. ARCHITECTURE ARCO : UNE REPONSE POUR LE COMPAGNON ARTIFICIEL	69
A. VOCABULAIRE	71
B. PRESENTATION D'ARCO	72
C. MICE : CADRICIEL POUR LA GESTION D'UN ENVIRONNEMENT MODULAIRE	77
1. <i>Modules et communication</i>	77
2. <i>Données gérées par le serveur</i>	79
3. <i>Protocole de communication</i>	82
4. <i>AmbiPath : Module pour la création du chemin logique</i>	85
D. ENTITES AGISSANTES : MODULES D'ENTREES ET SORTIES	86
E. AMBIPROG : OUTIL POUR LA CREATION DE SCENARIO.....	88
1. <i>Scénario d'interaction</i>	88
2. <i>Configuration d'AmbiProg : un exemple</i>	89
3. <i>Présentation générale</i>	92
4. <i>Grammaire du langage généré par AmbiProg</i>	93
5. <i>Exemple de scénario</i>	94
F. AMBILIVE : MODULE D'INTERPRETATION DES SCENARII	95
1. <i>Gestion des entrées et sorties du module d'interprétation</i>	96
2. <i>Interprétation des perceptions symboliques</i>	97

Table des Figures

G.	AMBI COP : MODULE D'ARBITRAGE DES ACTIONS.....	98
1.	Organisation générale d'AmbiCop.....	98
2.	Gestion des conflits.....	99
3.	Gestion des actions génériques.....	101
4.	Synchronisation des actions.....	103
CHAPITRE IV. STIMCARDS : UNE APPLICATION D'ARCO		105
A.	STIMCARDS : UN JEU DANS LE CADRE DE LA STIMULATION COGNITIVE	107
B.	STIMCARDS : SON FONCTIONNEMENT	109
1.	Présentation	109
2.	Modules utilisés	111
C.	STIMCARDS : UN EXEMPLE.....	113
CHAPITRE V. DIFFERENTES EXPERIMENTATIONS INTERACTIVES MENEES AVEC ARCO		119
A.	ETUDE SUR LA GESTUELLE D'UN ROBOT.....	122
B.	EVALUATION D'AMBI PROG	124
C.	EVALUATION DE STIMCARDS (ENFANTS ET PERSONNES AGEES).....	126
D.	EVALUATION DU MEILLEUR PARTENAIRE NUMERIQUE POUR STIMCARDS	128
E.	EVALUATION D'AMBI COP	130
CONCLUSION.....		133
REFERENCES		137
PUBLICATIONS		150

Table des Figures

FIGURE 1 : ROBOT KOMPAÏ DE ROBOSOFT	19
FIGURE 2 : UNE PERSONNE ET SON COMPAGNON ARTIFICIEL	20
FIGURE 3 : MODELE BDI	26
FIGURE 4 : ARCHITECTURE BAGHERA POUR DEUX UTILISATEURS CONNECTES.....	27
FIGURE 5 : ARCHITECTURE SEMAINE	28
FIGURE 6 : ARCHITECTURE DU MODELE DE PERSONNALITE BASE SUR L'ENNEAGRAMME	29
FIGURE 7 : ARCHITECTURE UBI-LEARN	30
FIGURE 8 : PANORAMA D'UN ENSEMBLE DES ROBOTS EXISTANTS	32
FIGURE 9 : ROBOTS DE SERVICES	33
FIGURE 10 : ROBOTS ANIMALOÏDES	34
FIGURE 11 : ROBOTS IMAGINAIRES.....	35
FIGURE 12 : ROBOTS HUMANOIDS	36
FIGURE 13 : ARCHITECTURE PRS	37
FIGURE 14 : ARCHITECTURE DE MAGGIE.....	38
FIGURE 15 : ARCHITECTURE BERRA.....	39
FIGURE 16 : ARCHITECTURE ATLANTIS.....	40
FIGURE 17 : ARCHITECTURE DE BIRON.....	41
FIGURE 18 : ARCHITECTURE DE ROBOVIE	42
FIGURE 19 : ARCHITECTURE TAME	43
FIGURE 20 : ARCHITECTURE IGRACE	44
FIGURE 21 : UNE ARCHITECTURE AFFECTIVE GENERALE	45
FIGURE 22 : ARCHITECTURE EGO	46
FIGURE 23 : AGENT VIRTUEL DIVA.....	47
FIGURE 24 : AGENT VIRTUEL HERMAN THE BUG	47
FIGURE 25 : AGENT PAT	48
FIGURE 26 : À GAUCHE, LE ROBOT MUU2. À DROITE, REPRESENTATION VISUEL DE MUU2	48
FIGURE 27 : ROBOT ET REPRESENTATION VISUELLE CHERRY	48

FIGURE 28 : ROBOT VALERIE ET SA TETE VIRTUELLE.....	49
FIGURE 29 : AGENT VIRTUEL MAX.....	49
FIGURE 30 : AGENTS GRETA ET SASO'S SMARTBODY.....	50
FIGURE 31 : ARCHITECTURE DE SAIBA.....	50
FIGURE 32 : ARCHITECTURE DE GRETA	51
FIGURE 33 : PLATEFORME D'AGENTS EMOTIONNELS MARC.....	51
FIGURE 34 : ARCHITECTURE MULTI-USER VIRTUAL ENVIRONMENT.....	52
FIGURE 35 : ARCHITECTURE D'UN ASSISTING CONVERSATIONAL AGENT.....	53
FIGURE 36 : ROBOT MAMORU	66
FIGURE 37 : ORGANISATION DE L'ARCHITECTURE ARCo POUR LE COMPAGNON ARTIFICIEL.....	71
FIGURE 38 : PRESENTATION GENERALE DU COMPAGNON ARTIFICIEL.....	72
FIGURE 39 : MODULES PRINCIPAUX D'ARCo	73
FIGURE 40 : ORGANISATION GENERALE D'ARCo AVEC LE CADRICIEL MICE	75
FIGURE 41 : ORGANISATION GENERALE DE MICE.....	78
FIGURE 42 : ORGANISATION D'UN MODULE DE MICE	78
FIGURE 43 : FICHIERS STOCKES PAR LE SERVEUR	79
FIGURE 44 : LIEN LOGIQUE ENTRE LES MODULES DE L'EXEMPLE	85
FIGURE 45 : PERCEPTIONS ET ACTIONS DANS ARCo	86
FIGURE 46 : ORGANISATION D'AMBIProg	92
FIGURE 47 : SCENARIO VISUEL DE LA MACRO "SURVEILLANCE_CHUTE"	94
FIGURE 48 : SCENARIO VISUEL SIMPLE.....	94
FIGURE 49 : ORGANISATION GENERALE DE L'INTERPRETEUR	95
FIGURE 50 : ORGANISATION GENERALE D'AMBIcOP	98
FIGURE 51 : EXEMPLE DE GESTION DE PRIORITE DE SCENARIO	100
FIGURE 52 : ACTIONS GENERIQUES DANS AMBIProg.....	101
FIGURE 53 : EXEMPLE DE GESTION D'UNE ACTION GENERIQUE	102
FIGURE 54 : EXEMPLE D'UTILISATION DE STIMCARDS	109
FIGURE 55 : AFFICHAGE DE STIMCARDS EN FONCTION DU TYPE DE LA QUESTION	110
FIGURE 56 : STIMCARDS AFFICHANT UNE QUESTION A CHOIX MULTIPLES	110
FIGURE 57 : EXEMPLE DE CHEMIN LOGIQUE DE STIMCARDS.....	114
FIGURE 58 : SCENARIO DE STIMCARDS - PARTIE 01	115
FIGURE 59 : SCENARIO DE STIMCARDS - PARTIE 02	115
FIGURE 60 : SCENARIO DE STIMCARDS - PARTIE 03	115
FIGURE 61 : SCENARIO DE STIMCARDS - PARTIE 04	116
FIGURE 62 : SCENARIO DE STIMCARDS - PARTIE 05	116
FIGURE 63 : SCENARIO DE STIMCARDS - PARTIE 06	116
FIGURE 64 : SCENARIO DE STIMCARDS - PARTIE 07	117
FIGURE 65 : SCENARIO DE STIMCARDS - PARTIE 08	117
FIGURE 66 : MODULES DE LA PREMIERE EVALUATION DE LA GESTUELLE DU ROBOT	123
FIGURE 67 : MODULES DE LA DEUXIEME EVALUATION DE LA GESTUELLE DU ROBOT	123
FIGURE 68 : MODULES DE LA DEUXIEME EVALUATION D'AMBIProg	125
FIGURE 69 : MODULES DE L'EVALUATION DE STIMCARDS AVEC LES PERSONNES AGEES	127
FIGURE 70 : MODULES DE L'EVALUATION DE STIMCARDS AVEC LES ENFANTS	129
FIGURE 71 : MODULES DE L'EVALUATION D'AMBIcOP	131
FIGURE 72 : COOPERATION AVEC LE LIG - INTEGRATION D'UN AGENT VIRTUEL DANS ARCo	132
FIGURE 73 : ARCHITECTURE ARCo	133

Table des Tableaux

TABLEAU 1 : POSITIONNEMENT DE L'INTELLIGENCE AMBIANTE, SELON AARTS ET RUYTER	58
TABLEAU 2 : SYSTEM INTELLIGENCE VERSUS SOCIAL INTELLIGENCE, SELON AARTS ET RUYTER	58
TABLEAU 3 : ETUDES SUR LES COMPARAISONS ENTRE LES EFFETS DU REEL ET DU VIRTUEL.....	65
TABLEAU 4 : CAPACITES DES GROUPES DE ROBOTS	66
TABLEAU 5 : LISTE DES MESSAGES MICE RESERVES.....	84
TABLEAU 6 : EXEMPLE DE LISTE D'ATTENTE D'UN ENTITY MANAGER.....	100
TABLEAU 7 : LES JEUX COGNITIFS ACTUELS	108

Introduction

Aujourd'hui, la technologie s'impose dans notre quotidien et l'améliore. L'éclairage à la bougie et le chauffage au feu de bois ont été supplantés par de nouveaux dispositifs comme la lampe et le chauffage électrique. Notre mode de vie a été enrichi progressivement. Aujourd'hui, nous utilisons tout un ensemble d'objets numériques : caméra, microphone, téléphone mobile, GPS, tablettes tactiles, liseuses numériques, robots, télévisions, appareils ménagers, objets domotiques...

Le progrès de la technologie ne se limite pas au matériel et à ses fonctionnalités. Depuis quelques années, le nombre de personnages virtuels qui envahissent nos écrans augmente. Par exemple, des conseillers virtuels sont mis en place sur des sites web comme EDF ou Ameli. Ces conseillers virtuels sont des « Chatbot » et permettent de répondre aux questions les plus fréquentes grâce à une interface textuelle. Il existe des personnages plus évolués, qui sont animés et qui peuvent aider les internautes. Par exemple, on trouve *Inès* de Nespresso, *Michelin Man* de Michelin, *Emilie* de Numéricable ou *Voki*, un personnage proposé aux concepteurs de site web. Les personnages virtuels émergent également dans les jeux vidéo, les solutions de formation, les visites virtuelles, etc.

D'autres personnages, cette fois réels, connaissent aussi une grande évolution depuis ces dernières années, ce sont les robots. Ils sont utilisés pour rendre des services comme *Roomba* l'aspirateur « intelligent », *Rovio* le robot caméra de surveillance ou *Jazz* un robot de téléprésence. Et dernièrement, l'univers du jouet s'y intéresse pour les émotions qu'ils sont capables de provoquer. L'entreprise Woowee propose par exemple des robots humanoïdes télécommandés, des dinosaures, ou encore un phoque ressemblant à un animal réel.

La « course au meilleur » personnage virtuel ou robotique a été lancée depuis plusieurs années. La compétition *RoboCup@Home* permet de déterminer « le meilleur robot de service autonome » et la compétition *Laval Virtual Awards* « récompense les meilleurs projets de réalité virtuelle et des technologies convergentes ». La compétition existe même entre les personnages virtuels et les robots. Leur impact sur les humains est étudié. Et des expérimentations comparent la préférence des utilisateurs pour l'un ou l'autre des personnages. Mais les résultats n'aboutissent à aucun consensus. Ni les personnages réels, ni les personnages virtuels ne s'imposent. Il semble que pour certaines tâches, il soit préférable d'interagir avec un interlocuteur virtuel et dans d'autres cas, avec un interlocuteur réel. Chacun d'entre eux serait le plus efficace dans des rôles différents. Il n'existe pas de personnage réel ou virtuel qui puisse répondre à toutes les attentes d'un utilisateur.

Nous partons donc du constat suivant : il existe différents objets numériques qui ont des rôles différents les uns des autres. A défaut d'avoir un objet polyvalent, l'utilisateur n'a pas d'autres choix que de posséder plusieurs objets en fonction de ses besoins. En effet, on peut imaginer qu'un humain, qui effectue des tâches variées dans son quotidien, utilise différents objets réels ou virtuels. Cela peut rapidement poser des problèmes de synchronisation ou d'organisation.

Prenons l'exemple d'un homme qui possède un assistant virtuel au travail et un robot d'aide à domicile chez lui. Pendant la journée de travail, l'assistant virtuel récolte un ensemble de données sur l'homme qui ne sont pas accessibles au robot. Et si l'homme demande à son assistant virtuel d'enregistrer un rendez-vous privé, ce n'est pas possible car ce dernier n'a pas accès à son agenda privé, géré par le robot à domicile. Ceci est un exemple de problème de synchronisation.

En utilisant deux objets numériques différents, l'utilisateur peut avoir à répéter des informations ou choisir à quel objet les informations doivent être envoyées (par exemple la prise d'un rendez-vous privé). Et si l'utilisateur possède 20 objets différents, cela augmente

considérablement la difficulté : comment s'utilise chacun des objets, quels informations leur envoyer, quel objet gère quoi... Ceci est un exemple de problème d'organisation.

Il est possible de diminuer ces deux types de problèmes si les objets numériques sont connectés entre eux et gèrent eux-mêmes la répartition des informations. Il devient même possible d'en tirer un certain confort si l'ensemble des fonctionnalités peut être exploité en dehors de l'utilisation des objets. Par exemple, si l'utilisateur boit un café chaque matin, la machine à café pourrait se lancer lorsque le réveil sonne. Cette vision des choses implique que les objets numériques partagent la même base de données d'informations (la même connaissance) et qu'ils puissent communiquer et se coordonner les uns avec les autres.

Une question qui se pose est de savoir si ces objets doivent effectivement être coordonnés afin de simplifier leur utilisation et de disposer de toutes les fonctionnalités existantes. Des questions se posent alors : comment s'effectue cette coordination ? Qui arbitre et décide le comportement des objets numériques ? Quelle est la légitimité de telle ou telle action des objets numériques ? Enfin, quelle place l'utilisateur a-t-il dans ce système ? Ce manuscrit propose une architecture informatique qui permet de mettre en place cette coordination des objets numériques.

Ce travail s'inscrit dans le projet pluridisciplinaire Robadom¹ dont l'objectif est d'étudier l'impact d'un robot d'aide à domicile sur l'état psychoaffectif et cognitif de personnes âgées ayant des troubles cognitifs légers. Cet impact est étudié lors d'exercices de stimulation cognitive. Des robots d'aide à domicile ont déjà été développés dans le cadre d'autres projets. On trouve, par exemple, le robot *Nursebot* (Pollack et al. 2002), un robot d'assistance pour les personnes âgées issu d'un projet américain ou alors *Hector* (Gross et al. 2011), le robot du projet Européen *CompanionAble*, conçu pour aider les personnages âgés qui souffrent d'un handicap mental. D'autres projets utilisent des robots existants pour étudier des problématiques d'interaction homme-machine. Par exemple, le robot *Kompaï*² est utilisé par le projet européen *MobiServ* (Nani et al. 2010) pour surveiller la santé d'un utilisateur. Le robot *Nao*³ est, quant à lui, utilisé par le projet européen *ALIZ-E* (Baxter et al. 2011) pour créer une relation à long terme avec les humains. La création d'une relation à long terme n'est pas un problème simple et plusieurs autres projets tentent de le résoudre, comme les projets européens SERA (Hudlicka et al. 2009) et LIREC (G. Castellano et al. 2008).

Le projet Robadom est différent car le robot utilise un certain nombre d'objets numériques pour fonctionner : des capteurs qui obtiennent des informations sur la personne et son environnement et des actionneurs qui peuvent agir sur cet environnement. Le robot est d'ailleurs un des éléments de l'interaction, mais pas le seul. Pour simplifier la communication entre l'utilisateur et les objets numériques, toutes les fonctionnalités des capteurs et des actionneurs sont uniformisées et rendues disponibles. L'utilisateur peut alors piloter son environnement en utilisant ces fonctionnalités. Il interagit alors avec son Compagnon Artificiel, qui est une entité abstraite capable de répondre à ses besoins en utilisant les objets numériques nécessaires.

Deux catégories de questions se posent alors. Premièrement comment fonctionne ce Compagnon Artificiel ? Un objet numérique peut tomber en panne ou être momentanément indisponible. Par exemple, si la personne se trouve dans le salon, alors la télévision de la cuisine est inutile. Ou alors il peut y avoir une coupure de réseau qui rende impossible la communication avec un objet numérique. Comment le Compagnon Artificiel s'adapte-t-il à cette présence ou absence ?

¹ Projet TecSan ANR : ANR-09-TECS-01202. Responsable du projet : Pr Anne-Sophie Rigaud (hôpital Broca)

² Le robot de service Kompaï est construit par la société française Robosoft. Il est utilisé dans différents projets de robots d'aide à domicile.

³ Le robot humanoïde Nao est construit par la société française Aldebaran. Il est actuellement proposé pour la recherche et l'enseignement.

Secondement, comment configurer le Compagnon Artificiel ? Comment permettre à la personne, à son entourage ou à son médecin de déterminer le comportement du Compagnon Artificiel ?

Pour affiner cette problématique, la question que nous adressons maintenant est : comment construire une architecture informatique pour assurer l'activité du Compagnon Artificiel ? L'architecture informatique doit permettre à la personne, son entourage ou son médecin de configurer le Compagnon Artificiel. Ces personnes sont appelées *prestataires* car elles permettent l'ajout de services dans l'environnement. Il n'est pas concevable de leur demander des compétences en programmation informatique pour hiérarchiser les actions que doivent effectuer les objets numériques. L'architecture informatique doit fournir un outil qui permette aux prestataires de configurer facilement le Compagnon Artificiel. Mais que doit faire cet outil pour permettre de mettre en place une interaction entre les objets numériques et l'utilisateur ?

Pour résumer, le Compagnon Artificiel doit permettre de coordonner des objets numériques avec un outil que des prestataires peuvent utiliser. Ce type de système est soumis à un certain nombre de contraintes : maintenir la sécurité des informations et des utilisateurs, permettre et optimiser la communication entre les objets numériques, disposer d'un protocole permettant l'interopérabilité entre les dispositifs, gérer la présence et l'absence des dispositifs, fournir des fonctionnalités permettant de gérer l'interaction avec les humains... Il existe également des propriétés d'évolution. Comment, dans le cadre d'une interaction à long terme, le Compagnon Artificiel peut-il évoluer avec la personne en fonction de ses préférences et de ses exigences ?

Dans le travail présenté ici, nous nous limitons à l'étude de deux types de propriétés :

- **Structurelles** : le Compagnon Artificiel pilote un environnement dynamique. Lorsque l'humain est à son travail ou à son domicile, le Compagnon Artificiel ne gère pas toujours les mêmes objets numériques. Comment le Compagnon Artificiel doit-il gérer la dynamique du système ?
- **Fonctionnelles** : Quels sont les outils nécessaires pour permettre aux prestataires d'utiliser le Compagnon Artificiel ? Comment le Compagnon Artificiel interagit avec les objets numériques ?

D'un point de vue structurel, nous nous intéressons à trois notions :

- **Dynamique** du système : tous les objets numériques ne sont pas toujours disponibles : ils peuvent être en panne, déjà utilisés ou éteints. Comment le Compagnon Artificiel s'adapte-t-il au nomadisme de la personne ? Comment gérer la présence et l'absence des objets numériques connus de façon dynamique (informatique ambiante) ?
- **Adaptabilité** du système : le système doit prendre en compte des objets numériques qui n'existaient pas dans l'environnement avant. Comment intègre-t-il ces objets inconnus de façon transparente ?
- **Conception** : comment permettre à des concepteurs de créer de nouveaux capteurs et actionneurs compatibles au Compagnon Artificiel le plus simplement possible ?

D'un point de vue fonctionnel, nous nous intéressons à deux notions :

- **Conflit** : il ne faut pas qu'un objet numérique ait à effectuer deux actions, utilisant les mêmes ressources, en même temps (conflits). Par exemple, on ne peut pas demander à un personnage virtuel de sourire ou d'afficher un visage neutre en même temps. Comment l'architecture informatique détecte-t-elle et gère-t-elle un conflit ?

- **Prestataire** : des personnes non expertes en informatique doivent pouvoir personnaliser et utiliser le Compagnon Artificiel. Comment permettre à ces personnes de le faire de façon simple ? Comment l'ensemble des objets numériques est-il configuré ?

Ce manuscrit propose une solution à ces cinq notions.

Le Chapitre I introduit la notion de Compagnon Artificiel. A travers cette présentation destinée à un public large, les termes « architecture informatique », « interaction homme-machine » et « Compagnon Artificiel » sont définis. Elle est suivie par une présentation du contexte de ce travail : le projet Robadom. Et enfin, les acteurs et l'environnement du Compagnon Artificiel sont décrits.

Le chapitre II présente l'état de l'art dans lequel l'architecture informatique proposée y trouve son fondement scientifique. Il est constitué de trois parties. La première partie propose un classement des architectures existantes et évalue leur pertinence par rapport à nos propriétés structurelles et fonctionnelles. La deuxième partie présente les acteurs de l'interaction qui sont des objets numériques capables d'interagir avec l'homme, en se focalisant sur les robots et les personnages virtuels⁴. Enfin, la dernière partie présente l'intelligence ambiante et s'intéresse à la résolution des propriétés structurelles. Cet état de l'art montre qu'il n'existe pas de solution aux propriétés structurelles et fonctionnelles du Compagnon Artificiel et qu'il est donc nécessaire de créer une architecture informatique pour gérer le Compagnon Artificiel.

Le Chapitre III présente une solution conforme à nos propriétés structurelles et fonctionnelles. Il définit le vocabulaire utilisé et présente l'architecture ArCo, pour le Compagnon Artificiel. Cette architecture est mise en place grâce à un cadriciel⁵ MICE qui permet de gérer les notions de dynamique, d'adaptabilité et de conception. Il permet également aux développeurs d'enrichir l'architecture ArCo. Enfin ce chapitre présente une interface de programmation AmbiProg permettant aux prestataires de configurer le Compagnon Artificiel pour un utilisateur, ainsi que deux modules AmbiLive et AmbiCop qui détectent et gèrent les conflits.

Le Chapitre IV présente notre contribution pour le projet Robadom : StimCards, un jeu de stimulation cognitive créé avec l'architecture ArCo.

Le Chapitre V fait la synthèse⁶ des évaluations qui ont été menées grâce à l'architecture ArCo. Elles ont permis de valider expérimentalement le fonctionnement d'ArCo et de répondre à des problématiques d'interaction homme-machine.

⁴ Nous ne présentons que des objets numériques ayant été utilisés dans le cadre de travaux de recherche.

⁵ Le mot cadriciel est la traduction du terme *Framework* proposé par l'Office québécois de la langue française. Il représente un ensemble de composants logiciels permettant de faciliter la conception des logiciels.

⁶ Une présentation détaillée est disponible dans un document annexé à ce manuscrit.

Chapitre I.

Le Compagnon Artificiel

CHAPITRE I.	LE COMPAGNON ARTIFICIEL	13
A.	DEFINITIONS	15
1.	<i>Les architectures informatiques</i>	15
2.	<i>L'interaction homme-machine</i>	15
3.	<i>Le Compagnon Artificiel</i>	16
B.	CONTEXTE : LE PROJET ROBADM	17
1.	<i>Importance de l'autonomie à domicile</i>	17
2.	<i>Importance de la technologie pour l'autonomie à domicile</i>	17
3.	<i>Objectifs du projet</i>	17
4.	<i>Axes étudiés</i>	18
5.	<i>Participation des partenaires</i>	19
C.	INTERACTIONS DU COMPAGNON ARTIFICIEL	20
1.	<i>Rôle de l'homme dans le Compagnon Artificiel</i>	20
2.	<i>Environnement ambiant du Compagnon Artificiel</i>	21

La notion de Compagnon Artificiel est récente et ne possède pas encore de définition standard. Pour éviter toute ambiguïté, il nous a donc semblé important d'explicitier ce terme ainsi que les problèmes qui en découlent. Pour cela, le Chapitre I.A propose une explication des termes du sujet : « Une **architecture informatique** pour un **Compagnon Artificiel** en **interaction** avec un utilisateur ». Le 0 présente le projet Robadm dans lequel ce travail s'inscrit. Enfin, le Chapitre I.C écrit les différents acteurs et l'environnement du Compagnon Artificiel et les problèmes liés, que ce soit d'un point de vue technique, conceptuel ou éthique.

A. Définitions

L'objectif du travail présenté ici est de construire une architecture informatique capable de créer une interaction homme-machine « naturelle⁷ » avec un ensemble d'objets numériques gérés par le Compagnon Artificiel d'une personne.

Pour atteindre cet objectif, nous nous appuyons sur trois notions importantes : les architectures informatiques, l'interaction homme-machine et le Compagnon Artificiel.

1. Les architectures informatiques

Elles représentent la structure générale des systèmes informatiques. Chaque domaine de l'informatique propose un panorama riche d'architectures répondant à des problèmes variés. Pour optimiser les systèmes informatiques, elles sont souvent spécialisées dans un domaine ou une tâche précise. Pour permettre une interaction avec un Compagnon Artificiel, il faut déterminer l'architecture qui est la mieux adaptée à ce type d'application. La spécification de cette architecture nécessite de bien comprendre ce qu'est une interaction avec un Compagnon Artificiel.

2. L'interaction homme-machine

Le domaine de l'interaction homme-machine⁸ a beaucoup évolué depuis 40 ans. A l'origine, les ordinateurs n'étaient que des outils pour les scientifiques utilisés en tant que supercalculateurs. L'apparition de la micro-informatique a donné un sens nouveau aux ordinateurs et a permis une émergence de l'informatique dans tous les foyers.

Mais pour permettre d'intéresser un nombre maximum de personnes, les systèmes informatiques se sont adaptés aux utilisateurs. Ils sont devenus plus faciles et plus agréables à utiliser. Ce travail sur l'interaction entre l'homme et la machine a fait naître des environnements graphiques comme Xerox, Mac OS® ou Windows® ; des jeux vidéo utilisés par des millions de personnes ; des technologies permettant d'interagir de façon plus « naturelle » comme la Kinect™. Aujourd'hui, les ordinateurs ne sont plus les seuls périphériques informatiques existants. L'informatique est présente en tous lieux, et l'homme interagit chaque jour avec elle : dans sa voiture, à son domicile, dans les ascenseurs...

L'interaction homme-machine est un domaine vaste et complexe. C'est également un domaine pluridisciplinaire. Les sciences humaines étudient les utilisateurs (comportements, usages, besoins...). L'électronique construit des systèmes de plus en plus puissants. L'ergonomie travaille sur l'acceptabilité et l'utilisabilité des systèmes. Beaucoup de domaines apportent leur pierre à l'édifice. L'informatique utilise ces données pour proposer des systèmes adaptés à l'être humain et qui respectent les contraintes définies par les autres domaines.

Aujourd'hui, l'interaction homme-machine se doit d'être naturelle. En effet, l'homme souhaite pouvoir communiquer avec ses propres canaux de communication rendant la technologie la plus transparente possible. Cela permet aux utilisateurs de limiter le temps d'apprentissage de chaque nouvel objet avec lequel il doit interagir car ces objets sont nombreux, de natures et de formes différentes.

⁷ Le terme « naturel » se rapporte à l'humain qui évalue les interactions en fonction de ses propres canaux de communication (voie de circulation des messages). On qualifie de « naturelle » une interaction qui est effectuée avec la voix, la gestuelle... sans avoir à utiliser d'objets numériques.

⁸ L'interaction homme-machine est l'ensemble des moyens mis en œuvre pour qu'un humain puisse contrôler et communiquer avec une machine, que ce soit un ordinateur, un personnage virtuel, un robot, etc.

3. Le Compagnon Artificiel

La notion de Compagnon Artificiel n'a pas la même origine que les deux autres (architecture informatique et interaction homme-machine). Elle n'appartient pas au monde de l'informatique. L'intégrer au domaine de l'interaction homme-machine demande de s'intéresser à sa définition afin de pouvoir l'adapter. D'après Larousse (2012), un compagnon est celui qui accompagne quelqu'un ; qui partage ses joies, ses peines, ses occupations, ses idées, sa vie... La notion de compagnon peut être définie de plusieurs façons, mais elle semble toujours faire référence à un être vivant et est étroitement liée à la vie d'une personne.

Dans le domaine de l'interaction homme-machine les compagnons sont matérialisés de deux manières principales : sous forme de robot et sous forme d'agent virtuel. En robotique, les robots liés à l'homme, en dehors des robots industriels, sont des robots de service ou des robots compagnons. Les robots de service permettent d'assister l'homme dans les tâches de la vie quotidienne. Ils améliorent le quotidien des personnes en effectuant des tâches dans l'environnement. Les robots compagnons permettent d'apporter, en plus, un réconfort psychologique. Ils agissent sur les émotions pour favoriser le bien être des utilisateurs. Très souvent, ils visent les personnes fragilisées ou malades. L'objectif des années futures est de combiner services et compagnons pour que les robots puissent rendre service tout en étant capables de converser de façon intelligente et empathique. C'est un des objectifs que tente également d'atteindre la communauté des agents virtuels. Un agent virtuel est un personnage qui vit à travers un écran d'ordinateur, de téléphone mobile, de télévision... On peut considérer que le personnage peut être transporté partout avec l'humain (via son canal de communication).

Le Chapitre III montre que les humains ne préfèrent ni les robots, ni les personnages virtuels. Le robot apporte une présence physique (rassurante) aux êtres humains. Le personnage virtuel, généralement plus expressif, offre une qualité de relation supérieure (grâce aux émotions, à l'empathie...). En plus des robots et des agents virtuels, il existe de nombreuses applications sur ordinateur qui permettent d'accompagner un être humain : les environnements de formations, les visites virtuelles... En informatique, un compagnon peut donc être un robot, un agent virtuel ou un logiciel.

La technologie n'est pas encore suffisamment avancée pour fournir à l'utilisateur un compagnon qui puisse répondre à toutes ses attentes. Ainsi, un utilisateur possède plusieurs compagnons informatiques qui fonctionnent indépendamment. Il pourrait être intéressant de combiner les fonctionnalités de chacun des compagnons pour créer une sorte de « super compagnon ». Pour cela, il faut créer un système informatique qui acquière des informations sur l'utilisateur et son environnement, en utilisant les capteurs disponibles (caméra, microphone, thermomètre, ...), et qui agit sur l'environnement grâce aux actionneurs présents (robot, bras mécanique, personnage virtuel, volet automatique, ...). Ce système informatique est une entité abstraite capable de répondre aux besoins de l'utilisateur en utilisant les capteurs et les actionneurs nécessaires. **C'est le Compagnon Artificiel de l'utilisateur.**

B. Contexte : le projet Robadom

Ce travail s'inscrit dans l'ANR TecSan Robadom (ANR-09-TECS-012-02). Ce projet a pour titre : « impact d'un Robot « Majordome » à domicile sur l'état psychoaffectif et cognitif de personnes âgées ayant des troubles cognitifs légers ». C'est un projet simplifié de la problématique du Compagnon Artificiel. Il s'agit d'un robot majordome que l'on utilise dans un contexte précis.

1. Importance de l'autonomie à domicile

L'autonomie des personnes à domicile est une préoccupation économique et sociale. D'un point de vue économique, le nombre de personnes âgées ne cesse de croître avec l'augmentation de l'espérance de vie. Proportionnellement, le nombre de personnes atteintes d'Alzheimer augmente également. Plus leurs facultés cognitives diminuent, plus ces personnes ont besoin d'une assistance spécifique. Actuellement la limite des structures d'accueil est un problème majeur. Trouver le moyen de maintenir les personnes à domicile serait une solution pour permettre aux personnes âgées de conserver une bonne qualité de vie. D'un point de vue social, l'autonomie à domicile est primordiale. Les personnes âgées ne veulent pas quitter leur domicile, leurs habitudes, leur vie pour aller vivre dans une maison de retraite. Cependant, malgré cette volonté marquée, les personnes qui vivent seules souffrent souvent de la solitude et développent des signes inquiétants de dépressions. De plus, les personnes atteintes d'Alzheimer ont plus que les autres besoins de repères et d'habitudes. Il est donc crucial de les maintenir dans leur cadre de vie habituel.

2. Importance de la technologie pour l'autonomie à domicile

Le maintien à domicile des personnes ne développant pas de troubles cognitifs peut être mis en place grâce à la visite à domicile des assistants de vie. Ce système ne peut pas être adapté aux personnes souffrant de troubles cognitifs car celles-ci requièrent une attention constante et il n'y a pas assez d'assistants de vie pour assurer un suivi permanent de ces personnes. Utiliser la technologie pour aider à l'autonomie des personnes âgées, dont les personnes atteintes de troubles cognitifs, semble être la solution la plus adaptée. La technologie permet de régler les problèmes économiques et sociaux cités ci-dessus. D'un point de vue économique, il n'est pas possible de chiffrer *a priori* le coût de la technologie nécessaire. Il faudrait d'abord investir dans ce matériel technologique. Mais ensuite, les coûts d'utilisation de cette technologie seraient négligeables en comparaison de la valeur du placement dans une structure spécialisée. Et, plus la technologie permettra de centraliser l'aide apportée par des personnes extérieures moins les coûts seront importants car, par exemple, un assistant de vie prendrait en charge la surveillance de beaucoup de personnes. D'un point de vue social, la technologie permet déjà de développer et maintenir son réseau social. On peut imaginer qu'une personne âgée puisse régulièrement faire des vidéos conférences avec sa famille, ses amis, son médecin... La personne serait moins isolée. Si la technologie offre en plus la possibilité d'aider la personne dans l'organisation de son quotidien, de lui assurer une présence quotidienne et de lui fournir un suivi adapté alors une partie du défi du maintien à domicile sera relevé.

3. Objectifs du projet

L'objectif du projet Robadom est de proposer une technologie qui permette le maintien à domicile des personnes âgées souffrant de troubles cognitifs légers. Dans ce cas, la technologie doit pouvoir apporter trois types de services :

- **Services d'aide matérielle** : la technologie permet d'aider la personne dans la gestion de sa vie quotidienne. Par exemple, elle peut gérer les rendez-vous de la personne ; gérer les moments importants de la journée, comme la prise de médicaments ; surveiller que la personne boive suffisamment ; donner des informations variées comme les actualités, la météo, le programme TV... ; rechercher des objets comme les lunettes, les clefs, les médicaments...
- **Services de relais d'informations** : la technologie peut être un intermédiaire pour les proches de la personne. Grâce aux systèmes de vidéosurveillance, la personne pourrait discuter avec son médecin, sa famille, ses amis, ses voisins à tout moment. Sans même parler d'interaction direct, le système informatique pourrait tout simplement envoyer des alertes spécifiques à son médecin ou sa famille en fonction des informations à relayer. Par exemple, on pourrait imaginer que si la personne refuse de prendre ses médicaments, alors une alerte serait envoyée au médecin qui pourrait agir en conséquence.
- **Services de soutien psychologique et cognitif** : l'intérêt de la technologie est son omniprésence au sein de la maison et à travers le temps. C'est elle qui est présente pendant les moments de solitude de la personne. Ainsi, le système informatique pourrait offrir des services d'animation de vie et notamment, dans le cas des personnes atteintes de troubles cognitifs, des exercices de stimulation cognitive qui permettent de ralentir le déclin cognitif.

4. Axes étudiés

Le projet Robadom propose de concevoir un robot d'aide à domicile et d'évaluer les bénéfices apportés par ce type de technologie. Pour y parvenir, il propose d'explorer les différents axes suivants :

- **Définir l'apparence du robot** : un groupe de personnes âgées réuni au début du projet Robadom a permis d'établir que le robot ne doit pas être de type humanoïde. Il est possible que la science-fiction a peint une image très noire des robots, qui volent la vie des humains ou qui essaient de les remplacer. Ce préjugé semble très présent dans l'esprit de ce groupe qui montre une réticence certaine concernant les robots. Ainsi, le robot d'aide à domicile doit ressembler à un objet classique de la vie de ces personnes. L'une des missions du projet est de proposer un robot qui sera accepté par elles.
- **Définir le comportement du robot** : en interaction homme-machine, il est nécessaire que la technologie soit adaptée à l'être humain. En premier lieu, le robot doit être facile à utiliser et répondre aux besoins des personnes. Il doit offrir une interaction « naturelle » (cf. note de bas de page 7) pour faciliter l'adaptation des personnes. Et pour finir, il ne doit pas être perçu comme un système intrusif. S'il doit être différent de l'homme par son apparence, il doit au contraire être similaire à l'homme dans sa façon d'être. Cela signifie qu'il doit adapter son comportement verbal et non verbal, exprimant des émotions, de l'empathie, de façon à être accepté.
- **Étudier l'impact** de ce robot dans la vie de tous les jours : Est-ce que le robot apporte un confort ou un réconfort aux personnes ? Est-ce qu'il améliore leur qualité de vie ? Est-ce qu'il améliore ou n'aggrave pas les troubles cognitifs des personnes ?
- **Étudier la perception** qu'ont les personnes du robot : quelles sont les conditions d'acceptation de ce type de robot ? Est-ce que le robot est un simple objet qu'on pose dans un coin et qu'on oublie ? Au contraire est-il devenu un compagnon pour la personne, un intrus... ?

5. Participation des partenaires

Le projet Robadom compte quatre partenaires : l'hôpital Broca à Paris avec le laboratoire LUSAGE (Anne-Sophie Rigaud), le laboratoire ISIR à Paris (Mohamed Chetouani), l'entreprise Robosoft à Bidart (Meftah GHRISSI) et le laboratoire VALORIA (équipe aujourd'hui au laboratoire Lab-STICC) à Vannes (Dominique Duhaut). Les rôles de ces partenaires sont les suivants :

- **Hôpital Broca et laboratoire LUSAGE** : cette équipe est composée de médecins et de psychologues spécialisés dans la neurologie et la gériatrie. Ils sont chargés de spécifier les besoins et d'effectuer les évaluations des technologies utilisées dans le projet auprès des personnes âgées. Les personnes qui participent aux expérimentations ont des faibles troubles cognitifs.
- **Laboratoire ISIR** : cette équipe est composée d'informaticiens et de roboticiens spécialisés dans le traitement du signal, dans la reconnaissance de visage, de formes..., pour l'analyse de l'interaction et dans l'étude des capteurs en général. Par exemple, cette équipe fournit, dans le cadre du projet, un gestionnaire d'attention qui est capable de déterminer si une personne est concentrée dans une tâche ou non. Il est aussi possible de déterminer si une personne interagit avec le robot, si elle est en train de parler ou non, si elle est loin ou près, quel mouvement elle effectue...
- **Entreprise Robosoft** : cette entreprise est une PME française spécialisée dans la conception de robot mobile. Un de leur dernier produit, le robot Kompai, illustré sur la Figure 1, est fabriqué pour l'assistance des personnes âgées. Il fournit un ensemble de services permettant de gérer leur quotidien : gestion des rendez-vous, de la liste de course, discussion par vidéo conférences, jeux... C'est la base de ce robot qui est utilisé pour le projet. Il est prévu de remplacer le haut actuellement inerte par un corps et un visage expressif afin de lui faire effectuer un comportement non verbal et des expressions faciales.
- **Laboratoire VALORIA (2010-2011) puis Lab-STICC (2012-2013)** : cette équipe est spécialisée dans l'interaction homme-machine et dans la conception de robots expressifs et émotifs. Elle a pour objectif de concevoir une architecture informatique multimodale permettant de gérer l'interaction entre la personne âgée et le robot. Cette architecture doit répondre aux besoins de BROCA, intégrer le travail de l'ISIR et utiliser le robot de Robosoft.



Figure 1 : Robot Kompai de Robosoft

C. Interactions du Compagnon Artificiel

Le chapitre A.1 a expliqué que le Compagnon Artificiel permet à un utilisateur d'avoir une interaction avec l'ensemble des objets numériques contenus dans l'environnement. La Figure 2 illustre l'environnement d'un utilisateur. Elle montre qu'une personne est entourée par un certain nombre de dispositifs numériques. Ces dispositifs peuvent être des capteurs, capables de donner des informations sur l'environnement (caméras, micros, thermomètres...) ou des actionneurs, capables d'effectuer des actions (un robot, un agent virtuel, une télévision...). Chaque capteur et actionneur a un rôle défini et participe à l'interaction, même s'il n'est pas en interaction directe avec l'utilisateur. Le Compagnon Artificiel de la personne peut être représenté par les liens (en gris sur la Figure) entre chaque dispositif. Cela forme une entité abstraite qui utilise les capteurs et les actionneurs présents.



Figure 2 : Une personne et son Compagnon Artificiel

La Figure 2 montre que le Compagnon Artificiel est en perpétuelle interaction avec les êtres humains et l'environnement. Concernant les êtres humains, l'utilisateur n'est pas la seule personne en relation avec le Compagnon Artificiel. Ce chapitre met en évidence trois types d'humains concernés. De plus, concernant l'environnement du Compagnon Artificiel, il est soumis à un certain nombre de contraintes. En effet, l'environnement est ambiant et les objets numériques ne sont pas toujours les mêmes (présence, absence ou intégration d'un nouvel élément).

1. Rôle de l'homme dans le Compagnon Artificiel

Différentes personnes sont en relation avec le Compagnon Artificiel :

- Les concepteurs : ils fournissent le matériel et hébergent le système. Ce sont les créateurs des objets numériques.
- Les prestataires : ils créent de nouveaux services en utilisant les fonctionnalités des objets numériques. Il devient possible d'utiliser un objet numérique d'une façon non prévue initialement par son concepteur. Dans le cas de la surveillance à domicile, des médecins peuvent donner des instructions pour que le système informatique détecte un danger et appelle les secours si besoin.
- L'utilisateur final : il interagit avec le système et utilise ses fonctionnalités. Dans le cas de la domotique (automatisation d'une maison), il peut programmer un certain nombre d'événements pour gérer son matériel.

Chaque intervenant humain peut participer à l'évolution du système informatique en créant des scénarii d'usage.

Exemple

Dans le cas d'un thermostat :

- Concepteur : fournit un thermomètre et un chauffage.
- Prestataire de service : crée un scénario : « si la température atteint la température maximum autorisée alors le chauffage s'éteint sinon si la température atteint la température minimum alors le chauffage s'allume ».
- Utilisateur : personnalise le scénario en indiquant quelles sont les températures minimum et maximum souhaitées.

Au moment où un concepteur écrit le comportement de son objet numérique, il l'imagine dans un contexte précis. Cependant, un utilisateur peut vouloir utiliser l'objet numérique dans un autre contexte. Il est intéressant de proposer un système qui permette à un prestataire de service ou à un utilisateur d'écrire des nouveaux scénarii d'usage sans devoir créer un programme informatique complexe et sans devoir connaître, *a priori*, le contexte d'exécution de l'objet numérique. La création de scénario d'usage doit être simple et uniforme pour tous les capteurs et actionneurs de l'environnement et ne doit pas nécessiter des connaissances en informatique.

2. Environnement ambiant du Compagnon Artificiel

L'environnement du Compagnon Artificiel est composé d'un certain nombre d'objets numériques comme des caméras et des micros. Il contient aussi des objets augmentés qui « sont des objets en apparence classiques, conservant leurs fonctions habituelles, mais disposant de fonctionnalités supplémentaires apportées par l'informatique. Par exemple, un four qui calcule automatiquement la durée de cuisson des aliments peut être qualifié d'augmenté : il garde sa fonction originale, à laquelle il ajoute une nouvelle fonctionnalité » (Jacquet 2006). L'environnement inclut également des objets connectés. Grâce aux progrès de la technologie et à l'émergence des communications sans fil, des objets comme les télévisions, les frigos, les serrures peuvent communiquer directement avec d'autres dispositifs. Par exemple, la société américaine Apigy, propose Lockitron, un appareil qui permet de verrouiller ou déverrouiller une serrure à distance. Le Compagnon Artificiel dispose donc d'un éventail riche de fonctionnalités pour assister l'utilisateur. Oui, mais...

L'utilisateur n'est pas statique, en conséquence l'environnement du Compagnon Artificiel n'est pas statique non plus. Dans une journée, l'utilisateur n'a pas accès à tous les objets numériques en même temps. De plus, il se déplace et peut changer d'environnement ou alors effectuer des modifications. Par exemple, il peut jeter des objets, les changer de place... Le Compagnon Artificiel doit tenir compte de ces situations et s'adapter. Il doit utiliser des objets numériques en contact avec l'utilisateur pour lui assurer les meilleurs services. Par exemple, dans le contexte d'une surveillance à domicile, s'il existe plusieurs caméras, le Compagnon Artificiel doit récolter des informations uniquement en provenance de la caméra qui « voit » l'utilisateur.

Pour permettre au Compagnon Artificiel d'utiliser l'ensemble des fonctionnalités, il faut déjà que tous les objets numériques soient branchés en réseau et puissent communiquer ensembles. Cette informatique omniprésente est appelée **l'informatique ubiquitaire**. L'utilisateur interagit avec certains objets sans se rendre compte de tous les traitements informatiques.

Le Compagnon Artificiel ne peut assister l'utilisateur que si l'homme est au cœur du système. L'environnement doit posséder des capteurs capables de « représenter » l'environnement. Cette **perception de l'environnement** est une contrainte forte du Compagnon Artificiel. En effet, si une

personne âgée tombe et que les capteurs ne détectent pas cette chute, cela peut représenter un grave danger pour la personne. Le Compagnon Artificiel doit se faire une représentation constante de l'état de l'environnement. Ceci passe par une modélisation de l'environnement.

Enfin le Compagnon Artificiel doit maintenir une **interaction naturelle** avec les objets habituellement utilisés par l'utilisateur. Le principe consiste à utiliser les canaux de communication de l'homme pour créer une interaction qui lui semble naturelle : reconnaissance et synthèse vocales, reconnaissance des gestes, des formes, interaction émotionnelle... Le domaine de l'interaction homme-machine a beaucoup évolué. Les ordinateurs ne sont plus les seuls objets informatiques utilisés par une personne dans son quotidien. On compte les appareils électroménagers « intelligents », mais aussi les GPS, les voitures (remplies d'électroniques), les consoles de jeux, les télécommandes en tout genre. Non seulement, chacun de ces objets nécessite un temps d'apprentissage, mais en plus les utilisateurs peuvent développer de la lassitude si les systèmes sont trop complexes. Le Compagnon Artificiel permet de simplifier l'utilisation de tous les objets numériques par l'utilisateur en uniformisant leur accès et en minimisant le nombre d'« interlocuteurs ».

Le domaine de l'interaction homme-machine nous apprend qu'il est nécessaire de prendre en compte les utilisateurs. Ainsi, pour assurer une interaction à long terme, nous nous intéressons au problème de l'acceptabilité de l'interaction avec le Compagnon Artificiel. Bien que les notions de migration (Syrdal et al. 2009) et de mémoire à long terme (Ho et al. 2009) soient fondamentales pour créer une relation à long terme, notre travail ne traite pas ces sujets. L'acceptabilité est un sujet de recherche vaste et en plein essor. Il n'existe pas encore de règles standardisées à suivre. Mais il existe plusieurs études et modèles. L'annexe A en présente quelques travaux effectués dans le domaine de l'acceptabilité au sens large. Notre travail prend en compte quelques aspects de l'acceptabilité à travers les expérimentations que nous avons menées.

On retient donc deux problèmes principaux qui doivent être gérés :

- Au fil du temps, des nouveaux objets numériques sont créés. Lorsqu'ils sont introduits dans le système informatique, ils doivent pouvoir interagir avec les objets déjà en place.
- L'humain évolue dans son environnement. Il faut tenir compte du fait qu'un objet numérique n'est pas toujours accessible ou nécessaire.

Synthèse

Nous avons vu qu'il existe trois intervenants différents du Compagnon Artificiel :

- Les concepteurs des objets numériques,
- Les prestataires de services (qui définissent la configuration du système),
- L'utilisateur final.

Chacun de ces acteurs a un rôle différent qui doit être prévu lors de la conception du Compagnon Artificiel.

Le Compagnon Artificiel utilise des objets numériques qui peuvent être connectés ou déconnectés, présents ou absents. Sa conception doit prendre en compte la dynamique du système et son adaptabilité.

Enfin, si nous créons un produit à destination d'un utilisateur, il faut s'assurer que ce produit sera effectivement utilisé. Le Compagnon Artificiel est conçu pour un utilisateur spécifique. Cela signifie qu'il doit respecter ses besoins. La conception du Compagnon Artificiel doit prendre en compte l'utilisateur et respecter au maximum les contraintes liées à l'acceptabilité des technologies.

Chapitre II.

Etat de l'art

CHAPITRE II. ETAT DE L'ART	23
A. <u>COMPAGNONS LOGICIELS</u>	25
B. <u>COMPAGNONS ROBOTIQUES</u>	31
1. <u>Introduction</u>	31
2. <u>Architectures</u>	37
C. <u>COMPAGNONS VIRTUELS</u>	47
D. <u>INTELLIGENCE AMBIANTE</u>	54
1. <u>Origines de l'intelligence ambiante</u>	54
2. <u>Définitions de l'intelligence ambiante</u>	56
3. <u>Problématiques pertinentes</u>	59
4. <u>Applications</u>	60
5. <u>Perspectives de recherche en intelligence ambiante</u>	63
E. <u>DISCUSSION</u>	64

Le Compagnon Artificiel utilise des objets numériques qui peuvent être classés en trois catégories : les logiciels, les robots et les personnages virtuels, qui peuvent être déployés sur tout type de support : ordinateur, téléphone portable, tablette, ... La question principale qui se pose alors est : comment construire une architecture informatique pour assurer l'activité du Compagnon Artificiel en utilisant tous ces objets numériques ?

Pour répondre à cette question, nous nous intéressons aux trois types de compagnons : compagnons logiciels, compagnons robotiques et compagnons virtuels (Chapitre II.A, B et C). En quoi sont-ils utiles pour le Compagnon Artificiel ? Est-ce qu'il existe une architecture qui peut être utilisée pour le Compagnon Artificiel ? Le D présente les travaux menés en intelligence ambiante dont la problématique est proche de celle du Compagnon Artificiel. Enfin le Chapitre II.E propose une discussion pour indiquer s'il existe une architecture (d'un compagnon ou en intelligence ambiante) qui apporte une réponse aux propriétés structurelles et fonctionnelles du Compagnon Artificiel.

A. Compagnons logiciels

Nous appelons ici logiciels ou compagnons logiciels des programmes qui ne s'interfaçent pas avec des robots ou des personnages virtuels. Par exemple, cela peut être des interfaces graphiques sur ordinateur, des environnements de formation, des applications de téléphones mobiles ou de tablettes etc. Par exemple, Verstockt *et al* ont développé une application sur smartphone pour assister les personnes ayant des handicaps (Verstockt et al. 2009). Qualifiés de compagnon logiciel, on trouve également Clavie (Laperrousaz and Teutsch 2003), un logiciel d'autoformation, capable de dialoguer avec l'utilisateur pour répondre à ses besoins et un assistant de fitness développé sur un téléphone mobile qui seconde les personnes pratiquant un sport (Stahl et al. 2008). Dans tous les cas, l'utilisateur interagit directement avec le logiciel via son support. C'est ce qui le distingue des compagnons robotiques où l'utilisateur interagit via un robot et des compagnons virtuels où l'utilisateur interagit avec un personnage virtuel.

Ce chapitre présente les architectures de six logiciels ou compagnons logiciels : l'architecture BDI, Baghera, la plateforme SEMAINE, un modèle de personnalité basé sur l'ennéagramme, Ubi-learn et ICARE. Ces architectures, issues de domaines différents, permettent de se faire une représentation globale de l'existant. Il est intéressant de comprendre leur organisation et leur structure afin de voir si elles peuvent être la base du Compagnon Artificiel ou, si ce n'est pas le cas, comment elles s'interfaçent avec le Compagnon Artificiel.

Modèle BDI

Le modèle BDI (Belief-Desire-Intention) (Bratman 1999) a été créé pour la programmation d'agents intelligents. Plus précisément, il permet de décrire le comportement des agents cognitifs. Il a été créé par Michael Bratman, professeur de philosophie à Stanford en 1999. La Figure 3 montre que ce modèle utilise trois concepts pour permettre un raisonnement de l'agent : les croyances (belief), les désirs (desire) et les intentions (intention). Les croyances d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement. Une croyance n'est pas forcément vraie (contrairement à une connaissance) et peut évoluer au fur et à mesure que l'agent recueille des informations. Les désirs représentent les motivations de l'agent, c'est-à-dire les objectifs et les situations que l'agent souhaite atteindre. Les désirs ne sont pas forcément réalisés. L'agent effectue un processus de délibération dans lequel il confronte ses désirs et ses croyances et choisit un ensemble de désirs qui peuvent être satisfaits. Les intentions sont des désirs que l'agent a décidé d'accomplir. Les intentions sont persistantes, cela signifie qu'un agent ne va pas les abandonner.

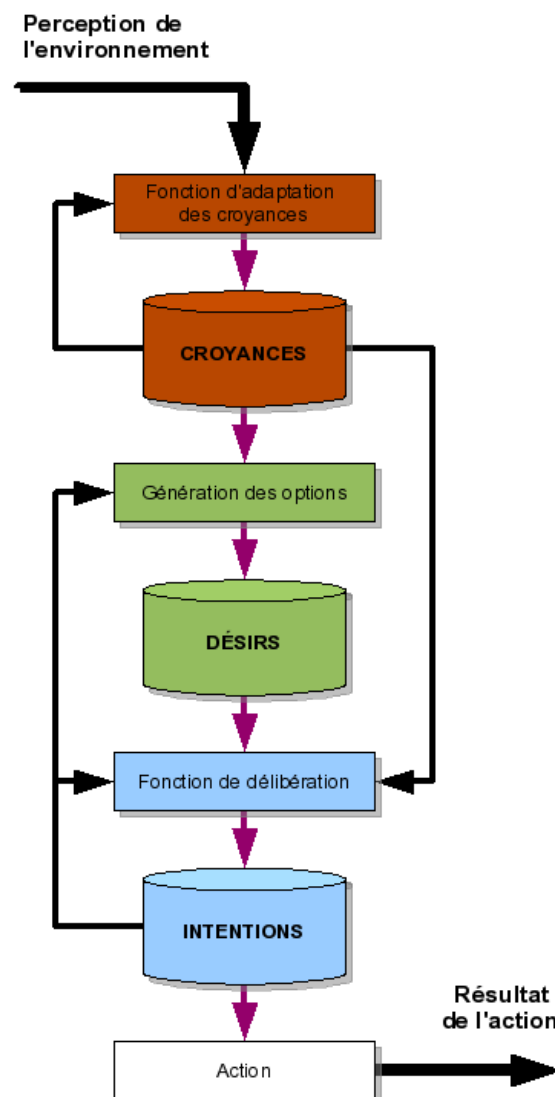


Figure 3 : Modèle BDI

API SEMAINE

L'API SEMAINE (Schröder 2010) est un cadriciel *open source* qui permet de construire des systèmes émotionnels à base de composants. Il a été utilisé, par exemple, pour créer un agent sensible SAL (*Sensitive Artificial Listener*). L'API utilise des formats de représentation standards de façon à permettre l'interopérabilité et la réutilisabilité des composants développés. Il est possible d'écrire des composants dans différents langages (Java et C++). Les composants s'envoient des messages, ce qui permet de s'affranchir du système d'exploitation et du langage utilisé. La Figure 5 montre l'architecture de l'API SEMAINE. Elle est composée de trois parties principales : l'analyse des capteurs, l'interprétation de ces données et la génération d'actions appropriées. L'entrée du système est donc constituée de capteurs qui permettent l'acquisition des données en provenance de l'utilisateur (analyse du visage, de la voix, de la gestuelle...). Les données sont fusionnées et interprétées, ce qui permet d'établir l'état de l'utilisateur, de l'agent et du dialogue. En fonction de ces trois données, le système propose des actions au générateur de comportement. La sortie du système est l'écran sur lequel s'affiche un agent conversationnel qui effectue l'action demandée.

L'API SEMAINE est intéressante car les composants sont interopérables et réutilisables. De plus, l'envoi de message entre les composants permet des échanges d'informations en plus de la tâche que doit effectuer chaque composant. Enfin, il est intéressant de pouvoir récupérer des données multimodales et de les fusionner pour les utiliser de façon globale.

Pendant l'API n'est pas dynamique et ne permet pas de prendre en compte la présence et l'absence des objets numériques. De plus, il n'est pas indiqué si l'utilisateur peut contrôler son environnement et s'il peut interférer sur les données d'entrée du système.

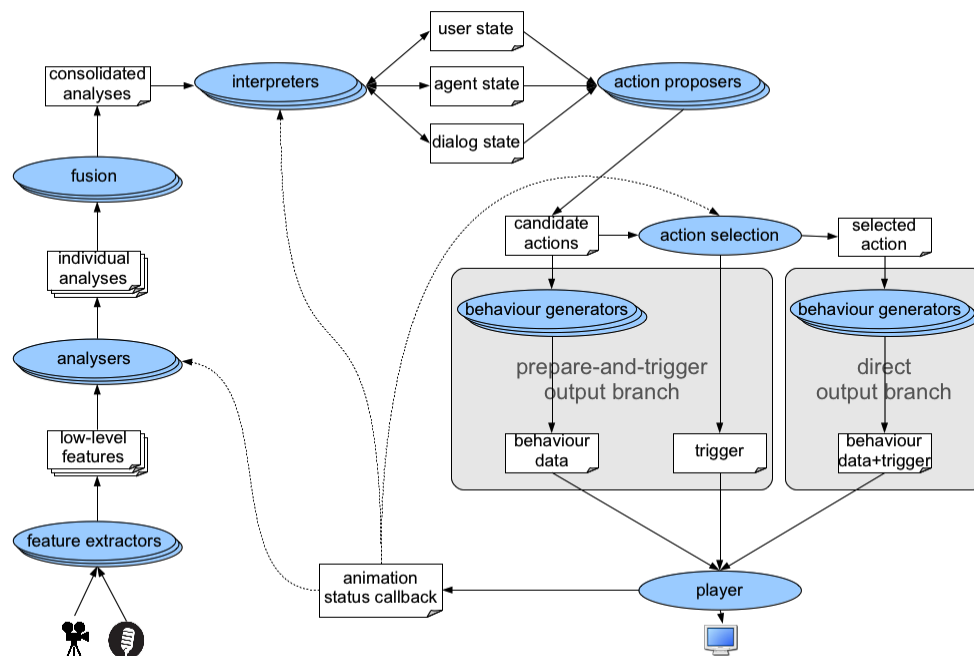


Figure 5 : Architecture SEMAINE

Modèle de personnalité

Soueina a proposé un modèle de personnalité (Soueina 2003) qui permet en observant un utilisateur de déterminer sa personnalité à travers neuf types de personnalité prédéterminés. L'objectif est d'éviter de faire passer de lourds questionnaires et de deviner la personnalité en fonction des indices donnés par les modalités d'entrées : reconnaissance du visage, de regard, gestuelle, reconnaissance automatique de la parole... Les modalités de sorties peuvent être du texte, de la synthèse de parole, des graphiques, du braille...

La Figure 6 montre l'architecture développée. Celle-ci prend en compte un utilisateur pour construire la réponse à lui apporter. L'ensemble des données des capteurs sont fusionnées par un module spécialisé (MMIF) et transmis à un gestionnaire de dialogue. Les données des capteurs sont également utilisées par un module (MMPPE) chargé d'extraire la personnalité supposée de l'utilisateur. Le gestionnaire de dialogue utilise l'analyse de ces deux modules pour générer la sortie appropriée à un module spécialisé (MMOG). C'est ce module qui est chargé d'afficher le résultat sur la sortie appropriée.

Le point fort de cette architecture est d'avoir un module spécialisé dans la gestion des données d'entrée et un module spécialisé dans la gestion des données de sortie. Il semble important de pouvoir prendre en compte des données multimodales. Mais il ne semble pas possible d'enrichir le système facilement et il n'est pas indiqué comment il est possible de contrôler les réponses à donner en sortie.

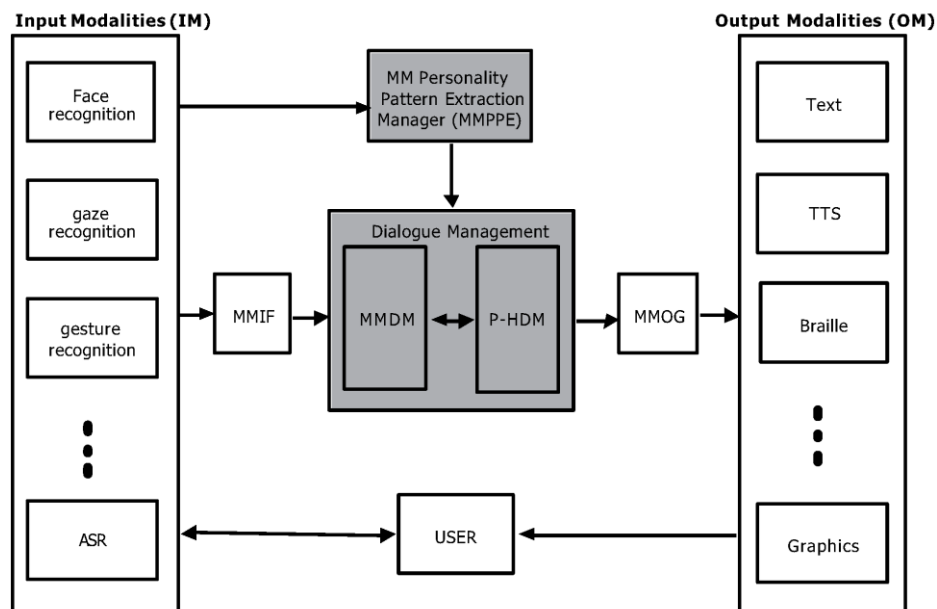


Figure 6 : Architecture du modèle de personnalité basé sur l'ennéagramme

Ubi-learn

Ubi-learn (Derycke et al. 2005) est une architecture qui permet la création d'applications d'e-formation. L'architecture est multicanale et multimodale. Cela signifie que l'application est faite pour pouvoir s'afficher sur n'importe quel mobile (téléphone, PDA, etc.) et qu'elle est capable d'utiliser n'importe quelle modalité disponible sur ce périphérique. L'interfaçage entre l'architecture *Ubi-learn* et les services applicatifs est fait par les services web. Ce type de système est composé de trois couches principales : une couche concerne les données (mise en œuvre et stockage), une couche services applicatifs et services communs (gestion des processus d'apprentissage), et une couche d'intermédiation qui fournit l'interface avec les usagers. Ubi-learn permet la composition dynamique d'e-services et utilise un système multi-agent. Cela permet de supporter simultanément plusieurs types de transactions, pour un nombre d'apprenants relativement importants et pour plusieurs types de canaux disponibles.

Les auteurs de ce projet sont parties de l'hypothèse que les aspects techniques qui permettent l'interopérabilité entre les services, leur ouverture et leurs réutilisation seront sans doute acceptés prochainement comme un standard et se sont focalisés sur la couche d'intermédiation.

Cette architecture ne semble pas évolutive et semble trop spécifique au cadre de l'e-formation pour être utilisée pour le Compagnon Artificiel. Les services web peuvent considérablement compliquer l'architecture. Il est nécessaire que le Compagnon Artificiel soit le plus simple possible.

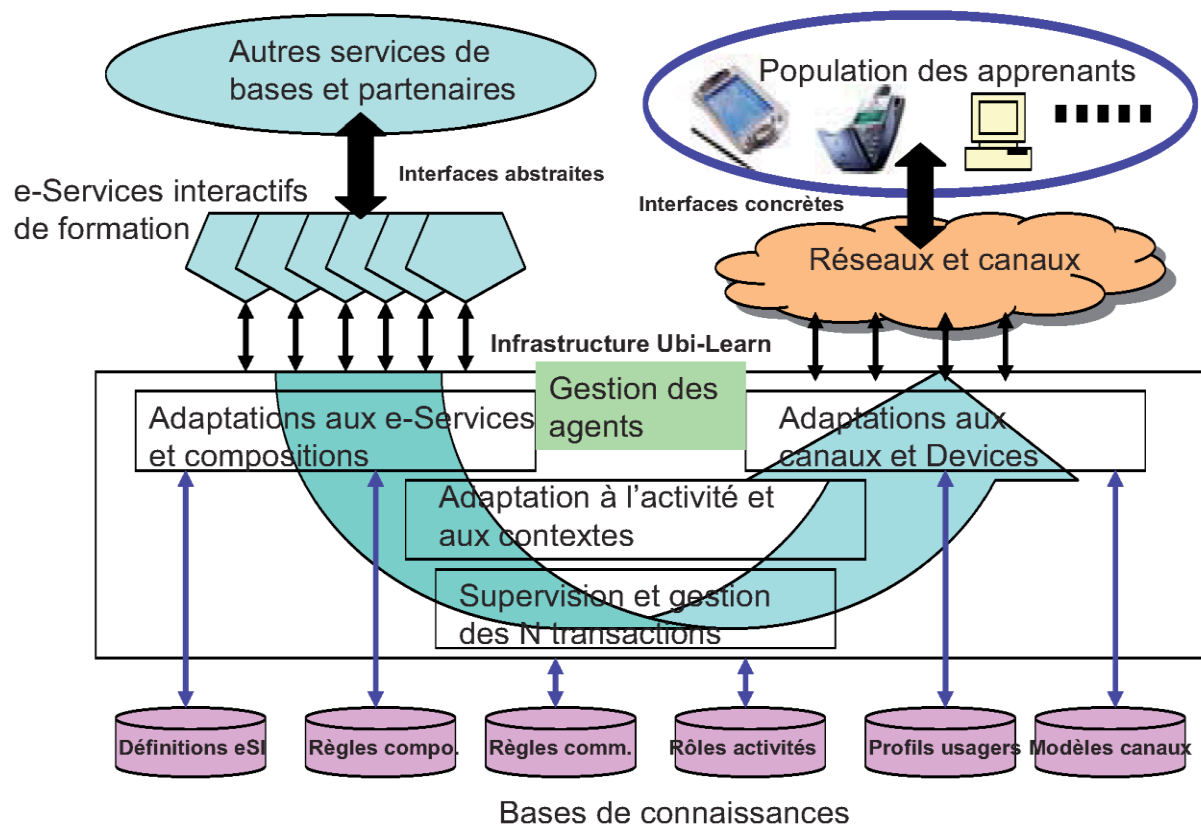


Figure 7 : Architecture Ubi-Learn

Architecture ICARE

L'architecture ICARE (*Interaction Complementarity Assignment Redundancy and Equivalence*) (Bouchet, Nigay, and Ganille 2004) est une approche basée composants permettant de créer rapidement des interfaces multimodales. A partir d'une spécification simple, le code est automatiquement généré. Il existe deux composants principaux. Le premier permet de définir des modalités et le deuxième permet de spécifier les relations entre ces modalités. Une modalité est la couche supérieure d'un matériel physique. Elle est représentée par des informations de haut niveau. Par exemple, une souris peut être représentée par « bouton pressé », « bouton relâché », etc. A chaque action, la modalité peut indiquer des informations relatives à l'interaction en cours. Par exemple, si le bouton est pressé, l'information peut être « sélection menu ».

Cette architecture permet de créer des logiciels de façon automatique et ne peut pas s'adapter au Compagnon Artificiel qui utilise différents types d'objets numériques. Cependant il est intéressant de représenter les modalités avec un niveau d'abstraction qui permet de s'affranchir des connaissances techniques des capteurs d'entrées. Et il est également intéressant de pouvoir déterminer des relations entre ces modalités d'entrées.

B. Compagnons robotiques

1. Introduction

Au début des années 60, les robots sont utilisés par les industries pour automatiser les tâches des hommes. Ce sont des robots industriels. Dans une revue de littérature (Shibata 2004), Shibata décrit l'émergence des robots industriels à partir des années 70, avec un pique en 1991. C'est à partir de ce moment que l'utilisation des robots se démocratise. Les robots de service naissent alors. Ils ne sont pas seulement utilisés par les professionnels. En effet, il existe de plus en plus de robots de services fabriqués pour une utilisation personnelle et privée. Il existe donc deux catégories de robots : les robots professionnels qui sont évalués par des mesures objectives (vitesse, précision, haute performance, fiabilité...) et les robots personnels qui sont aussi évalués par des mesures subjectives. En effet, ils doivent apporter un certain confort à l'utilisateur.

Les premières expérimentations sur les mesures subjectives de ces robots ont été effectuées dans un contexte médical par Shibata (Wada et al. 2004) avec Paro, un robot représentant un bébé phoque. L'objectif était d'évaluer l'impact de ces robots sur des personnes fragilisées. Ce pan de recherche s'appelle la Thérapie Assistée par Robots. Paro, qui ressemble à une peluche, est non seulement considéré comme un jouet mais également comme un compagnon pour les patients et a une influence significative sur leur humeur. De cette étude est né le courant des robots compagnons, qui apportent une dimension affective à la relation entre l'homme et le robot. Actuellement, il existe beaucoup d'études sur les robots compagnons qui ont pour objectif d'améliorer la qualité de l'interaction afin d'aider les personnes.

Selon Shibata, il y a quatre catégories de robots : les robots de services, et les robots compagnons qui se décomposent en trois catégories : les robots qui représentent des animaux réels, les robots qui représentent des créatures ou objets imaginaires et les robots humanoïdes (Shibata 2004). La Figure 8 montre un panorama d'un ensemble de robots existants actuellement dans chacune de ces quatre catégories (services, animaloïdes, imaginaires et humanoïdes).

Tous ces robots sont susceptibles d'être utilisés par le Compagnon Artificiel. En effet, le Compagnon Artificiel doit s'adapter au matériel que possède l'utilisateur. L'architecture doit pouvoir intégrer des robots différents techniquement.

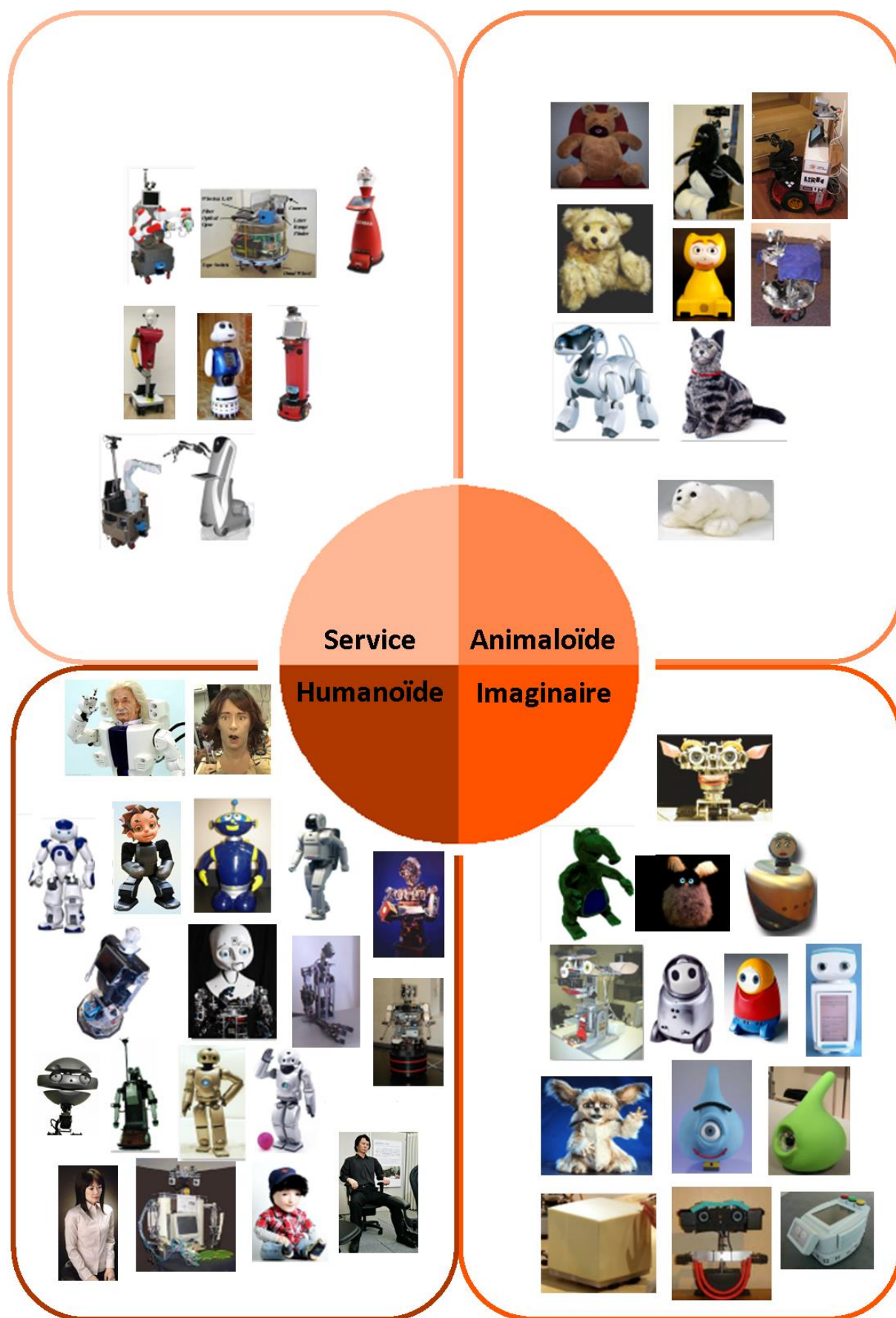


Figure 8 : Panorama d'un ensemble des robots existants

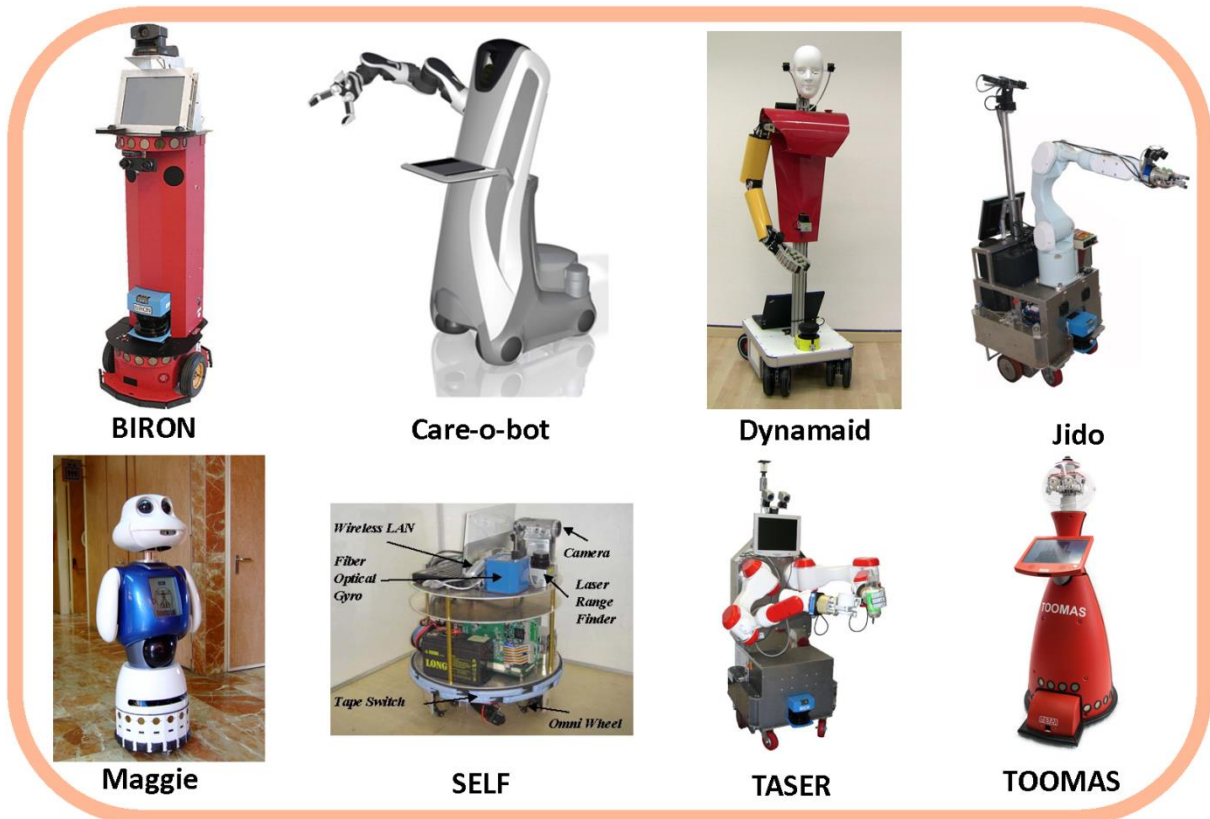


Figure 9 : Robots de services

BIRON (Shuyin Li et al. 2004) (Lohse et al. 2008), Care-O-bot I, II et III (Hans, Graf, and Schraft 2002), Care-O-bot 3 (Graf et al. 2009), Dynamaid (Stückler et al. 2009), Jido (Clodic et al. 2007), Maggie (Salichs et al. 2006), SELF (Mori et al. 2007), TASER (Baier et al. 2006) et TOOMAS (Gross et al. 2009).

L'objectif des robots de service est d'effectuer des tâches à la place de l'homme pour lui simplifier le quotidien. Ils doivent agir efficacement mais leur objectif premier n'est pas d'apporter une dimension sociale à la relation. Ainsi, ils n'ont pas de visages expressifs et ne peuvent pas exprimer les émotions. Leur but n'est pas de créer un lien émotionnel avec les humains. Ces robots ont généralement des roues, qui leur permettent de se déplacer de façon autonome. Certains ont des bras pour prendre des objets, d'autres intègrent un écran d'ordinateur. La Figure 9 montre un ensemble (non exhaustif) de ces robots.

Selon le point de vue du Compagnon Artificiel, ces robots peuvent effectuer des actions complexes. Il faut que l'utilisateur connaisse la liste des actions possibles et puisse y accéder simplement.



Figure 10 : Robots animaloïdes

Aibo (Arkin et al. 2003), Eml (Saint-Aime et al. 2009), Paro (Takanori Shibata et al. 2008), Mel (Sidner, Lee, and Lesh 2003), Huggable (Stiehl et al. 2005a) (Stiehl et al. 2005b), iCat (A. J. N. van Breemen 2005) (A. Van Breemen 2004) (Kessens et al. 2009) , NeCoRo (Libin 2003), Yuppy, un robot animal émotionnel (Velásquez 1998).

L'objectif des robots animaloïdes est de créer un lien affectif entre l'humain et le robot. Ces robots copient le comportement des animaux qu'ils représentent. Cependant, deux problèmes majeurs apparaissent dans ce groupe. Premièrement, la plupart d'entre eux représentent un animal connu, appelé les *animaux familiers* par Shibata (Shibata 2004). Souvent, les humains s'attendent à ce que les robots agissent exactement comme l'animal qu'ils représentent. Mais lorsque le robot n'arrive pas à reproduire le comportement de l'animal réel, les utilisateurs peuvent être déçus et rejeter le robot. Deuxièmement, ces robots ne peuvent ni communiquer ni être compris facilement, éléments pourtant fondamentaux dans l'interaction homme-machine. La Figure 10 montre un ensemble (non exhaustif) de ces robots.

Pour le Compagnon Artificiel, ces robots peuvent jouer le rôle de peluche ou de jouet. Ils ne peuvent pas rendre de service car ils n'ont pas les compétences physiques pour le faire. Par contre, ces robots pourraient récupérer des infos sur l'utilisateur et les transmettre au Compagnon Artificiel ou exprimer des émotions pour guider l'utilisateur dans son interaction.

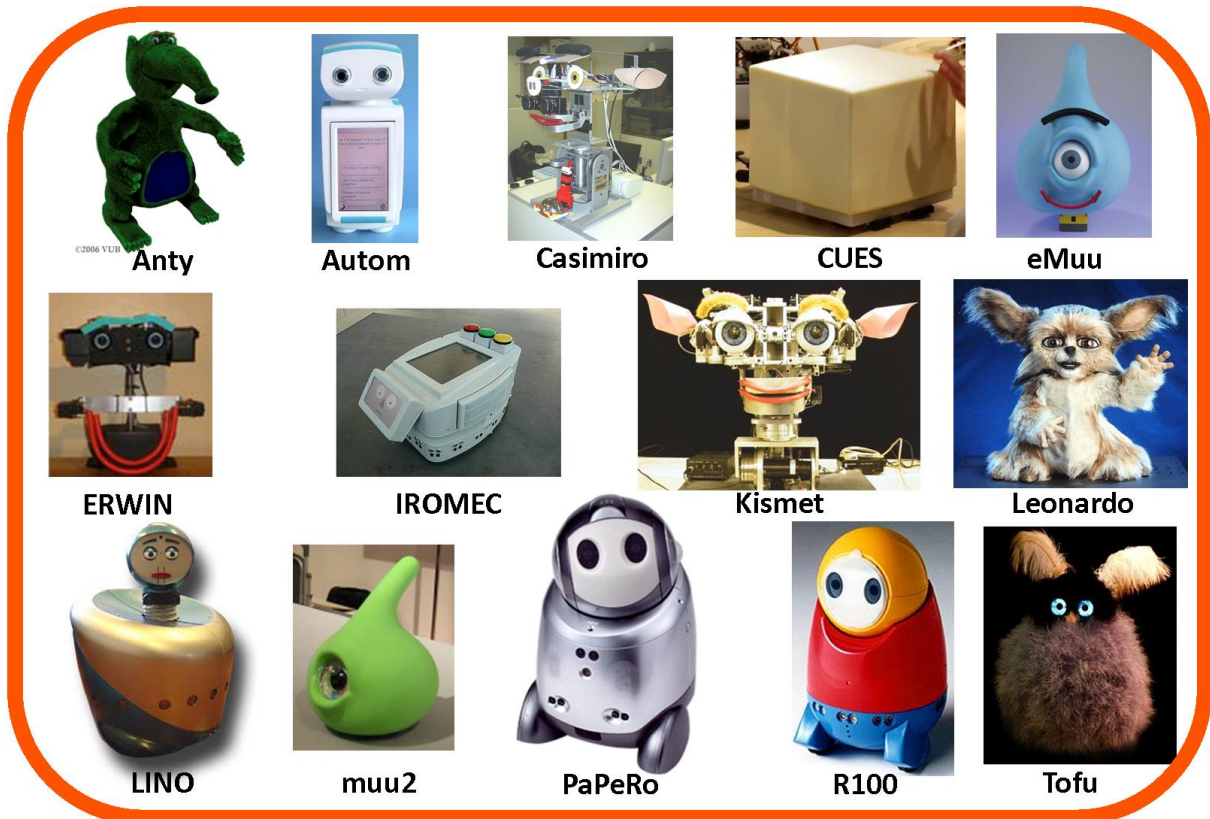


Figure 11 : Robots imaginaires

Anty/Probo (Saldien et al. 2006) (Saldien et al. 2008), Autom un robot coach pour la perte de poids (Cory D. Kidd and Breazeal 2008), CASIMIRO (Déniz et al. 2012), Cues (Yoshiike, De Silva, and Okada 2010), eMuu (Bartneck and Michio 2001) (Bartneck 2002), ERWIN, une tête de robot (Murray et al. 2009), IROMEC (Patrizia et al. 2009), Kismet (Cynthia Breazeal 2003) (C. Breazeal and Scassellati 1999), Leonardo (C. Breazeal et al. 2004), LINO (A. Van Breemen 2004), Muu2, NEC PaPeRo (C. Kidd and Breazeal 2006), R100, et Tofu (Wistort and Breazeal 2009).

L'objectif des robots imaginaires est d'apporter un lien affectif à l'humain. Comme leur apparence est totalement inventée, les humains ne s'attendent pas à un comportement particulier car ils ne sont pas familiers de ce genre de personnages. Cela semble favoriser l'acceptabilité de ces robots. Les robots peuvent utiliser des expressions et des comportements qui ne se calquent pas sur un modèle connu. Les humains doivent apprendre à les connaître de la même façon que nous devons apprendre la personnalité et le caractère d'une nouvelle personne quand nous la rencontrons. Mais, tout comme le deuxième groupe, ces robots ont un problème de communication car celle-ci ne ressemble pas forcément à celles des humains et il peut être difficile de comprendre le robot. La Figure 11 montre un ensemble (non exhaustif) de ces robots.

De par les possibilités qui sont offertes par ce type de robots, leur rôle dans le cadre du Compagnon Artificiel reste à définir. Il faut que le Compagnon Artificiel permette à chaque constructeur de robots d'imaginer eux même le rôle que doit avoir leur robot.

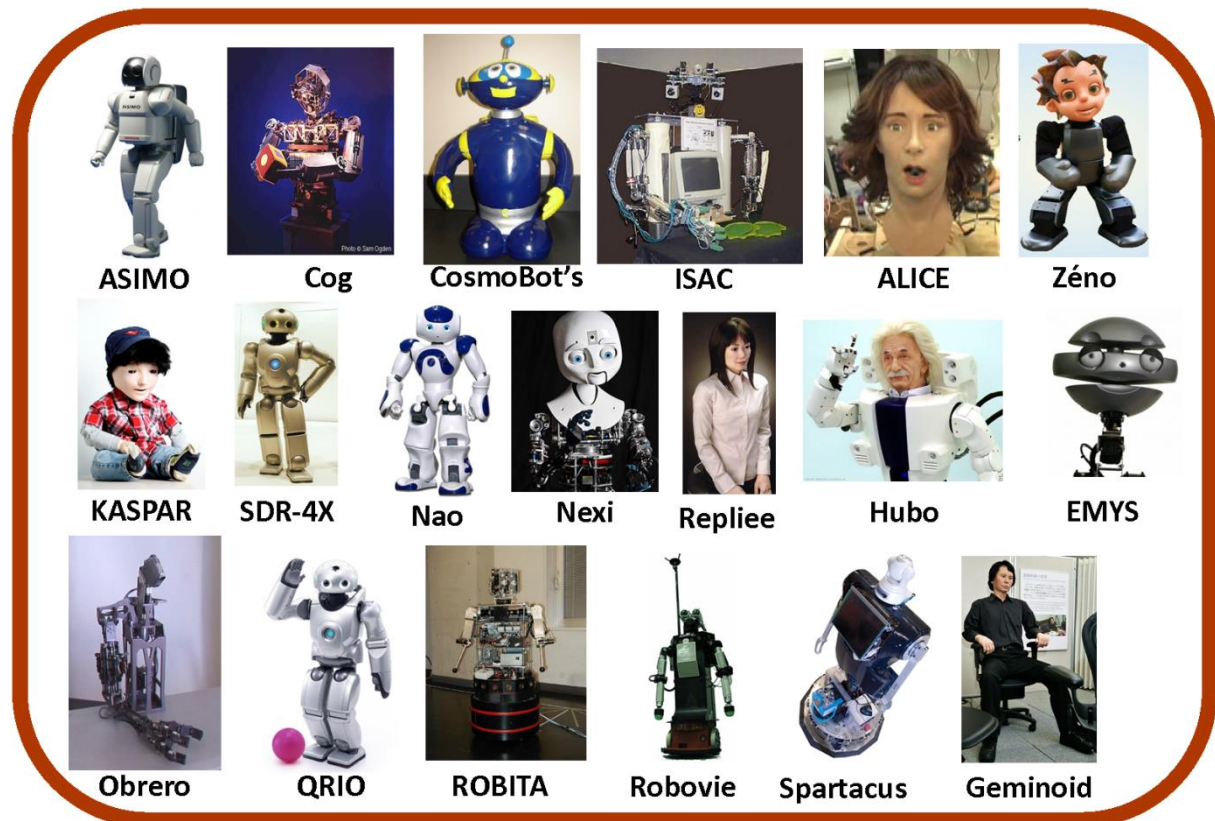


Figure 12 : Robots humanoïdes

ASIMO (Sakagami et al. 2002), Cog (R. A. Brooks et al. 1999), CosmoBot's (Lockerd, Brisben, and Lathan 2004) (Brisben et al. 2005), ISAC (Peters et al. 1999), ALICE (Hanson et al. 2012), Zeno (Hanson et al. 2009), KASPAR (Dautenhahn et al. 2009), SDR-4X (Arkin et al. 2003) (Fujita et al. 2003), Nao (Kulk and Welsh 2008), Nexi (C. Breazeal 2009), Repliee (MacDorman and Ishiguro 2004), Hubo (Oh et al. 2006), EMYS (Kędzierski et al. 2013), Obrero (Torres-Jara, Natale, and Fitzpatrick 2005), QRIO (A. G. Brooks and Arkin 2006), ROBITA (Tojo et al. 2000), Robovie (Kanda and Ishiguro 2005), Spartacus (Michaud et al. 2006) et Geminoid (Nishio, Ishiguro, and Hagita 2007).

Les robots humanoïdes peuvent communiquer plus facilement avec les humains grâce à leur anthropomorphisme. Ils peuvent utiliser le langage verbal ou les codes sociaux des humains et en ce sens, ils sont facilement compréhensibles. La Figure 12 montre un ensemble (non exhaustif) de ces robots.

Cette catégorie de robot est important pour l'interaction homme-robot car l'anthropomorphisme des machines augmente le jugement positif des humains et la qualité de l'interaction (Y. Kim and Sundar 2012). De plus, lorsque le robot imite le comportement humain, l'interaction peut être fluide et naturelle (Kirby, Forlizzi, and Simmons 2010), ce qui augmente l'acceptabilité du robot. Du point de vue du robot, être capable de communiquer naturellement revient à être capable de communiquer verbalement et non verbalement. Aucun canal n'est prédominant sur un autre dans la communication homme-homme (Ekman et al. 1980). Ainsi le robot doit s'exprimer en utilisant plusieurs modalités, et il doit également être capable de décoder l'humain. Pour cela, il est plus facile d'analyser plusieurs canaux simultanément (Ginevra Castellano, Kessous, and Caridakis 2008).

Enfin, les robots ne seraient pas de bons communicants s'ils n'étaient pas pourvu d'émotions (Scherer (Scherer 1984), James (James 1884), Darwin (Darwin 2002)). Même si le processus émotionnel n'est pas encore bien connu, il n'y a aucun doute sur le bienfait des émotions dans la

communication. Les études se portent majoritairement sur les six émotions de bases d'Ekman qui sont la joie, la peur, la tristesse, la surprise, le dégoût et la colère (Ekman and Friesen 2003). Les émotions peuvent s'exprimer à travers les expressions faciales, la gestuelle, la voix, le discours etc. Cette multimodalité complique la mise en place de processus émotionnel sur le robot, car même si les expressions faciales semblent maintenant comprises et reproduites sur des robots (Cynthia Breazeal 2003) (Saint-Aimé et al. 2009), ce n'est pas le cas de la voix par exemple. En effet la synthèse vocale ne permet pas pour l'instant d'exprimer des émotions car il est difficile de trouver le ton, la vitesse et la prosodie associés à chacune d'entre elles (C. Breazeal 2001). La recherche sur la communication homme-robot est ainsi très active comme l'explique en détail Breazeal dans son ouvrage (C. Breazeal and Brooks 2005).

Comme l'utilisateur peut s'identifier à ces robots, le Compagnon Artificiel pourrait les utiliser comme partenaire de travail ou de jeu de l'utilisateur.

2. Architectures

La partie précédente montre qu'il existe déjà beaucoup de compagnons robotiques pour les utilisateurs. Certains d'entre eux ont été créés avec des architectures plutôt génériques qui peuvent être réutilisées dans d'autres contextes. Il est intéressant de les étudier pour voir si une de ces architectures peut être celle du Compagnon Artificiel. Ce chapitre présente un ensemble d'architectures de robots ou ayant un lien avec des robots : les architectures PRS, AD, BERRA, ATLANTIS, TAME, iGrace et les robots Maggie, BIRON, Robovie et QRIO.

Architecture PRS

L'architecture PRS (*Procedural Reasoning System*) (Georgeff and Ingrand 1989) peut être vue comme une extension du modèle BDI. Elle permet de créer des applications capables de raisonner et d'établir des plans dans un environnement dynamique. La Figure 13 montre qu'elle est composée de :

- Une base de données qui contient les croyances actuelles ou les faits concernant l'environnement,
- Un ensemble de buts courants qui doivent être réalisés,
- Un ensemble de plans qui décrit comment certaines séquences d'actions peuvent être exécutées pour atteindre un but donné et réagir dans une situation particulière,
- Une structure d'intention qui contient les plans choisis pour être exécutés éventuellement,
- Un interpréteur qui dirige ces quatre composants en sélectionnant les plans appropriés (basé sur le système des croyances et des buts), les plaçant dans la structure d'intention et les exécutant.

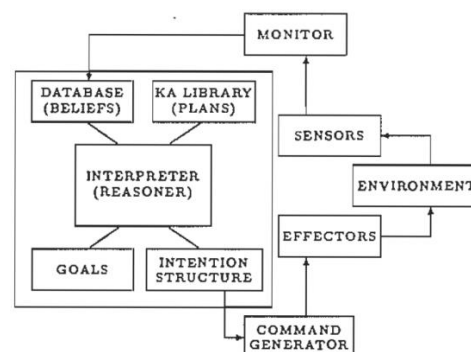


Figure 13 : Architecture PRS

Le système interagit avec son environnement, incluant d'autres systèmes. Sa base de données est mise à jour lorsque qu'il acquière de nouvelles croyances en réponse à un changement de l'environnement. Enfin il s'exprime à travers des actions qu'il accomplit au fur et à mesure qu'il exécute ses intentions.

Le Compagnon Artificiel peut s'inspirer de l'architecture PRS en ce qui concerne les plans et l'interpréteur qui dirige l'ensemble des composants. Par contre, l'architecture PRS n'est pas adapté en ce qui concerne les croyances et les intentions puisque le Compagnon Artificiel doit simplement obéir aux ordres donnés par l'utilisateur. Il ne faut pas que les objets numériques soient des agents autonomes du point de vue du Compagnon Artificiel.

Architecture AD

Plusieurs robots sont implémentés avec une architecture qui est basée sur un modèle psychologique : l'architecture AD (Barber and Salichs 2001). Le niveau A (Automatique) représente les compétences sensorielles et réactives du robot. Ce sont les compétences qui s'effectuent de façon automatique comme les réflexes. Tandis que le niveau D (délibératif) représente la capacité du robot à effectuer une tâche de haut niveau comme la planification, la gestion de mots, la gestion d'une mémoire etc.

Robot MAGGIE

On trouve une première application de l'architecture AD, sur le robot Maggie (Salichs et al. 2006), au sein du projet R2H, comme le montre la Figure 14. Ce projet a pour objectif de développer un robot réellement autonome avec un comportement basé sur ses propres impulsions et motivations. L'architecture propose une approche particulière où l'humain est au même niveau que les autres objets dans l'environnement. Les capteurs récupèrent des informations sur l'environnement et les transmettent au niveau Automatique sans faire de distinction. De plus, l'architecture AD est enrichie par un système de contrôle émotionnel du robot.

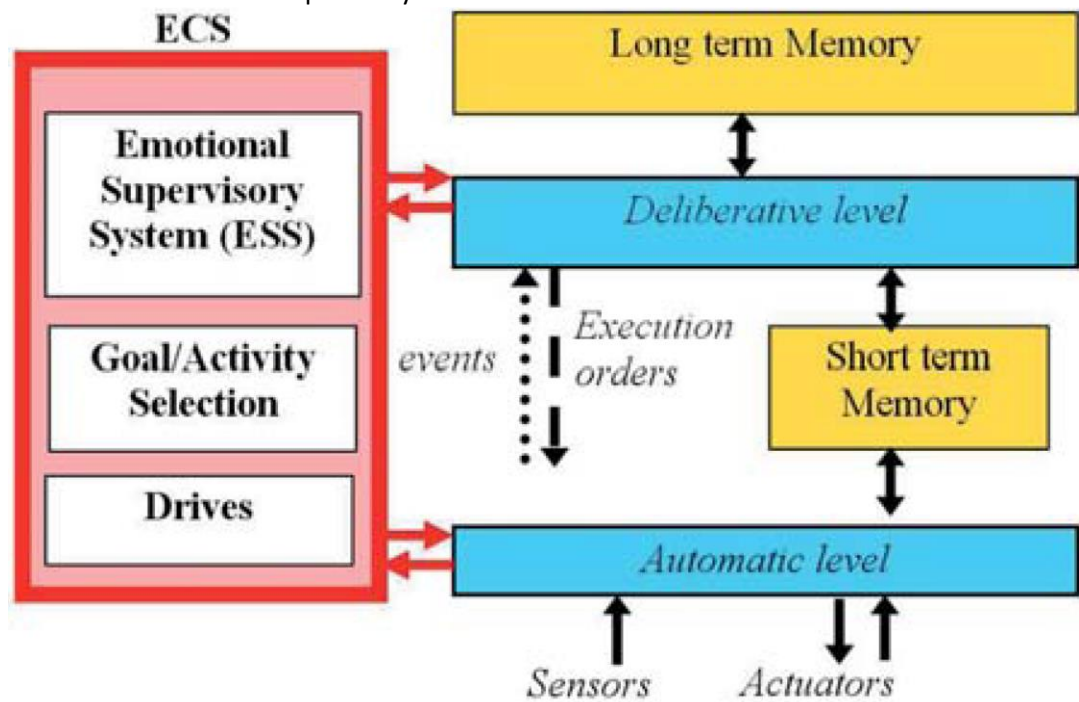


Figure 14 : Architecture de Maggie

Architecture BERRA

Lindstrom *et al* ont proposé une extension de l'architecture AD en proposant l'architecture BERRA (Lindstrom, Oreback, and Christensen 2000) qui signifie *BEhavior-based Research Architecture*. Cette architecture doit permettre de développer un robot capable de mener un ensemble de missions dans une maison ordinaire ou dans un bureau. Les auteurs ont basé leur architecture sur l'architecture AD car ils pensent qu'un robot de service doit à la fois avoir des réponses instantanées dans des situations inattendues mais également savoir délibérer pour fournir des réponses plus abouties. La Figure 15 montre qu'en plus des niveaux Automatique (nommée ici Réactif) et Délibératif, l'architecture BERRA possède un niveau d'exécution des tâches. Ce niveau permet de suivre la position du robot et de surveiller l'exécution des tâches. Le niveau Délibératif fait le lien entre l'homme et le robot tandis que le niveau Automatique récupère les informations des capteurs et envoient des ordres aux actionneurs.

L'architecture AD est trop spécifique pour être utilisée pour le Compagnon Artificiel. Elle ne répond qu'à une partie des besoins, qui nous semblent à l'heure actuelle optionnels (les émotions). Cependant, il est intéressant de reprendre l'idée de la mémoire à long terme pour permettre un apprentissage des réactions passées pour une meilleure adaptation du système à des événements imprévus.

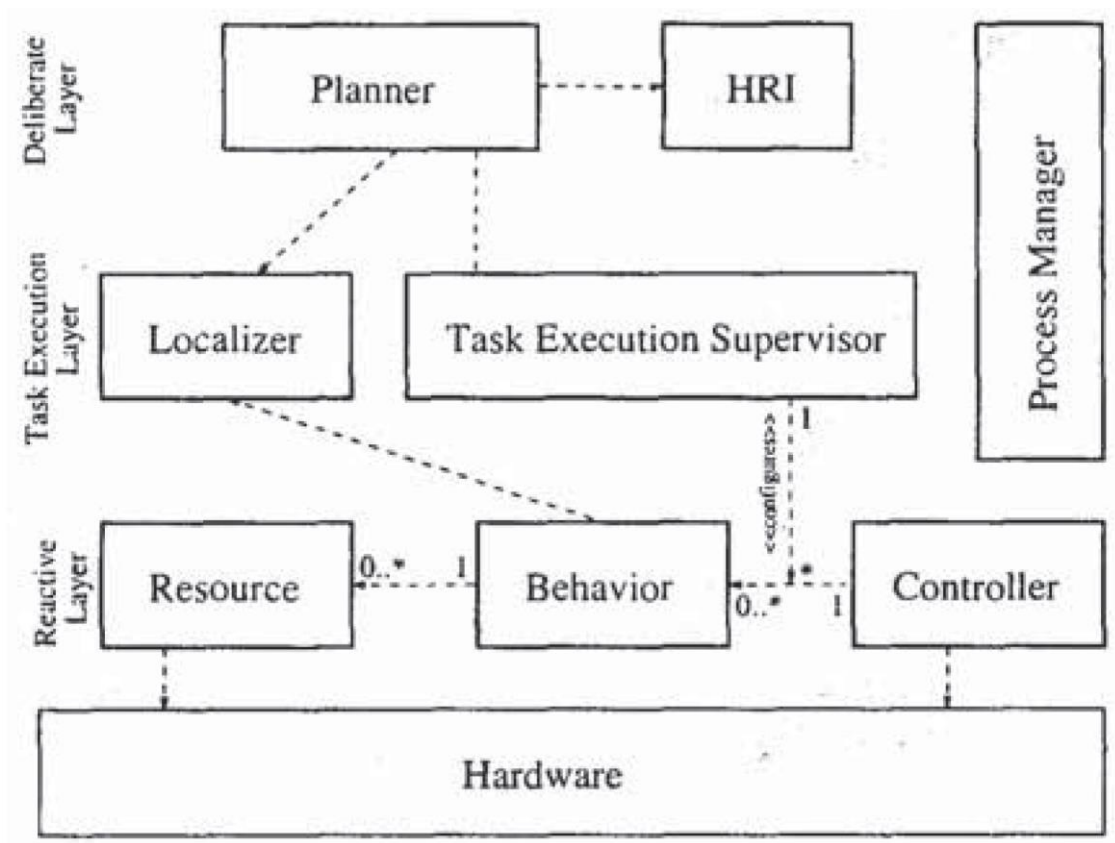


Figure 15 : Architecture BERRA

Architecture ATLANTIS

En tant qu'architecture à trois niveaux, l'architecture BERRA n'est plus vraiment une architecture AD. Il est possible de faire un rapprochement avec l'architecture *Three-Layer* (E. Gat and others 1998), et l'architecture ATLANTIS (Erann Gat 1991) qui signifie : *A Three-Layer Architecture for Navigating Through Intricate Situations*. La Figure 16 montre les trois niveaux : Délibératif, Séquencement et Contrôle. Le niveau contrôle est équivalent au niveau Automatique de l'architecture AD. Le niveau Délibératif est similaire à son homologue de l'architecture AD. Le niveau Séquencement contrôle les activités. Il est capable de définir des stratégies pour assigner des responsabilités aux autres niveaux. C'est un peu le chef d'orchestre de l'architecture.

Le Compagnon Artificiel peut s'inspirer de cette architecture en déterminant une sorte de chef d'orchestre qui pilote l'ensemble des objets numériques afin d'exécuter les ordres de l'utilisateur.

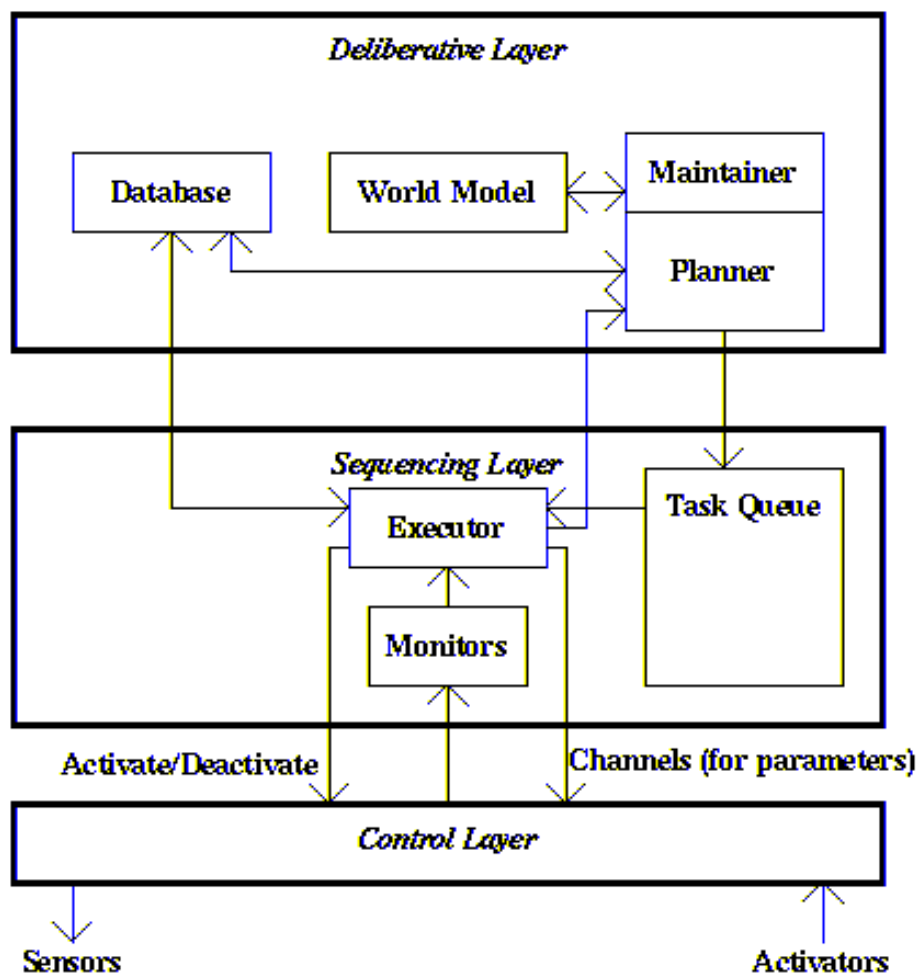


Figure 16 : Architecture ATLANTIS

Robot BIRON

Le robot BIRON (Fritsch et al. 2005) est basé sur une architecture à trois niveaux. La Figure 17 montre que l'on retrouve bien les trois niveaux : Délibératif, Intermédiaire (ou Séquencement) et Réactif (ou Automatique). Le vocabulaire employé pour désigner les niveaux n'est pas toujours le même, mais le rôle de ces niveaux est toujours le même. Par exemple, on retrouve un module chargé de coordonner les opérations (*execution supervisor*) comme pour l'architecture ATLANTIS. On retrouve également des modules chargés de générer les plans (*planner*) et de récupérer les instructions données par l'utilisateur (*dialog control*).

L'architecture du robot BIRON est trop spécifique pour être utilisée pour le Compagnon Artificiel mais l'idée de fournir une interface de contrôle à l'utilisateur est intéressante. Le Compagnon Artificiel doit permettre à l'utilisateur de donner des instructions au système.

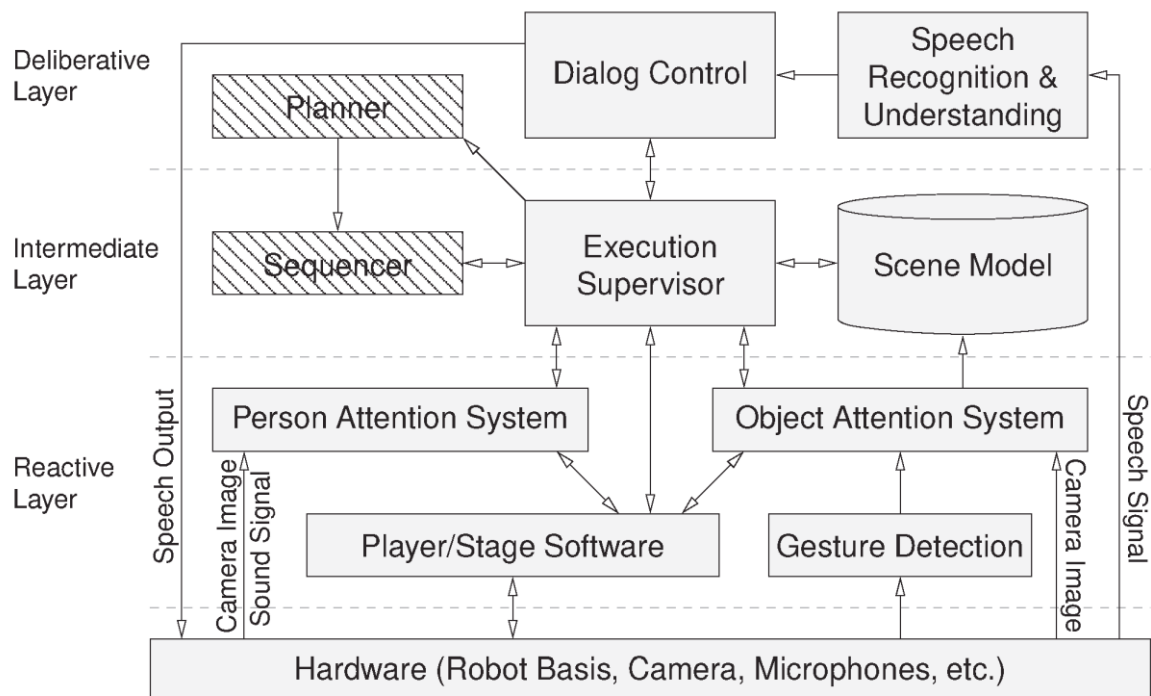


Figure 17 : Architecture de BIRON

Robot Robovie

Malgré tous ces exemples, les architectures à deux et trois niveaux ne sont pas un standard en robotique puisqu'il existe un nombre non négligeable de robots qui n'ont pas d'architecture standardisée. C'est le cas, par exemple, du robot autonome Robovie (Kanda and Ishiguro 2005). Ce robot a été utilisé dans des écoles en tant que partenaire d'élèves. Il utilise des modules qui gèrent l'interaction avec les humains (gestes et paroles), les puces RFID pour identifier les personnes avec qui il communique. Comme le montre la Figure 18, l'architecture contient quatre bases de données chargées de stocker des informations sur les personnes et de maintenir une communication avec elles. Ainsi le robot peut suivre la progression de chaque élève. L'architecture contient des modules réactifs qui semblent avoir un rôle proche d'une couche Automatique. Elle contient aussi des modules spécialisés dans le traitement des capteurs d'une part et des actionneurs d'autre part.

L'architecture de Robovie est spécifique à son utilisation et ne peut pas être utilisée pour le Compagnon Artificiel. Cependant on retrouve l'utilisation de modules dédiés à l'analyse des données d'entrées et à la génération des données de sorties. Le Compagnon Artificiel doit organiser les modules de la même façon pour permettre d'enrichir rapidement et facilement l'architecture avec des nouveaux modules d'entrées et de sorties. Il est également intéressant de pouvoir générer des règles d'interaction, de gérer une mémoire des événements et de prendre en compte l'utilisateur.

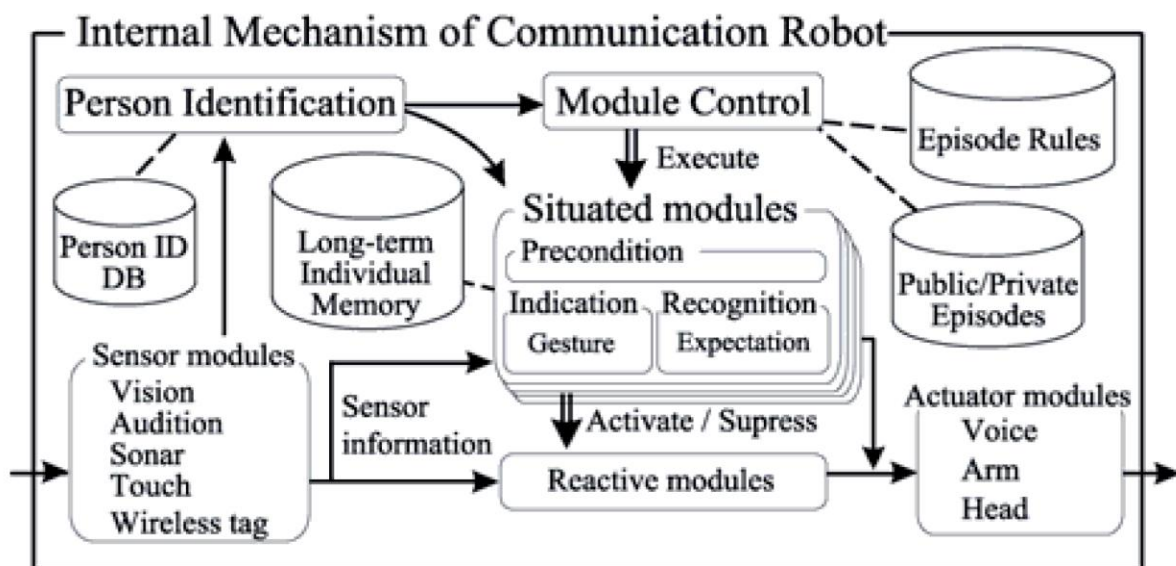


Figure 18 : Architecture de Robovie

Architecture TAME

En interaction homme-robot, la dimension sociale est importante. Un ensemble d'architectures propose une solution pour gérer la robotique affective. Par exemple, l'architecture TAME (Moshkina and Arkin 2003), présentée par la Figure 19, permet de calculer le comportement affectif des robots. Elle est composée de quatre composants qui gèrent les traits de personnalités, les émotions, l'humeur et l'attitude du robot. Ces quatre composants déterminent le comportement du robot en fonction des données de l'environnement qui sont envoyées par des modules de perception.

Cette architecture est limitée car elle est dédiée à la gestion des émotions. Les auteurs ont dû faire des choix subjectifs concernant le modèle d'émotion à implémenter. L'architecture est dépendante de ce modèle et il n'est pas possible d'en utiliser un autre. L'architecture n'est pas assez complète pour le Compagnon Artificiel pour qui la partie émotionnelle est optionnelle à ce stade.

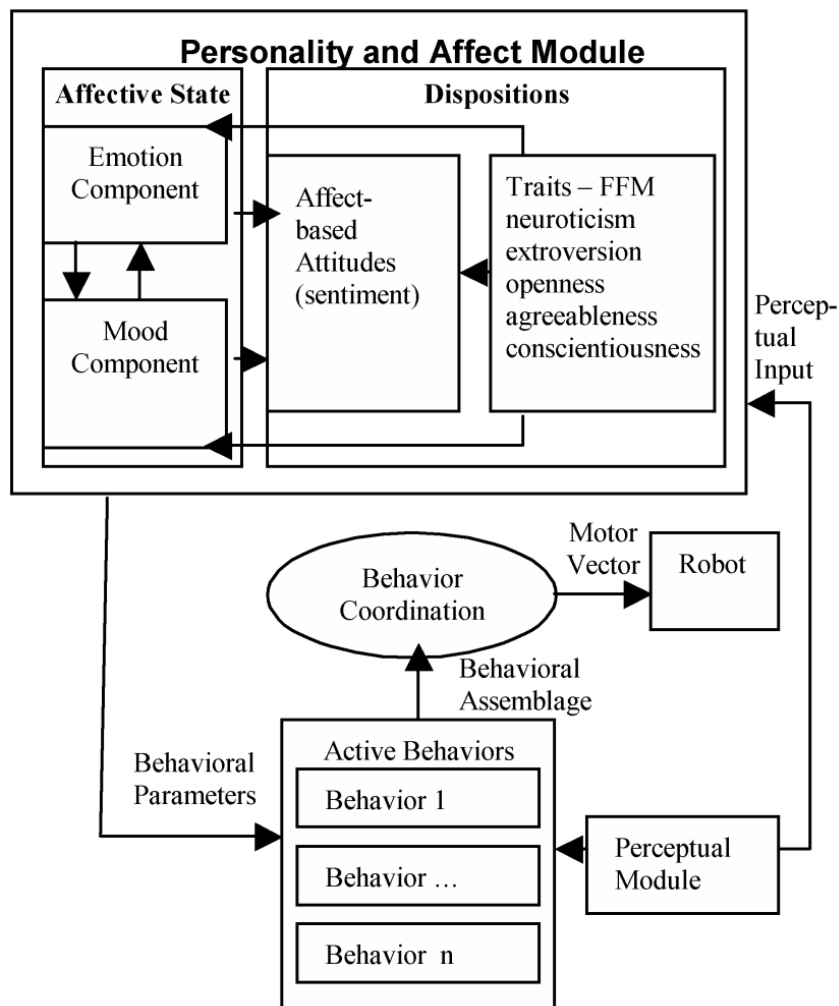


Figure 19 : Architecture TAME

Architecture iGrace

De façon complémentaire, Saint-Aimé *et al* ont défini un modèle computationnel des émotions iGrace (Saint-Aimé et al. 2009), permettant de définir le comportement approprié d'un robot ours en peluche en fonction de l'état émotionnel d'une personne. La Figure 20 montre l'architecture qui est iGrace qui est composée de trois modules :

- Le module des entrées : gère les paramètres d'entrées du système (les données résultantes des différents capteurs,
- Le module d'interaction émotionnelle : est chargé de calculer la réponse émotionnelle du robot,
- Le module des sorties : permet l'expression du comportement du robot.

Cette architecture est capable de déterminer l'émotion sous-jacente d'une phrase dite par un enfant (l'utilisateur du robot) et de générer une réponse émotionnelle appropriée.

Cette architecture est également limitée dans le cadre du Compagnon Artificiel. Cependant elle montre une organisation intéressante avec ces trois types de modules : gestion des entrées, calcul de la réponse, gestion des sorties. Cette organisation est suffisamment générale pour pouvoir ajouter facilement des entrées, des sorties et déterminer des calculs de réponses personnalisés.

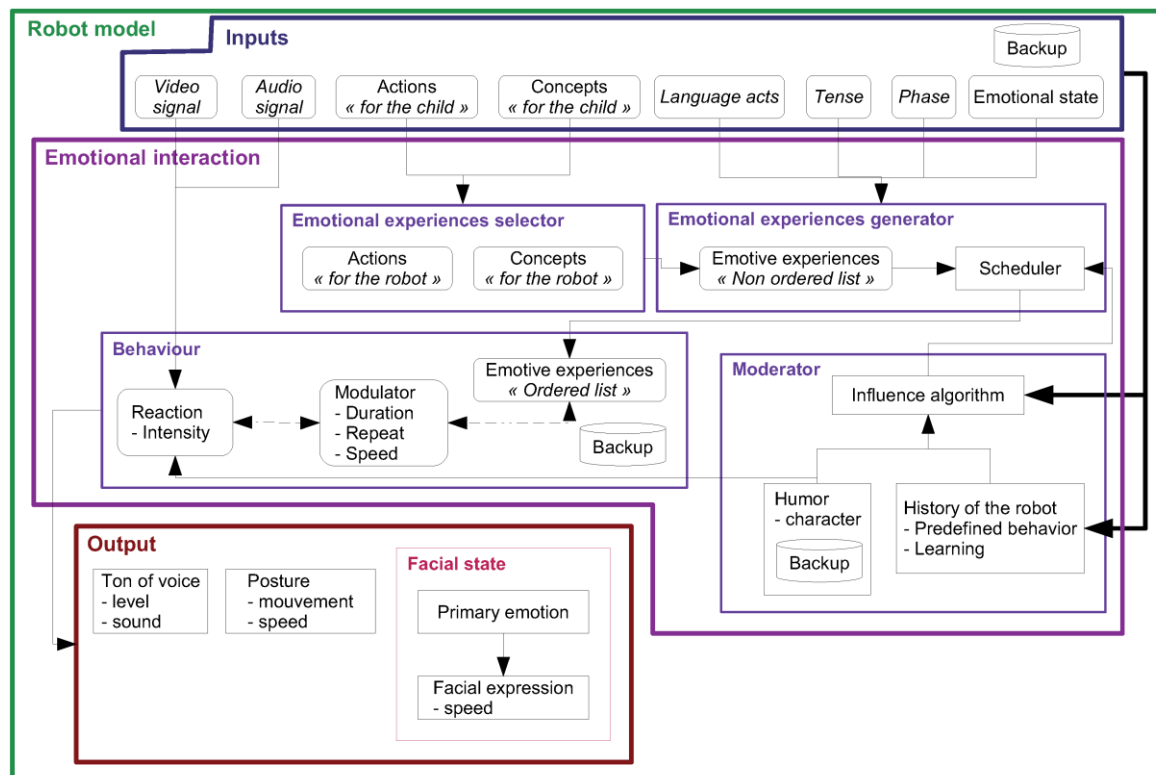


Figure 20 : Architecture iGrace

Cadriciel général pour application affective

Enfin, KangWoo *et al* ont proposé un cadriciel général (KangWoo Lee et al. 2005) qui permet également de générer un comportement en fonction d'émotions. Comme le montre la Figure 21, cette architecture affective se base sur trois modules :

- Le module d'interaction multimodale : prend en compte le caractère multimodal des entrées et des sorties. Par exemple, un humain ne communique pas seulement avec sa voix. Il utilise aussi de la gestuelle et d'autres indices non verbaux. Le robot peut aussi utiliser plusieurs modalités pour s'exprimer,
- Le module d'interaction cognitive : permet au robot et à l'humain d'avoir une connaissance partagée. Ainsi, ce module gère un modèle de connaissance, de tâche, de besoins, d'interaction, ainsi que les modèles représentant le mental du robot et la représentation de l'utilisateur,
- Le module d'interaction émotionnelle : gère les différentes réponses émotionnelles du robot. Ce module est composé d'une couche réactive, responsable des réactions immédiates (réflexes), et d'une couche délibérative, responsable des émotions indirectes suscitées par les tâches et la relation sociale avec l'utilisateur.

Concernant cette architecture, elle est également très orientée et trop spécifique pour le Compagnon Artificiel malgré sa généricité car les auteurs ont fait des choix, notamment en ce qui concerne les émotions, qui ne peuvent pas être modifiées. Elle ne peut donc pas s'adapter aux autres travaux. Mais, on retrouve cette structure intéressante de gestion des entrées, calcul de la réponse et gestion des sorties.

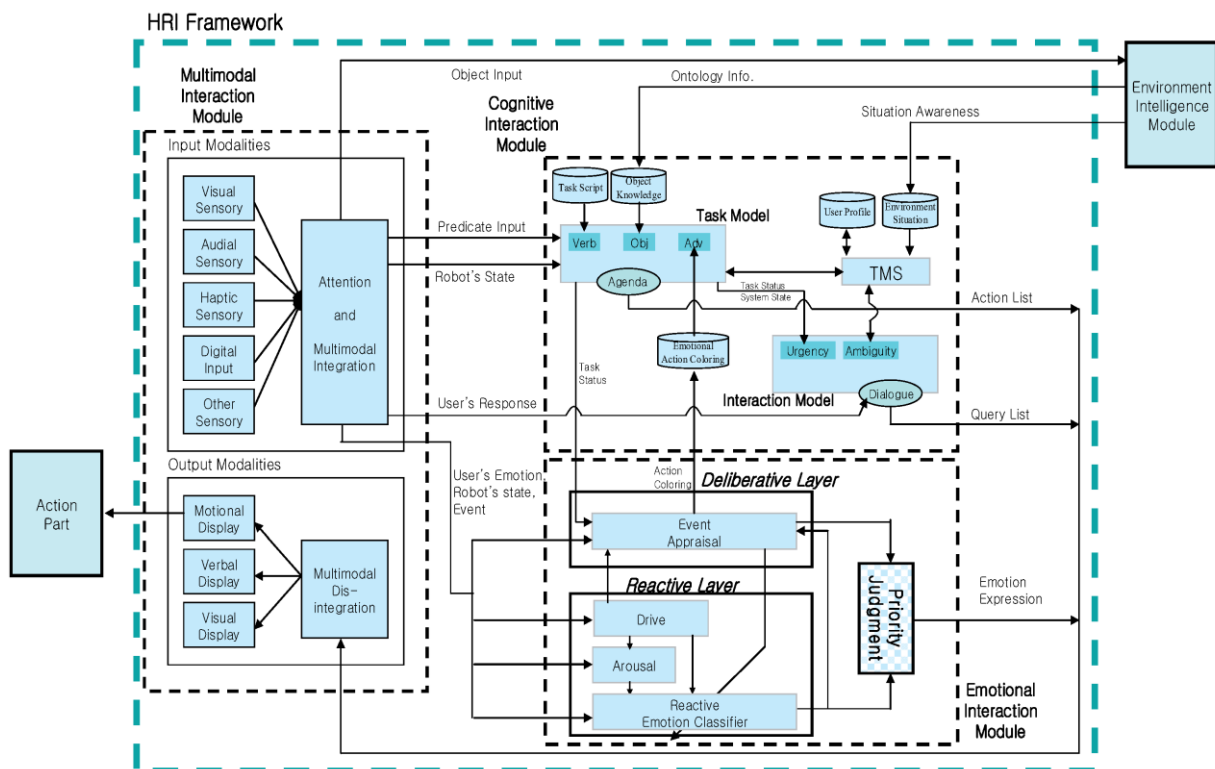


Figure 21 : Une architecture affective générale

Architecture EGO

Pour finir, Brooks et Arkin (A. G. Brooks and Arkin 2006) mettent l'accent sur l'importance de la communication non verbale dans l'interaction homme-homme ainsi que de la proxémie. C'est pour cela qu'ils s'intéressent à ces caractéristiques dans la communication homme-robot et qu'ils ont développé le concept de *behavioral overlays* pour incorporer l'affichage de la communication non verbale à des comportements préexistants. Cela signifie que le robot effectue sa tâche normale et qu'en même temps, il affiche des comportements non-verbaux. C'est le cas du robot QRIO de Sony sur lequel a été développée l'architecture EGO (A. G. Brooks and Arkin 2006), présentée sur la Figure 22. Cette architecture est composée de cinq parties principales. Une partie est chargée de gérer les différents capteurs du système, tandis qu'une autre gère les actionneurs. Une partie gère la mémorisation des informations. Une autre partie gère l'état interne et émotionnel du robot. Et enfin, une dernière partie est chargée de définir le comportement que doit exprimer le robot.

Cette architecture n'est pas assez générique pour le Compagnon Artificiel mais la structure est très intéressante car chaque groupe de module est spécialisé dans un type d'analyse. On retrouve la gestion des entrées, de la génération de la réponse et des sorties, comme la plupart des architectures. La mémorisation des informations est également gérée. Le module qui gère les émotions est superflu pour le Compagnon Artificiel. Les émotions et les comportements non verbaux résultants doivent être laissés à chaque objet numérique, ce n'est pas le rôle du Compagnon Artificiel que de s'en occuper.

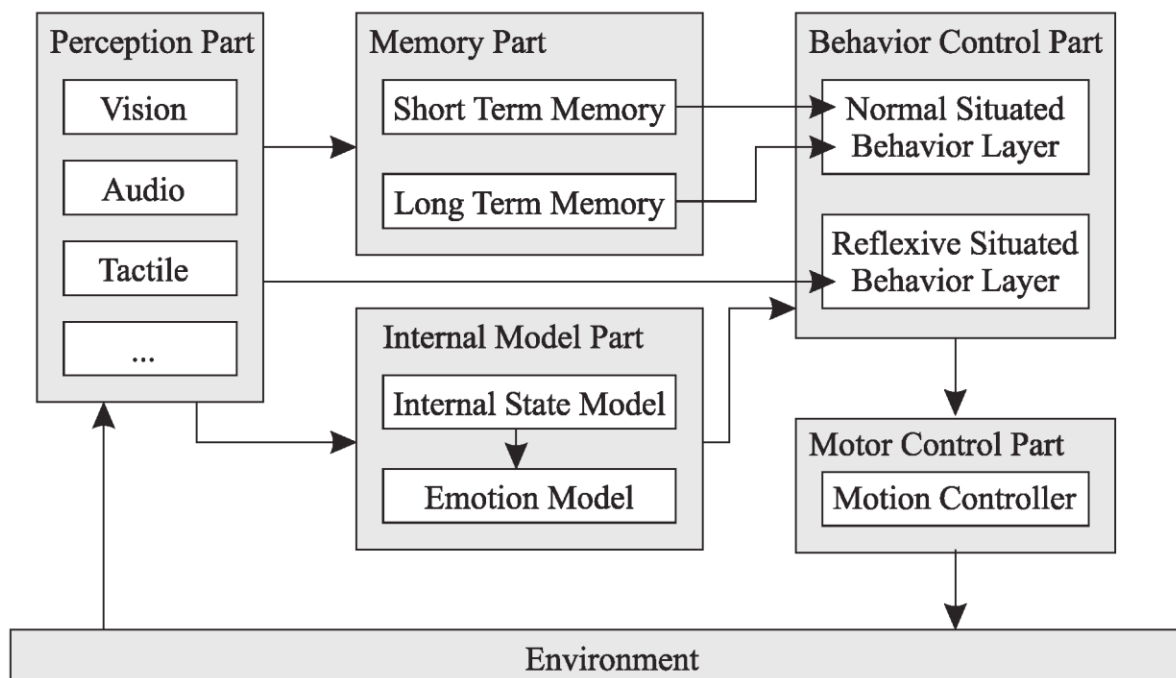


Figure 22 : Architecture EGO

C. Compagnons virtuels

Bartneck a proposé une revue de littérature (Bartneck 2002) qui explique le rôle et l'intérêt des agents virtuels et qui rappelle leurs premières utilisations dans la vie quotidienne. Ils sont apparus dans des jeux, comme Les Sims, Créatures ou Lemmings puis ensuite, en tant qu'agents pour aider les utilisateurs. Bartneck défend le point de vue que l'interaction avec des personnages de jeu peut être engageante et joyeuse car ces derniers utilisent des émotions afin d'améliorer la relation avec les utilisateurs.

Les agents virtuels sont également utilisés dans des environnements d'apprentissage. Dans ce domaine, Sansonnet *et al* ont créé un agent virtuel DIVA (Gockley et al. 2005), présentée sur la Figure 23, qui interagit sur des sites web. Ils considèrent que les agents virtuels sont intéressants car les hommes les jugent sympathiques et parce qu'ils sont des tuteurs efficaces dans le domaine de l'éducation.

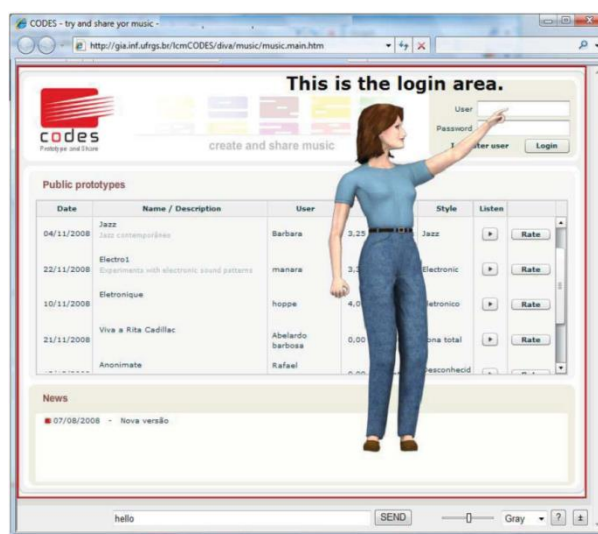


Figure 23 : Agent virtuel DIVA

Dans ce domaine, il existe aussi des agents pédagogiques animés, comme Herman the bug (Lester et al. 1997), présenté sur la Figure 24, qui est un tuteur pour des enfants.

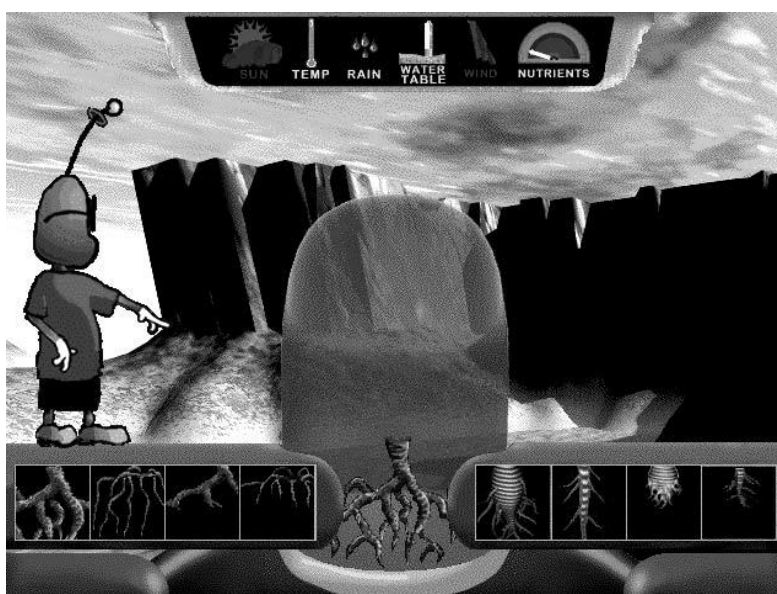


Figure 24 : Agent virtuel Herman the bug

Le domaine de la formation possède également son personnage virtuel PAT (Pesty, Webber, and Balacheff 2001) qui est basé sur le même principe que la plateforme Baghera (Chapitre II.A). C'est un agent virtuel qui permet d'aider des personnes à apprendre.



Figure 25 : Agent PAT

Parfois les agents virtuels sont utilisés en complément de robots. Par exemple, Bartneck a développé la représentation d'un robot existant Muu2 présenté dans le Chapitre II.B.1 (Bartneck 2002). La Figure 26 montre le robot et sa représentation visuelle. Le personnage virtuel a été réalisé pour faire partie d'une maison ambiante intelligente. Contrairement à un robot qui ne peut pas se déplacer, une représentation visuelle peut être disponible dans plusieurs pièces. De plus, grâce à cette migration, il peut interagir avec plusieurs utilisateurs en même temps. Et enfin, il peut exprimer des émotions plus facilement qu'un robot puisqu'il n'est pas limité physiquement.



Figure 26 : À gauche, le robot Muu2. À droite, représentation visuel de Muu2

Généralement, les personnages virtuels anthropomorphiques sont créés pour faciliter l'interaction sociale avec les humains puisqu'ils peuvent utiliser les mêmes codes. Ils sont de ce fait mieux compris et mieux acceptés. Brown *et al* (Cerezo et al. 2008) ont créé un avatar, représentant un robot guide Cherry, afin d'améliorer la communication homme-robot. Le robot et sa représentation visuelle sont présentés sur la Figure 27.



Figure 27 : Robot et représentation visuelle Cherry

Pour éviter de multiplier les représentations, Gockley *et al* ont utilisé les avantages du réel et du virtuel en mettant un visage virtuel à un robot réceptionniste : Valérie (Cassell 2001), comme le montre la Figure 28. Selon les auteurs, utiliser un visage animé à la place d'un visage mécanique est intéressant car un visage animé est plus expressif. De plus, ils considèrent qu'un visage robotique est moins fiable dans ses mouvements du fait des problèmes de mécanique. Pour finir, un personnage virtuel peut facilement changer d'apparence (coupe de cheveux, couleur des yeux), ce qui n'est pas le cas d'un robot. Or il semble important pour un utilisateur de pouvoir personnaliser la relation avec le personnage virtuel ou le robot.



Figure 28 : Robot Valérie et sa tête virtuelle

Selon Yoshikawa, pour qu'un ordinateur soit capable d'interagir avec un humain, il doit avoir des compétences de communication (Yoshikawa 1999). Plus généralement, un agent virtuel doit respecter les règles de communication des humains (Brown, Lisetti, and Marpaung 2002). C'est la spécialité des ECA (*Embodied Conversational Agents*) qui sont capables d'exprimer des émotions via le langage verbal et des expressions non verbales. Les ECA peuvent avoir une conversation naturelle avec un utilisateur (De Sevin *et al.* 2010).

Les ECA s'appliquent à tous les domaines. Par exemple, la Figure 29 montre Max, un agent virtuel permettant de contrôler un environnement domotique.



Figure 29 : Agent virtuel Max

Dans des applications plus spécialisées en interaction homme-machine, on trouve Greta (De Sevin et al. 2010), présenté à gauche sur la Figure 30. C'est un agent conversationnel animé qui peut afficher des expressions faciales et prononcer un discours. Elle prend en entrée un fichier BML ou FML, tout comme l'agent SASO's SmartBody à droite sur la Figure 30. Ces agents peuvent facilement s'interfacer avec d'autres applications et peuvent être utilisés dans différents contextes.



Figure 30 : Agents Greta et SASO's SmartBody

Greta est basée sur l'architecture SAIBA (De Sevin et al. 2010) qui signifie *Situation Agent Intention Behavior Animation*. C'est une architecture standardisée pour la génération de comportements interactifs des agents virtuels animés. La Figure 31 montre que SAIBA est composée de trois différents modules :

- Planification des intentions : définit les buts courants, l'état émotionnel et les croyances de l'agent. Les intentions sont codées en FML (*Function Markup Language*) et envoyées au module de Planification du comportement,
- Planification du comportement : définit des signaux communicatifs (paroles, expressions faciales, gestes...) qui sont encodés en BML (*Behavior Markup Language*) et envoyés au module de génération du comportement,
- Génération du comportement : réalise les comportements générés en calculant des fichiers d'animations de l'agent virtuel (en FAP-BAP).

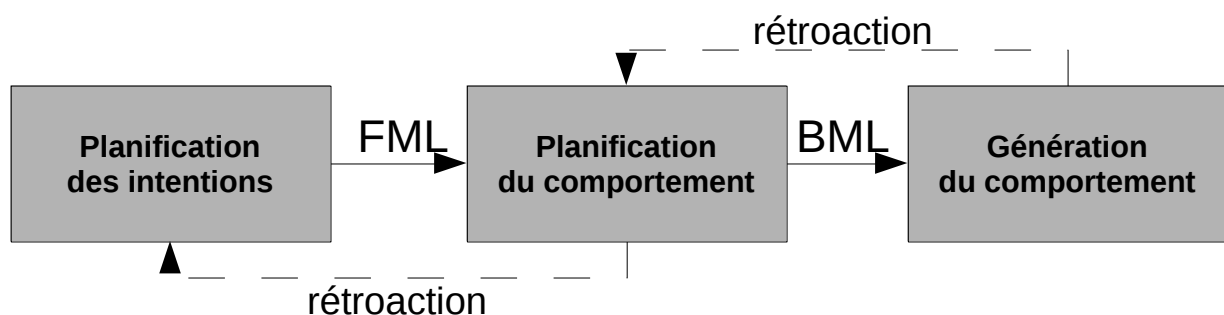


Figure 31 : Architecture de SAIBA

Greta est une adaptation de l'architecture SAIBA, comme le montre la Figure 32. Le module de planification du comportement comporte une large palette de comportements non verbaux générés à partir d'une seule intention communicative. Lorsque Greta interagit avec un utilisateur, elle est capable d'effectuer des rétroactions réactives et cognitives ainsi que des imitations de l'utilisateur, améliorant ainsi son acceptabilité.

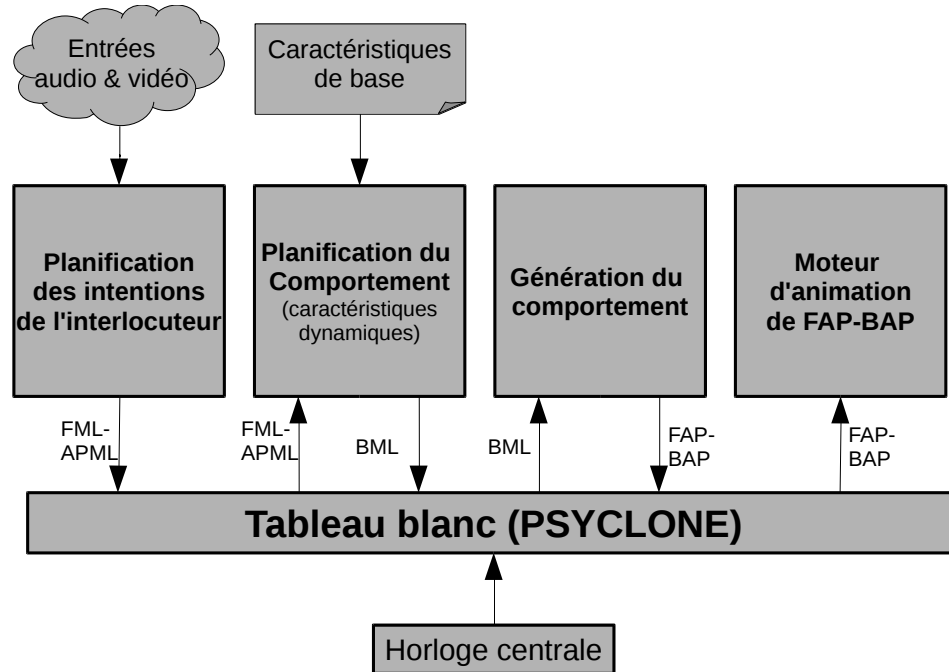


Figure 32 : Architecture de GRETA

Pour aller plus loin, Courgeon *et al* ont créé une plateforme d'agents émotionnels MARC (Courgeon, Martin, and Jacquemin 2008) dont un exemple est présenté sur la Figure 33. C'est une plateforme qui permet de contrôler en temps réel un personnage animé capable d'exprimer des expressions faciales émotionnelles. Elle permet de définir plusieurs profils expressifs pour les personnages virtuels afin de moduler en temps réel les réactions de l'agent aux entrées de l'utilisateur. Cette plateforme propose une interface de contrôle adaptable et réutilisable dans différents contextes.



Figure 33 : Plateforme d'agents émotionnels MARC

La littérature ne présente pas beaucoup d'architectures pour les personnages virtuels. Chen et Li proposent une architecture pour modéliser un environnement virtuel multi-utilisateur (Chen and Li 2006). Cette architecture apporte une autre vision de l'interaction homme-avatar en proposant un nouveau mécanisme d'interaction entre utilisateurs réels et utilisateurs virtuels basés sur un simple modèle psychologique permettant d'interagir de façon personnelle et émotionnelle. Le modèle psychologique se fonde sur les émotions, la mémoire et les traits personnels. Il affecte le comportement (mouvements et gestuelle) de l'agent. La Figure 34 montre que le système est composé de quatre modules principaux :

- *Scene and behavior input module* : utilise une description géométrique de la scène et génère les mouvements que les agents doivent effectuer. Ce module gère également les dialogues disponibles, une bibliothèque de mouvements prédéfinis et l'état mental par défaut de tous les agents virtuels,
- *System monitoring module* : surveille le statut des agents (lieux, futurs mouvements, état émotionnel,
- *Agent simulation module* : est en charge de la création des agents virtuels et de la simulation de leurs comportements lorsqu'ils interagissent avec des avatars (agents virtuels et utilisateurs réels),
- *Client module* : fournit une interface graphique pour afficher le monde 3D et pour permettre à l'utilisateur réel d'interagir avec les agents virtuels.

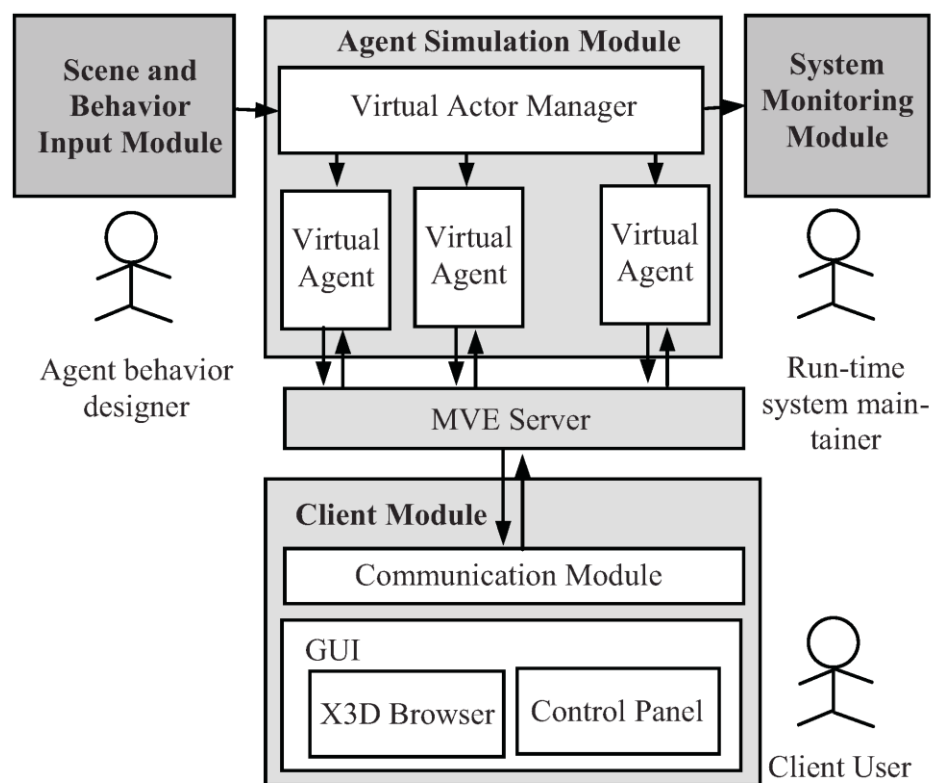


Figure 34 : Architecture Multi-user Virtual Environment

Enfin, Sansonnet *et al* proposent un cadriciel pour les *Assisting Conversational Agent* (Sansonnet, Leray, and Martin 2006). Le rôle d'un assistant est d'aider une personne lorsqu'elle se trouve en position d'échec. Selon ces auteurs, il doit exécuter trois fonctions majeures :

- La fonction de compréhension : elle analyse le discours de l'utilisateur pour comprendre sa requête. Il semblerait que la voie orale soit la plus importante car c'est celle qu'un être humain utilise dans un moment de difficulté,
- La fonction de compétence : elle fournit la réponse à la requête, il doit donc être capable de raisonner,
- La fonction de présence : il est montré que la présence est très importante dans le cas d'utilisateurs novices.

La Figure 35 montre que le cadriciel est composé de ces trois outils sous la forme de trois agents : le *Dialogical Agent* (Module *Analyser*), le *Rational Agent* (Module *Mediator*) et l'*Embodied Agent* (Module *Presenter*). L'*Analyser* doit traduire la langue naturelle en une requête formelle. Le *Mediator* est une représentation symbolique dynamique de la structure et du fonctionnement du composant actuel, sur lequel le *Rational Agent* doit résoudre la requête formelle. Enfin, le *Presenter* doit retourner la réaction de l'agent en utilisant des modalités multiples.

Les travaux sur les agents virtuels incluent beaucoup les émotions de l'homme ou de la machine. L'idée défendue est que l'homme doit être pris en compte. Les architectures proposées sont trop spécifiques pour le Compagnon Artificiel. Il semblerait qu'un agent virtuel est autonome et doit gérer lui-même sa personnalité et ses émotions, comme pour les robots.

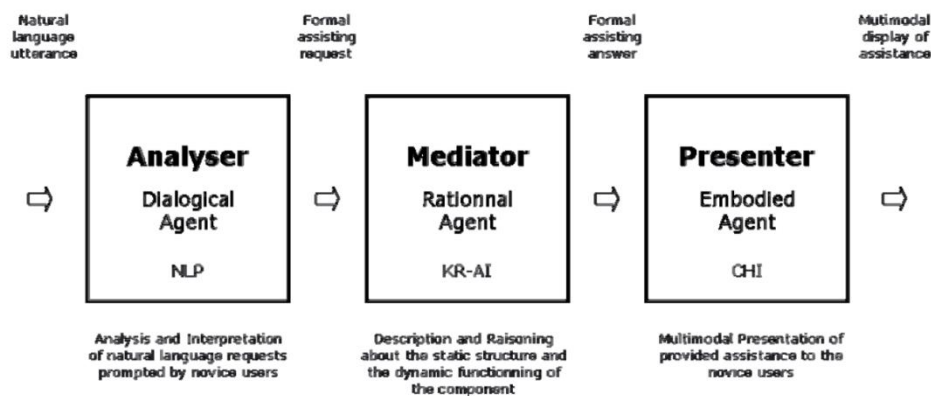
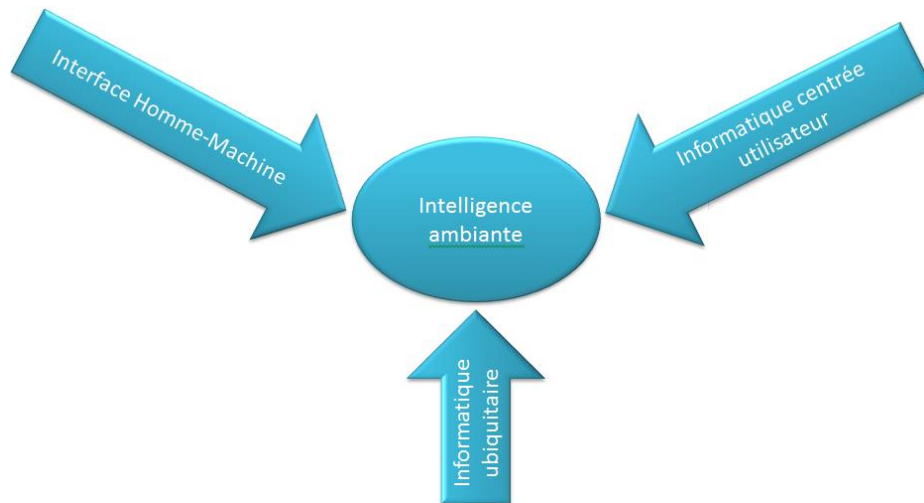


Figure 35 : Architecture d'un *Assisting Conversational Agent*

D. Intelligence ambiante

Ce chapitre introduit la notion d'intelligence ambiante. La première partie explique les origines de l'intelligence ambiante à travers trois disciplines : l'interaction homme-machine, l'informatique centrée utilisateur et l'informatique ubiquitaire. La deuxième partie explique ce qu'est l'intelligence ambiante. La troisième partie présente les travaux apportant des solutions ou des recommandations dans certains domaines de l'intelligence ambiante. La quatrième partie présente des applications dont la problématique se rapproche en tout ou partie de celle du Compagnon Artificiel. Enfin, la dernière partie présente les perspectives de recherche qui peuvent être effectuées dans ce domaine.



1. Origines de l'intelligence ambiante

a) Interaction homme-machine

A ses débuts, dans les années 70, l'interaction homme-machine n'est possible qu'à travers des interfaces textuelles, en ligne de commande. Au début des années 80, la première interface graphique est créée par Xerox. Quelques années plus tard, Apple commercialise des ordinateurs avec interfaces graphiques et souris, démocratisant ainsi l'outil informatique qui depuis ne cesse d'évoluer. Le schéma « clavier-souris » tend même à disparaître pour des moyens de communication plus naturels comme la communication tactile.

Les mobiles sont de plus en plus utilisés pour l'interaction homme-machine. Par exemple, Ballagas *et al* (Ballagas et al. 2006) ont utilisé un smartphone en tant que télécommande d'un système complexe. Selon eux, les téléphones mobiles sont tellement présents dans notre quotidien qu'il faut les utiliser pour d'autres usages. En effet, ils peuvent permettre de rendre un système *affordant* puisque les utilisateurs n'ont plus besoin d'apprendre le fonctionnement de leur « télécommande ». Mais puisque les téléphones s'utilisent différemment des ordinateurs, la philosophie des interfaces doit changer. Gong *et al* (Gong and Tarasewich 2004) ont fourni un ensemble de règles pour créer des interfaces pour les mobiles. De nouveaux concepts ont été utilisés comme le retour tactile (Poupyrev, Maruyama, and Rekimoto 2002) qui permet de donner une meilleure compréhension de la machine. Rico *et al* (Rico and Brewster 2010) ont étudié la gestuelle des humains en fonction des lieux et des personnes présentes pour connaître les mouvements acceptables en interaction homme-téléphone. Scoditti *et al* (Scoditti et al. 2011) utilisent la surface tactile et l'accéléromètre pour améliorer la sélection d'information tandis que Francone *et al* (Francone and Nigay 2011) exploitent le suivi du visage pour créer une nouvelle technique d'interaction avec les téléphones.

Les téléphones mobiles ne sont pas seuls responsables de l'évolution des interfaces graphiques. Les manettes sans fil, par exemple, ont ouvert des champs d'applications grâce à leurs

accéléromètres qui calculent la position d'un curseur en fonction des mouvements, aux secousses ou à l'inclinaison du périphérique (Levin and Yarin 1999). Cela permet d'imaginer de nouveaux types d'applications dont le contrôle est impossible avec une simple souris.

La technologie a également permis au corps entier de devenir un élément de contrôle d'interface, comme dans des environnements virtuels de collaboration (Shae, Tseng, and Leung 2001). Ces types d'interface, qui n'utilisent plus la souris ni le clavier, mais qui tentent d'utiliser des modalités d'interaction plus naturelles, sont appelés *Natural User Interface* ou NUI (Goth 2011). Un objet a rendu possible une évolution très rapide des NUI depuis 2010 : la Kinect de Microsoft. D'après Goth (Goth 2011), c'est « le début de la fin de la souris ». Cet objet a tout pour plaire dans le monde de la recherche puisque qu'il est bon marché et qu'il propose un SDK. L'homme entier devient le clavier et la souris. Elle a permis de créer facilement des applications de reconnaissance de gestes pour contrôler la télévision (Boulabiar et al. 2011) par exemple, des systèmes de réhabilitation physique pour les personnes avec handicap moteur (Huang 2011), des nouvelles approches pour faciliter l'apprentissage (Villaroman, Rowe, and Swan 2011) ou la construction de nouvelles interactions avec des téléphones mobiles (Norrie and Murray-Smith 2011).

Enfin, l'évolution également évolué en terme de nombre d'utilisateurs. Par exemple, les tables collaboratives ont permis à plusieurs utilisateurs d'interagir ensemble et en même temps (S.-G. Kim, Kim, and Lee 2012).

L'évolution n'est pas terminée et le domaine de l'interaction homme-machine va sans doute encore progresser beaucoup ces prochaines années. Jain et al (Jain, Lund, and Wixon 2011) indiquent que l'interaction entre l'homme et le reste du monde est multimodale. Ainsi, pour que l'interaction soit une expérience naturelle pour l'homme (cf. note de bas de page 7), il faut préserver cette richesse de modalité et rendre la technologie transparente⁹. En ce sens, ils citent un ensemble de travaux concernant le discours, la gestuelle, le toucher et la manipulation 3D au sein de la réalité augmentée. Même certains outils vieux de 40 ans comme les télécommandes des télévisions sont revisités (Vo et al. 2011). Certaines interfaces deviennent immersives et imposantes par leur taille, comme par exemple Tweetrix (Freeman et al. 2012), un jeu de Tétris où l'homme joue avec son corps ou encore the WILD Room (Beaudouin-Lafon 2011), une pièce contenant un mur composé d'écrans d'ordinateur pouvant être contrôlés par une table interactive, par un système de détection de mouvement ou encore par des PDAs, des tablettes tactiles... Ce système réussit le pari des NUI en permettant à l'humain qui le contrôle des mouvements naturels.

Mais pour l'instant ces interfaces ne sont pas utilisées par le grand public, certainement parce que les prix sont trop élevés et parce que leurs dimensions ne sont pas encore à l'échelle d'un domicile. Il est probable que la réalité économique freine également l'intégration de ces nouvelles interfaces.

b) Informatique centrée utilisateur

A la fin des années 90, la société TEPCO (domaine du nucléaire) a constaté que les hommes étaient responsables de la plupart des accidents. Ils ont alors créé un système de prédiction pour éviter ces accidents (Kawano 1997). Pour ce faire, ils ont dû rajouter les utilisateurs du système dans le système lui-même. Cela s'appelle l'informatique centrée utilisateur. A la fin des années 2000, il est devenu évident qu'il fallait rajouter l'humain dans la boucle computationnelle (Tedre 2008), au point que l'humain devrait même être une part du programme informatique (Elgammal 2006). Cela implique de connaître l'être humain et de pouvoir en faire un modèle. Malheureusement, même si l'humain est au cœur de nombreuses recherches (Lew et al. 2012), même si ces recherches sont liées à

⁹ L'« informatique transparente » vient de l'informatique ubiquitaire qui signifie que l'informatique est omniprésente, intégrée à des objets de la vie courante. L'homme interagit directement avec ces objets sans se préoccuper de la technologie qui se cache derrière.

différentes disciplines (philosophie, psychologie, biologie, neurologie, médecine...), la connaissance actuelle n'est pas complète.

Ainsi, il n'est pas possible de créer un programme qui observe l'humain et qui en déduit des informations. A ce stade, la communication est fondamentale car elle permet de réels échanges entre la machine et l'humain. Capelli *et al* (Cappelli and Giovannetti 2004) ont étudié les principales modalités utilisées : la parole, la gestuelle, les expressions faciales, le regard et la communication non verbale. Ils ont conclu que la parole est le moyen de communication le plus efficace car elle est naturelle pour l'humain. Mais la reconnaissance vocale n'est pas encore au point (Ndiwalana et al. 2003) car le système doit connaître en avance les mots à reconnaître. Par exemple, Kuhn *et al* (Kuhn et al. 2004) ont développé un système qui permet de choisir le programme TV, sans utiliser de télécommande, uniquement basé sur les mots. En effet les auteurs considèrent qu'une télécommande n'est pas adaptée à toutes les personnes et que les interfaces entre les humains et les objets doivent changer car la technologie évolue mais pas les interfaces. Cependant ce système fonctionne seulement si l'utilisateur prononce les phrases prévues et c'est une limitation importante de leur solution.

Ainsi l'idée qui semble la plus adaptée actuellement est que les nouveaux systèmes doivent être déployés sur des objets existants de façon transparente car les humains ne doivent pas s'adapter à de grands changements mais doivent apprendre progressivement. Ainsi, les systèmes centrés utilisateurs doivent respecter les besoins des utilisateurs et devraient être un complément des capacités des humains (Talbert 1997).

Cependant l'informatique centrée utilisateur pose des problèmes d'acceptabilité dont il faut tenir compte. En effet, le système doit proposer une interaction naturelle que l'humain peut prévoir (Kawano 1997). Les humains doivent être capables de prédire le comportement du système et *vice versa*.

c) Informatique ubiquitaire

En 1991, Weiser (Weiser 1991) a proposé une nouvelle vision des systèmes informatiques en se basant sur l'infrastructure des téléphones mobiles. Il a imaginé un monde où les humains pouvaient accéder à toutes les sources d'informations tout le temps et partout, un réseau distribué de milliers de dispositifs numériques intégrés qui pourraient communiquer ensemble. Ce serait une informatique invisible, centrée sur l'utilisateur et non invasive. Ce fut la première vision de l'informatique ubiquitaire.

2. Définitions de l'intelligence ambiante

L'intelligence ambiante Aml est un domaine né de l'informatique ubiquitaire et de l'interaction homme-machine (Jacquet 2006). Son objectif est d'améliorer la qualité de vie des gens en répondant aux besoins spécifiques des utilisateurs (E. Aarts 2004) sans leur imposer la technologie (Pruvost and Bellik 2009). Cela nécessite une prise en compte du contexte (E. Aarts 2004) afin de pouvoir anticiper les besoins des utilisateurs. CAMPS (Qin, Suo, and Shi 2006) est par exemple un *middleware* multi-agent spécialisé dans la prise en compte du contexte.

Aarts (Emile Aarts and Wichert 2009) résume un environnement intelligent comme :

- La perception de l'environnement par un réseau de capteurs qui récupère les données pertinentes de l'environnement,
- Un accès ubiquitaire où les informations doivent être accessibles n'importe quand, n'importe où et sur n'importe quel support d'information,
- Une interaction naturelle, où le contrôle de l'environnement n'est plus centralisé sur des PCs. Une nouvelle forme naturelle d'interaction aura un impact crucial dans le quotidien des utilisateurs, comme le discours, la reconnaissance de geste, la vision.

L'Aml favorise les interactions sociales (De Ruyter 2011) car elle apporte une sorte de présence à l'utilisateur. Les recherches en Aml ont débuté en 2001, lorsque l'ISTAG¹⁰ (Ducatel et al. 2001) a fait un appel à des groupes de recherches pour réfléchir à la place future de la technologie, à partir de 4 scénarii de la vie quotidienne de 2010. Les résultats de ces réflexions ont permis de mettre en avant les premières caractéristiques qui peuvent permettre l'acceptation de l'Aml par la société. Celle-ci devrait :

- Faciliter le contact humain¹¹,
- Être orientée vers l'amélioration de la communauté et de la culture,
- Aider à construire une connaissance et des compétences de travail, une meilleure qualité de travail, la citoyenneté et le choix des clients,
- Inspirer la confiance,
- Être durable et facile à vivre,
- Être contrôlable par des gens ordinaires.

L'intelligence ambiante a plusieurs définitions. Par exemple, pour Sadri (Sadri 2011), l'intelligence ambiante signifie que les environnements aident les personnes tout en étant invisibles, interconnectés, adaptables, dynamiques, intégrés et intelligents. Cela suit le même principe que les NUI avec la disparition de la souris et du clavier. Il considère que les habits, les appareils ménagers et les meubles pourraient communiquer entre eux, avec les humains, et avec les objets des autres. Pour Aarts (E. Aarts 2004) (Emile Aarts and Wichert 2009), l'intelligence ambiante vise à utiliser l'intégration de l'informatique ubiquitaire pour réaliser des environnements sensibles et adaptatifs qui réagissent aux actions des personnes et des objets pour répondre aux besoins des utilisateurs. Enfin, pour Issarny et al (Issarny et al. 2005), l'intelligence ambiante permettrait aux utilisateurs d'accéder à des contenus et des services de façon immédiate et universelle. Cette dernière vision des choses engendre des problèmes de sécurité (Sadri 2011) liés à la possession de données personnelles facilement accessibles par les périphériques de l'environnement. Toma *et al* (Toma et al. 2011) indiquent que l'intelligence ambiante doit également apporter des solutions économiquement viables.

Pour que le système informatique soit facile à utiliser pour les utilisateurs, les actions ne sont plus déterminées par les fonctionnalités des dispositifs numériques mais par des objectifs (Emile Aarts and Wichert 2009). C'est le système qui déterminerait automatiquement les actions à effectuer en fonction des objectifs énoncés par l'utilisateur.

L'intelligence ambiante pose des nouveaux défis concernant la gestion des dispositifs numériques qui doivent communiquer les uns avec les autres et coopérer pour répondre aux besoins des utilisateurs (Emile Aarts and Wichert 2009):

- Chaque dispositif numérique doit être autonome,
- L'intégration de l'informatique doit être transparente dans des équipements existants,
- Il faut éviter de concevoir des composants centraux,
- L'intégration des nouveaux dispositifs est dynamique,
- Les dispositifs se reconfigurent automatiquement,
- Il faut apporter une solution aux conflits.

Aarts et Marzano ont déterminé les 5 éléments clefs de l'Aml (E. Aarts 2004) :

- Intégration (*embedded*) : plusieurs dispositifs numériques sont intégrés à l'environnement

¹⁰ European Community's Information Society Technology Advisory Groups

¹¹ Aarts et de Ruyter (Emile Aarts and De Ruyter 2009) mettent en garde sur le fait que les nouvelles technologies ne devraient pas se complexifier mais devraient contribuer au développement de produits et de services *easy to use* et *simple to experience*.

Prise en compte du contexte (*context_aware*) : le système peut reconnaître les utilisateurs et le contexte. La prise en compte du contexte (lieux et activités) est un élément clef dans la construction d'applications de l'intelligence ambiante (Cook, Augusto, and Jakkula 2009),

- Personnalisation (*personalized*) : le système peut s'ajuster à l'utilisateur pour répondre à ses besoins,
- Adaptation (*adaptive*) : le système peut changer en réponse à l'utilisateur,
- Anticipation (*anticipatory*) : le système peut anticiper les désirs de l'utilisateur sans qu'il lui demande.

Les Tableau 1 et Tableau 2 présentent le point de vue d'Aarts et de Ruyter (Emile Aarts and De Ruyter 2009) sur la progression des éléments clefs des infrastructures existantes. Selon Aarts (E. Aarts 2004), les environnements Aml devraient manifester une forme d'émotion pour sembler vraiment intelligents. Comme le montre le Tableau 2, l'intelligence sociale intègre des critères liés au domaine social et aux émotions. Une étude (Dybala et al. 2010) a comparé un agent neutre et un agent qui racontait des blagues, à des moments opportuns. Les études ont montré que le deuxième agent était mieux évalué. De plus, l'environnement devrait être capable de détecter les humeurs des utilisateurs et d'y réagir. Ce domaine de recherche s'appelle l'*affective computing*.

Tableau 1 : Positionnement de l'intelligence ambiante, selon Aarts et Ruyter

Mobile	Pervasive	Ambient
• Portable • Wireless • Networked • Location sensitive • Secure	• Ubiquitous • Interactive • Interoperable • Distributed • Scalable	• Embedded • Context aware • Personalized • Adaptive • Anticipatory

Tableau 2 : System Intelligence versus Social Intelligence, selon Aarts et Ruyter

System intelligence	Social intelligence
• Context aware • Personalized • Adaptive • Anticipatory	• Socialized • Empathic • Conscious

L'organisation générale des Aml est composée de trois types d'éléments (Emile Aarts and Wichert 2009) :

- Les éléments qui gèrent les entrées (perception de l'état de l'environnement),
- Les éléments qui gèrent les sorties (action sur l'environnement),
- Les éléments qui sont capables de déterminer les actions à effectuer en fonction des entrées. Ces troisièmes éléments incluent la modélisation de l'utilisateur, la prédiction et la reconnaissance des activités et la prise de décision (Cook, Augusto, and Jakkula 2009).

Catenazzi *et al* (Catenazzi et al. 2012) a proposé en 2012 un ensemble de directives pour la création de système d'intelligence ambiante pour le partage d'activités. Le système doit avoir les propriétés suivantes :

- *Perceivable* : les utilisateurs doivent être capables de percevoir les informations qui viennent de l'environnement,
- *Understandable* : les utilisateurs doivent comprendre les informations perçues,

- *Actable* : les utilisateurs doivent pouvoir agir de façon intentionnelle sur leur environnement. Cela signifie qu'ils doivent disposer d'outils pour envoyer des informations à l'environnement,
- *Comfortable* : les utilisateurs ne doivent pas ressentir de contraintes physiques et doivent être à l'aise,
- *Pleasurable* : les utilisateurs doivent avoir une expérience d'interaction positive en utilisant le système,
- *Safe and secure* : la conception du système doit minimiser et prévenir les actions dangereuses et éviter les failles de sécurité du système,
- *Interoperable* : Les utilisateurs doivent pouvoir échanger des actions et des messages.

3. Problématiques pertinentes

Le domaine de l'intelligence ambiante pose des problématiques qui sont proches de celles du Compagnon Artificiel notamment en ce qui concerne l'acceptabilité, l'éthique, l'intelligibilité, l'interaction entre l'humain et son environnement, l'utilisation du monde réel et du monde virtuel, la personnalisation de l'interaction et enfin la sécurité du système. Cette partie présente des travaux qui apportent une solution ou des recommandations pour chacune de ces problématiques.

Acceptabilité

Selon Pruvost et Bellik (Pruvost and Bellik 2009), l'acceptation des utilisateurs est cruciale et ne peut être atteinte que si le système fournit une interface efficace pour contrôler l'environnement. Deux aspects majeurs affectent l'acceptation des environnements Aml (E. Aarts 2004) :

- La façon dont l'interface se conforme aux attentes et habitudes des utilisateurs,
- La capacité du système à s'adapter à l'utilisateur et à son environnement.

Ethique

Les environnements intelligents posent de nombreux problèmes éthiques, comme le souligne Coroama *et al* (Coroama, Bohn, and Mattern 2004). En effet, il faut faire attention au respect de la vie privée lors de la conception de tels systèmes. Comment assurer que les informations sont en sécurité ? Les utilisateurs vont-ils se sentir surveiller constamment ? Comment les mettre en confiance face à ce type de système ?

Intelligibilité

Dey *et al* (Dey 2009) expliquent que les utilisateurs se font un modèle mental du fonctionnement d'une application. Si le comportement de l'application ne respecte pas ce modèle et si l'application a un fonctionnement que l'utilisateur ne peut pas comprendre, alors elle manque d'intelligibilité. Ce manque provoque un manque de confiance des utilisateurs et probablement un problème d'acceptabilité.

Interaction humain-environnement

L'interaction humain-environnement est la nouvelle tendance née de l'informatique ambiante, à cause de l'invisibilité de l'informatique. L'interaction n'est plus effectuée entre l'homme et un ordinateur, mais entre l'homme et son environnement (Streitz 2012). Le système informatique doit d'ailleurs prendre en compte plusieurs personnes dans l'interaction (Crandall and Cook 2009). Plusieurs personnes interagissent avec plusieurs objets et cela pose de nombreux problèmes.

Monde réel et monde virtuel

Les environnements intelligents ne sont pas réservés au monde physique. Gardner *et al* (Gardner, Richter, and Härmä 2012) soulignent que les nouvelles habitudes des hommes doivent être prises en compte, comme l'utilisation des smartphones et que le monde virtuel fait partie intégrante de l'environnement.

Personnalisation

Il est important que les systèmes d'intelligence ambiante connaissent les préférences des utilisateurs, leurs intentions et leurs besoins pour savoir comment y répondre (Cook, Augusto, and Jakkula 2009). Ils doivent également respecter les codes sociaux des humains et ne pas intervenir à un moment inopportun. Ainsi, le système s'adapte vraiment à la personne qui peut en tirer un bénéfice maximum.

Sécurité

Les problèmes de sécurité doivent être gérés pour protéger la vie privée des utilisateurs. Ce sujet de recherche a donné lieu à un projet européen SERENITY (*System Engineering for Security and Dependability*) (Law and Havinga 2011) qui avait pour objectif de mettre en place un cadre permettant de sécuriser les infrastructures des systèmes Aml. Il existe également l'architecture AETHER (Argyroudis and O'Mahony 2004) qui est spécialisée dans la sécurité des environnements intelligents à domicile.

4. Applications

La création d'applications intelligentes complique le processus du développement logiciel (Preuveneers and Novais 2012). Il a donc fallu repenser les développements. La recherche dans le domaine des Aml est très active et il existe maintenant beaucoup d'applications : assistance d'opération virtuelle, des environnements domotiques pour les personnes âgées, des maisons intelligentes, des applications de surveillance et d'assistance, dans le domaine de l'éducation (Emile Aarts and Wichert 2009)(Cook, Augusto, and Jakkula 2009)(Aldrich 2012)(Cook 2009).

Cette partie présente quelques applications dont la problématique est proche de celle du Compagnon Artificiel : gestion des préférences des utilisateurs, gestion de la proximité, gestion du contexte et interface adaptative, appartement intelligent pour la sécurité et la protection des habitants, systèmes de dialogue, gestion flexible d'une maison intelligente, de contrôle de l'environnement, interface de contrôle pour la maison et collaboration d'objets numériques.

Préférences des utilisateurs

Les recherches sur l'étude des préférences des utilisateurs permettent de créer des systèmes qui peuvent s'adapter aux utilisateurs. C'est une sorte de personnalisation et permet de faciliter l'acceptabilité. San Martín *et al* (San Martín et al. 2010) fournissent une architecture de gestion de contexte avec des techniques d'apprentissage afin d'inférer automatiquement les préférences des utilisateurs. Les informations du contexte sont représentées par des ontologies et le système est organisé avec une architecture orientée service. Le système informatique étudie le comportement de l'utilisateur pour apprendre ses préférences et pouvoir anticiper ultérieurement.

Des travaux menés par Pruvost et Bellik (Pruvost and Bellik 2009) s'intéressent également aux préférences des utilisateurs mais leur laisse la possibilité de choisir eux-mêmes leurs préférences. Les auteurs proposent une vision de l'intelligence ambiante basée sur la collaboration multi-agent et sur des *widgets* multimodaux qui fournissent des interfaces utilisateurs mobiles. Un cadre a été créé pour prendre en compte les propriétés des modalités d'entrée, des modalités de sortie et la relation entre les entrées et les sorties. Soit le système s'adapte automatiquement en fonction des habitudes

et des préférences des utilisateurs, soit ce sont les utilisateurs qui configurent le système. Les utilisateurs peuvent relier les entrées et les sorties et sont les architectes de leur système.

Gestion de la proximité

Ce travail (Jacquet 2006) permet de présenter des informations en fonction des personnes qui se trouvent à proximité des dispositifs numériques. Par exemple, dans les aéroports, les panneaux d'affichage indiqueraient les informations des vols en fonction des personnes à proximité des panneaux. Cette plateforme est mise en place grâce au modèle KUP (Jacquet, Bellik, and Bourda 2007). Ce travail se base sur deux phases : le choix de l'information en fonction de l'utilisateur présent et le choix du dispositif sur lequel afficher l'information. Ces deux phases sont opportunistes : elles surviennent au gré des déplacements des utilisateurs.

Gestion du contexte et interface adaptative

Des travaux effectués dans le cadre du projet ATRACO (Goumopoulos et al. 2008) ont permis de mettre en place une interface graphique (Dooley et al. 2011) qui s'adapte en fonction du contexte. L'affichage graphique s'effectue sur un périphérique proche de la personne et affiche des informations pertinentes par rapport au contexte.

Appartement intelligent pour la sécurité et la protection des habitants

L'entreprise Philips a créé CareLab (Ruyter and Pelgrim 2007), un appartement équipé d'un réseau de capteurs pour tester l'acceptation de ce type de solutions avant de les déployer. Leur point de vue sur l'Aml se concentre sur la sécurité et la protection de l'environnement personnel et la stimulation et la maintenance d'une vie active des personnes âgées.

Systèmes de dialogue

Des travaux (Minker, López-Cózar, and McTear 2009) ont permis de mettre au point un système de dialogue pour les environnements intelligents afin de permettre aux utilisateurs d'interagir de façon naturelle, en dialoguant. Selon Minker *et al* : « *Des systèmes intelligents devraient être facile à utiliser, non intrusifs et exploiter les moyens de communication les plus naturels. Indéniablement, améliorer la communication et les capacités d'assistance augmente la convivialité et l'acceptabilité sociale de ce genre de système* ».

Gestion flexible d'une maison intelligente

Ce travail (Turner 2011) fournit un cadre pour gérer de façon flexible les maisons intelligentes. Il permet de contrôler les dispositifs numériques et les services à plusieurs niveaux :

- Niveau composant : les dispositifs et les services sont intégrés dans une architecture flexible. Cela permet d'ajouter de modifier ou de supprimer facilement des dispositifs,
- Niveau de contrôle (*police*) : des règles peuvent être définies pour définir la façon dont la maison doit réagir à des événements,
- Niveau but : les utilisateurs peuvent définir des buts de hauts niveaux pour déterminer comment la maison doit se comporter. Lorsqu'un but doit être réalisé, le système choisit l'ensemble des règles à suivre.

L'utilisateur peut intervenir à deux niveaux : but et contrôle. Le langage utilisé n'est pas le même dans les deux cas.

Langage de contrôle de l'environnement

Ce travail (García-Herranz, Alamán, and Haya 2010) permet à l'utilisateur de décider du comportement de son environnement en utilisant un langage basé sur des règles ECA (*Evenement Condition Action*).

Interface de contrôle pour la maison

Turner (Weingarten, Blumendorf, and Albayrak 2010) propose en 2011 une interface utilisateur multimodale du système d'exploitation de la maison. L'interface de contrôle permet d'afficher quatre types d'informations :

- Le *lobby* fournit des informations générales sur l'environnement : pièces disponibles, données des capteurs et connexion des dispositifs numériques,
- Le *Home Control Center* fournit un accès direct à tous les périphériques simples. Il est possible de contrôler chaque dispositif mais également de contrôler leur consommation électrique individuellement,
- L'*Assistants area* permet de lancer les applications disponibles via le *Home Control Center*,
- Le *Message Center* stocke toutes les informations qui doivent être transmises à l'utilisateur pendant l'interaction avec le système venant du Home Control Center et des assistants. Par exemple, il peut y avoir une notification indiquant qu'un nouveau dispositif numérique a été trouvé.

Cette interface se focalise sur la simplicité. Il ne faut pas polluer l'application avec des informations compliquées qui n'intéressent pas forcément les utilisateurs. Leur système est développé avec le Multi-Access Service Platform (MASP, masp.dai-labor.de) qui permet de créer des interfaces ubiquitaires pour les environnements intelligents et permet de créer des interactions multimodales distribuées. L'interface peut être personnalisée et est pensée pour être facile à utiliser.

Collaboration d'objets numériques

Les projets permettant de faire collaborer des objets numériques sont nombreux et utilisent souvent des techniques différentes. On trouve par exemple :

- Des services web : le langage WSAMI (Web Service For Ambient Intelligence) (Issarny et al. 2005) permet la spécification de services web qui peuvent être composés dynamiquement en fonction de l'environnement dans lesquels les services sont demandés,
- Un système multi-agent : Le projet LAICA (Cabri et al. 2008) est un *middleware* ad-hoc qui utilise un système multi-agent pour gérer l'Aml. Ce *middleware* contient 4 éléments principaux :
 - Les agents *Sensor* gèrent les informations acquies des dispositifs numériques,
 - Les agents *Effector* peuvent effectuer des actions sur l'environnement ambiant,
 - Une base de données enregistre toutes les informations pour garder une trace des événements qui se sont déroulés,
 - Un portail Web contrôle l'intégralité du système. Il publie différentes sortes d'information aux autres agents. C'est un composant centralisé sur lequel les auteurs émettent ce commentaire : « *Note that, even if this is a centralized component, the intelligence of the whole ambient infrastructure is spread across the agents and the middleware* ».

5. Perspectives de recherche en intelligence ambiante

Les perspectives en intelligence ambiante sont nombreuses et il se pose des questions similaires à celles posées par les problématiques du Compagnon Artificiel. Cette partie décrit des problèmes techniques qui n'ont pas encore été résolus et des caractéristiques de l'intelligence ambiante qui ne sont pas encore intégrés dans les systèmes existants, ainsi que des nouvelles idées qui semblent prometteuses comme celle de l'A.I.vatar et des méta-appareils domestiques.

Problèmes techniques

En intelligence ambiante, trois points sont à considérer (Emile Aarts and Wichert 2009) :

- Interopérabilité : comment permettre à n'importe quel dispositif indépendant d'échanger des données avec les autres ?
- Hétérogénéité : comment le système peut-il être exécuté sur n'importe quel dispositif ?
- Dynamique : comment le système peut-il s'adapter aux changements ?

Caractéristiques de l'intelligence ambiante

Aarts et de Ruyter ont déterminé 7 perspectives de recherche (Emile Aarts and De Ruyter 2009):

- **Contrôle ambiant** : comment peut-on accéder et contrôler les dispositifs de l'environnement Aml ? Ce problème s'intéresse à la façon d'intégrer le monde physique dans le monde digital.
- **Interfaces tangibles** : quels sont les concepts d'interaction multimodale qui peuvent être appliqués aux environnements Aml ?
- **Programmation des utilisateurs finaux** : comment les utilisateurs finaux peuvent programmer leurs propres fonctionnalités dans l'environnement Aml ? Le problème s'intéresse à la conception d'environnement de programmation qui permet à des non-experts en informatique de piloter l'environnement Aml.
- **Expériences sensorielles** : comment les utilisateurs traitent les informations dans l'interaction avec les environnements Aml ?
- **Présence sociale** : comment prendre en compte l'aspect social dans les interactions avec les environnements Aml ?
- **Confiance** : comment le système peut inspirer la confiance des utilisateurs ?
- **Éthique** : comment les gens peuvent accepter qu'un environnement surveille leur moindre mouvement, attendant le bon moment pour intervenir ?

A.I.vatar

La personnalisation est un élément clef de l'Aml. Suivant cette idée, l'A.I.vatar (De Andrade 2011) est un nouveau type d'agents qui est hautement personnalisé, intuitif et intelligent. C'est une sorte de clone numérique de la personne, l'imitant, reflétant sa personnalité et simulant, de façon autonome, ce que serait son comportement dans un contexte donné. L'A.I.vatar est également capable de percevoir et d'interpréter l'état affectif d'un utilisateur pour prendre des décisions.

Les méta-appareils domestiques

Chin *et al* (Chin, Callaghan, and Clarke 2009) ont introduit en 2009 une nouvelle vision de la gestion des appareils domestiques dans le domaine de la domotique. Dans cette vision, on ne détermine plus les appareils domestiques par leur nom, comme TV, mais par leur fonction « affiche », « capte le son »... Ainsi, les services peuvent être combinés avec d'autres services, et les utilisateurs peuvent reconfigurer les services pour créer de nouvelles configurations. Cela forme des *meta-appliances* (méta-appareil domestique). Cette nouvelle vision crée la notion de MAPs (*Meta-*

Appliances/Applications), qui est un appareil domestique virtuel formé par un ensemble d'entités coordonnées du réseau. C'est un ensemble de règles et d'association sémantique. Un utilisateur peut interagir avec son environnement via son *proctor* (*programmer-communicator*) qui contient ses préférences privées et la description de son *soft-appliances*. Cette vision imagine que toutes les fonctions et tous les appareils sont définis par une ontologie. L'auteur explique que la façon de programmer son environnement est un défi et qu'il faudra faire un choix entre des agents autonomes ou laisser la possibilité à l'utilisateur de programmer son environnement. Quelques évaluations ont montré que ce genre de vision peut être accepté des utilisateurs et que ces derniers trouvent cela plutôt simple.

A ce jour, le système ne semble pas encore implémenté :

"Finally, whilst this is only a brief overview of the vision and agenda for soft-appliances, we hope it may inspire other people to add to the body of research that will turn this research vision into a commercial reality."

E. Discussion

Nous avons vu que les robots et les personnages virtuels sont utilisés en interaction homme-machine pour interagir avec l'homme. Mais si l'homme ne devait avoir qu'un seul compagnon, lequel serait-il ? La littérature montre que la réponse n'est pas simple puisque trois points de vue s'opposent :

1. Les objets réels sont « meilleurs » que les objets virtuels,
2. Les objets virtuels sont « meilleurs » que les objets réels,
3. Ni les objets réels, ni les objets virtuels ne sont « meilleurs ».

1. Selon Kidd (C.D. Kidd 2003), la présence physique d'un robot est importante, car le robot est plus crédible et persuasif qu'un objet virtuel. Plusieurs études montrent cette importance des robots. Par exemple, Nomura *et al* (Nomura and Sasa 2009) ont comparé l'impact d'un robot réel et virtuel sur des personnes jeunes (moyenne d'âge : 20,6 ans) et des personnes âgées (moyenne d'âge : 68,7 ans). Les résultats ont montré que les personnes âgées prenaient plus en compte le robot réel que les personnes jeunes. L'impression des personnes âgées, concernant le robot, était plus positive que celles des personnes jeunes. Dans une autre étude, Tapus *et al* (Tapus, Tapus, and Mataric 2009) ont comparé les performances obtenues lors d'un jeu cognitif musical qu'il soit présenté par un robot réel ou par un robot virtuel. L'étude a révélé que l'interaction sociale et la performance à réaliser une tâche sont améliorées avec le robot réel.

2. Concernant l'autre point de vue, les études scientifiques semblent moins nombreuses. Nous pouvons citer Hsu *et al* (Hsu *et al.* 2007) qui ont fait la comparaison entre un robot réel et un robot virtuel dans un système compagnon d'apprentissage. L'étude a révélé que les participants étaient plus concentrés avec le robot virtuel.

3. Enfin, la majorité des études montrent que le réel n'est pas « meilleur » que le virtuel et vice versa. Deux études (Shinozawa *et al.* 2005)(Yamato *et al.* 2001) ont fait la comparaison entre un robot et un agent virtuel qui donnaient des recommandations sur la prise de décision dans une tâche de sélection de couleur. Les expérimentations n'ont pas déterminé de différence d'agissement des participants en fonction de l'agent réel ou virtuel. Par contre, il semblerait que l'environnement agisse sur l'effet de l'agent. Un personnage en 3D a un effet dans un environnement 3D, un personnage en 2D a un effet dans un environnement 2D, mais n'a pas d'effet dans un environnement 3D et inversement. Une autre étude, réalisée par Lee *et al* (Lee *et al.* 2006), a montré que l'interaction tactile est plus importante que le fait d'être en présence d'un personnage virtuel ou réel. Ainsi, ils indiquent que le personnage réel est évalué plus positivement que le personnage virtuel si l'humain peut toucher des objets. Si ce n'est pas le cas, c'est le personnage virtuel qui est évalué plus positivement. Le réel ou le virtuel a peu d'importance. Enfin Ohara (Ohara *et al.* 2009) a étudié

l'impression que se font les personnes d'un robot réel et virtuel en fonction de sa présence ou non dans leur espace vital. Il semblerait qu'il n'y ait pas de différences d'impression mais que les personnes ont besoin d'un plus grand espace vital en présence d'un robot réel.

Le Tableau 3 résume les sept évaluations présentées ci-dessus pour faire ressortir les points de comparaison. (O = observation, Q = questionnaire, R = Réel, V = Virtuel)

Tableau 3 : Etudes sur les comparaisons entre les effets du réel et du virtuel

Etude	Année	Lieu	Participants	Méthode	« Meilleur » interlocuteur
(Shinozawa et al. 2005)(Yamamoto et al. 2001)	2004	Japon	Pas de caractéristiques précises	O + Q	R = V
(Lee et al. 2006)	2006	USA	Étudiants	Q	R = V
(Ohara et al. 2009)	2009	Japon	Jeunes	Q	R = V
(Hsu et al. 2007)	2007	Taiwan	Élèves de primaire	Q	V
(Nomura and Sasa 2009)	2009	Japon	Personnes âgées et jeunes	O + Q	R
(Tapus, Tapus, and Mataric 2009)	2009	USA	Personnes âgées avec démences	O + Q	R

Ces résultats montrent que les interlocuteurs réels ou virtuels sont aussi importants les uns que les autres. L'impression que s'en font les personnes semblent plus être liées au contexte d'évaluation qu'à l'interlocuteur même. Notre hypothèse est que la présence physique est recommandée dans certaines tâches, tandis que la présence virtuelle dans d'autres. D'autres études devraient être réalisées pour comprendre le rôle des robots et des agents virtuels dans notre quotidien en fonction des cultures. En effet, les études présentées ici ne montrent qu'une partie de la réalité car elles ont toutes été réalisées en Asie et en Amérique. Elles ne représentent pas un échantillon de toutes les cultures, or il est communément admis que la relation de l'homme à la technologie est sous-tendue par sa culture.

Concernant le projet Robadom, le choix du robot compagnon n'a pas pu être possible. En effet un focus groupe (Wu, Fassert, and Rigaud 2012), réalisé par l'hôpital Broca et composé d'un groupe d'une vingtaine de personnes âgées françaises a été interrogé sur ce sujet et a montré une réticence concernant les robots humanoïdes. Les personnes âgées imaginent une interaction avec des petits robots qui ressemblent à une « théière ». Ce qualificatif a été employé pour parler de Mamoru, le robot de la Figure 36. De plus, elles perçoivent le robot comme un remplaçant de l'homme et cela leur fait peur. Enfin, elles ont exprimé le fait qu'un robot devait être un objet et garder sa condition d'objet. Il n'est donc pas possible, à l'heure actuelle, de leur proposer un robot humanoïde, même si ce type de robot est un meilleur communiquant que les autres.



Figure 36 : Robot Mamoru

Prenons l'hypothèse qu'il est important pour les personnes âgées d'avoir une assistance physique, de créer un lien émotionnel avec le robot pour favoriser une bonne relation (acceptabilité), que le robot puisse communiquer verbalement et non verbalement. Le Tableau 4 propose notre vision des capacités de chaque type de robot. Il n'existe pas de robot qui aient toutes ces capacités simultanément. Il semble pourtant nécessaire pour les personnes âgées d'avoir un robot qui ait les capacités d'un robot de service pour assister l'homme et qui construise le lien émotionnel des robots compagnons, car les émotions ont un impact positif sur la santé physique et mental (Scherer 1984).

Tableau 4 : Capacités des groupes de robots

	Groupe 1	Groupe 2	Groupe 3	Groupe 4
Assistance physique	++	-	-	+
Lien émotionnel	-	++	++	+
Communication verbale	++	--	-	++
Communication non verbale	-	++	++	++
Acceptabilité française	++	-	++	--

Groupe 1 : Robots de service, Groupe 2 : robots animaloïdes, Groupe 3 : robots imaginaires, Groupe 4 : robots humanoïdes

Concernant les architectures des logiciels, compagnons robotiques et compagnon virtuels, l'état de l'art montre la diversité des approches et des modèles, toujours réalisés en fonction des besoins. La plupart des solutions manque de pluridisciplinarité et sont généralement trop dédiées pour notre usage. Cette approche nous semble un peu trop restrictive pour une interaction avec un Compagnon Artificiel. Si le but est de gérer un système de façon autonome, créer une architecture pour un domaine précis n'est pas le moyen d'y parvenir. Des architectures basées sur le processus mental de l'être humain semblent être une bonne solution car elles permettent d'imiter l'humain dans ces réactions et permettent d'établir des bases communes entre l'homme et le système. Les architectures à plusieurs niveaux montrent qu'il y a deux types de réactions essentielles : les réactions réflexes et les réactions liées à la prise de décision. Est-ce que le Compagnon Artificiel doit intégrer ces réactions ? Les implémentations du modèle BDI, l'architecture PRS et l'architecture SAIBA montre que l'intention du système doit être modélisée, pour qu'il puisse effectuer des actions d'une part et qu'il puisse être prédictible d'autre part. De plus, l'architecture SAIBA est intéressante car elle offre une solution standardisée en utilisant des fichiers XML pour faire communiquer les différents modules entre eux. De plus, elle est largement utilisée par la communauté, puisqu'elle est à la base de Greta et qu'elle s'insère dans le projet SEMAINE. Néanmoins, ces projets sont dédiés à des agents virtuels alors que le Compagnon Artificiel peut utiliser différents objets numériques.

Un autre aspect émerge de cet état de l'art concernant la gestion des entrées et des sorties du système. Le travail sur le modèle de personnalité basé sur l'ennéagramme montre qu'un module gérant toutes ces entrées et qu'un module gérant toutes ces sorties est important pour recentrer toutes les informations en un seul point et pouvoir plus facilement prendre des décisions. Cela permet au système de connaître l'environnement qui l'entoure et de s'en faire une représentation.

En conclusion, nous remarquons qu'un humain est entouré de plusieurs compagnons (Pesty and Duhaut 2011) avec lesquels il interagit tout au long de sa vie. Un compagnon est un objet

communicant : personnage virtuel, robot, télévision, téléphone, ordinateur... Les logiciels, les robots et les personnages virtuels sont complémentaires les uns des autres. Il est donc nécessaire de construire une architecture qui peut gérer un ensemble d'objets numériques pour exploiter l'ensemble de leurs fonctionnalités. L'architecture doit également être modulaire et facilement adaptable afin d'être enrichie au fur et à mesure de l'avancement de la recherche. Une nouvelle découverte ne doit pas remettre en cause le fonctionnement global du Compagnon Artificiel mais doit pouvoir être ajoutée simplement.

L'intelligence ambiante pose les mêmes problématiques que celle de la conception du Compagnon Artificiel : interopérabilité, hétérogénéité, adaptation aux changements, contrôle des objets numériques par l'utilisateur, personnalisation du système, confiance dans le système, éthique, acceptabilité...

Synthèse

Dans ce chapitre nous avons montré qu'il n'existe pas de solution pour le problème du Compagnon Artificiel tel que nous l'avons défini.

Concernant les architectures informatiques, elles sont trop spécifiques à un domaine. L'architecture pour un Compagnon Artificiel se place à un plus haut niveau d'abstraction où « tout » ne doit pas être modélisé. Notre idée est de fournir une architecture qui puisse être enrichie avec le temps, elle ne doit donc pas, par sa structure, présupposer de l'utilisation qui en sera faite. Nous retenons l'idée d'une architecture avec au moins trois niveaux : un niveau spécialisé dans le traitement des entrées, un autre spécialisé dans le calcul de la réponse et un dernier spécialisé dans le traitement des sorties. Les échanges entre les différents niveaux pourraient se faire par des messages standardisés pour favoriser l'intégration d'un maximum d'objets numériques.

L'état de l'art montre qu'il existe une grande diversité de robots, de personnages virtuels et d'autres objets (smartphone, logiciel compagnon...) utilisés pour l'interaction homme-machine dans différents contextes. Ces trois types de compagnons ont des rôles différents en fonction des situations. Le Compagnon Artificiel doit utiliser cette diversité pour assister le plus efficacement possible l'utilisateur final du système. Il ne doit pas être prévu pour un type particulier d'objets numériques et doit prévoir une intégration facile d'objets *a priori* inconnus.

L'état de l'art a montré que le Compagnon Artificiel se situe dans le domaine de l'intelligence ambiante et qu'il pourra répondre à un certain nombre de problématiques notamment en ce qui concerne l'interopérabilité des différents composants du système, leur hétérogénéité, l'adaptation aux changements du Compagnon Artificiel, le contrôle des objets numériques par l'utilisateur, la personnalisation de son système, la confiance dans le système, l'éthique, l'acceptabilité... La vision apportée par Chin en 2009 est une approche intéressante pour le Compagnon Artificiel. Si chaque objet numérique est représenté par des informations de haut niveaux et une liste d'actions possibles, alors le système n'est plus dépendant de la technologie utilisée et cela permet l'intégration de tous types d'objets numériques. Le Compagnon Artificiel doit reprendre cette idée.

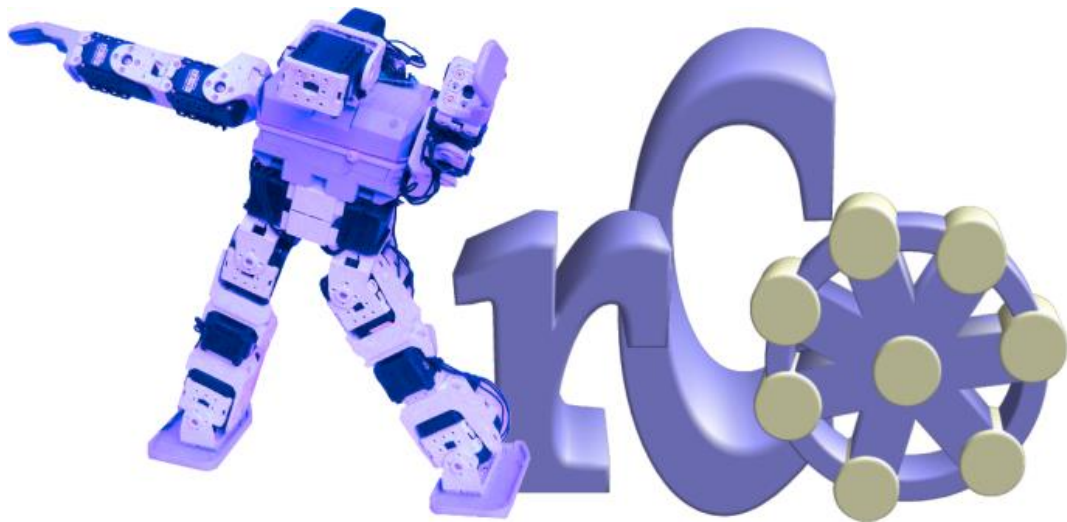
Pour la création de notre architecture ArCo, nous retiendrons les points suivants :

- L'utilisation de tous les robots, tous les personnages virtuels, tous les objets numériques disponibles,
- La notion des architectures 3 couches avec une gestion spécialisée des entrées, une gestion spécialisée des sorties et le calcul du lien entre les entrées et les sorties,
- La notion d'entités autonomes capables de communiquer avec d'autres entités mais sans propre en compte leurs propres décisions (seul l'utilisateur doit être le maître du système),
- La notion de représentation de l'environnement. Il est important de se représenter l'environnement pour pouvoir l'analyser et réagir,
- La notion de communication entre les entités via des fichiers XML standardisés au sein de l'architecture.

La notion de personnalisation de l'environnement pour un utilisateur final par le biais d'une interface de programmation visuelle simple.

Chapitre III.

Architecture ArCo : une réponse pour le Compagnon Artificiel



CHAPITRE III. ARCHITECTURE ARCO : UNE REPONSE POUR LE COMPAGNON ARTIFICIEL	69
A. VOCABULAIRE	71
B. PRESENTATION D'ARCO	72
C. MICE : CADRICIEL POUR LA GESTION D'UN ENVIRONNEMENT MODULAIRE	77
1. Modules et communication	77
2. Données gérées par le serveur	79
3. Protocole de communication	82
4. AmbiPath : Module pour la création du chemin logique	85
D. ENTITES AGISSANTES : MODULES D'ENTREES ET SORTIES	86
E. AMBIPROG : OUTIL POUR LA CREATION DE SCENARIO	88
1. Scénario d'interaction	88
2. Configuration d'AmbiProg : un exemple	89
3. Présentation générale	92
4. Grammaire du langage généré par AmbiProg	93
5. Exemple de scénario	94
F. AMBILIVE : MODULE D'INTERPRETATION DES SCENARI	95
1. Gestion des entrées et sorties du module d'interprétation	96
2. Interprétation des perceptions symboliques	97
G. AMBICOP : MODULE D'ARBITRAGE DES ACTIONS	98
1. Organisation générale d'AmbiCop	98
2. Gestion des conflits	99
3. Gestion des actions génériques	101
4. Synchronisation des actions	103

Ce chapitre présente ArCo, une architecture pour un Compagnon Artificiel en interaction avec un utilisateur qui offre une solution aux propriétés structurelles et fonctionnelles. ArCo est composée de trois éléments essentiels : l'interface de programmation AmbiProg qui permet la création des scénarii par des prestataires et deux modules : AmbiLive chargé d'interpréter un scénario et AmbiCop qui gère les conflits. Ils sont programmés avec un cadriciel MICE qui permet de construire des modules compatibles à ArCo grâce à un protocole de communication permettant une interopérabilité entre les modules, une représentation de l'environnement et une bibliothèque disponible en JAVA.

Le Chapitre III.A présente les définitions des concepts utilisés dans ce document. Cela pose les briques de base qui permettent au Chapitre III.B de faire une présentation générale du compagnon artificiel avec les trois éléments AmbiProg, AmbiLive et AmbiCop et leurs perceptions, actions et scénarii d'interaction. Cette présentation générale est suivie par une présentation du cadriciel MICE qui permet de mettre en œuvre ces éléments (cf. Chapitre III.C). Enfin, les Chapitre III.E, Chapitre III.F et Chapitre III.G présentent AmbiProg, AmbiLive et AmbiCop respectivement.

L'annexe H complète ce chapitre en présentant la bibliothèque MICE en détail, avec tous les diagrammes de classe la décrivant. Les développeurs trouveront également trois exemples de modules :

- *Un module de perception simple,*
- *Un module d'action simple,*
- *Le module que nous avons utilisé pour associer la version Psyclone de Greta à ArCo. Cet exemple montre qu'un tel module a été écrit avec un faible nombre de lignes de code et cela confirme la simplicité d'intégration d'un objet numérique ou personnage virtuel à MICE.*

A. Vocabulaire

Cette partie donne la définition des termes utilisés.

Un **utilisateur** est la personne pour qui le système est déployé. Dans le cas du projet Robadom, l'utilisateur est la personne à domicile atteinte d'Alzheimer.

Un **objet numérique** est un capteur (calcul de l'**environnement**) ou un actionneur (action sur l'**environnement**) : un robot domestique, une caméra, un ordinateur, une tablette, un téléphone, une télévision, un volet roulant automatisé... Un objet numérique contient une part d'informatique qui lui permet d'effectuer des calculs et de communiquer avec d'autres systèmes.

Une **entité agissante** est un programme informatique qui gère la communication avec un objet numérique. Un capteur est géré par une **entité agissante perceptive**, qui récupère les informations calculées, et un actionneur est géré par une **entité agissante active**, qui lui envoie l'ordre d'effectuer des actions. Une **entité agissante perceptive** fournit des perceptions et une **entité agissante active** fournit des actions.

Un **scénario** est un programme qui décrit l'ensemble des liens dynamiques entre les perceptions et les actions.

L'**environnement** est l'ensemble de tous les **objets numériques** pris en compte par les scénarii.

Un **Compagnon Artificiel** est l'ensemble des objets numériques de l'**environnement**. Techniquement, le **Compagnon Artificiel** est une entité informatique qui exécute les scénarii dans un **environnement** donné, comme le montre la Figure 37.

Un **prestataire** est une personne qui participe à la configuration du **Compagnon Artificiel** en écrivant des scénarii.

Un **concepteur** est une personne qui crée de nouveaux capteurs ou actionneurs ainsi que les entités agissantes associées.

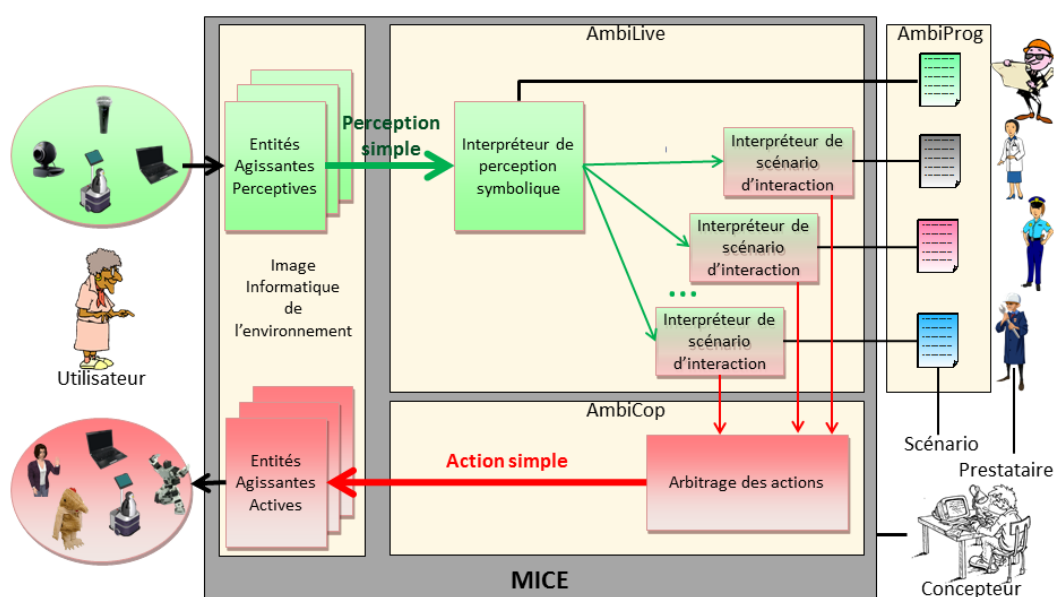


Figure 37 : Organisation de l'architecture ArCo pour le Compagnon Artificiel

B. Présentation d'ArCo

Le contexte d'utilisation du Compagnon Artificiel est celui d'un utilisateur en interaction avec un ensemble d'objets numériques représentés dans ArCo par des entités agissantes (cf. Chapitre III.D). La Figure 38 montre cet utilisateur en interaction (par intermédiaire des objets numériques) avec les entités agissantes qui sont utilisées par un ensemble de scénarii. Ces entités agissantes sont coordonnées pour aider et assister l'utilisateur dans sa vie quotidienne. C'est cet ensemble d'entités agissantes coordonnées qui est le Compagnon Artificiel de l'utilisateur, à travers l'apparence des objets numériques.

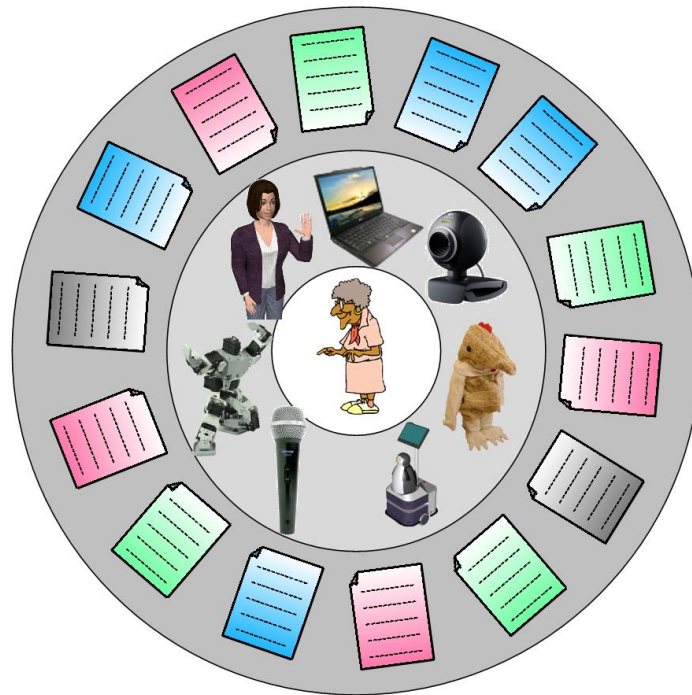
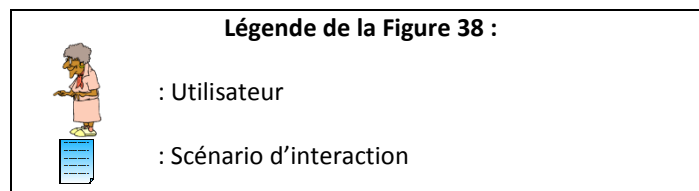


Figure 38 : Présentation générale du Compagnon Artificiel



est une architecture modulaire permettant de faire « vivre » ce Compagnon Artificiel dans un environnement. Elle est constituée de modules qui sont des programmes indépendants spécialisés dans le traitement d'une tâche spécifique. ArCo exécute un ensemble de scénarii d'interaction qui décrivent les actions que doivent effectuer les entités agissantes en fonction des données qui sont fournies.

Les scénarii sont écrits par des prestataires qui peuvent être des développeurs, des amis, la famille de l'utilisateur, des médecins ou l'utilisateur lui-même afin de fournir des services à l'utilisateur comme : la sécurité, le suivi médical, la lecture de flux d'information...

Pour ce faire, ArCo est composé de trois différents exécutifs, présentés sur la Figure 39 :

- AmbiProg¹² (cf. Chapitre III.E) permet d'écrire les scénarii d'interaction,
- AmbiLive (cf. Chapitre III.F) interprète¹³ les scénarii d'interaction,
- AmbiCop (cf. 0) coordonne les actions.

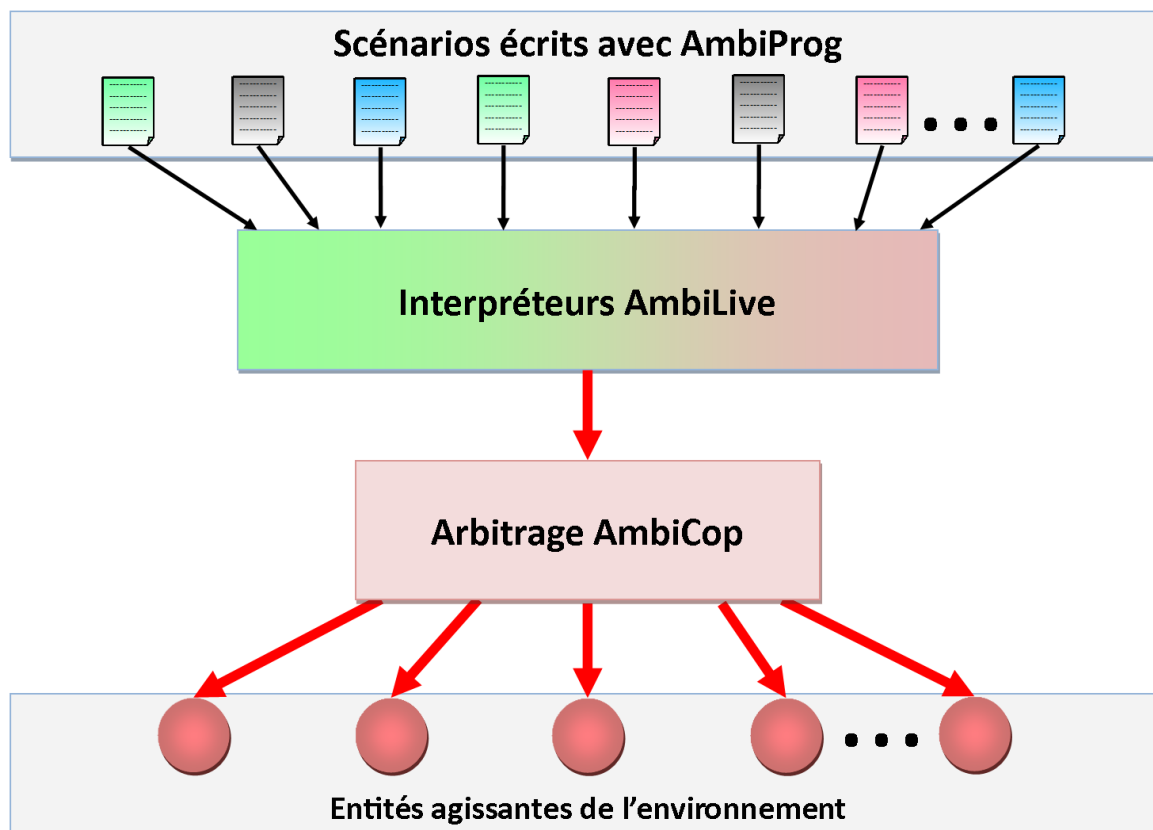


Figure 39 : Modules principaux d'ArCo

Les exécutifs AmbiLive et AmbiCop et les entités agissantes s'exécutent en utilisant le cadriciel MICE. Du point de vue logiciel, ArCo a quelques propriétés : il est possible d'ajouter en « temps réel » des entités agissantes perceptives et actives qui participent automatiquement aux scénarii en cours. ArCo est également un système ouvert qui peut être enrichi par des développeurs grâce aux bibliothèques fournies : MICE (cf. Chapitre III.C) et AmbiLib (cf. Annexe G). ArCo est d'ailleurs réalisée avec ces bibliothèques.

Les scénarii manipulés par ArCo sont la base opérationnelle du Compagnon Artificiel. Ils manipulent les entités agissantes pour obtenir des services utiles à l'utilisateur. Nous distinguons deux types d'entités agissantes : les entités agissantes perceptives qui fournissent des perceptions et les entités agissantes actives qui fournissent des actions.

¹² AmbiProg vient de Programmation Ambiante et propose d'écrire des scénarii grâce à un langage de programmation visuelle simple pour faciliter l'apprentissage par tous.

¹³ Le système évolue en interaction avec la personne, à une échelle de temps humaine.

MICE est un cadriciel qui permet de mettre en place des applications variées. Dans le cadre du travail présenté ici, qui utilise des perceptions et des actions pour mettre en place une interaction avec un utilisateur, l'utilisation d'ArCo est effectuée avec 4 types de modules

- Les modules de perceptions,
- Les modules d'interprétation (de perceptions symboliques et de scénarii d'interaction),
- Les modules d'arbitrage,
- Les modules d'actions.

Ces quatre types de modules sont les modules minimum d'ArCo qui assurent un fonctionnement de base dans la gestion du Compagnon Artificiel. Chaque développeur peut ajouter d'autres modules effectuant des calculs sur l'information qui transite.

La Figure 40 montre l'organisation minimum d'ArCo. Les modules d'entrées du système sont les modules de perceptions qui sont liés au module d'interprétation des perceptions symboliques. Le lien est unidirectionnel. Tous les messages envoyés sur le flux de sortie des modules de Perceptions sont envoyés sur le flux d'entrée du module d'interprétation des perceptions symboliques, mais la réciproque est fautive. Le module d'interprétation des perceptions symboliques envoient les perceptions à tous les modules d'interprétation des scénarii connectés à MICE. Ceux-ci et le module d'arbitrage des actions sont liés dans les deux sens. En effet, les modules d'interprétation des scénarii envoient les actions à effectuer au module d'arbitrage des actions et celui-ci leur envoie en retour les notifications de fin d'actions. Il est le relai entre l'interprétation et l'exécution et intercepte tous les messages qui peuvent être envoyés entre les deux. Les modules d'actions reçoivent les ordres du module d'arbitrage des actions, qui s'ayant inscrit aux notifications de fin d'action, reçoit directement de MICE les notifications de fin d'action des modules d'actions. La communication est donc unidirectionnelle.

La chaîne de traitement de l'information s'effectue en 5 étapes, à travers ces 5 groupes de modules :

1. **Les modules de perceptions** initient le système. Une perception est une notification d'un évènement s'étant produit dans l'environnement. Il existe deux types de perceptions : les perceptions brutes et les perceptions symboliques. Les perceptions brutes sont celles qui sont envoyées par les entités agissantes perceptives comme la température, la position de la personne, l'indication de prise de médicament. Les perceptions symboliques sont des perceptions personnalisées par des prestataires.
2. **Les modules d'interprétation de perceptions symboliques** permettent de calculer les perceptions symboliques en interprétant un scénario de perception symbolique. C'est un scénario construit avec AmbiProg. Il est possible d'utiliser tous les éléments de scénarii. La spécificité de ce scénario est qu'il permet de fournir des perceptions alors que les autres fournissent des actions.

Exemple

Un scénario de perception symbolique peut être : « Si l'accéléromètre détecte une position horizontale avec un pourcentage de certitude supérieur à 90% et que la caméra indique l'absence de lit ou assimilé avec un pourcentage supérieur à 95% alors il faut envoyer la perception P(camera, personne_tombée, 1, 100) ». Dans cet exemple, une perception **personne_tombée** est créée. C'est une perception symbolique.

3. **Les modules d'interprétation** de scénarii d'interaction sont chargés de faire le lien entre les perceptions reçues et les actions à envoyer. Plusieurs scénarii tournent en parallèle.
4. **Le module d'arbitrage** des actions permet d'éviter de solliciter une même entité agissante active pour réaliser plusieurs actions. Les actions qui résultent de l'interprétation sont envoyées à un module d'arbitrage des actions. Celui-ci est chargé de contrôler les ressources et de diriger les actions vers les entités agissantes concernées. De la même façon que pour les perceptions symboliques, il existe des actions génériques. En effet, une action peut être destinée à une entité agissante précise ou alors exiger une compétence sans préciser l'entité agissante qui doit effectuer l'action. Dans ce cas, ce sont des actions génériques que le module d'arbitrage des actions doit associer à une entité agissante disponible ou efficace pendant l'exécution des scénarii.
5. **Les modules d'actions** reçoivent les ordres et les exécutent. Ils doivent prévenir le serveur lorsqu'ils ont terminé d'effectuer les actions. C'est le dernier maillon de la chaîne.

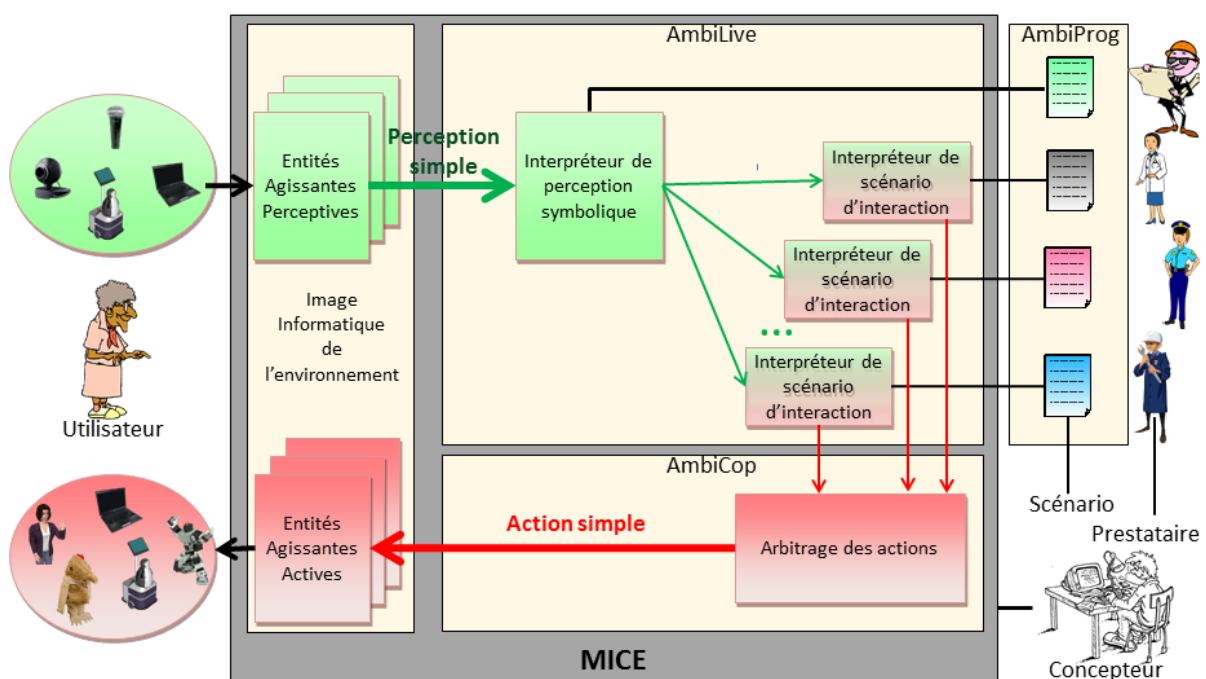


Figure 40 : Organisation générale d'ArCo avec le cadrice MICE

La suite de ce chapitre détaille l'organisation générale d'ArCo du point de vue de ses fonctionnalités et des exécutifs qui le composent.

Il se décompose en :

- MICE : cadriciel permettant de créer une architecture informatique modulaire, spécialisée dans la gestion d'un environnement. MICE impose un protocole de communication :
 - Permettant à chaque module de l'architecture de communiquer ensemble ,
 - Assurant l'interopérabilité entre les modules (*cf.* Chapitre III.C.3).
- L'ensemble des exécutifs :
 - AmbiProg : langage de programmation visuelle permettant de créer de scénarii d'interaction pour définir les actions du Compagnon Artificiel. Il permet à des prestataires de piloter un Compagnon Artificiel (*cf.* Chapitre III.D),
 - AmbiLive : module d'interprétation associé à AmbiProg, chargé d'exécuter des scénarii (*cf.* Chapitre III.F),
 - AmbiCop : module permettant de gérer les conflits d'accès aux entités (*cf.* 0).

C. MICE : Cadriciel pour la gestion d'un environnement modulaire

MICE (Machines Interaction Control in their Environment) est un cadriciel sur lequel est construit ArCo. MICE se compose de :

- Une architecture client/serveur modulaire : MICE propose un module *serveur* fonctionnel permettant de gérer un ensemble de modules *client*,
- Un protocole de communication : permet d'assurer une interopérabilité entre les modules en leur imposant un format de messages,
- Une représentation de l'environnement : MICE permet d'agir sur l'environnement par l'utilisation des perceptions et des actions,
- Une bibliothèque MICE : permet de fournir les outils nécessaires à l'intégration de nouvelles entités agissantes et permet à des concepteurs de définir leurs propres modules, disponibles à l'adresse <https://sites.google.com/site/compagnonartificiel/> en JAVA.

La première partie de ce chapitre détaille le fonctionnement général à base de modules qui constitue la base de l'architecture client/serveur. Pour être fonctionnel, le serveur stocke des informations sur tous les clients, lui permettant ainsi d'avoir une connaissance globale des entités agissantes existantes. La structure de ces informations est décrite dans la deuxième partie de ce chapitre. Pour que les modules puissent communiquer avec le serveur, ils doivent respecter un protocole de communication présenté dans la troisième partie. Les messages réservés à la configuration du serveur sont détaillés dans la quatrième partie. Pour permettre de déterminer de façon automatique le chemin logique de l'information entre les modules, un module AmbiPath est fourni par MICE et décrit dans la cinquième partie. La sixième partie présente l'utilisation de MICE dans le cadre de l'architecture ArCo.

1. Modules et communication

Les modules communiquent au moyen de messages spécifiques au format MICE (*cf.* chapitre III.B.3). La communication entre les modules est présentée sur la Figure 41. La communication entre les modules « M » et le serveur « S » s'effectue avec deux types de messages :

- Le flux noir : représente le **chemin physique** de l'information. Ce sont des messages personnels, échangés entre le serveur et un module, afin de configurer le système. Le flux noir représente les messages d'enregistrement auprès du serveur. Chaque module s'enregistre en donnant son identité et son type. Cela permet au serveur de pouvoir identifier chacun des modules afin de pouvoir leur envoyer des messages. Le flux noir représente aussi des messages permettant de créer le flux bleu, de s'abonner à la notification de certains événements, de récupérer la liste des modules connectés ou la liste des perceptions et des actions que gère chaque module.
- Le flux bleu : représente le **chemin logique** de l'information que doit suivre les messages en transit. Lorsqu'un module envoie un message, celui-ci suit le chemin. Chaque module du chemin peut alors lire et modifier le message. Les messages transitant sur le flux bleu sont des messages de perceptions, d'actions ou d'autres messages créés par les concepteurs.

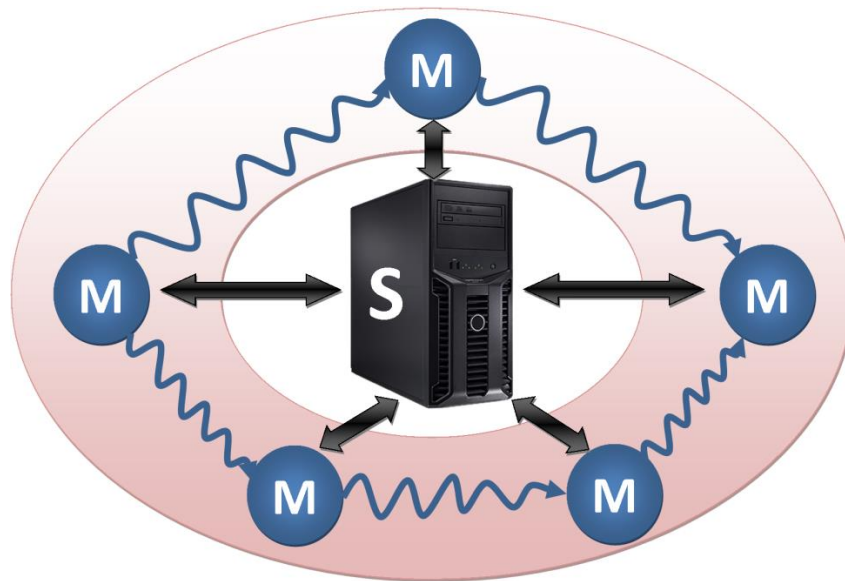


Figure 41 : Organisation générale de MICE

Le serveur est le seul à connaître le chemin représenté par le flux bleu. En effet, un module possède une entrée et une sortie « anonymes », comme le montre la Figure 42. Cela signifie qu'il reçoit des informations sans nécessairement connaître leur provenance, qu'il transforme ce message si besoin et qu'il renvoie l'information sans nécessairement connaître leur destinataire. Chaque message transite donc à travers tous les modules du chemin.

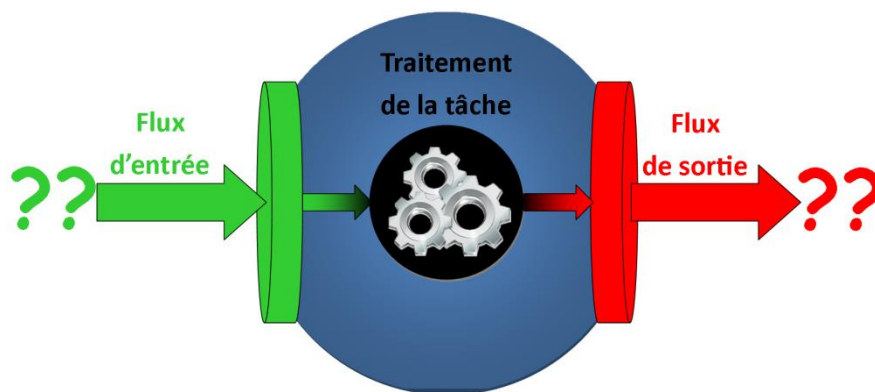


Figure 42 : Organisation d'un module de MICE

2. Données gérées par le serveur

Le serveur de MICE stocke des informations concernant les modules afin de se représenter l'environnement. Ces informations sont utilisées par AmbiProg qui se configure automatiquement en fonction des entités agissantes connues par le serveur. Les informations manipulées par le serveur sont les suivantes :

- Les perceptions et les actions gérées par les modules,
- La liste des modules connectés,
- La liste de tous les modules s'étant connectés au moins une fois à MICE.

MICE gère trois types d'informations statiques :

- Les modules connus par l'architecture (cf. Chapitre III.C.2.a),
- La liste des perceptions envoyées par les modules (cf. Chapitre III.C.2.c),
- La liste des actions que peuvent effectuer les modules (cf. Chapitre III.C.2.c).

Le serveur construit dynamiquement, en mémoire, trois types d'informations :

- Le chemin de l'information entre les modules (cf. Chapitre III.C.2.b),
- Les modules inscrits aux notifications des actions et des identifications (cf. Chapitre III.C.2.b),
- La liste des modules actuellement connectés (cf. Chapitre III.C.2.b).

a) Informations stockées statiquement

La Figure 43 montre l'arborescence des fichiers stockés par le serveur. Ces informations sont stockées car elles ne dépendent pas d'une exécution et sont les bases pour différentes exécutions. Ce sont des données fixes. Dans le fichier actions.desc se trouvent toutes les actions que sont capables d'effectuer les entités. Dans le fichier perceptions.desc se trouvent toutes les perceptions que sont capables d'envoyer les entités. Un exemple de ces deux fichiers se trouve au Chapitre III.E.1.

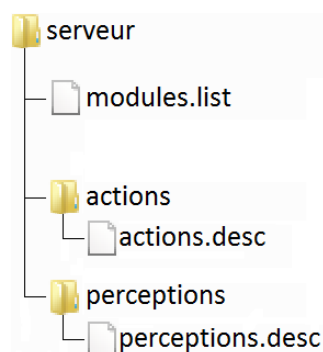


Figure 43 : Fichiers stockés par le serveur

Le fichier modules.list contient la liste des modules connus par le serveur, c'est-à-dire tous les modules qui se sont déjà connectés au moins une fois au serveur. Le fichier modules.list correspondant aux deux fichiers de descriptions est présenté ci-dessous.

Fichier modules.list

```
<?xml version="1.0" encoding="UTF-8"?>
<modules>
  <module id="camera" type="input" />
  <module id="micro" type="input" />
  <module id="robot" type="output" />
  <module id="agent_virtuel" type="output" />
</modules>
```

b) Informations stockées dynamiquement

Certaines informations dépendent d'une exécution de l'architecture ArCo. Par exemple, les liens entre les modules ne sont pas définis en avance. Les modules peuvent être connectés et déconnectés dynamiquement. Les modules s'inscrivent aux notifications des actions en fonction de leurs besoins, cette information ne peut pas être définie de façon statique. Ces deux informations ne sont pas partagées avec les modules. Ce sont des données qui ne sont utiles qu'au serveur. Ainsi, il n'existe pas de représentation XML. Les liens entre les modules et la liste des modules inscrits aux notifications des actions sont stockées en mémoire.

La liste des modules connectés peut être demandée par chaque module. Cette liste est fournie, sous la forme de la chaîne XML suivante :

```
<connected_modules>
  <module>camera</module>
  <module>micro</module>
  <module>robot</module>
  <module>agent_virtuel</module>
</connected_modules>
```

c) Fichiers de descriptions des entités

Pour être compatible avec ArCo, chaque entité numérique fournit un fichier XML qui contient la description de toutes les perceptions qu'elle peut fournir et de toutes les actions qu'elle peut effectuer.

Le fichier qui contient la liste des perceptions a le format suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<sensor_description>
  <sensor id="nom_entite ">
    <perception name="nom_perception" min="X" max="Y" type="un_type" />
  </sensor>
</sensor_description>
```

Ce fichier XML est constitué de trois balises différentes :

- **sensor_description** : c'est la balise racine du fichier de perceptions. Elle contient un nombre indéterminé de balises **sensor**,
- **sensor** : c'est la balise qui détermine une entité numérique. Elle contient un nombre indéterminé de balise **perception**. Le paramètre **id** indique le nom de l'entité. L'id utilisé doit être le même nom qui est utilisé par ArCo pour recenser cette entité,
- **perception** : c'est la balise qui décrit les perceptions pouvant être fournies par l'entité. Le paramètre **name** indique le nom de la perception. Il existe trois types possibles : string, boolean et number. Pour le dernier type, il est possible de borner une valeur minimum et/ou une valeur maximum.

Le fichier qui contient la liste des actions a le format suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<entity_description>
  <entity id="nom_entite">
    <fonction name="nom_fonction">
      <argument id="id_argument" type="un_type" />
      <ordre>
        un_ordre
      </ordre>
    </fonction>
  </entity>
</entity_description>
```

Ce fichier XML est constitué de cinq balises différentes :

- **entity_description** : c'est la balise racine du fichier d'actions. Elle contient un nombre indéterminé de balises **entity**,
- **entity** : c'est la balise qui détermine une entité numérique. Elle contient un nombre indéterminé de balises **fonction**. Le paramètre **name** indique le nom de la fonction et doit être unique au sein d'une même balise **entity**,
- **fonction** : une balise fonction est déterminée par un nombre indéterminé de balise **argument** et d'une balise **ordre**. Elle représente les actions que peuvent effectuer les entités,
- **argument** : une balise argument est composée d'une chaîne de caractère **id** et d'un **type**. L'id est le nom de l'argument. Cette chaîne de caractère apparaît dans la balise **ordre**, pour faire la correspondance entre l'argument de la fonction et son utilisation dans la balise **ordre**. Ainsi, deux balises **argument** ne doivent pas avoir la même **id**. Et cette chaîne de caractère ne doit pas être assez commune pour pouvoir se trouver dans la balise **ordre** par hasard,
- **ordre** : cette balise contient une chaîne au format XML représentant l'ordre à envoyer à l'entité.

Synthèse

Une entité agissante adaptée à ArCo est décrite par un fichier XML qui présente toutes les perceptions fournies ou toutes les actions qu'elle peut effectuer.

3. Protocole de communication

Les messages transitent à travers des sockets TCP/IP sous la forme d'une chaîne XML au format MICE. La communication est standardisée afin de faciliter l'ajout de nouveaux modules. Le format MICE, s'inspirant des emails, est le suivant :

```
<mice type="">
  <subject></subject>
  <from></from>
  <to></to>
  <size></size>
  <time></time>
  <frequency></frequency>
  <message></message>
</mice>
```

Un message MICE est une chaîne XML dont la racine est une balise **mice**. Cette balise possède deux types d'informations :

- Structurelles : ce sont les informations relatives au transport et au type des données. C'est une partie fixe du message et tous les modules sont capables de la lire. Il s'agit des balises :
 - **type** : représente le type de données et permet aux modules qui reçoivent le message de savoir rapidement s'ils doivent le traiter ou non, et comment le traiter,
 - **subject** : apporte une précision sur le contenu du message et permet aux modules de savoir comment interpréter le sens du message contenu dans la balise message,
 - **from** : indique l'expéditeur du message,
 - **to** : indique le destinataire du message,
 - **time** : indique l'heure à laquelle le message a été envoyé sous la forme suivante : yyyyMMddHHmmss. Par exemple si nous sommes le 24 juin 2012 et qu'il est 12h54 et 36 secondes, alors la balise time contient : 20120624125436,
 - **frequency** : indique si ce message revient périodiquement ou non en donnant la valeur de la fréquence,
 - **size** : indique la taille du message, c'est-à-dire le nombre de caractères contenus dans la balise message.
- Sémantiques : Il s'agit de la balise **message**. Ces informations ne sont pas standardisées, leur format est libre. Ainsi chaque module n'est pas capable d'interpréter ces informations. Seuls les modules qui ont été programmés pour déchiffrer le message sont capables d'interpréter le contenu et de le transformer.

Le Tableau 5 de la page 86 présente des messages réservés utilisés pour la configuration du système. Il existe 4 types de messages :

- Configuration : ces messages permettent de gérer les modules et le chemin de l'information,
- Informationnel : ces messages permettent de récupérer des informations concernant les modules (liste des actions et des perceptions et liste des modules connectés),
- Action : ces messages permettent de gérer les actions,
- Perception : ces messages permettent de gérer les perceptions.

a) Configuration

Lorsqu'un module se connecte, il envoie le message MICE_1 afin que le serveur enregistre son nom et qu'il puisse lui faire parvenir des messages. Un module peut s'inscrire à la réception des messages d'identification MICE_2. De cette façon, il reçoit également les messages d'identification des modules. Le module reçoit également un message lorsque le module s'est déconnecté du serveur. Il peut ainsi connaître la liste des modules connectés. Le message MICE_3 permet de se désinscrire à la réception des messages d'identification. Un module peut créer un chemin d'information entre deux modules en envoyant le message MICE_4 ou en détruire un avec le message MICE_5. Il est possible d'enlever toute référence au module en envoyant le message MICE_6. Dans ce cas, le module n'existe plus pour le serveur.

b) Informationnel

Lorsque le serveur reçoit un message d'identification, il vérifie dans un fichier spécifique si ce module s'est déjà connecté au serveur. Si ce n'est pas le cas alors le module est nouveau et le serveur l'inscrit dans le fichier des modules connus et demande sa description avec le message MICE_7. Le module fournit le contenu de ses fichiers de description (liste des actions et des perceptions) avec le message MICE_8 que le serveur peut enregistrer dans des fichiers spécifiques. Un module peut demander la description de tous les modules connus de MICE, actuellement connectés ou non. Il envoie le message MICE_7 pour faire sa demande. La réponse du serveur correspond au message MICE_8. Les modules peuvent également demander la liste des modules actuellement connectés auprès du serveur avec le message MICE_9. Le serveur envoie alors la liste des modules connectés avec le message MICE_10. Le client peut alors connaître la description de l'environnement en exécution.

c) Action

Lorsqu'un module envoie un message d'action MICE_13 à destination d'une entité agissante active, l'action est associée à un numéro d'identification et à un nom, ce qui lui permet d'être unique au sein de toute l'architecture. Lorsque l'entité agissante a fini de réaliser l'action, elle envoie un message MICE_14 au serveur indiquant que l'action est finie. A la réception de la notification de fin d'action, le serveur relaie le message à tous les modules qui en ont fait la demande. Pour qu'un module reçoive les notifications de fin d'action, il doit s'inscrire auprès du serveur avec le message MICE_11. Il peut se désinscrire à tout moment avec le message MICE_12. Ce fonctionnement basé sur les inscriptions permet d'éviter que tous les modules voient les messages de fin d'action. Cela permet de ne pas trop charger la bande passante.

d) Perception

Le message MICE_15 est un message de perception, envoyée par les entités agissantes perceptives, pour donner une information sur l'environnement.

e) Autre message

Il est toujours possible à chaque développeur de définir des messages personnels pour permettre à chaque module d'effectuer des calculs sur l'interaction. Un module n'est pas forcément lié à une entité agissante. Par exemple, il peut y avoir des modules de logs ou des modules d'affect chargés de calculer l'émotion d'une phrase et d'envoyer une perception d'émotion. Ce sont des modules intermédiaires qui peuvent déterminer leur propre sémantique du message MICE.

Tableau 5 : Liste des messages MICE réservés

Identifiant	Commande	Type	Emise par	Rôle
MICE_1	id	configuration	Modules	Connexion initial à MICE.
MICE_2	subscribe(id)			Souscrit aux notifications indiquant lorsque des modules se connectent au serveur.
MICE_3	unsubscribe(id)			Annule la souscription aux notifications des connexions.
MICE_4	connect			Etablit le chemin de l'information entre deux modules.
MICE_5	disconnect			Détruit le chemin de l'information entre deux modules.
MICE_6	delete			Indique que le module n'existe plus et doit être enlevé du serveur MICE.
MICE_7	get(description)	informationnel	Modules	Demande des descriptions des modules.
MICE_8	set(description)			Envoie des descriptions. Les modules envoient leur propre description. Le serveur envoie la description des modules qui se sont connectés au moins une fois.
MICE_9	get(modules)		Modules	Demande la liste des modules actuellement connectés.
MICE_10	set(modules)		Serveur	Envoie la liste des modules actuellement connectés.
MICE_11	subscribe(action)	action	Modules	Souscrit aux notifications indiquant qu'une action est terminée.
MICE_12	unsubscribe(action)			Annule la souscription aux notifications des actions.
MICE_13	action			Message d'action destiné aux entités agissantes actives.
MICE_14	action state			Indique la fin d'une action.
MICE_15	perception	perception	Modules	Message de perception destiné aux entités agissantes perceptives.

La description détaillée de l'ensemble de ces messages réservés est disponible dans l'annexe C.

4. AmbiPath : Module pour la création du chemin logique

AmbiPath est un module qui permet d'établir les liens entre les modules à partir d'un fichier passé en paramètre. Il permet d'éviter de faire ce travail à la main. Il est en effet possible d'établir les liens, les uns après les autres, manuellement. Cependant, pour des applications où les modules sont fixes (par exemple StimCards, cf. chapitre IV.B), il est intéressant de définir ces liens automatiquement.

AmbiPath est un programme Java, AmbiPath.jar, qui se lance de la façon suivante :

```
> java -jar AmbiPath.jar [adresse du serveur] [port du serveur] [chemin de fichier de liens]
```

Le fichier de liens est composé d'un ensemble de chaînes XML connect et disconnect. En ajoutant à notre exemple deux interpréteurs interpreter01 et interpreter02, voici le fichier de configuration des liens :

```
<connect><in>camera</in><out>interpreter_symbolique</out></connect>
<connect><in>micro</in><out>interpreter_symbolique</out></connect>
<connect><in>interpreter_symbolique</in><out>interpreter01</out></connect>
<connect><in>interpreter_symbolique</in><out>interpreter02</out></connect>
<connect><in>interpreter01</in><out>gestion_monde</out></connect>
<connect><in>interpreter02</in><out>gestion_monde</out></connect>
<connect><in>gestion_monde</in><out>interpreter01</out></connect>
<connect><in>gestion_monde</in><out>interpreter02</out></connect>
<connect><in>gestion_monde</in><out>robot</out></connect>
<connect><in>gestion_monde</in><out>agent_virtuel</out></connect>
```

Le fichier donne le résultat montré sur la Figure 44.

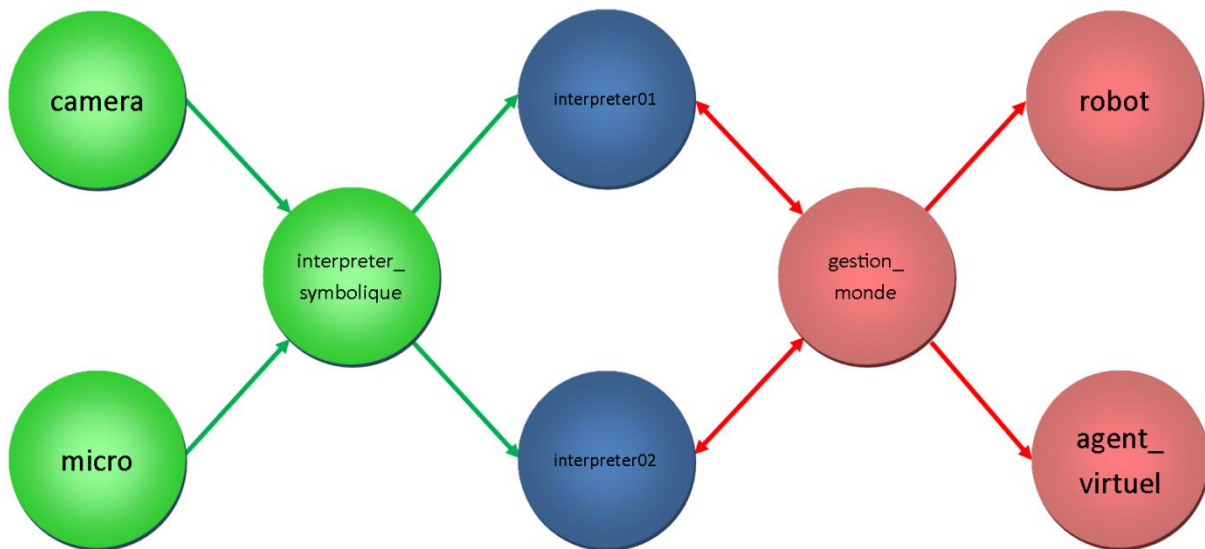


Figure 44 : Lien logique entre les modules de l'exemple

D. Entités agissantes : Modules d'entrées et sorties

Chacune des entités agissantes appartient à un même **environnement** qui est, par exemple, la maison dans le contexte d'une personne à domicile. La Figure 45 présente les deux types de dispositifs numériques contenus dans l'environnement :

- Les **entités agissantes perceptives** (caméra, micro, thermomètre, détecteur de présence...), mesurent l'environnement. Les données mesurées s'appellent des **perceptions**. Par exemple, une perception peut être : **personne_tombée**, **prise_medicament**, **personne_dit_bonjour** ... qui indiquent respectivement qu'une personne est tombée ou qu'il est l'heure de prendre un médicament.

Une perception $P(m, p, v, c)$ est un quadruplet composé de :

- m : Le nom du module émetteur de la perception,
- p : Le nom de la perception,
- v : Une valeur associée à la perception,
- c : Un coefficient de vraisemblance.

Exemple

La perception $P(\text{camera}, \text{personne_tombée}, 1, 100)$ indique que l'entité agissante perceptive « camera » a envoyé une perception nommée **personne_tombée**. La valeur 1 indique que la personne est tombée. Le coefficient 100 indique que l'entité estime à 100% la fiabilité de l'information.

- Les **entités agissantes actives** effectuent des **actions** sur l'environnement. Les entités agissantes peuvent être associées à des robots, des personnages virtuels ou des actionneurs spécifiques (capteur de présence, thermomètre, accéléromètre...) capables d'agir sur l'environnement. Les actions peuvent être, par exemple, **baisse_chauffage**, **ferme_porte**, **dit_bonjour**, **appel_secours**, **salue_personne** ...

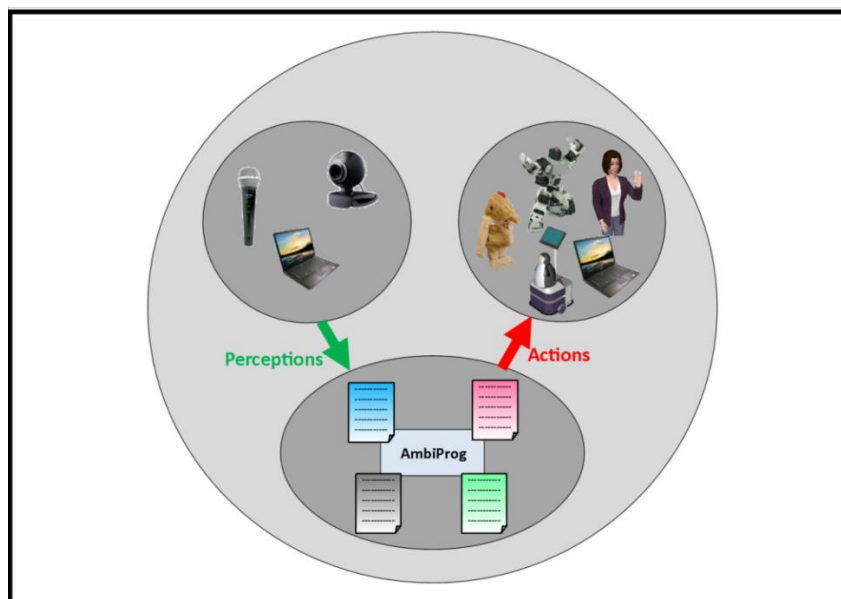


Figure 45 : Perceptions et actions dans ArCo

Le paragraphe précédent (III.A.1) indique que le comportement¹⁴ des entités agissantes est le résultat de l'exécution des scénarii d'interaction. Ces scénarii ont pour objectif de faire le lien entre les perceptions et les actions. Par exemple, avec une entité agissante perceptive appelée *micro* et une entité agissante active appelée *robot*, un scénario peut être « Si le micro détecte que la personne a dit bonjour, alors le robot salue la personne »¹⁵. Le scénario fait le lien entre la perception `personne_dit_bonjour` et l'action `salue_personne`.

Il existe un type particulier de perceptions appelé **perception symbolique**. Il s'agit de perceptions évaluées grâce à l'interprétation d'un scénario spécifique. Une perception symbolique est créée à partir de perceptions simples. Les perceptions symboliques servent à améliorer l'analyse de l'environnement à partir d'informations qui ne sont pas fournies *a priori*. Par exemple, « si l'accéléromètre indique une position horizontale et que la caméra n'affiche plus l'humain, alors la perception symbolique `personne_tombée` est créée ». Cependant, ce calcul de perception symbolique n'est pas toujours possible car l'environnement de l'utilisateur est dynamique. Il est possible que certaines perceptions nécessaires au calcul d'une perception symbolique ne soit pas fournies car l'entité agissante associée ne se trouve pas en interaction avec un utilisateur.

Lorsqu'un scénario cible une action simple, il spécifie l'entité agissante qui doit exécuter l'action. Mais il n'est pas toujours possible de savoir quelle entité agissante sera disponible pour l'effectuer ou alors quelle entité agissante sera la plus efficace pour l'effectuer. Ainsi, si plusieurs entités agissantes peuvent effectuer la même action, le scénario peut cibler une **action générique**, qui est un type particulier indiquant qu'une entité agissante précise n'est pas ciblée. Lorsqu'un scénario indique qu'une action générique doit être effectuée, au moment de l'exécution, AmbiCop choisit l'entité agissante qui convient le mieux pour effectuer cette action. Ainsi, si un nouveau robot est ajouté à l'environnement et que celui-ci est capable d'effectuer une action plus rapidement que les autres, il deviendra prioritaire pour l'exécution de cette action. La façon dont l'entité agissante est choisie est détaillée dans le 0. Cependant, s'il n'est pas disponible, c'est la meilleure entité agissante libre qui sera choisie. Il est également possible que toutes les entités agissantes soient déjà en train d'exécuter des actions. Dans ce cas, l'action générique risque de ne pas être exécutée.

Il existe un autre type d'action : les **actions synchronisées**, qui ciblent un groupe d'actions spécifiques. Cela signifie que les entités agissantes envoient l'ordre de démarrage de toutes les actions en même temps, et ne sont toutes libérées que lorsque l'ensemble des dispositifs numériques associés a fini d'exécuter leur action. Ainsi, l'interprétation du reste du scénario n'est possible que lorsque toutes les entités agissantes sont libérées.



Nous posons l'hypothèse que les actions sont effectivement correctement exécutées. Chaque entité agissante envoie une notification de fin d'action à l'interpréteur de scénario lorsque l'action doit être terminée mais nous ne vérifions pas la validité de cette information.

¹⁴ Le comportement d'une entité agissante est défini par l'ensemble des actions qui lui sont associées lors de l'interprétation des scénarii.

¹⁵ Nous notons entre guillemets une traduction française d'un scénario. Bien évidemment ce n'est pas le « micro qui détecte » et « le robot qui salue », mais l'entité agissante micro qui envoie une perception et l'entité agissante robot qui envoie une action au robot physique.

E. AmbiProg : Outil pour la création de scénario

Le logiciel AmbiProg offre une interface graphique permettant la réalisation de scénarii d'ArCo par une programmation visuelle des entités agissantes. Il se configure automatiquement en fonction des entités agissantes disponibles dans l'environnement. Cette configuration automatique se base sur un mécanisme de description des fonctionnalités des entités agissantes. Cette description est la liste de toutes les perceptions que peut effectuer une entité agissante perceptive et la liste de toutes les actions que peut effectuer une entité agissante active. Elle est contenue dans un fichier au format XML (cf. Chapitre III.C.2.c)). C'est ce fichier que charge AmbiProg à son démarrage. Il peut alors proposer la liste de toutes les perceptions et toutes les actions disponibles dans l'environnement. Le langage de programmation visuelle simple permet à des non-experts de créer des scénarii, sans connaissance approfondie en informatique, comme nous avons pu le vérifier au cours de deux expérimentations (cf. Chapitre V.B). La partie 1 présente les fichiers XML des entités qui seront utilisés, à titre d'exemple, dans la suite du document. La partie 2 montre l'organisation de l'interface graphique tandis que la partie 3 décrit le langage visuel proposé.

1. Scénario d'interaction

Un scénario d'interaction décrit donc des interactions possibles entre les humains et le système informatique, et entre les différents éléments du système informatique. La création des scénarii permet de répondre à la question : « Que doivent faire les entités agissantes en fonction des différents événements surgissant dans l'environnement ? ». Les scénarii sont les données d'ArCo. En leur absence, les entités agissantes ne sont pas coordonnées et effectuent seulement le rôle pour lequel elles ont été conçues. Les scénarii d'interaction utilisent les fonctionnalités existantes des entités agissantes pour effectuer de nouvelles actions.

Ainsi, les scénarii permettent aux utilisateurs de définir un grand nombre d'interactions et de personnaliser un Compagnon Artificiel pour un utilisateur. La personnalisation est une condition nécessaire à l'acceptation d'une technologie (Chatterjee 2007) par les humains. Notre objectif est de fournir un Compagnon Artificiel qui réponde à toutes les attentes de l'utilisateur, grâce à cette possibilité de personnalisation, et afin qu'il soit mieux accepté.

Les scénarii d'interaction, par l'utilisation des actions génériques (cf. Chapitre III.C.3.c), permettent d'unifier la façon de programmer chaque objet numérique et de limiter le temps d'apprentissage d'utilisation de chacun d'entre eux. À l'achat d'un matériel, chaque utilisateur doit passer un temps non négligeable à apprendre son utilisation. Et deux objets différents ne s'utilisent pas nécessairement de la même façon. Par exemple, l'interface de programmation d'un réveil et d'une machine à café ne présente pas les informations de la même façon. Il faut un temps d'apprentissage pour les deux objets. Avec ArCo, ces deux objets peuvent relier des informations et être programmés via l'interface AmbiProg. Un prestataire de service peut alors écrire un scénario qui permet de lancer la machine à café lorsqu'un utilisateur a stoppé la sonnerie du réveil. Par défaut, une machine à café et un réveil n'ont pas de liens et ne communiquent pas. ArCo offre la possibilité de mettre en relation tous types de dispositifs, de les rendre interopérable.

Les scénarii d'interaction se basent sur quatre concepts :

- La notion de **conditionnelle** : se présente sous la forme « si... alors...(sinon...) ». L'exemple précédent (avec le micro) utilise ce concept,
- La notion de **boucle** : permet de répéter un morceau de programme un certain nombre de fois ou jusqu'à une certaine perception soit envoyée,
- L'**attente de perception** : permet de définir un scénario qui s'exécute lorsqu'une perception spécifique est reçue. L'attente de perception bloque le programme jusqu'à réception de la perception,
- Le **parallélisme** : permet d'écrire plusieurs scénarii qui s'exécutent en même temps.

Synthèse

ArCo est une architecture qui permet de coordonner des entités agissantes perceptives et actives grâce à des scénarii d'interaction qui permettent d'adapter le comportement du Compagnon Artificiel pour un utilisateur.

2. Configuration d'AmbiProg : un exemple

Lors du démarrage d'AmbiProg, celui-ci charge deux fichiers de description fournis par le serveur de MICE utilisé dans l'architecture ArCo (cf. 0) qui regroupent tous les perceptions d'une part et toutes les actions d'autre part.

Dans la suite de ce document, nous utiliserons un exemple composé de quatre entités : une caméra, un micro, un robot et un agent virtuel. Les deux fichiers suivants font la liste des perceptions (fichier perceptions.desc) et des actions (fichier actions.desc) de ces entités.

Fichier perceptions.desc

```
<?xml version="1.0" encoding="UTF-8"?>
<sensor_description>
  <sensor id="camera">
    <perception name="nom_personne" type="string" />
    <perception name="personne_tombée" min="0" max="1" type="boolean" />
    <perception name="nombre_personnes" min="0" max="1000" type="number" />
  </sensor>
  <sensor id="micro">
    <perception name="dit_bonjour" min="0" max="1" type="boolean" />
    <perception name="dit_au_revoir" min="0" max="1" type="boolean" />
    <perception name="dit" type="string" />
  </sensor>
</sensor_description>
```

Le fichier précédent décrit deux entités agissantes perceptives : une caméra et un micro.

La **camera** peut envoyer trois perceptions :

- **nom_personne** : est une chaîne de caractère qui correspond au nom d'une personne.
- **personne_tombée** : est un booléen qui informe sur une chute possible de la personne. Si la perception vaut 0, cela indique que la personne n'est pas tombée. Si la perception vaut 1, cela indique que la personne est tombée.
- **nombre_personnes** : est un nombre entre 0 et 1000 qui correspond au nombre de personnes présentes dans le champ de la caméra.

Le **micro** peut envoyer trois perceptions :

- **dit_bonjour** : est un booléen qui indique si la personne a dit bonjour ou pas. Si la perception vaut 0, la personne n'a pas dit bonjour (mais a parlé). Si la perception vaut 1, la personne a dit bonjour.
- **dit_au_revoir** : est un booléen qui indique si la personne a dit au revoir ou pas. Si la perception vaut 0, la personne n'a pas dit au revoir (mais a parlé). Si la perception vaut 1, la personne a dit au revoir.
- **dit** : est une chaîne de caractère qui correspond à une parole prononcée par une personne.

Fichier actions.desc

```
<?xml version="1.0" encoding="UTF-8"?>
<entity_description>
  <entity id="robot">
    <fonction name="dit">
      <argument id="phrase" type="string" />
      <ordre>
        ...
      </ordre>
    </fonction>
    <fonction name="dit_bonjour">
      <ordre>
        ...
      </ordre>
    </fonction>
    <fonction name="dit_aurevoir">
      <ordre>
        ...
      </ordre>
    </fonction>
    <fonction name="appelle_secours">
      <argument id="numero_appel" type="string" />
      <ordre>
        ...
      </ordre>
    </fonction>
  </entity>
  <entity id="agent_virtuel">
    <fonction name="dit">
      <argument id="phrase" type="string" />
      <ordre>
        ...
      </ordre>
    </fonction>
    <fonction name="dit_bonjour">
```

```

    <ordre>
        ...
    </ordre>
</fonction>
<fonction name="dit_aurevoir">
    <ordre>
        ...
    </ordre>
</fonction>
<fonction name="appelle_secours">
    <argument id="numero" type="string" />
    <ordre>
        ...
    </ordre>
</fonction>
<fonction name="lit_informations">
    <ordre>
        ...
    </ordre>
</fonction>
</entity>
</entity_description>

```

Le contenu des balises `ordre` n'est pas explicité ici car il est dépendant du robot ou du personnage virtuel cible et n'a pas d'importance pour les exemples donnés. Ces exemples se focalisent sur la partie « fixe » des fichiers XML.

Le fichier précédent décrit deux entités agissantes actives : un **robot** et un **agent_virtuel**.

Ces deux entités peuvent effectuer les mêmes actions génériques :

- **dit** : cette action fait prononcer une phrase passée en paramètre au **robot** ou à **agent_virtuel**.
- **dit_bonjour** : le **robot** ou l'**agent_virtuel** disent bonjour à la personne.
- **dit_aurevoir** : le **robot** ou l'**agent_virtuel** disent au revoir à la personne.
- **appelle_secours** : le **robot** ou l'**agent_virtuel** appellent les secours en utilisant le numéro de téléphone passé en paramètre de la fonction.

L'**agent_virtuel** est capable d'effectuer une action en plus :

- **lit_informations** : l'agent virtuel donne une liste d'actualités.

En chargeant ces deux fichiers, AmbiProg peut créer des scénarii qui mettent en relation ces quatre entités agissantes. La partie suivante montre l'interface graphique une fois démarrée.

3. Présentation générale

AmbiProg est composé de trois parties, comme le montre la Figure 46. La partie 1 est le **menu** de l'application. La partie 2 est la **zone de construction du scénario**. Enfin, la partie 3 est la **zone d'éléments de scénarii**. Ces trois parties sont détaillées ci-dessous.

La partie 1 est le menu de l'application. La partie 2 est la zone de construction du scénario. Enfin, la partie 3 est la zone de programmation. C'est à cet endroit que l'on choisit les éléments du scénario à assembler dans la partie 2.

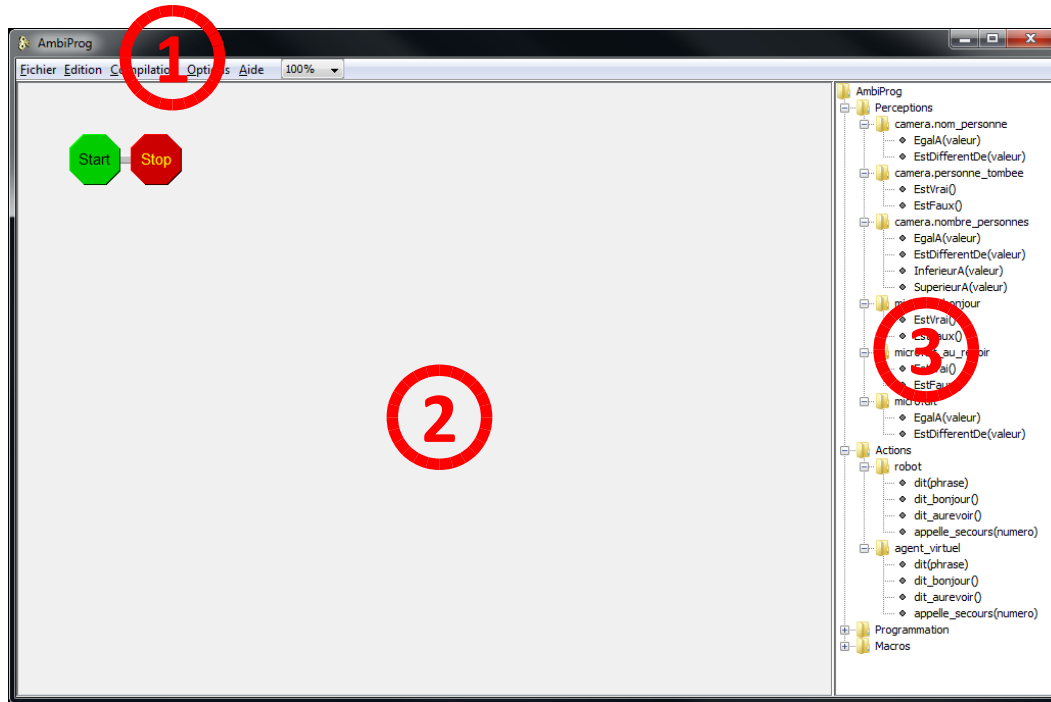


Figure 46 : Organisation d'AmbiProg

La zone de programmation propose les éléments suivants :

- PARALLELISME : cet élément est composée de plusieurs objets BRANCHE. Ainsi, il est possible d'écrire plusieurs sous-scénarii qui seront interprétés en même temps,
- BRANCHE : permet d'écrire un sous-scénario,
- ACTIONS SYNCHRONISEES : cet élément se compose de BRANCHE, comme l'élément PARALLELISME. il permet d'écrire un ensemble d'actions qui seront exécutées de façon synchronisées,
- REPETER : cet élément permet de faire répéter un sous-scénario un nombre de fois choisi par l'utilisateur,
- TANT QUE : cet élément permet d'indiquer qu'un sous-scénario sera interprété tant qu'une certaine condition sera vraie,
- SI ALORS : cet élément représente une conditionnelle « si alors »,
- SI ALORS SINON : cet élément représente une conditionnelle « si alors sinon »,
- EVENEMENT : cet élément est bloquant et indique qu'un sous-scénario sera interprété seulement lorsqu'une certaine perception sera reçue,

- ATTENDRE : cet élément permet de faire une pause d'interprétation pendant un laps de temps défini par l'utilisateur,
- BREAK : cet élément permet de sortir des boucles ou de l'élément PARALLELISME,
- ET, OU, NON, () : ces éléments permettent de créer des conditionnelles plus complètes.

Une présentation plus complète d'AmbiProg avec des captures d'écran et l'explication complète du langage visuel est disponible dans l'annexe D.

4. Grammaire du langage généré par AmbiProg

La grammaire du langage AmbiProg est présentée ci-dessous :

programme	::=	instructions
instructions	::=	instructions ; { instructions ; } * action répéter tantque parallèle conditionnelle événement WAIT(nb) BREAK
action	::=	ident (param ^{0/1})
ident	::=	ident.ident
param	::=	variable { , variable } *
variable	::=	nb ident
répéter	::=	nb *(instructions)
tantque	::=	*[condition](instructions)
condition	::=	perception !(condition) (condition) condition & condition condition condition
perception	::=	ident (param ^{0/1})
parallèle	::=	//(branches)
actions_sync	::=	/\(branches)
branches	::=	instructions , { instructions } *
conditionnelle	::=	[condition](instructions){ !(instructions) } ^{0/1}
événement	::=	< condition >(instructions)

Ce langage est construit de façon à pouvoir être écrit sur une seule ligne et sans mot (utilisation uniquement de caractères de ponctuation) afin de prendre le minimum de place et minimiser le débit sur la bande passante d'ArCo. Le terme ident représente un identificateur (une chaîne de caractères). Le terme nb représente un nombre entier. Le symbole * indique que l'élément concerné est présent 0, 1 ou plusieurs fois. Le symbole 0/1 indique que l'élément concerné est présent 0 ou 1 fois pris au sens habituel.

5. Exemple de scénario

L'annexe D a montré la création de la macro surveillance_chute dont le scénario visuel est représenté par la Figure 47. Ce scénario indique que l'interpréteur est en attente de la réception d'une perception **personne_tombée** en provenance du module **camera**. Lorsque cette perception est reçue, alors le module **robot** appelle les secours.

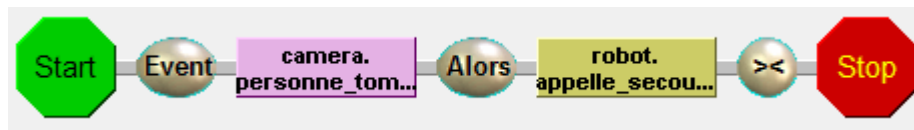


Figure 47 : Scénario visuel de la macro "surveillance_chute"

Le code généré par AmbiProg est le suivant :

```
<camera.personne_tombée.EstVrai()>(robot.appelle_secours(18));;
```

La Figure 48 montre un autre exemple un peu plus complet utilisant l'élément CONDITIONNELLE, l'élément ACTIONS SYNCHRONISEES, l'élément EVENEMENT et utilise une action générique.

Ce scénario indique que si une personne dit bonjour (information captée par le micro et envoyée en tant que perception) alors le robot et l'agent virtuel effectuent, de façon synchronisée, l'action **dit_bonjour()**. Ensuite, le scénario indique que la perception **micro.dit_aurevoir()** est attendue. Lorsque celle-ci est reçue, soit le robot, soit l'agent virtuel effectue l'action **dit_aurevoir()**. « all » indique qu'il s'agit ici d'une action générique.



Figure 48 : Scénario visuel simple

Le code généré par AmbiProg est le suivant :

```
[micro.dit_bonjour.EstVrai() ]
(\/(robot.dit_bonjour();,agent_virtuel.dit_bonjour();,);
  <micro.dit_aurevoir.EstVrai()>(
    all.dit_aurevoir();
  );
);
```

Ce code est écrit normalement sur une seule ligne, mais pour faciliter la lecture, il a été affiché ici sur plusieurs lignes et avec des indentations.

F. AmbiLive : Module d'interprétation des scénarii

AmbiLive est le module qui permet d'exécuter un scénario d'interaction construit par AmbiProg. La Figure 49 montre son organisation générale. AmbiLive charge un scénario et lance l'interprétation. Il existe dans ArCo autant de module AmbiLive qu'il y a de scénario à interpréter. Ce module reçoit des perceptions en entrée et génère des actions en sortie. Pour manipuler ces perceptions et ces actions, il gère deux listes fournies par MICE :

- Liste des entités agissantes perceptives et leurs perceptions possibles associées,
- Liste des entités agissantes actives et leurs actions possibles associées.

L'interpréteur effectue deux tâches principales en parallèle :

- La gestion des perceptions reçues,
- L'interprétation d'un scénario.

À la réception d'une perception, l'interpréteur vérifie qu'elle existe dans la liste et si c'est le cas, la rajoute à la mémoire interne. La fréquence fournie par le message MICE (cf. Chapitre III.C.3) est utilisée en tant que durée de vie de la perception. Ainsi lorsque la durée de vie est passée, la perception est enlevée de la mémoire.

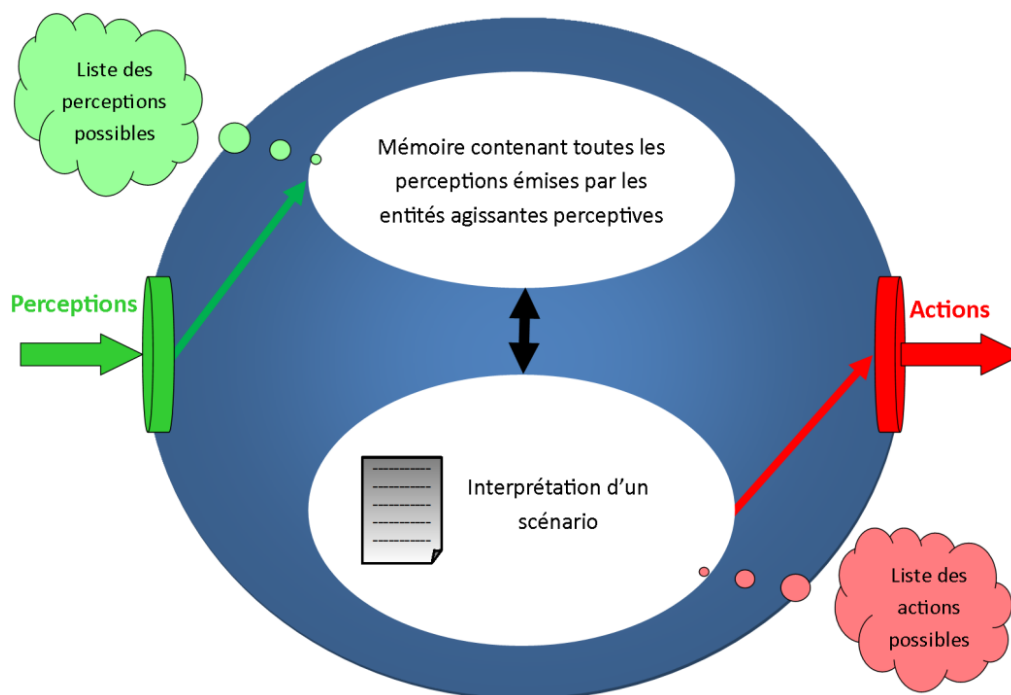


Figure 49 : Organisation générale de l'interpréteur

Dans le même temps, l'interpréteur lit le scénario qui peut contenir des conditionnelles et des fonctions, comme les langages de programmations classiques. Les conditionnelles utilisent les perceptions, tandis que les fonctions sont directement liées aux actions. L'interpréteur utilise les perceptions et les actions de la façon suivante :

- Les données d'entrées du scénario sont les perceptions. Chaque fois que l'interpréteur analyse une conditionnelle, il lit dans sa mémoire interne, la valeur de la perception concernée pour pouvoir effectuer le test,
- Les données de sorties du scénario sont les actions. Chaque fois que l'interpréteur analyse une fonction, il cherche dans la liste des actions possibles l'ordre à envoyer à l'entité agissante active. L'interprétation consiste à remettre en sortie la chaîne correspondante à la fonction dans la balise `ordre` du fichier XML de description de l'entité agissante (cf. *Chapitre III.D*).

L'interpréteur est fondé sur la grammaire décrite dans le Chapitre III.E.4. Il utilise cette grammaire pour effectuer une analyse syntaxique qui permet de transformer un scénario en arbre syntaxique (cf. Annexe E). Cet arbre syntaxique offre la possibilité d'interpréter simplement et automatiquement un scénario (cf. Annexe F).

L'interpréteur est construit à partir de la bibliothèque JAVA AmbiLib (cf. Annexe G). qui fournit une interprétation des éléments structurels du langage AmbiProg. Par exemple, l'interprétation d'une boucle, d'un parallélisme ou d'un break peut se faire de façon générale. Ainsi, AmbiLib contient l'interprétation de tous les éléments de programmation d'AmbiProg qui sont indépendante d'un contexte d'utilisation et donc indépendante du fonctionnement d'ArCo. Cette bibliothèque généralise les entrées et les sorties d'un programme. Les entrées sont les variables dans un langage de programmation classique ou les perceptions dans ArCo. Les sorties sont les fonctions ou les actions dans ArCo. Ainsi, grâce à AmbiLib, le langage AmbiProg peut facilement être réutilisé dans un autre contexte que celui d'ArCo. Et ainsi, on permet à d'autres développeurs de l'utiliser.

1. Gestion des entrées et sorties du module d'interprétation

Le module d'interprétation utilise la bibliothèque AmbiLib et surcharge les fonctions principales pour inclure le contexte lié à l'architecture ArCo : utilisation de perceptions et d'actions en tant qu'entrées et sorties du programme. L'interpréteur a accès aux fichiers de description des perceptions et des actions.

Les perceptions sont gérées par la classe Variables. Ainsi lorsqu'une perception est envoyée par une entité agissante perceptive, la perception est ajoutée dans **Variables**. Lors de l'interprétation des scénarii, l'interpréteur peut interroger l'objet de la classe Variables.

Les actions sont gérées par la classe Action. Une action se décompose en trois parties : une liste d'identificateurs, un nom de fonction, une liste de paramètres. Lorsqu'une action doit être effectuée, l'interpréteur cherche dans le fichier de description des actions, l'entité agissante qui doit envoyer la perception. Ensuite, il fait le lien entre l'action et le message à envoyer à l'entité agissante.

Par exemple, si l'on prend l'action suivante :

```
robot.appelle_secours(18)
```

Elle est décomposée en :

- Identificateur : robot
- Nom de fonction : appelle_secours
- Paramètre : 18

Dans le fichier de description, l'entité robot possède la fonction suivante :

```
<fonction name="appelle_secours">
  <argument id="numero_appel" type="string" />
  <ordre>
    <communication type="telephone" message="secours" numerotation="numero_appel" />
  </ordre>
</fonction>
```

L'argument passé en paramètre de la fonction est appelé **numero_appel** dans le message **ordre**. Ainsi, le terme **numero_appel** est modifié par **18** et le message à envoyer à l'entité robot est :

```
<communication type="telephone" message="secours" numerotation="18" />
```



Il existe un paramètre particulier qui permet de passer la valeur d'une perception ou la valeur d'un coefficient à une fonction afin d'avoir un système de variable dans AmbiProg. Pour cela, il suffit de passer `nom_module.nom_perception` pour que l'interpréteur remplace par la valeur de la perception ou `nom_module.nom_perception.coefficient` pour que l'interpréteur remplace par la valeur du coefficient.

2. Interprétation des perceptions symboliques

L'interpréteur des perceptions symboliques est un interpréteur particulier qui reçoit et qui génèrent exclusivement des perceptions. Son travail peut être décomposé en 3 étapes :

- Au démarrage, l'interpréteur récupère tous les scénarii contenus dans le répertoire passé en paramètre. Il crée un scénario global avec l'élément PARALLELISME. Sur chaque branche se trouve un des scénarii de perceptions symboliques,
- Ce scénario global est interprété en boucle, jusqu'à l'arrêt de l'interpréteur. L'interprétation interroge une mémoire interne qui contient l'ensemble des perceptions reçues par l'interpréteur,
- A chaque fois que le scénario doit interpréter la fonction `creePerceptionSymbolique`, il envoie la perception correspondante.

G. AmbiCop : Module d'arbitrage des actions

AmbiCop est un module qui reçoit les informations de tous les interpréteurs AmbiLive qui tournent dans l'architecture ArCo.

AmbiCop a trois rôles principaux :

- Gestion des conflits : dans un contexte de multi-scénarii, il est possible que deux scénarii demandent à une même entité agissante d'effectuer une action utilisant les mêmes ressources au même moment. C'est ce que nous appelons conflit. Le gestionnaire du monde doit gérer ces conflits,
- Gestion des actions génériques : comme les perceptions symboliques, il est possible de créer des actions symboliques qui sont basées sur des actions réelles. Une action symbolique n'est pas associée à une entité agissante précise. Cela permet au gestionnaire du monde de choisir quelle entité est la plus adaptée pour effectuer cette action. En effet, lorsqu'une personne écrit un scénario, elle ne connaît pas nécessairement en avance la meilleure entité pour effectuer l'action,
- Synchronisation des actions : l'élément ACTIONS_SYNC permet de déterminer plusieurs actions qui doivent commencer en même temps et être exécutées en même temps. Dans ce cas, il est important que les entités, qui sont utilisées par le scénario, ne soient pas déjà utilisées par d'autres scénarii. La gestionnaire du monde s'assure de lancer les actions lorsque les entités sont libres.

Ce chapitre présente l'organisation générale du gestionnaire du monde, puis décrit le fonctionnement des trois rôles du gestionnaire du monde. Ce travail a été réalisé, dans le cadre d'un stage de master, avec Germain Lemasson.

1. Organisation générale d'AmbiCop

Le module d'arbitrage des actions prend des actions qui viennent des interpréteurs en entrée et génère des actions en sortie à destination des entités agissantes actives. Il filtre toutes les actions.

La Figure 50 montre l'organisation générale d'AmbiCop

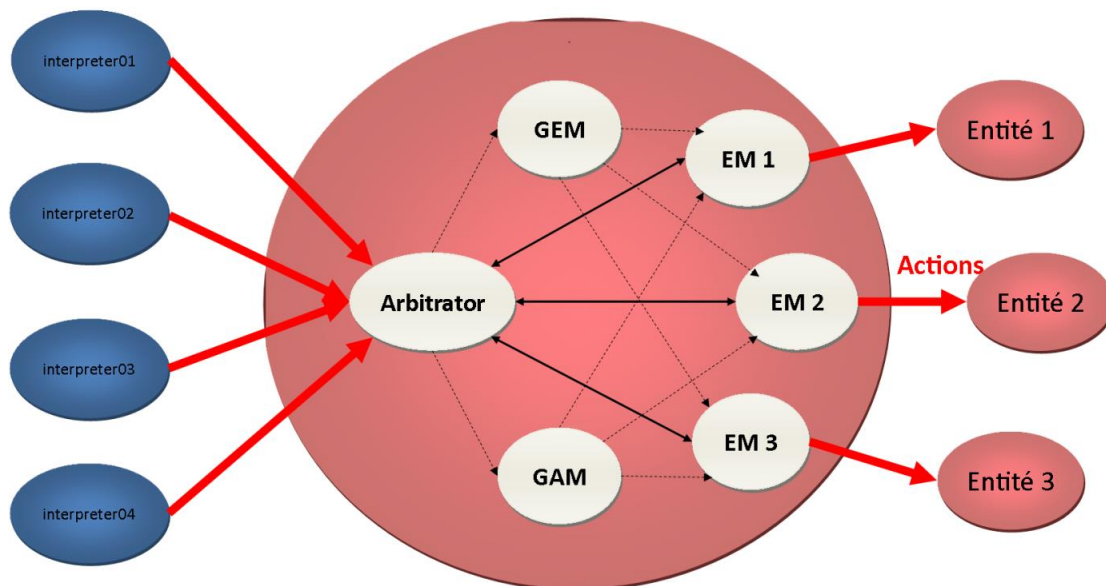


Figure 50 : Organisation générale d'AmbiCop

Un arbitre (*arbitrator*) récupère les messages des interpréteurs. L'arbitre reçoit les actions et les redirige en fonction de leur cible (entité réelle ou symbolique) et de leurs types (simples, symboliques ou synchronisées). Il coordonne les autres éléments du module d'arbitrage des actions. C'est l'arbitre qui est inscrit aux notifications de fin d'actions et qui retransmet les notifications aux interpréteurs lorsqu'elles ont été traitées.

AmbiCop prend en compte les connexions dynamiques des entités agissantes. Il est inscrit aux notifications d'identification des entités. Si des actions ont été envoyées à une entité pendant qu'elle était déconnectée, elles sont stockées. Lorsque l'entité se connecte, les actions lui sont envoyées en fonction des priorités et de sa qualité.

Un gestionnaire d'entité (Entity Manager EM) est chargé de gérer une entité. C'est le seul qui peut envoyer des actions à l'entité. Il y a autant de gestionnaires d'entités que d'entités agissantes dans l'environnement. Un gestionnaire d'entité générique (GenericEntityManager GEM) gère les actions symboliques. Et un gestionnaire d'actions synchronisées (GroupActionManager GAM) gère les entités de façon à pouvoir synchroniser les actions associées.

Le module d'arbitrage des actions utilise deux dimensions pour assurer ces trois rôles principaux : priorité et qualité :

- L'indice de priorité est envoyé par les scénarii car ceux-ci ont des priorités différentes qui peuvent être liés à :
 - l'auteur des scénarii : un scénario écrit par un médecin est plus prioritaire qu'un scénario écrit par un ami,
 - le type des scénarii : scénario de vie, de jeu, de surveillance, de santé...,
 - l'état de l'environnement : danger, calme...
- L'indice de qualité est fourni par les entités agissantes. En effet, celles-ci ont différentes fonctionnalités et chaque entité est plus ou moins efficace pour réaliser une même fonctionnalité.

2. Gestion des conflits

La gestion des conflits concerne l'envoi d'action simple. Il faut éviter que deux actions qui ciblent une même entité agissante soient envoyées en même temps. Lorsqu'une action simple est envoyée par l'interpréteur, elle est redirigée vers l'EM qui gère l'entité agissante concernée.

Un EM est composé de deux éléments :

- Une référence vers l'action actuellement exécutée par l'entité agissante,
- Une liste d'attente dont un exemple est décrit par le Tableau 6. Cette liste stocke les actions en attentes d'être exécutées, regroupées en fonction de la priorité de leur scénario. Les scénarii sont les points d'entrée du tableau. A l'intérieur d'un même scénario, les actions sont triées en fonction de leur qualité. Seule la liste avec la plus haute priorité est triée et maintenue dans l'ordre.

Tableau 6 : Exemple de liste d'attente d'un Entity Manager

Scénarii		Actions	
Nom	Priorité	Nom	Qualité
Scénario 1	10	Act 1	90
		Act 27	60
Scénario 2	5	Act 4	20
		Act12	60
		Act13	40
Scénario 3	1	Act9	90

La priorité du scénario est considérée plus importante que la qualité de l'entité agissante. Par exemple, une action d'un scénario de sécurité doit être exécutée avant une action d'un scénario de jeu, même si l'action de jeu a une meilleure qualité que l'action de sécurité.

La Figure 51 montre un exemple de gestion de priorité. Deux actions doivent être effectuées : l'action 01 du scénario 01 et l'action 02 du scénario 02 par la même entité. L'arbitrage des actions demande les priorités des deux scénarii. Comme le scénario 02 a une priorité plus élevée, l'action 02 est effectué en premier. Lorsque l'entité est libre, l'action 01 est effectuée.

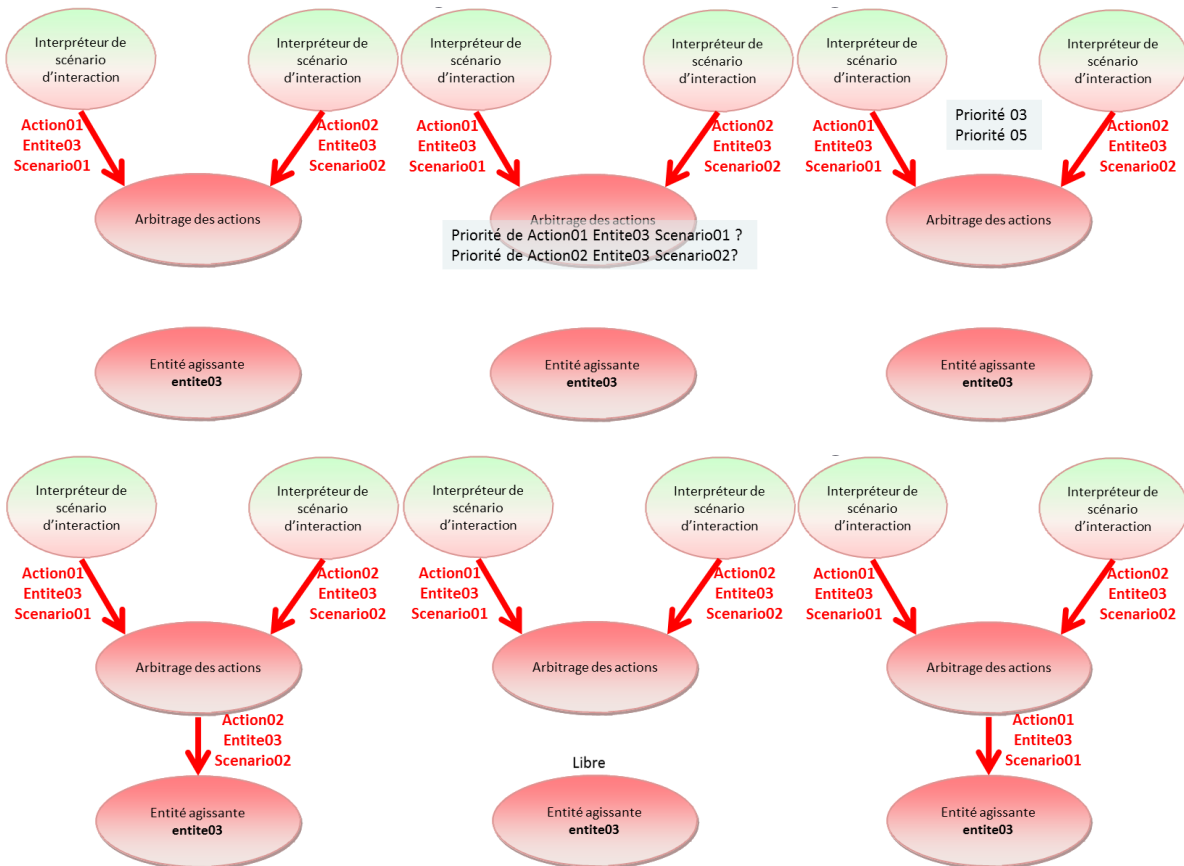


Figure 51 : Exemple de gestion de priorité de scénario

Si la priorité du scénario change pendant l'exécution, chaque EM, qui contient une action du scénario, met à jour sa liste d'attente. Si la qualité d'une entité change, seule la liste de plus haute priorité est triée à nouveau.

Cette structure permet à chaque entité de ne recevoir qu'un ordre à la fois, ainsi, il ne peut pas y avoir de conflit d'actions.

3. Gestion des actions génériques

Dans AmbiProg, les actions génériques sont automatiquement chargées lorsque plusieurs entités ont une action qui porte le même nom. Elles se présentent comme le montre la Figure 52, dans la sous branche **all**.

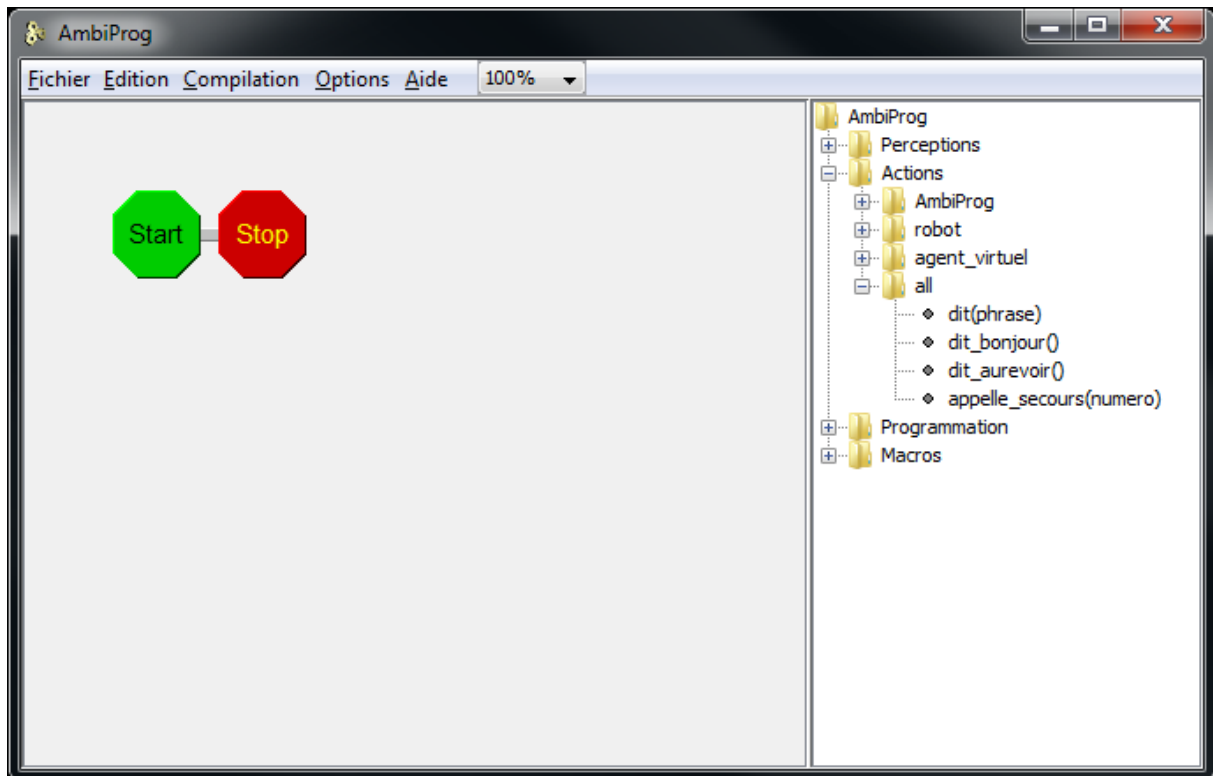


Figure 52 : Actions génériques dans AmbiProg

Lorsqu'il crée son scénario, l'utilisateur peut spécifier si une entité précise doit effectuer l'action (choix de **robot** ou **agent_virtuel**) ou alors si l'action doit être réalisée avec la meilleure entité disponible (choix de **all**).

Les actions génériques sont gérées par le Generic Entity Manager (GEM). La réception d'une action générique enclenche un traitement en 5 étapes :

1. Création d'une liste L des EM dont l'entité associée est capable d'exécuter cette action générique A.
2. Tri de L dans l'ordre descendant des qualités des entités associées.
3. Parcours de L où la disponibilité des entités est vérifiée pour chaque EM. Une entité est libre si elle n'a pas d'action en cours et si elle est connectée.
 - Lorsqu'une entité libre est trouvée, le GEM passe à l'étape 4,
 - Si aucune entité n'est libre, alors après un certain temps, le GEM envoie une réservation à tous les EM. Dès qu'un EM se libère, il vérifie auprès du GEM si l'action

générique est toujours disponible. Si elle est toujours disponible, le GEM passe à l'étape 4. Sinon, si l'action générique a déjà été envoyée, il l'annule auprès de l'EM.

4. Le GEM transmet l'action générique à l'EM sélectionné.
5. L'EM envoie une confirmation au GEM lorsque l'action est envoyée.

Le GEM attend alors la notification de fin de cette action. Ce procédé assure que l'action générique est envoyée à la meilleure entité libre.

La Figure 53 montre un exemple de gestion d'une action générique. L'arbitrage des actions vérifie parmi les entités agissantes libres, celles qui sont capables de réaliser l'action. Il choisit alors celle qui a la meilleure qualité.

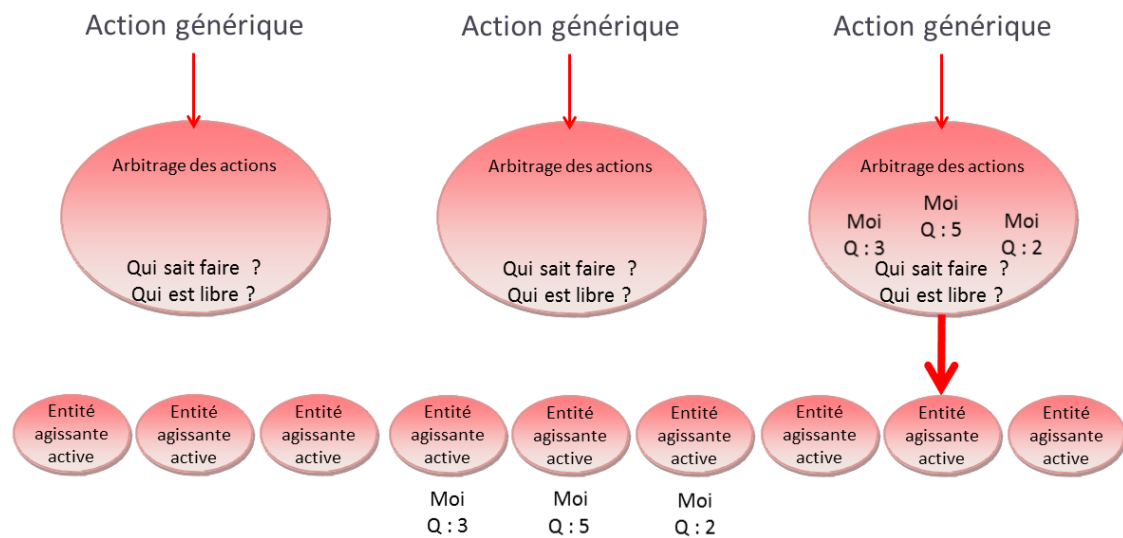


Figure 53 : Exemple de gestion d'une action générique

4. Synchronisation des actions

Les actions à synchroniser (élément `actions_sync` de la grammaire) appartiennent à un groupe spécifique dont le nom est la valeur de l'attribut **id_action** dans le message d'action. Toutes les actions de ce groupe doivent commencer en même temps sur leurs entités agissantes respectives. Ces entités doivent être libérées par leur EM uniquement lorsque toutes les actions sont terminées.

La synchronisation des actions s'effectue en six étapes :

1. L'arbitre du gestionnaire de monde dirige les messages d'actions synchronisées vers le Group Action Manager (GAM).
2. Le GAM transmet les actions à leur EM respectif, en précisant le type de l'action (simple, générique ou synchronisée).
3. Dans le cas des actions synchronisées, un EM ne lance jamais une action de sa propre initiative mais attend l'ordre du GAM.
4. Le GAM ordonne aux EM d'envoyer l'ordre des actions si toutes les actions d'un même groupe sont la première de la liste d'attente et si tous les EM sont libres. Cette condition est vérifiée chaque fois qu'une action devient la première de la liste d'attente et que l'EM associé est libre.
5. Lorsque l'action est terminée, l'EM prévient le GAM mais ne libère pas encore l'entité. En effet, les entités ne doivent être libérées que lorsque toutes les actions sont terminées.
6. Lorsque toutes les actions sont terminées, le GAM demande aux EM de libérer les entités agissantes. Ainsi, toutes les EM utilisées par la synchronisation sont libérées en même temps.

Synthèse

Ce chapitre a présenté l'**architecture ArCo** qui sert à coordonner un ensemble de **dispositifs numériques** présents dans l'**environnement**. ArCo est constitués de modules qui s'appuient sur l'exécutif **MICE**. Les dispositifs numériques sont gérés par deux types de modules d'ArCo :

- Les **entités agissantes actives** qui sont associées aux dispositifs numériques capables de calculer des données de l'environnement,
- Les **entités agissantes perceptives** qui sont associées aux dispositifs numériques capables d'agir sur l'environnement.

L'architecture ArCo est composée de trois éléments essentiels :

- **AmbiProg** qui permet à des prestataires de programmer des scénarii d'interaction entre un utilisateur et le Compagnon Artificiel,
- **AmbiLive** qui permet l'interprétation de chaque scénario d'interaction,
- **AmbiCop** qui permet de coordonner des actions issues des scénarii que doivent effectuer les entités agissantes actives.

Une **bibliothèque MICE** permettant à des développeurs de créer ses propres modules est disponible à cette adresse : <https://sites.google.com/site/compagnonartificiel/>

Chapitre IV.

StimCards : une application d'ArCo

CHAPITRE IV. STIMCARDS : UNE APPLICATION D'ARCO	105
A. STIMCARDS : UN JEU DANS LE CADRE DE LA STIMULATION COGNITIVE	107
B. STIMCARDS : SON FONCTIONNEMENT	109
1. Présentation	109
2. Modules utilisés	111
C. STIMCARDS : UN EXEMPLE	113

Un des objectifs du projet Robadom (Wu1 et al. 2011) est de diminuer les troubles cognitifs des personnes atteintes d'Alzheimer. En effet, leurs capacités cognitives devraient être améliorées pour assurer un niveau minimum de qualité de vie aussi longtemps que possible, grâce à la stimulation cognitive, l'entraînement cognitif et/ou la réhabilitation cognitive (Clare et al. 2003). L'informatique peut aider à créer des systèmes pour pratiquer ces activités de façon complètement automatique. Les personnes pourraient alors travailler de façon indépendante.

Ce chapitre présente notre contribution pour le projet Robadom : un jeu de cartes de stimulation cognitive : StimCards. C'est un jeu qui permet une participation active de l'utilisateur et qui peut être complètement configuré par les médecins, thérapeutes, psychologues... Des expérimentations auprès d'enfants et de personnes âgées ont montré que StimCards est apprécié (cf. Chapitre V.C). D'autres expérimentations, réalisées avec des enfants, ont permis d'évaluer l'importance d'un robot en tant que compagnon de jeu (cf. Chapitre V.D).

Le Chapitre IV.A explique l'intérêt de créer un jeu pour aider à la stimulation cognitive. Il explique les bénéfices apportés par les jeux et le besoin d'un jeu configurable. Le Chapitre IV.B présente le jeu qui est une instance de l'architecture ArCo. Le Chapitre IV.B.2 montre les modules MICE qui sont utilisés pour StimCards et le Chapitre IV.C présente un scénario d'interaction possible.

L'annexe I montre comment il est possible de configurer le jeu.

A. StimCards : un jeu dans le cadre de la stimulation cognitive

La création d'un système informatique qui interagit avec un utilisateur se heurte à un problème de taille : l'acceptabilité (cf. Annexe A pour une introduction au problème). De plus, une étude de Lazar (Lazar 2007) montre que la passivité peut provoquer un sentiment de lassitude et une perte d'attention. Pour éviter la passivité des personnes, le système informatique devrait prendre en compte l'humain qui devrait participer activement à la tâche et obtenir un retour positif immédiat concernant sa participation.

L'acceptabilité est difficile à mesurer. On sait déjà qu'il n'y a pas de corrélation entre les capacités sociales et l'acceptation des technologies (Heerink et al. 2006). Il est très difficile de savoir ce dont les personnes âgées ont besoin et ce qu'elles veulent. D'après Vanden *et al* (Vanden Abeele and Van Rompaey 2006), les personnes âgées rejettent les ordinateurs car ils ne doivent pas remplacer les personnes réelles. Et pourtant, si ces préjugés sont dépassés, l'ordinateur n'est pas un obstacle. Une étude sur des personnes ayant Alzheimer a indiqué que les patients pouvaient apprendre à interagir avec un ordinateur (Berenbaum, Lange, and Abramowitz 2011). Ils en tirent même un bénéfice lorsque cet ordinateur leur permet, non seulement d'effectuer des exercices de stimulation cognitive, mais encore de communiquer avec leur famille et médecins. C'est nécessaire d'après Vanden *et al* (Vanden Abeele and Van Rompaey 2006) qui explique que les personnes âgées ont besoin de se sentir utile, de se cultiver et de rester connectées à la société. La solitude est la pire des situations pour elles. Il est donc nécessaire de créer un système qui respecte ces trois conditions de bien-être : se sentir utile, se cultiver et rester connecté. L'idée des jeux de stimulation cognitive semble être une bonne piste. En effet, les jeux apportent des bénéfices cognitifs pendant que les utilisateurs se divertissent (Brandão et al. 2010) et sont souvent utilisés dans l'éducation car ils représentent un processus d'apprentissage intéressant (Wei et al. 2010). Les jeux qui allient exercices d'entraînement et technologies génèrent plus de plaisir que des jeux sur papiers (Imbeault, Bouchard, and Bouzouane 2011) et donc favorisent l'amélioration de l'état cognitif des personnes. Un jeu efficace devrait suivre les considérations des NUI (Natural User Interface, cf. Chapitre II.A). La question est de savoir comment créer un tel jeu de façon à maximiser son impact (Pesty and Duhaut 2011).

La littérature fournit un ensemble riche de jeux utilisant des technologies variées et appliqués à des domaines variés. La majorité des jeux utilisent les ordinateurs et sont loin des interfaces du futur promis par le domaine des NUI. Par exemple, on trouve une application de jeux de stimulation cognitive nommée SAVION (Berenbaum, Lange, and Abramowitz 2011), une application de gestion de budgets (Lopez-Martinez et al. 2011) et une application web pour les enfants (Hussaan, Sehaba, and Mille 2011). Tous les trois sont utilisés pour aider les personnes ayant des troubles cognitifs. Il existe également un jeu, du domaine des serious game, pour stimuler les personnes atteintes d'Alzheimer en leur présentant des situations de la vie quotidienne (Imbeault, Bouchard, and Bouzouane 2011). La personne reste dans un environnement connu, ce qui est important pour ce type de troubles cognitifs. Ces jeux ont été testés sur les personnes cibles ou par simulation et obtiennent de bons résultats. Mais, ce sont des applications classiques qui risquent d'être rapidement limitées. Pour essayer d'intéresser d'avantage les utilisateurs, certains jeux introduisent un personnage virtuel, dans un monde 3D, se rapprochant du domaine des jeux vidéo. Certains sont très généralistes et dédiés à toutes les personnes en difficultés (Abreu et al. 2011). L'avantage est de laisser le patient évoluer seul pendant que le thérapeute le surveille à distance. D'autres applications, comme Jecripe (Brandão et al. 2010) sont spécialisées pour les enfants ayant un syndrome de Down (trisomie 21). Même si Jecripe se distingue en ne laissant pas l'utilisateur passif devant l'ordinateur, en l'invitant à imiter le personnage, aucune interaction réelle n'est faite avec l'utilisateur et le jeu ne contrôle pas les actions faites par l'utilisateur et ne lui donne donc aucun retour alors que « l'humain devrait être le cœur du système » (Tedre 2008). Au contraire, certains jeux sont trop invasifs pour les personnes en les obligeant à porter des capteurs sur elles pour pouvoir surveiller leur constantes comme ZPLAY

(Makedon et al. 2010). Ces technologies semblent trop invasives et devraient être transparentes (Ndiwalana et al. 2003) pour l'utilisateur.

Dans le domaine des NUI, il existe un certains nombres de jeux qui tentent de résoudre les problèmes des jeux suscités. On trouve des jeux de stimulation cognitive dans le domaine de la réalité augmentée (Zapirain, Zorrilla, and Larra 2010), dans le domaine des jeux vidéo avec la console Wii, d'une part (Morelli et al. 2010) ou avec un vélo d'appartement relié à un écran, d'autre part (Chilukoti et al. 2007). On trouve un jeu urbain de construction d'histoire, basé sur une infrastructure ubiquitaire permettant de faire jouer des centaines de joueurs en même temps (Scheible, Tuulos, and Ojala 2007). Des iPod Touch ont été utilisés pour le traitement de l'amblyopie (Long To et al. 2011). Et enfin, des robots également proposent des jeux. Une étude (Tapus, Tapus, and Mataric 2009) a même montré que les participants préfèrent largement le robot pour jouer plutôt que sa version sur ordinateur.

Le Tableau 7 montre la liste des jeux cités en précisant leur domaine cible et les technologies utilisées. Tous ces jeux sont spécialisés et ne peuvent pas être adaptés à un autre domaine. Cinq des six jeux les plus récents utilisent des interfaces graphiques traditionnelles alors que la communauté de l'Interaction Homme-Machine a montré qu'ils devaient utiliser des NUI (Goth 2011) (Jain, Lund, and Wixon 2011). Peu de jeux sont innovants du point de vue de leur technologie : périphérique mobile, vélo augmenté, réalité augmentée, console de jeux. Mais le problème de ces jeux est qu'il n'est pas possible de les adapter à un autre contexte.

Tableau 7 : Les jeux cognitifs actuels

Noms	Année	Domaine cible	Technologies
Story Mahsup (Scheible, Tuulos, and Ojala 2007)	2007	Tout le monde	Mobiles + PC
Pas de nom (Chilukoti et al. 2007)	2007	Alzheimer	Vélo augmenté + PC
Pas de nom (Tapus, Tapus, and Mataric 2009)	2009	Troubles cognitifs	Robot
Jecripe (Brandão et al. 2010)	2010	Syndrome de Down	Ordinateur
ZPLAY (Makedon et al. 2010)	2010	Alzheimer	Ordinateur + capteurs sensoriels
Pas de nom (Zapirain, Zorrilla, and Larra 2010)	2010	Personnes âgées	Réalité augmentée + tangram
VI-Tennis (Morelli et al. 2010)	2010	Personnes aveugles	Son + vibration (Wii)
Pas de nom (Abreu et al. 2011)	2011	Personnes en difficulté	Ordinateur + personnage 3D
SAVION (Berenbaum, Lange, and Abramowitz 2011)	2011	Démences	3D
Pas de nom (Imbeault, Bouchard, and Bouzouane 2011)	2011	Alzheimer	Ordinateur
Pas de nom (Hussaan, Sehaba, and Mille 2011)	2011	Difficultés cognitives	Ordinateur
Pas de nom (Long To et al. 2011)	2011	Amblyopie	iPod Touch
Pas de nom (Lopez-Martinez et al. 2011)	2011	Troubles cognitifs	Ordinateur

StimCards est un jeu du domaine des NUI, permettant de faire de la stimulation cognitive, tout public et entièrement configurable pour essayer d'apporter une solution à ce problème de jeu non adaptable.

B. StimCards : son fonctionnement

1. Présentation

StimCards s'utilise de la façon suivante :

1. Un utilisateur dispose d'un paquet de cartes de jeux. Le visuel de chaque carte contient un ou plusieurs codes-barres appelés « code QR ». Lorsque le joueur présente une carte à la caméra, celle-ci détecte le code QR et décode son contenu. Son contenu est une fiche XML qui contient des informations sur l'exercice (type d'exercice, questions, proposition de réponses, indices...). Les exercices peuvent être des questions ouvertes, à choix multiples etc.
2. Si l'interface de StimCards est active, elle affiche la question et des propositions de réponses s'il y en a. En l'absence d'interface, un robot ou un personnage virtuel peut énoncer la question. (Cette possibilité est laissée libre à l'utilisateur qui configure lui-même son scénario de jeu).
3. L'utilisateur saisit sa réponse avec le dispositif actif (cela peut être une tablette tactile, une manette de jeu, une boîte de réponse...).
4. La réponse est analysée et l'utilisateur configure le comportement de StimCards en cas de bonne ou de mauvaise réponse.

La Figure 54 montre deux exemples de StimCards. Sur l'exemple de gauche, StimCards est accompagné de deux robots : un bioloïd (robot humanoïde à gauche) et un Nabaztag (robot lapin à droite) et l'utilisateur saisit sa réponse avec le clavier d'un ordinateur. Sur l'exemple de droite, StimCards est accompagné d'un bioloïd pour communiquer avec l'utilisateur et d'une tablette tactile pour la saisie de la réponse.

StimCards est développé avec l'architecture ArCo, ce qui signifie qu'il est possible d'intégrer des robots, des agents virtuels, des tablettes tactiles, des manettes de jeux... Il est possible de brancher tout type de dispositifs numériques d'entrées et de sorties. La Figure 54 n'est qu'un exemple d'implémentation de StimCards.

StimCards a été évalué afin d'étudier son acceptabilité auprès d'enfants et de personnages âgées (cf. Chapitre V.C et Chapitre V.D).

Il est possible de créer beaucoup de scénarii d'interaction avec AmbiProg pour diversifier le jeu.



Figure 54 : Exemple d'utilisation de StimCards

Les questions posées lors d'exercices de stimulation cognitive sont variées. Il peut y avoir des questions ouvertes, à choix multiples, des classements à faire, des mots à relier etc. StimCards est prévu pour adapter son affichage en fonction du type de la question posée. La Figure 55 montre trois exemples. L'écran de gauche montre l'affichage d'une carte de type « question à choix multiple ». Les deux autres exemples montrent une question de type « stroop ».



Figure 55 : Affichage de StimCards en fonction du type de la question

L'interface de StimCards est composée de :

- Un menu qui permet de charger une carte à la main si l'utilisateur joue seul sur son ordinateur,
- Une zone de titre avec affichage d'une image et d'un titre,
- Une zone d'affichage de cartes. Par exemple, pour la Figure 56, la question, les réponses proposées et une zone d'aide,
- Une zone de saisie des réponses si l'utilisateur joue seul sur son ordinateur.

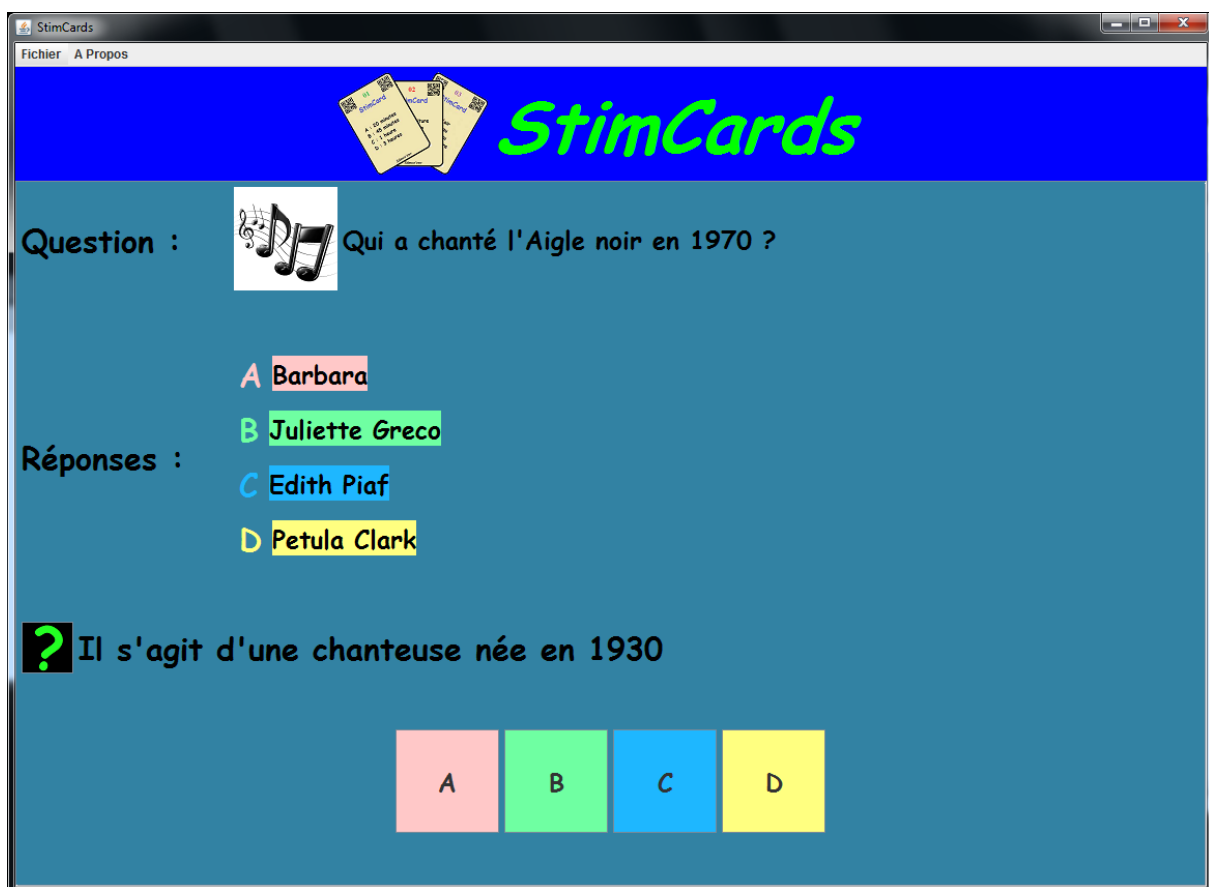


Figure 56 : StimCards affichant une question à choix multiples

2. Modules utilisés

Comme l'explique le Chapitre IV.B, StimCards est un jeu modulable développé avec l'architecture ArCo. Cela signifie qu'un utilisateur peut y ajouter des entités agissantes perceptives et actives. Dans sa forme la plus simple, StimCards est composé de quatre modules :

- *reader* : module chargé de décoder le code QR d'une carte de jeu,
- *organizer* : module chargé d'indiquer la fiche XML de la question,
- *stimcoquestions* : le moteur de jeu,
- *stimcoenv* : l'interface de StimCards.

La Figure 57 montre ces quatre modules dans une configuration de jeu un peu plus complète.

a) Module *reader*

Le module **reader** utilise la caméra afin de décoder un code QR. Dès qu'un code QR est détecté, sa valeur **v** est envoyée par une **perception** `P(reader, stimcoquestion_id, v, 100)`.

b) Module *organizer*

Le module **organizer** reçoit la perception `stimcoquestion_id`. Ce module est associé à un fichier XML qui fait le lien entre les valeurs connues des codes QR et des noms de fichier. Ces noms de fichier représentent les fiches XML des questions de jeu. Ainsi, le module cherche quel est le fichier associé à la valeur de la perception reçue. Le nom **n** du fichier est envoyé par une **perception** `P(organizer, stimcoquestion_path, n, 100)`.

Dans la version actuelle de ce module, la fiche XML de la question n'est pas encodée dans le code QR car les bibliothèques gratuites permettant de coder et décoder ne sont pas aussi efficaces que les produits commerciaux. La solution consistant à utiliser les modules reader et organizer est une version temporaire et doit impérativement être optimisée. La fiche XML de la question doit être contenue dans le code QR pour permettre aux cartes de jeu une réelle autonomie.

c) Module *stimcoquestions*

Le module **stimcoquestions** est le moteur de jeu. Il gère le chargement des questions, l'évaluation des propositions de réponses, l'accès à la question, à la vraie réponse etc. Il est prévu pour recevoir la **perception** `stimcoquestion_path`. A la réception de cette perception, le module effectue deux actions :

1. Il accède à la fiche XML et charge la question en mémoire.
2. Si l'accès au fichier s'est bien déroulée, il envoie la **perception** `P(stimcoquestions, card_scanned, TRUE, 100)`, qui indique qu'une carte de jeu a été chargée.

Le module peut effectuer les actions suivantes :

- `read_question()` : cette action permet de lire la question,
- `read_suggested_answers(texte)` : cette action permet de donner la liste des réponses proposées après avoir dit un texte passé en paramètre,
- `read_true_answers(texte)` : cette action permet de donner la bonne réponse après avoir dit un texte passé en paramètre,
- `read_clue()` : permet de lire l'indice en cours. A chaque fois que l'action est exécutée, l'indice suivant est lu,

- `read(texte)` : cette action permet de lire un texte passé en paramètre,
- `raz(nom module, nom perception)` : cette action permet de « remettre à zéro » la perception passée en paramètre. Cela permet de se mettre en attente d'une carte scannée par exemple. Puisque la perception vaut `FALSE`, alors la carte n'a pas été scannée et l'attente peut commencer. Lorsque la `perception card_scanned` est envoyée, la perception vaut `TRUE` et l'attente de carte scannée s'arrête.

Le module de gestion de question peut envoyer, en plus de `card_scanned`, les `perceptions` suivantes :

- `P(stimcoquestions, good_answer, TRUE, 100)` : cette perception est envoyée lorsque le module a reçu une perception `stimco_answer` et que la réponse envoyée est une bonne réponse,
- `P(stimcoquestions, wrong_answer, TRUE, 100)` : cette perception est envoyée lorsque le module a reçu une perception `stimco_answer` et que la réponse envoyée est une mauvaise réponse.

d) Module *stimcoenv*

Ce module est l'interface de StimCards. Il est prévu pour recevoir la `perception stimcoquestion_path`. A la réception de cette `perception`, il charge la question en mémoire.

Le module peut effectuer les actions suivantes :

- `display_question()` : cette action permet d'afficher la question dans l'interface de StimCards,
- `efface_questions()` : cette action permet d'effacer la question actuellement affichée dans l'interface,
- `display_clue()` : cette action permet d'afficher un indice dans l'interface.

Ce module peut envoyer deux perceptions :

- `P(stimcoenv, stimco_answer, ..., 100)` : perception envoyée lorsque l'utilisateur clique sur un des boutons de l'interface. La perception est envoyée au module `stimcoquestions` qui vérifie ensuite si la réponse est bonne ou non,
- `P(stimcoenv, stimco_clue, ..., 100)` : perception envoyée lorsque l'utilisateur clique sur le bouton d'aide de l'interface. Cela permet d'écrire avec AmbiProg le scénario : « Si l'utilisateur a cliqué sur le bouton d'indice, alors le module `stimcoquestions` lit l'indice pendant que le module `stimcoenv` affiche l'indice.

e) Module de réponse

Un module gérant les réponses des utilisateurs peut être ajouté à StimCards. Ce module doit envoyer les `perceptions stimco_answer` et `stimco_clue`, qui peuvent être reçues par les modules `stimcoquestions` et `stimcoenv`.

C. StimCards : un exemple

Le scénario présenté ici est celui qui a été utilisé dans le cadre d'une expérimentation de StimCards (cf. Chapitre V.D) avec des personnes âgées (à l'hôpital Broca). Les personnes âgées utilisaient un boîtier de réponse pour interagir avec leur interlocuteur, l'Agent Conversationnel Animé : Greta. Le boîtier de réponse contient cinq boutons de couleurs qui envoient chacun une perception différente.

Le scénario se déroulait de la façon suivante :

Greta :	« Bonjour. Appuyez sur le bouton blanc du boîtier de réponse lorsque vous êtes prêt. »
<i>Ce message est à destination du thérapeute de la personne âgée. Le système attend que le thérapeute appuie sur le bouton blanc.</i>	
Greta :	« Bonjour. Nous allons faire une série de 10 questions ensemble. Vous utiliserez le boîtier de réponse noir pour poser votre carte et saisir votre réponse. Vous utiliserez aussi la caméra pour scanner la carte. L'exercice commence maintenant.»
<i>La suite du scénario est effectuée 10 fois</i>	
Greta :	« Vous pouvez montrer une carte à la caméra. »
<i>La personne âgée présente une carte à la caméra.</i>	
Greta :	« Une carte a été scannée. Posez la carte sur le boîtier de réponse et appuyez sur le bouton blanc lorsque vous êtes prêt. »
<i>La personne âgée appuie sur le bouton blanc. Greta lit la question. Le système attend la réponse de la personne âgée qui doit appuyer sur le bon bouton du boîtier de jeu.</i>	
<i>Si la réponse est bonne :</i>	
Greta :	« Bravo ! C'est la bonne réponse ! »
<i>Si la réponse est mauvaise :</i>	
Greta :	« Désolée ce n'est pas la bonne réponse. La bonne réponse est : » puis <i>Greta énonce la bonne réponse.</i>
<i>Une fois que la personne âgée a répondu à 10 questions, Greta annonce la fin du jeu.</i>	
Greta :	« Les exercices sont terminés. Merci d'avoir participé. »

Le scénario du jeu a été créé avec AmbiProg.

Les modules disponibles étaient ceux de la Figure 57. Les modules **reader**, **organizer**, **stimcoenv** et **stimcoquestions** ont été présentés dans le Chapitre IV.B.2.

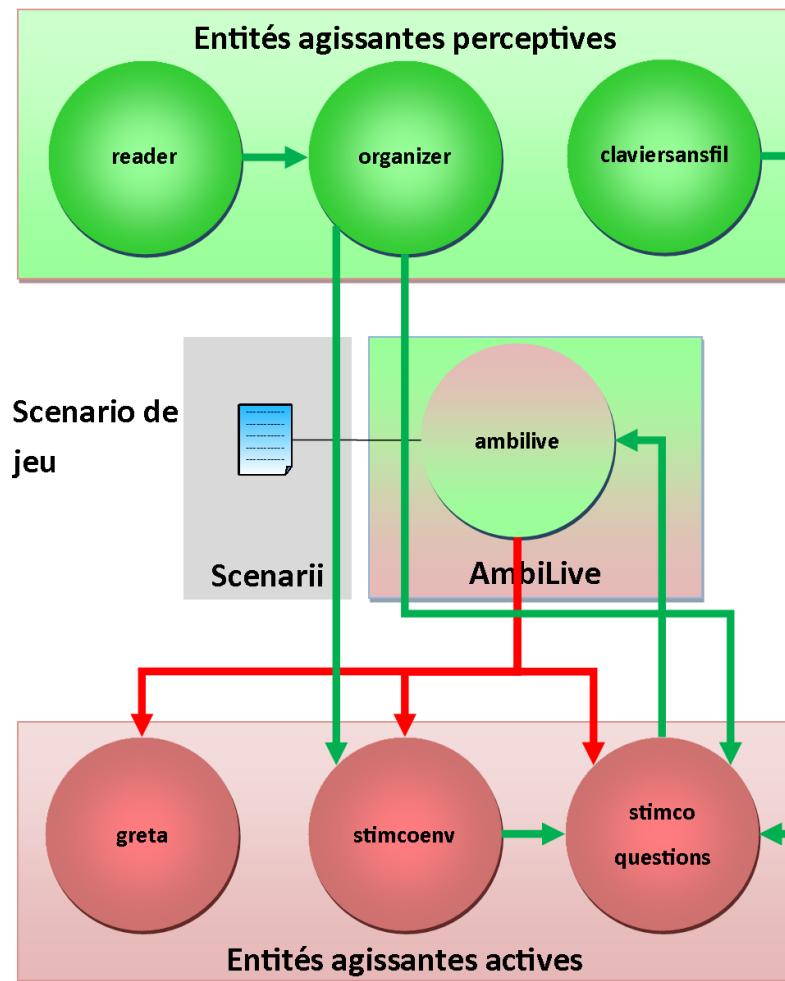


Figure 57 : Exemple de chemin logique de StimCards

Le module **claviersansfil** est celui qui a été fourni aux participants pour interagir avec le système. Il peut envoyer les perceptions suivantes :

- **stimcoanswer** : cette perception était envoyée lorsque l'utilisateur appuyait sur un bouton de réponse,
- **stimcovalidate** : cette perception était envoyée lorsque l'utilisateur appuyait sur un bouton spécial du clavier indiquant qu'il était prêt.

Le module **greta** permet de contrôler l'agent conversationnel animé Greta. Il peut effectuer un ensemble d'actions, représentés par des mouvements non verbaux, qui ne sont pas décrits dans ce document.

Le scénario utilisé pour ce jeu est représenté de la Figure 58 à la Figure 65. Le scénario commence (Figure 58) par mettre à zéro les perceptions **card_scanned**, **good_answer** et **wrong_answer** du module **stimcoquestions** et la perception **stimcovalidate** du module **claviersansfil**. Il envoie l'ordre à Greta d'effectuer une action *agreement*. Ce mouvement fait hocher

la tête de Greta. Il a été utilisé pour initialiser la connexion avec le personnage virtuel. Le scénario dédie une demi-seconde à l'initialisation du système puis passe à la suite.

Dans la suite, le module **stimcoquestions** lit la phrase « Bonjour. Appuyez sur le bouton blanc du boîtier de réponse lorsque vous êtes prêt. ». Le « bouton blanc » représente celui qui envoie la **perception stimcovalide**. Pendant ce temps, Greta effectue l'action **bonjour01**.

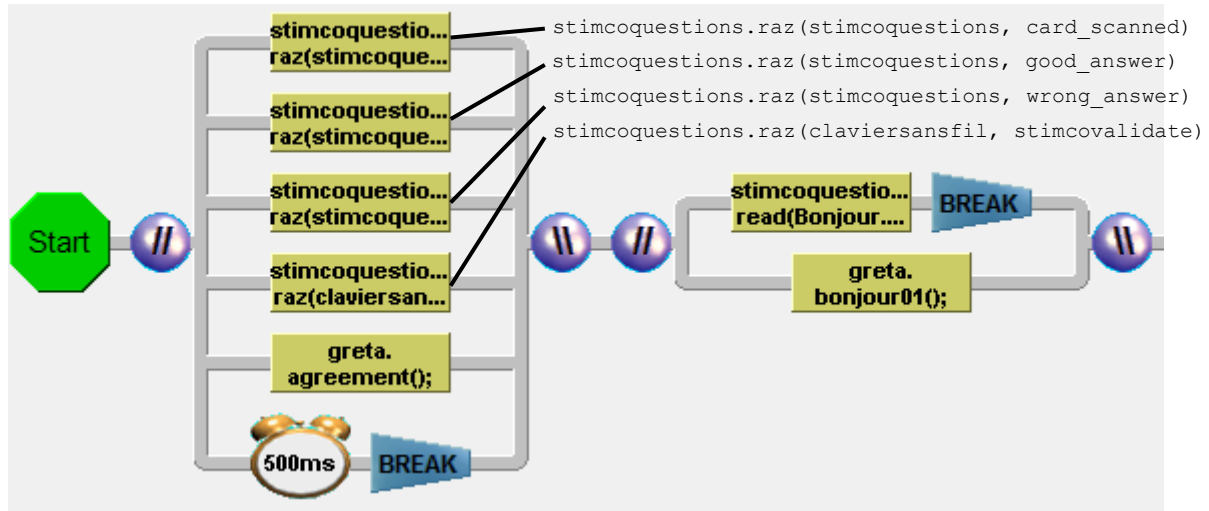


Figure 58 : Scénario de StimCards - partie 01

Dans la suite du scénario, représenté par la Figure 59, l'interpréteur est en attente de la **perception stimcovalide** valant VRAI (appui sur le bouton blanc). Lorsque la perception est reçue, la première action effectuée est la remise à zéro de la **perception stimcovalide**. Ensuite, le module **stimcoquestions** lit la phrase « Bonjour. Nous allons faire une série de 10 questions ensemble. ». Pendant ce temps, Greta effectue l'action **bonjour02**. Le « Break » permet de passer à la suite, même si Greta n'a pas fini de réaliser son action.

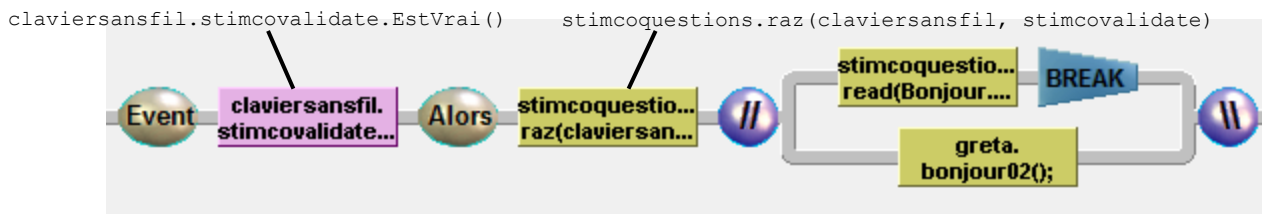


Figure 59 : Scénario de StimCards - partie 02

Ensuite (Figure 60), le module **stimcoquestions** lit la phrase « Vous utiliserez le boîtier de réponse noir pour poser votre carte et saisir votre réponse. » pendant que Greta effectue l'action **indication01**. Puis le module **stimcoquestions** lit la phrase « Vous utiliserez aussi la caméra pour scanner la carte. » pendant que Greta effectue l'action **indication02**.

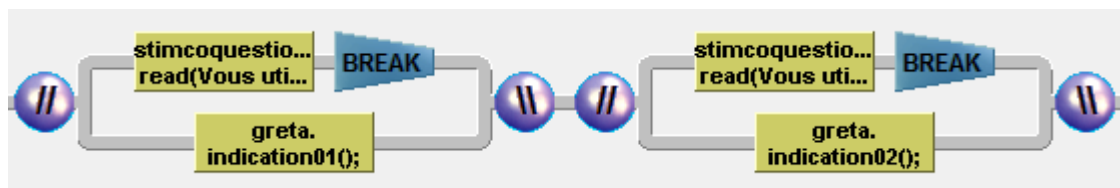


Figure 60 : Scénario de StimCards - partie 03

Enfin (Figure 61), le module **stimcoquestions** lit la phrase « L'exercice commence maintenant. » pendant que Greta effectue l'indication03. Ceci termine la présentation du jeu à l'utilisateur. La suite

du scénario est composé d'une boucle de 10 itérations. C'est le jeu qui commence. Il a été prévu une série de 10 questions. La série commence par la lecture de la phrase « Vous pouvez montrer une carte à la caméra. » par le module **stimcoquestions** pendant que Greta effectue l'action indication04.

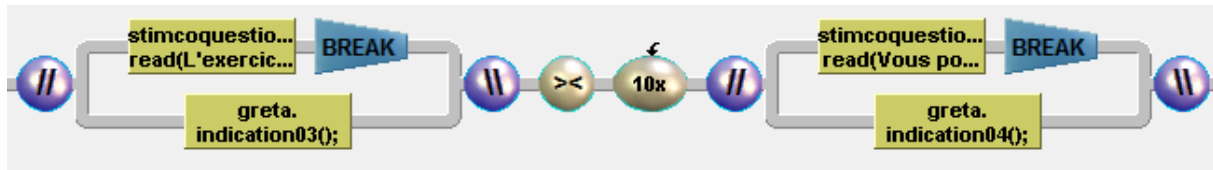


Figure 61 : Scénario de StimCards - partie 04

La suite du scénario, représenté par la Figure 62, est une attente de la **perception card_scanned**. Lorsque l'utilisateur a montré une carte à la caméra et que la **perception card_scanned** a été envoyée alors le module lit la phrase « Une carte a été scannée. Posez la carte sur le boîtier de réponse et appuyez sur le bouton blanc lorsque vous êtes prêt. » pendant que Greta effectue l'action indication05. Ensuite le module **stimcoquestions** remet à zéro la **perception card_scanned**.

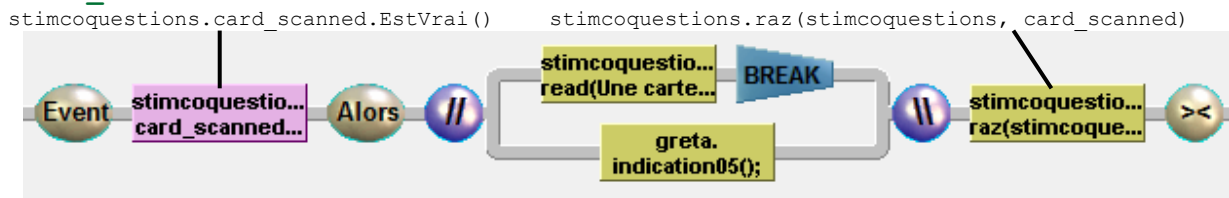


Figure 62 : Scénario de StimCards - partie 05

La Figure 63 montre la suite du scénario où l'interpréteur attend la réception de la **perception stimcovalidate** qui indique que l'utilisateur a posé la carte sur le boîtier de réponse et est prêt pour la suite. Lorsque la perception est reçue, le module **stimcoquestions** remet à zéro la **perception stimcovalidate** et lit la question pendant que Greta effectue l'action **pose_Question**.

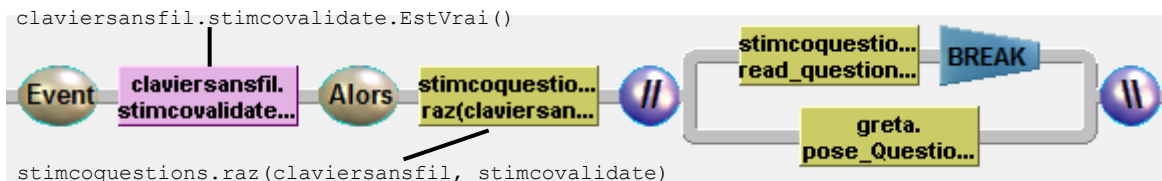


Figure 63 : Scénario de StimCards - partie 06

La suite du scénario (voir Figure 64) est une attente de perceptions en parallèle. L'interpréteur attend la réception de la perception **good_answer** ou **wrong_answer**. Dans les deux cas, la valeur de la perception reçue est remise à zéro. Si l'utilisateur a donné la bonne réponse, alors le module **stimcoquestions** lit la phrase « Bravo ! C'est la bonne réponse ! » pendant que Greta effectue l'action **Bravo**. Si l'utilisateur a donné la mauvaise réponse, le module **stimcoquestions** lit la phrase « Désolé ce n'est pas une bonne réponse. La bonne réponse est : » puis donne la bonne réponse pendant que Greta effectue l'action **Désolé**. Une fois que la branche a été interprétée, le Break permet de continuer le scénario.

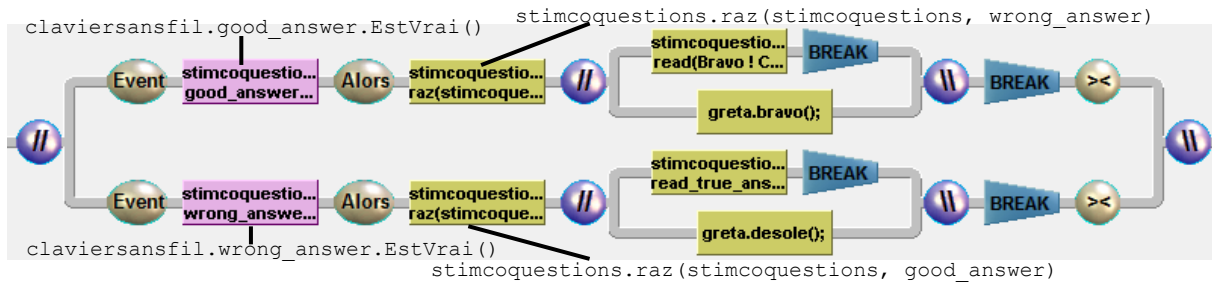


Figure 64 : Scénario de StimCards - partie 07

La Figure 65 montre le scénario exécuté après 10 itérations. Le module **stimcoquestions** lit la phrase « Les exercices sont terminés. Merci d'avoir participé. » pendant que Greta effectue l'action fin. Le scénario se termine après cette action.

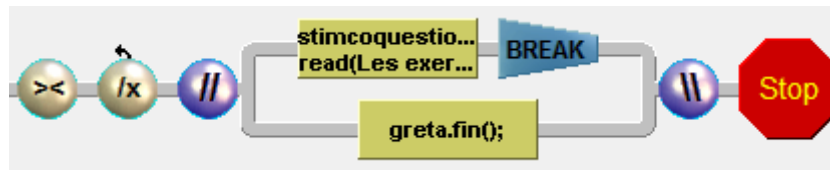


Figure 65 : Scénario de StimCards - partie 08

Synthèse

L'objectif du projet Robadom est d'étudier l'impact d'un robot d'aide à domicile sur l'état psychoaffectif des personnes âgées atteintes de troubles cognitifs légers. Nous avons conçu StimCards, un jeu de stimulation cognitive, implémenté avec notre architecture ArCo, permettant au robot d'être une des entités agissantes de l'architecture. Sur le même principe que l'exemple présenté avec un personnage virtuel, le robot du projet a pu proposer des jeux de stimulation cognitive à des personnes âgées souffrant de troubles cognitifs légers.

Nous avons pu mettre en place et expérimenter StimCards qui est une réponse au problème posé par le projet Robadom par l'utilisation d'ArCo. Les prestataires de l'hôpital Broca ont pu configurer et utiliser StimCards avec succès que ce soit avec le robot ou avec un personnage virtuel.

Chapitre V.

Différentes expérimentations interactives menées
avec ArCo

CHAPITRE V.	DIFFERENTES EXPERIMENTATIONS INTERACTIVES MENEES AVEC ARCO	119
A.	ETUDE SUR LA GESTUELLE D'UN ROBOT	122
B.	EVALUATION D'AMBIPROG	124
C.	EVALUATION DE STIMCARDS (ENFANTS ET PERSONNES AGEES)	126
D.	EVALUATION DU MEILLEUR PARTENAIRE NUMERIQUE POUR STIMCARDS	128
E.	EVALUATION D'AMBIPOP	130

Ce chapitre présente le résumé de sept expérimentations réalisées grâce à l'architecture ArCo dans un environnement spécifique. Le détail des évaluations se trouve en annexe.

Deux premières études se sont intéressées aux conditions d'acceptation d'un robot par l'homme. Pour apporter un élément de réponse, deux critères ont été évalués : la crédibilité et la sincérité d'un robot en fonction d'une gestuelle adaptée ou non au discours (Annexe : J)

Deux autres études ont permis de valider l'interface de programmation visuelle AmbiProg. La première étude évaluait si l'organisation graphique des éléments de l'interface favorisait une prise en main rapide de l'application. La deuxième étude évaluait si le langage était suffisamment simple et accessible pour être utilisée, sans difficulté, par des non-experts en informatique (Annexe : K).

Deux autres études ont permis d'évaluer StimCards, l'application présentée dans le chapitre IV, qui a été proposée en tant qu'outil de stimulation cognitive. Ces deux études ont été réalisées de façon similaire avec des enfants et des personnes âgées. L'objectif de cette étude était de vérifier si le jeu était à la fois simple à utiliser, facile à comprendre et intéressant pour les joueurs (Annexe : L).

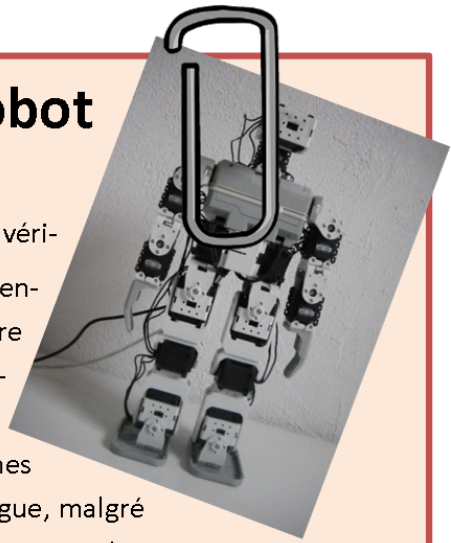
Une autre étude, utilisant StimCards, et réalisée avec des enfants, permettait de tester trois types d'interface jeu-enfant : un ordinateur, un robot et un agent virtuel. Cette étude visait à étudier quel type d'interface était préférée dans le cadre de jeux de stimulation cognitive (Annexe : M).

Pour finir, ArCo a été utilisé lors d'un stand de démonstration afin de tester le fonctionnement d'AmbiCop (Annexe : N).

A. Etude sur la gestuelle d'un robot

Etude sur la gestuelle d'un robot

Objectifs : Cette évaluation avait pour objectif de vérifier si la crédibilité et la sincérité d'un robot peut être dépendante des gestes qui accompagnent sa parole. Une première évaluation a étudié si une gestuelle non adaptée à un discours amène un problème de crédibilité et de sincérité des robots. Une deuxième évaluation a étudié ces mêmes conditions, dans le cadre d'une interaction un peu plus longue, malgré quelques incohérences dans le discours et la gestuelle. Les protocoles expérimentaux utilisés viennent du LIG.



Participants : La première évaluation a été effectuée par 60 personnes (17 femmes et 43 hommes) âgées de 10 à 57 ans. La deuxième évaluation a été effectuée par 26 personnes (9 femmes et 17 hommes) âgées de 18 à 28 ans.

Méthodes : La première évaluation s'est déroulée lors d'un concours de robotique. Le robot était placé sur une table. Les participants devaient regarder le robot faire une série de gestes accompagnant une phrase et répondre à un questionnaire. La deuxième évaluation s'est déroulée dans une pièce isolée. Chaque participant était isolé avec le robot, devait observer une série de mouvement et remplir un questionnaire. Les questionnaires mesuraient si le robot semblait crédible et sincère.

Résultats : Lors de la première évaluation, le robot a été jugé crédible (81,10%) et sincère (88,35%) lorsque le discours étaient en adéquation avec la gestuelle. Lors de la deuxième évaluation, le robot a été jugé crédible et sincère malgré quelques incohérences.

Conclusions principales : Chaque geste doit être en adéquation avec le discours pour que le robot semble sincère et crédible. Mais, dans le cas d'une interaction un peu plus longue, quelques erreurs n'altèrent pas la crédibilité et la sincérité du robot. Ces expérimentations nous ont permis de confirmer les résultats obtenus par le LIG sur des personnages virtuels.

Détail de l'expérimentation : annexe J

La première évaluation sur la gestuelle du robot ne contenait pas beaucoup de modules car le robot effectuait des mouvements sans aucune interaction. Une interface graphique, pilotée par le module **adminframe**, permettait à l'expérimentateur d'envoyer l'ordre au module **bioloid** d'effectuer des mouvements. Les mouvements avaient été programmés en avance puisque l'évaluation consistait à comparer trois mouvements de base. Le module **bioloid** faisait appel au module nommé **hautparleur**, qui utilisait un système de synthèse vocale pour faire dire au robot des phrases prédéfinies à l'avance. L'ensemble de ces modules est présenté sur la Figure 66.

Le fichier de description du chemin logique contenait les informations suivantes :

```
<connect><in>adminframe</in><out>bioloid</out></connect>
<connect><in>bioloid</in><out>hautparleur</out></connect>
```

La deuxième évaluation de la gestuelle du robot contenait le module d'interprétation de scénarii : **ambilive**. Un nouveau module **adminframe** envoyait des perceptions au module **ambilive**. Ce dernier envoyait des actions au module **bioloid** et au module **hautparleur**. Cette évaluation contenait peu de modules car c'est l'expérimentateur qui la pilotait entièrement, en magicien d'Oz. Chaque participant pouvait observer le robot et l'expérimentateur dialoguer ensemble. En réalité, l'expérimentateur cliquait sur un bouton de l'interface graphique du module **adminframe** pour envoyer une perception, qui déclenchait le comportement du robot via l'interprétation du scénario d'interaction. Ces modules sont présentés sur Figure 67.

Le fichier de description du chemin logique contenait les informations suivantes :

```
<connect><in>adminframe</in><out>ambilive</out></connect>
<connect><in>ambilive</in><out>bioloid</out></connect>
<connect><in>ambilive</in><out>hautparleur</out></connect>
```

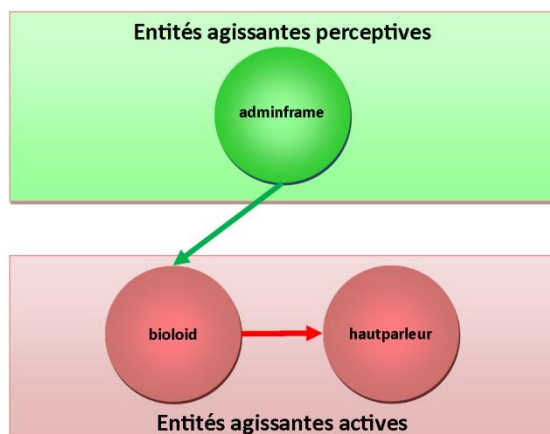


Figure 66 : Modules de la première évaluation de la gestuelle du robot

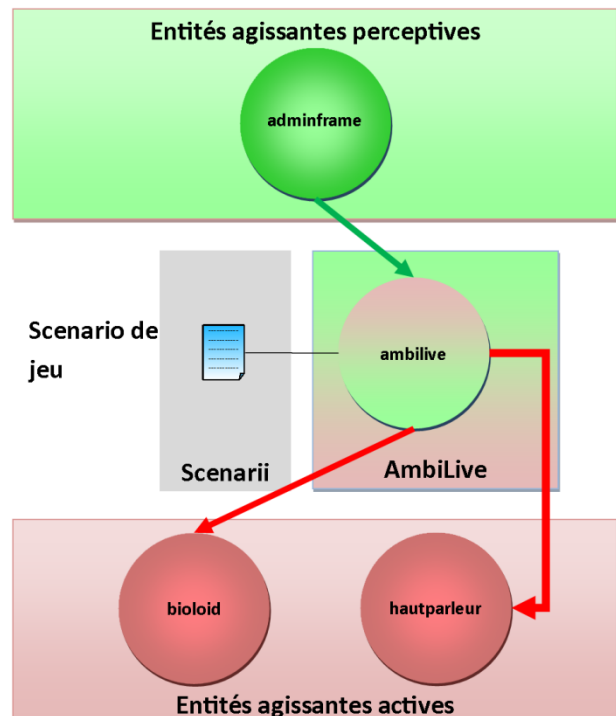
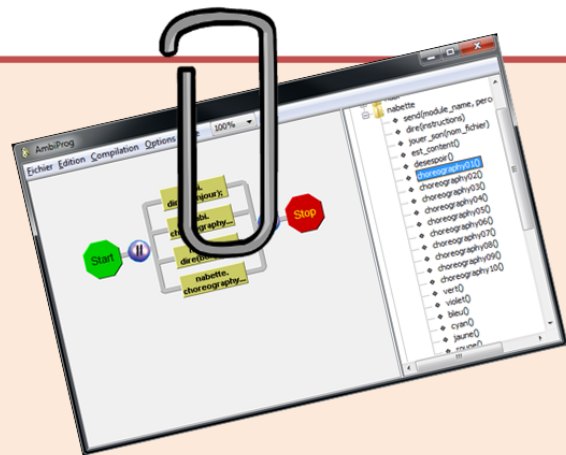


Figure 67 : Modules de la deuxième évaluation de la gestuelle du robot

B. Evaluation d'AmbiProg

Evaluation d'AmbiProg



Objectifs : Cette évaluation avait pour objectif de tester l'interface de programmation AmbiProg pour vérifier si les non experts en informatique sont capables de créer des scénarios. Deux évaluations ont été réalisées. La première permettait de vérifier si l'organisation visuelle d'AmbiProg était facile à comprendre ou si des compétences en informatique étaient nécessaires pour créer des scénarios. La deuxième étudiait la capacité des enfants à réaliser des scénarios pour contrôler deux robots.

Participants : La première évaluation a été réalisée avec 14 personnes (5 femmes et 9 hommes) âgés de 23 à 58 ans. La deuxième évaluation a été réalisée avec 16 enfants de CM2 (8 filles et 8 garçons) âgés de 10 à 11 ans, par groupe de deux.

Méthodes : Lors des deux évaluations, les participants et l'expérimentateur étaient isolés dans une pièce spécifique aux évaluations. L'expérimentateur avait le rôle d'enseignant puis d'évaluateur. D'abord, l'expérimentateur expliquait les concepts. Ensuite les participants devaient réaliser des exercices. Pendant la deuxième évaluation, les participants faisaient bouger des robots. Ils pouvaient à tout moment poser des questions à l'expérimentateur. Après l'expérimentation, les participants remplissaient un questionnaire.

Résultats : Les non experts et les experts ont pu réaliser des scénarios. Plus les personnes étaient âgées et plus le temps de réalisation de la tâche était longue. Tout le monde a trouvé qu'AmbiProg était facile à comprendre et à utiliser (Score : 4,56/5). La majorité des enfants ont réussi à effectuer les exercices et à comprendre le fonctionnement du langage de programmation.

Conclusions principales : AmbiProg est perçu positivement par les utilisateurs. Il n'est pas nécessaire d'avoir des compétences en informatique pour créer des scénarios. Les enfants ont eu besoin de 15 minutes pour apprendre chaque élément de scénario.

Détail de l'expérimentation : annexe K

La première évaluation consistait à utiliser uniquement AmbiProg, hors contexte. Aucun module d'ArCo n'a été utilisé.

La deuxième évaluation d'AmbiProg contenait les modules utilisés par les enfants pour réaliser l'expérimentation. Le module **fenetre_boutons** gérait la tablette tactile et les modules **nabi** et **nabette** géraient les deux Nabaztag. Il y avait deux interpréteurs **ambilive1** et **ambilive2**, un pour participant enfant, noté « enfant 1 » et « enfant 2 » sur la Figure 68. Les modules de l'évaluation sont présentés sur la Figure 68.

Le fichier de description du chemin logique contenait les informations suivantes :

```
<connect><in>fenetre_boutons</in><out>ambilive1</out></connect>
<connect><in>fenetre_boutons</in><out>ambilive2</out></connect>
<connect><in>ambilive1</in><out>nabi</out></connect>
<connect><in>ambilive1</in><out>nabette</out></connect>
<connect><in>ambilive2</in><out>nabi</out></connect>
<connect><in>ambilive2</in><out>nabette</out></connect>
```

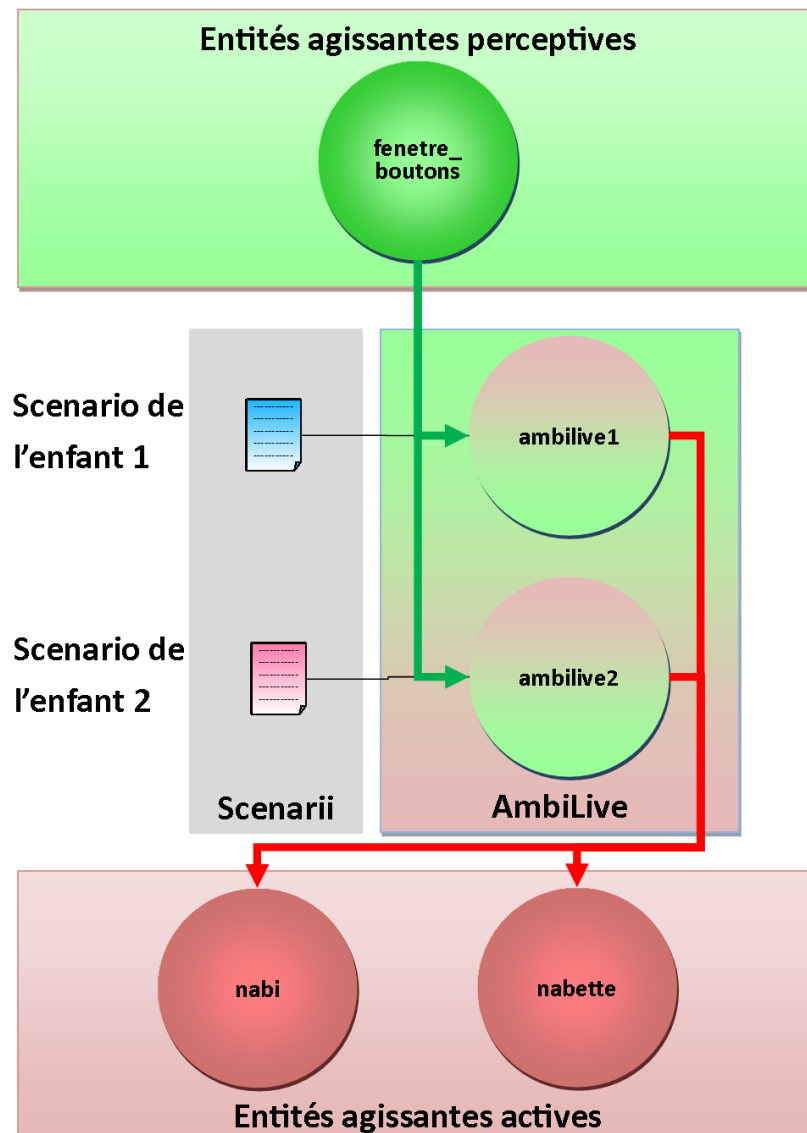


Figure 68 : Modules de la deuxième évaluation d'AmbiProg

C. Evaluation de StimCards (enfants et personnes âgées)

Evaluation de StimCards

Objectifs : L'objectif de cette évaluation était de tester StimCards auprès d'enfants et de personnes âgées. L'intérêt était de vérifier si les règles du jeu étaient simples à comprendre et si ce type de jeu pouvait intéresser un grand nombre de personnes.

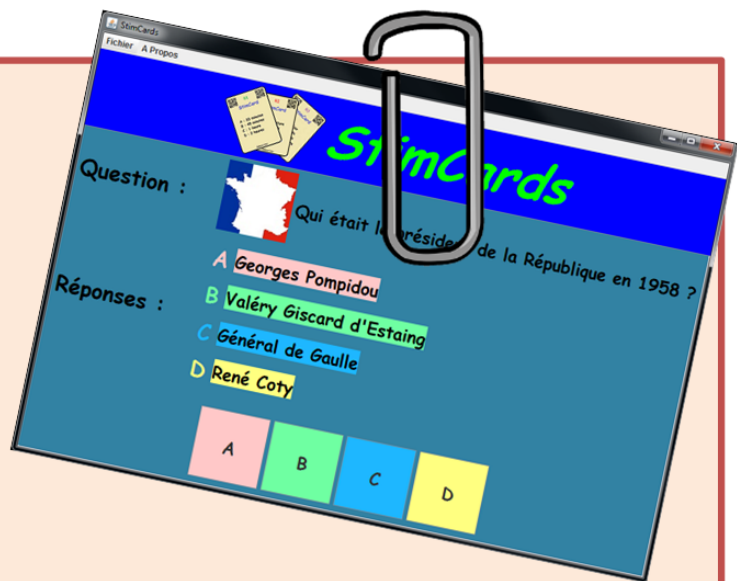
Participants : Une première évaluation a été effectuée par 52 enfants (27 filles et 25 garçons) âgés de 10 à 11 ans. Une deuxième évaluation a été effectuée auprès de 18 personnes âgées (16 femmes et 2 hommes) âgés de 63 à 88 ans.

Méthodes : Les participants étaient isolés avec StimCards associé à un interlocuteur numérique : un ordinateur, un agent virtuel et un robot. Les participants devaient effectuer un exercice puis remplir un questionnaire composé de 6 questions.

Résultats : Les enfants et les personnes âgées ont largement aimé jouer au jeu et ont trouvé les règles du jeu faciles. Les enfants ont d'avantage envie de continuer à utiliser StimCards que les personnes âgées même si personne ne l'a rejeté. Par contre, les enfants sont intéressés pour avoir une interaction personnalisée alors que les personnes âgées n'en veulent pas.

Conclusions principales : StimCards est un jeu apprécié qui évite la passivité des utilisateurs et qui augmente leur motivation.

Détail de l'expérimentation : annexe L



Lors de l'évaluation de StimCards avec les personnes âgées, les modules de base, présentés dans le Chapitre IV.B.2., ont été utilisés. Pour la saisie des réponses, les participants ont utilisé une boîte de réponse, fabriquée à l'aide d'un clavier sans fil, gérée par le module **claviersansfil**. La Figure 69 montre les modules utilisés pour l'expérimentation avec Greta. Lors de l'évaluation avec le robot ou avec l'ordinateur, le module **greta** était remplacé par le module **ordinateur** ou le module **robot**.

Le fichier de description du chemin logique contenait les informations suivantes :

```
<connect><in>reader</in><out>organizer</out></connect>
<connect><in>organizer</in><out>stimcoquestions</out></connect>
<connect><in>organizer</in><out>stimcoenv</out></connect>
<connect><in>stimcoquestions</in><out>ambilive</out></connect>
<connect><in>ambilive</in><out>stimcoquestions</out></connect>
<connect><in>ambilive</in><out>stimcoenv</out></connect>
<connect><in>ambilive</in><out>greta</out></connect>
<connect><in>stimcoenv</in><out>stimcoquestions</out></connect>
<connect><in>claviersansfil</in><out>stimcoquestions</out></connect>
```

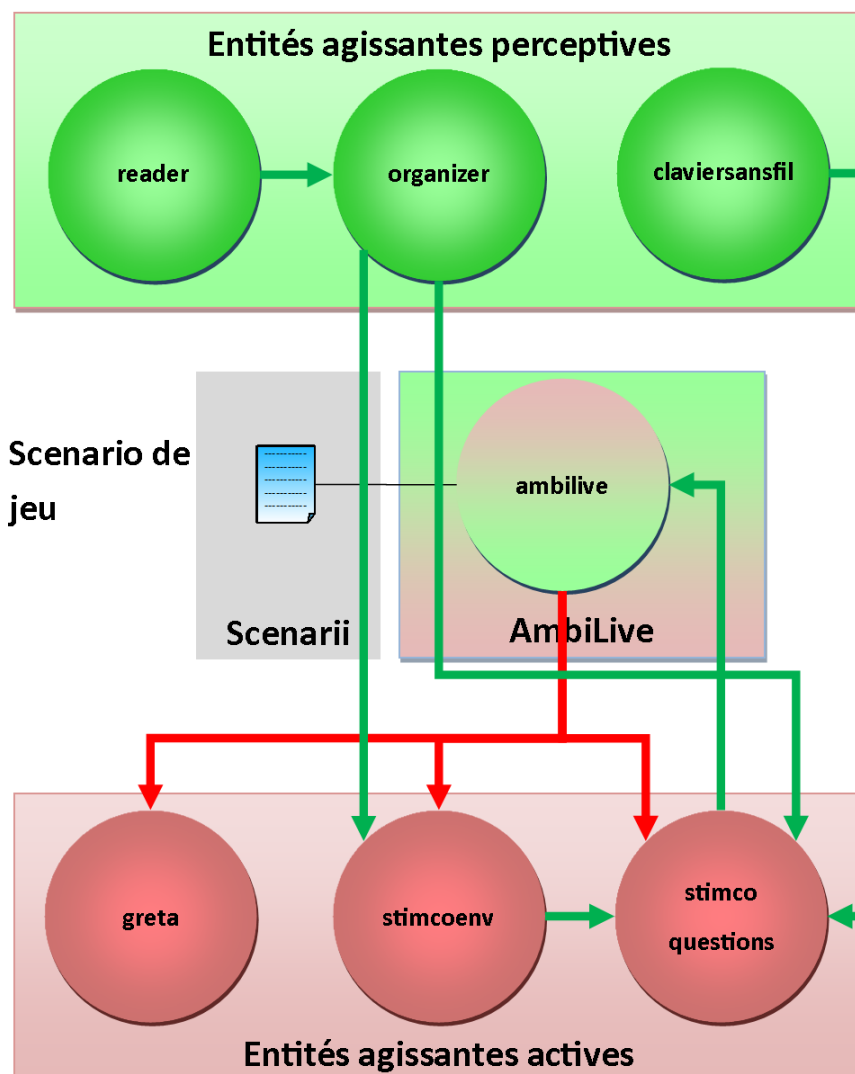


Figure 69 : Modules de l'évaluation de StimCards avec les personnes âgées

D. Evaluation du meilleur partenaire numérique pour StimCards

Evaluation du meilleur partenaire numérique pour StimCards



Objectifs : Cette évaluation

avait pour objectif de tester si l'impact de StimCards augmente avec la présence d'un partenaire numérique : un ordinateur, un agent virtuel et deux robots (un robot métallique et un robot peluche).

Participants : L'évaluation a été réalisée par 52 enfants de CM2 (27 filles et 25 garçons) âgés de 10 à 11 ans.

Méthodes : Chaque enfant a participé à quatre sessions de dix minutes où il devait faire une session d'exercices mathématiques avec StimCards et un partenaire potentiel différent à chaque fois. L'évaluation a été réalisée avec des questionnaires, des calculs de performance et de l'observation. Les sessions étaient filmées et permettaient de faire des analyses de vidéos. Après chaque évaluation et à la fin des évaluations, les participants devaient remplir un questionnaire.

Résultats : Les exercices (à difficulté croissante) n'ont jamais été jugés difficiles. Peu importe le partenaire, aucun d'entre eux n'a été jugé désagréable pour les participants. Les partenaires ont aimé jouer avec tous les partenaires. L'analyse des questionnaires a montré que les enfants ont préféré interagir avec les deux partenaires robotiques. L'analyse des vidéos a montré que les enfants ont préféré le robot peluche.

Conclusions principales : Le robot favorise l'impact de StimCards.

Détail de l'expérimentation : annexe M

L'évaluation de StimCards avec les enfants contenait 9 modules, répartis de la façon suivante :

- Trois entités agissantes perceptives :
 - **participant_response** : la fenêtre de réponse fournie aux participants,
 - **participant_response2** : la fenêtre de réponse utilisée par l'expérimentateur pour cliquer sur les boutons « oui » ou « non » lorsque les enfants répondaient à l'oral aux questions posées par le robot, l'agent virtuel ou l'ordinateur,
 - **reader** : le module décodant les codes QR.
- Trois modules de type delibetator : **stimcoquestions**, **ambilive** et **organizer** dont le rôle a été expliqué dans le Chapitre IV.B.2,
- Trois entités agissantes actives : **bioloid**, **greta** et **hautparleur** pour effectuer les actions requises. Le module **bioloid** était utilisé à la fois pour le robot métallique et pour le robot peluche.

Les modules de l'évaluation sont présentés sur la Figure 70.

Le fichier de description du chemin logique contenait les informations suivantes :

```
<connect><in>participant_response</in><out>ambilive</out></connect>
<connect><in>participant_response2</in><out>ambilive</out></connect>
<connect><in>stimcoquestions</in><out>ambilive</out></connect>
<connect><in>ambilive</in><out>bioloid</out></connect>
<connect><in>ambilive</in><out>greta</out></connect>
<connect><in>ambilive</in><out>hautparleur</out></connect>
<connect><in>ambilive</in><out>question</out></connect>
<connect><in>reader</in><out>organizer</out></connect>
<connect><in>organizer</in><out>stimcoquestions</out></connect>
<connect><in>participant_response</in><out>question</out></connect>
<connect><in>participant_response2</in><out>question</out></connect>
```

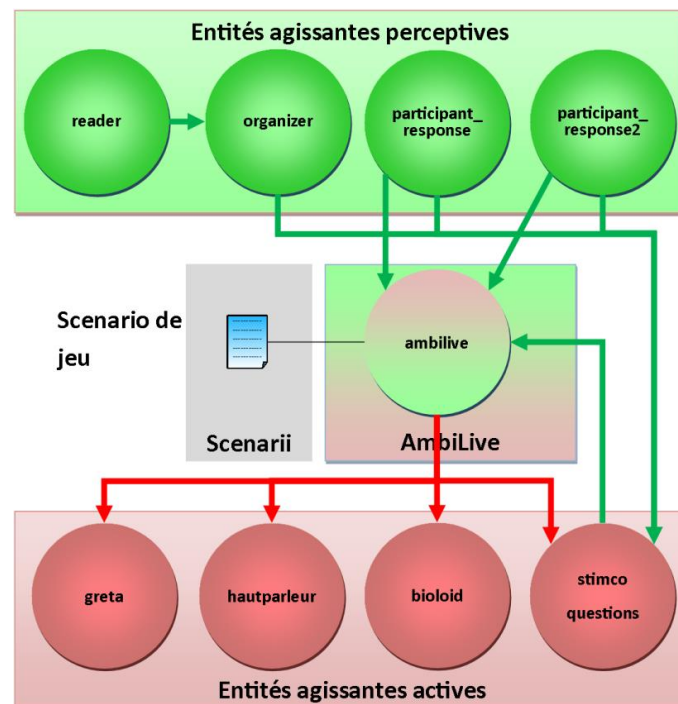
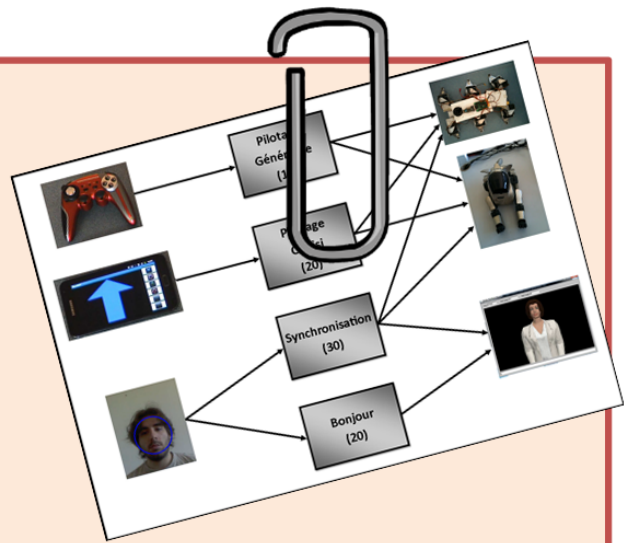


Figure 70 : Modules de l'évaluation de StimCards avec les enfants

E. Evaluation d'AmbiCop

Evaluation d'AmbiCop

Objectifs : L'objectif de cette évaluation était de tester AmbiCop pour vérifier si les conflits étaient évités, si les actions symboliques étaient bien redirigées et si les actions synchronisées démarraient en même temps et libéraient les entités agissantes en même temps. Cette évaluation consistait à observer si le système se comportait correctement. Des participants ont pu utiliser le système mais ils n'étaient pas les évaluateurs.



Participants : Les participants ont été choisis sur la base du volontariat lors d'un concours de robotique : Robofesta.

Méthodes : L'architecture ArCo a été mise en place avec une manette de PS3, un smartphone, une caméra, un hexapode, un aibo et Greta. Il y avait 4 scénarii :

- Scénario générique : la manette de PS3 prenait le contrôle soit de l'hexapode, soit de l'aibo, en fonction des disponibilités.
- Scénario spécifique : le smartphone pilotait un robot spécifique (au choix du participant).
- Scénario Greta : lorsque la caméra détectait un visage, Greta saluait la personne.
- Scénario synchro : lorsque la caméra détectait une carte de jeu, les deux robots et Greta saluaient, de façon synchronisée, l'utilisateur.

évaluation consistait à observer si les ordres envoyés étaient correctement réalisés. Le questionnaire du monde a été soumis à un banc d'essai où toutes les combinaisons ont été essayées.

Conclusions principales : AmbiCop a permis d'éviter les conflits, de rediriger des actions symboliques et de synchroniser des actions entre elles.

Détail de l'expérimentation : annexe N

L'évaluation d'AmbiCop contenait :

- Trois entités agissantes perceptives : **manette_ps3**, **manette_android** et **camera**,
- Cinq modules intermédiaires : 4 interpréteurs chargés d'interpréter les 4 scénarios de l'évaluation et un module ambicop,
- Trois entités agissantes actives : **aibo**, **greta** et **hexapode**.

Les modules de l'évaluation sont présentés sur la Figure 71.

Le fichier de description du chemin logique contenait les informations suivantes :

```
<connect><in>manette_ps3</in><out>ambilive_generique</out></connect>
<connect><in>manette_android</in><out>ambilive_specifique</out></connect>
<connect><in>camera</in><out>ambilive_greta</out></connect>
<connect><in>camera</in><out>ambilive_sync</out></connect>
<connect><in>ambilive_greta</in><out>ambicop</out></connect>
<connect><in>ambilive_specifique</in><out>ambicop</out></connect>
<connect><in>ambilive_generique</in><out>ambicop</out></connect>
<connect><in>ambilive_sync</in><out>ambicop</out></connect>
<connect><in>ambicop</in><out>aibo</out></connect>
<connect><in>ambicop</in><out>greta</out></connect>
<connect><in>ambicop</in><out>hexapod</out></connect>
```

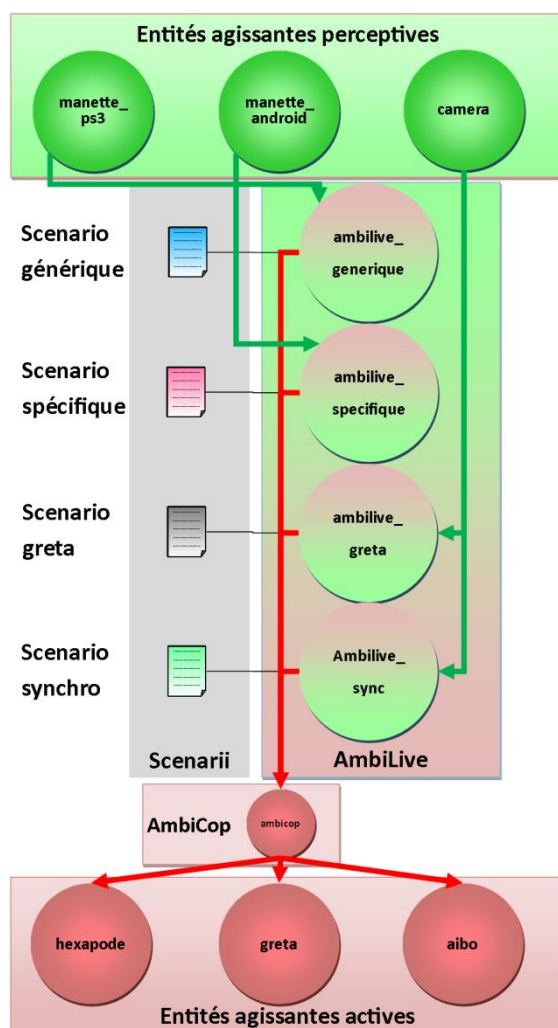


Figure 71 : Modules de l'évaluation d'AmbiCop

Synthèse

Sept expérimentations ont été développées avec l'architecture ArCo. Nous avons pu valider expérimentalement 4 des 5 dimensions cités dans les objectifs de notre travail :

- **Prestataires** : les évaluations sur AmbiProg ont montré que les non informaticiens peuvent facilement créer des scénarii. Et la majorité d'entre eux ont exprimé de l'intérêt à l'utiliser. Certains ont même imaginé les services qu'ArCo pourrait leur rendre dans leur vie quotidienne.
- **Conflits** : les évaluations effectuées à Robofesta ont permis de vérifier que les conflits étaient gérés avec succès par AmbiCop.
- **Dynamique** : lors des expérimentations avec StimCards, tous les modules n'étaient pas connectés en même temps puisque le but était de comparer trois configurations entre elles. Nous avons pu vérifier que la dynamique était correctement prise en compte.
- **Adaptabilité** : la mise en place de chaque cas particuliers d'ArCo a permis de valider l'adaptabilité puisque dans chaque cas, le serveur était « remis à zero » et ne connaissait aucun module à la première utilisation.

La **facilité** de conception n'a pas été validée lors d'expérimentation même si nous avons proposé un cadriciel MICE dans ce but dont un exemple d'utilisation est montré en annexe. Cependant, nous pouvons faire part de notre propre expérience car deux intégrations ont été effectuées :

- La Figure 72 montre l'intégration à ArCo d'une expérimentation du LIG. L'expérimentation initiale ne comprenait que l'ordinateur portable. Nous avons fourni la tablette tactile, en tant qu'entité agissante perceptive et le robot en tant qu'entité agissante active. Le développeur du LIG a pu, en moins d'une heure, connecter un agent virtuel de la plateforme MARC (Courgeon 2011), en tant qu'entité agissante active, à ArCo. Le robot et l'agent virtuel recevait les mêmes ordres.
- Un développeur du laboratoire ISIR a pu utiliser notre protocole de communication pour nous fournir une entité agissante perceptive « gestionnaire d'attention » capable de détecter une personne. Les perceptions fournies donnent la position de la personne. Ce qui permet de mettre en place le suivi d'une personne par un robot.

Pour conclure, ArCo apporte une solution aux cinq dimensions présentés dans les objectifs.



Figure 72 : Coopération avec le LIG - intégration d'un agent virtuel dans ArCo

Conclusion

Ce travail s'inscrit dans le projet pluridisciplinaire Robadom dont l'objectif est d'étudier l'impact d'un robot d'aide à domicile sur l'état psychoaffectif et cognitif de personnes âgées ayant des troubles cognitifs légers.

Nos objectifs étaient de créer une architecture plus générique et capable de prendre en compte différents objets numériques composant le Compagnon Artificiel. L'architecture devait répondre aux cinq points suivants :

- **Dynamique** du système : tous les objets numériques ne sont pas toujours disponibles : ils peuvent être en panne, déjà utilisés ou éteints. Comment le Compagnon Artificiel s'adapte-t-il au nomadisme de la personne ? Comment gérer la présence et l'absence des objets numériques connus de façon dynamique (informatique ambiante) ?
- **Adaptabilité** du système : le système doit prendre en compte des objets numériques qui n'existaient pas dans l'environnement avant. Comment intègre-t-il ces objets inconnus de façon transparente ?
- **Conception** : comment permettre à des concepteurs de créer de nouveaux capteurs et actionneurs compatibles au Compagnon Artificiel le plus simplement possible ?
- **Conflit** : il ne faut pas qu'un objet numérique ait à effectuer deux actions, utilisant les mêmes ressources, en même temps (conflits). Par exemple, on ne peut pas demander à un personnage virtuel de sourire ou d'afficher un visage neutre en même temps. Comment l'architecture informatique détecte-t-elle et gère-t-elle un conflit ?
- **Prestataire** : des personnes non expertes en informatique doivent pouvoir personnaliser et utiliser le Compagnon Artificiel. Comment permettre à ces personnes de le faire de façon simple ? Comment l'ensemble des objets numériques est-il configuré ?

Nous avons proposé une architecture logicielle ArCo (voir Figure 73) permettant de coordonner un ensemble de d'objets numériques qui construisent le Compagnon Artificiel d'un utilisateur. Un des objectifs de cette architecture est de permettre, à des prestataires, de créer des scénarii qui configurent le fonctionnement des dispositifs numériques. Chacun d'entre eux est géré par des entités agissantes perceptives ou actives.

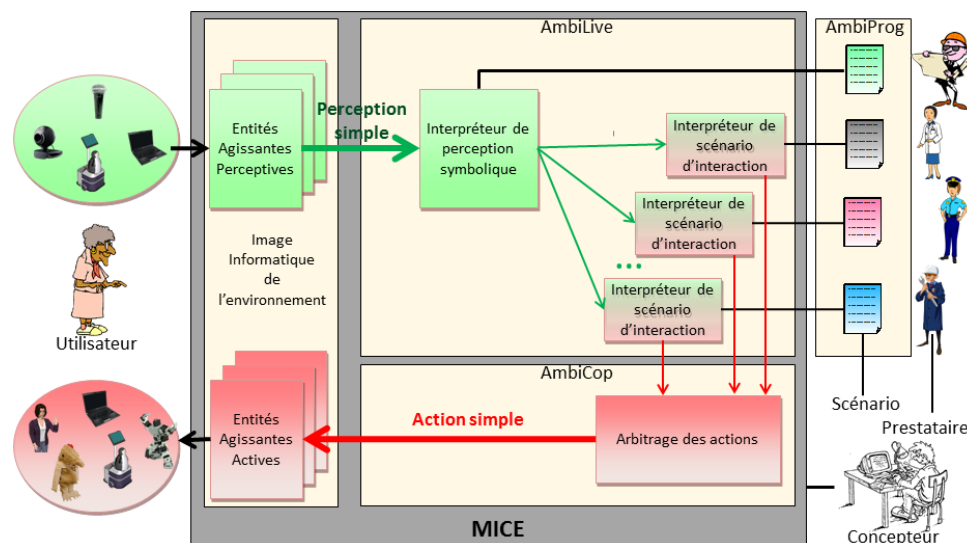


Figure 73 : Architecture ArCo

Pour permettre aux prestataires de créer de façon simple des scénarii, ArCo offre un environnement de programmation AmbiProg. Des expérimentations ont montré que l'usage d'AmbiProg est considéré comme simple et accessible à des personnes n'ayant pas de fortes connaissances en informatique. (cf. Chapitre V.B.).

Par ailleurs, ArCo permet la gestion dynamique des objets numériques par des mécanismes d'identification, de connexion et de déconnexion qui sont nécessaires en milieu ouvert, ambiant. En outre, il est également possible, avec ArCo, de tenir compte de la dynamique des dispositifs numériques de l'environnement. Ainsi, un dispositif numérique peut être rendu compatible à l'architecture ArCo en créant une entité agissante associée, par l'utilisation d'une bibliothèque fournie par le cadriciel MICE (cf. Annexe H). La prise en compte de ce nouveau dispositif numérique est alors automatique. Enfin, ArCo permet de gérer les conflits survenant dans l'environnement par un gestionnaire AmbiCop. Il choisit la « meilleure » entité pour réaliser les actions génériques, c'est-à-dire celle qui est la mieux évaluée pour ce service parmi toutes les entités agissantes disponibles. Il gère également les conflits d'accès aux entités agissantes.

Neuf d'expérimentations ont pu être réalisées grâce à ArCo en ayant deux objectifs :

- la validation expérimentale de l'architecture ArCo,
- l'évaluation de diverses interactions entre les entités agissantes et l'utilisateur.

Parmi ces neuf expérimentations, une a été réalisée par l'hôpital Broca pour évaluer un robot d'aide à domicile et l'autre (voir Figure 72). a été réalisée par le LIG pour évaluer la pertinence de la réaction d'un agent virtuel (Riviere et al. 2011). Aucune évaluation n'a été réalisée dans le but de valider l'architecture, la communication entre les composants, la bonne réception des messages... Mais l'absence de dysfonctionnement pendant l'utilisation d'ArCo a permis de valider expérimentalement l'architecture.

Concernant l'interaction homme-machine, le LIG avait effectué une expérimentation (Riviere et al. 2011) avec des Agents Conversationnels Animés . Nous l'avons reproduit avec un robot et confirmé les résultats obtenus. Cette expérimentation a permis de conclure qu'un robot doit également avoir une gestuelle cohérente avec son discours pour paraître crédible et sincère.

L'évaluation du logiciel AmbiProg (interface graphique de saisie de scénario) a permis de vérifier que son langage de programmation visuelle ne nécessite pas de compétences en informatique pour créer des scénarii. AmbiProg a été perçu positivement par ses utilisateurs. Les enfants n'ont eu besoin que de 15 minutes pour s'approprier chaque élément de programmation.

Nous avons également évalué StimCards et il en ressort que c'est un jeu apprécié, à la fois par les enfants, et par les personnes âgées. Un jeu qui fait participer activement le joueur permet de diminuer l'ennui des utilisateurs et augmente leur motivation. Nous avons pu étudier le Compagnon Artificiel avec diverses interfaces : agent virtuel, ordinateur et robots. Et dans le cadre d'un jeu de stimulation cognitive, nous constatons que le robot favorise l'impact du jeu et est le « partenaire » préféré des utilisateurs.

Perspectives

Ce travail nous a permis de créer une première version d'ArCo (voir <https://sites.google.com/site/compagnonartificiel/>), qui est une architecture informatique pour la gestion d'un Compagnon Artificiel. Un ensemble de perspectives sont envisageables pour améliorer ArCo et créer une deuxième version plus complète.

1. Calcul dynamique du chemin logique de l'information, de l'indice de priorité et de l'indice de qualité

Le travail sur cette première version d'ArCo avait pour but de créer les modules nécessaires à la gestion du Compagnon Artificiel : interpréteur de perceptions symboliques, interpréteur de scénario et gestionnaire de conflits. Certaines données utilisées par ces modules sont actuellement fixes :

- La création du chemin logique
- L'indice de priorité et l'indice de qualité.

La première amélioration d'ArCo sera de mettre en place l'**automatisation du chemin logique** de l'information entre les modules. Pour cela, chaque scénario pourrait être accompagné d'un méta-scénario. L'objectif serait lors de l'activation du scénario que son analyse détermine l'ensemble de toutes les perceptions et actions nécessaires à son usage. En cas d'absence de l'un d'entre eux un scénario pourrait ne pas être activé par exemple et donc gérer des problèmes de dysfonctionnement potentiels. L'ensemble des scénarii et l'ensemble des fiches de descriptions pourraient être utilisés pour le calcul du chemin logique de l'information entre les entités agissantes. Celui-ci serait réalisé automatiquement en fonction des entités agissantes connectées. Lorsqu'une entité agissante, utilisée dans un scénario, se déconnecte, le chemin logique serait recalculé. L'utilité d'une telle construction serait la possibilité de pouvoir télécharger des scénarii conçus pour d'autres compagnons

Le **calcul dynamique de l'indice de priorité** entre deux scénarii nécessite qu'on puisse différencier l'importance des scénarii. Pour cela chaque scénario pourrait contenir un type de scénario, par exemple scénario de jeu, de surveillance, de santé, d'information. De plus, il pourrait contenir le nom et le type de l'auteur du scénario. Le type peut être médecin, ami, utilisateur... Ainsi on pourrait établir des règles. Par exemple, un scénario écrit par un médecin est plus prioritaire qu'un scénario écrit par un ami. Pour pouvoir créer un module de calcul de l'indice de priorité, il est nécessaire de créer préalablement une ontologie concernant les types de scénario et les types d'auteur afin de déterminer une classification ou une hiérarchie entre tous les types existants.

Le **calcul dynamique de l'indice de qualité** des entités agissantes doit également être implémenté pour qu'ArCo s'adapte dynamiquement aux qualités des entités agissantes.

2. AmbiProg

Concernant **AmbiProg**, trois améliorations principales seront mises en place :

- Un nouvel élément **Aléatoire** sera créé. Il s'utilisera comme l'élément Parallélisme et Actions Synchronisées. AmbiLive interprétera de façon aléatoire une seule des branches de l'élément. Cela permettra, par exemple, de créer des variations pour que les robots ou les personnages virtuels n'effectuent pas les actions toujours de la même façon.
- Actuellement, il est possible de passer des perceptions en paramètre des fonctions en saisissant `nom_module.nom_perception` pour passer la valeur des perceptions ou

`nom_module.nom_perception.coefficient` pour passer le coefficient de la perception. Pour faciliter le passage de perception en paramètre, il faudrait créer un type d'argument spécifique, par exemple « perception » dans les fiches de description des entités. Lorsqu'AmbiProg charge les fichiers de description, pour chaque argument de type « perception », une liste déroulante contenant la liste de toutes les perceptions sera présentée à l'utilisateur afin de lui éviter de saisir lui-même les données et éviter ainsi les erreurs de saisi.

- Le langage AmbiProg ne possède pas de variables. L'élément le plus similaire est la perception. Actuellement il est possible, avec AmbiProg, de créer des perceptions et/ou de leur affecter une valeur. Pour utiliser les perceptions comme des variables, il faudrait rajouter les deux fonctions suivantes :
 - `incrimente_perception(module, perception)` : qui permet d'incrémenter la valeur d'une perception s'il s'agit d'un nombre.
 - `decremente_perception(module_perception)` : qui permet de décrémenter la valeur d'une perception s'il s'agit d'un nombre.

3. Interface de gestion des utilisateurs

Il semble important pour les utilisateurs de visualiser « ce qu'il se passe ». Ainsi, une amélioration consisterait à créer une interface graphique qui afficherait l'ensemble des entités agissantes connectées ainsi que le chemin logique de l'information. Les utilisateurs pourraient visualiser les perceptions et les actions gérées par ces entités agissantes. Ils pourraient également visualiser l'état de l'environnement (la liste des perceptions envoyées par les entités agissantes perceptives et leur valeur) et les actions en cours. D'une façon plus prospective, il serait souhaitable de mettre en place une interface vocale qui permette à l'utilisateur d'interagir directement avec son Compagnon Artificiel pour le configurer ou pour obtenir des informations sur l'état de son environnement.

Références

- Aarts, E. 2004. "Ambient Intelligence: a Multimedia Perspective." *MultiMedia, IEEE* 11 (1) (March): 12 – 19. doi:10.1109/MMUL.2004.1261101.
- Aarts, Emile, and Boris de Ruyter. 2009. "New Research Perspectives on Ambient Intelligence." *Journal of Ambient Intelligence and Smart Environments* 1 (1) (January 1): 5–14. doi:10.3233/AIS-2009-0001.
- Aarts, Emile, and Reiner Wichert. 2009. "Ambient Intelligence." In *Technology Guide*, edited by Hans-Jörg Bullinger, 244–249. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-540-88546-7_47.
- Abreu, Priscilla F. de, Vera Maria B. Werneck, Rosa Maria E. Moreira da Costa, and Luis Alfredo V. de Carvalho. 2011. "Employing Multi-agents in 3-D Game for Cognitive Stimulation." In *SVR 2011*, 73–78. Brésil: IEEE. doi:10.1109/SVR.2011.32. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5951837>.
- Aldrich, Frances K. 2012. "Smart Homes: Past, Present and Future." In *Inside the Smart Home*, edited by Richard Harper, 17–39. London: Springer-Verlag. Accessed September 23. http://www.springerlink.com/index/10.1007/1-85233-854-7_2.
- Argyroudis, Patroklos G., and Donal O'Mahony. 2004. "Securing Communications in the Smart Home." In *Embedded and Ubiquitous Computing*, edited by Laurence T. Yang, Minyi Guo, Guang R. Gao, and Niraj K. Jha, 3207:891–902. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/index/10.1007/978-3-540-30121-9_85.
- Arkin, Ronald C., Masahiro Fujita, Tsuyoshi Takagi, and Rika Hasegawa. 2003. "An Ethological and Emotional Basis for Human–robot Interaction." *Robotics and Autonomous Systems* 42 (3-4) (March): 191–201. doi:10.1016/S0921-8890(02)00375-5.
- Baier, Tim, Markus Huser, Daniel Westhoff, and Jianwei Zhang. 2006. "A Flexible Software Architecture for Multi-modal Service Robots." In *Multiconference on Computational Engineering in Systems Applications*, 587–592. Chine: IEEE. doi:10.1109/CESA.2006.4281721. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4281721>.
- Ballagas, R., J. Borchers, M. Rohs, and J.G. Sheridan. 2006. "The Smart Phone: A Ubiquitous Input Device." *IEEE Pervasive Computing* 5 (1) (January): 70–77. doi:10.1109/MPRV.2006.18.
- Barber, R., and MA Salichs. 2001. "A New Human Based Architecture for Intelligent Autonomous Robots." In *The Fourth IFAC Symposium on Intelligent Autonomous Vehicles*, 85–90.
- Bartneck, C. 2002. *eMuu: An Embodied Emotional Character for the Ambient Intelligent Home*. Technische Universiteit Eindhoven.
- Bartneck, C., and O. Michio. 2001. "eMuu-An Emotional Robot." *Demonstration at Robo Festa, Osaka*.
- Baxter, P., T. Belpaeme, L. Cañamero, P. Cosi, Y. Demiris, V. Enescu, A. Hiole, et al. 2011. "Long-term Human-robot Interaction with Young Users." In *IEEE/ACM Human-Robot Interaction 2011 Conference (Robots with Children Workshop)*.
- Beaudouin-Lafon, Michel. 2011. "Lessons Learned from the WILD Room, a Multisurface Interactive Environment." In *IHM 2011*, 1. France: ACM Press. doi:10.1145/2044354.2044376. <http://dl.acm.org/citation.cfm?doid=2044354.2044376>.
- Berenbaum, Rakel, Yehudit Lange, and Leah Abramowitz. 2011. "Augmentative Alternative Communication for Alzheimer's Patients and Families' Using SAVION." In *PETRA 2011*, 1. Grèce: ACM Press. doi:10.1145/2141622.2141677. <http://dl.acm.org/citation.cfm?doid=2141622.2141677>.
- Bouchet, Jullien, Laurence Nigay, and Thierry Ganille. 2004. "ICARE Software Components for Rapidly Developing Multimodal Interfaces." In *ICMI'04*, 251. Espagne: ACM Press. doi:10.1145/1027933.1027975. <http://portal.acm.org/citation.cfm?doid=1027933.1027975>.
- Boulabiar, Mohamed-Ikbel, Thomas Burger, Franck Poirier, and Gilles Coppin. 2011. "A Low-Cost Natural User Interaction Based on a Camera Hand-Gestures Recognizer." In *Human-*

- Computer Interaction. Interaction Techniques and Environments*, edited by Julie A. Jacko, 6762:214–221. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/index/10.1007/978-3-642-21605-3_24.
- Brandão, André, Lenisa Brandão, Giancarlo Nascimento, Bruno Moreira, Cristina Nader Vasconcelos, and Esteban Clua. 2010. "JECRIPE: Stimulating Cognitive Abilities of Children with down Syndrome in Pre-scholar Age Using a Game Approach." In *International Conference on Advances in Computer Entertainment Technology*, 15. Taiwan: ACM Press. doi:10.1145/1971630.1971635. <http://portal.acm.org/citation.cfm?doid=1971630.1971635>.
- Bratman, M.E. 1999. "Intention, Plans, and Practical Reason."
- Breazeal, C. 2001. "Emotive Qualities in Robot Speech." In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference On*, 3:1388 –1394 vol.3. doi:10.1109/IROS.2001.977175.
- . 2009. "Role of Expressive Behaviour for Robots That Learn from People." *Philosophical Transactions of the Royal Society B: Biological Sciences* 364 (1535) (November 2): 3527–3538. doi:10.1098/rstb.2009.0157.
- Breazeal, C., A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Mulanda. 2004. "Humanoid Robots as Cooperative Partners for People." *Int. Journal of Humanoid Robots* 1 (2): 1–34.
- Breazeal, C., and R. Brooks. 2005. "Robot Emotion: A Functional Perspective." *Who Needs Emotions*: 271–310.
- Breazeal, C., and B. Scassellati. 1999. "How to Build Robots That Make Friends and Influence People." In *IROS 1999*, 2:858–863. Corée du sud: IEEE. doi:10.1109/IROS.1999.812787. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=812787>.
- Breazeal, Cynthia. 2003. "Emotion and Sociable Humanoid Robots." *International Journal of Human-Computer Studies* 59 (1-2) (July): 119–155. doi:10.1016/S1071-5819(03)00018-1.
- Brisben, A., C. Safos, A. Lockerd, J. Vice, and C. Lathan. 2005. *The Cosmobot System: Evaluating Its Usability in Therapy Sessions with Children Diagnosed with Cerebral Palsy*.
- Brooks, Andrew G., and Ronald C. Arkin. 2006. "Behavioral Overlays for Non-verbal Communication Expression on a Humanoid Robot." *Autonomous Robots* 22 (1) (September 12): 55–74. doi:10.1007/s10514-006-9005-8.
- Brooks, R.A., C. Breazeal, M. Marjanović, B. Scassellati, and M.M. Williamson. 1999. "The Cog Project: Building a Humanoid Robot." In *Computation for Metaphors, Analogy, and Agents*, 52–87.
- Brown, S.M., C.L. Lisetti, and A.H. Marpaung. 2002. "Cherry, the Little Red Robot with a Mission and a Personality." In *AAAI Fall Symposium on Human-Robot Interaction, North Falmouth, Massachusetts*.
- Cabri, Giacomo, Francesco De Mola, Luca Ferrari, Letizia Leonardi, Raffaele Quitadamo, and Franco Zambonelli. 2008. "The LAICA Project: An Ad-hoc Middleware to Support Ambient Intelligence." *Multiagent and Grid Systems* 4 (3) (January 1): 235–247.
- Cappelli, A., and E. Giovannetti. 2004. "Human-robot Interaction." *Intelligenza Artificiale* 1 (2): 18–36.
- Cassell, J. 2001. "Embodied Conversational Agents: Representation and Intelligence in User Interfaces." *AI Magazine* 22 (4): 67.
- Castellano, G., R. Aylett, K. Dautenhahn, A. Paiva, P.W. McOwan, and S. Ho. 2008. "Long-term Affect Sensitive and Socially Interactive Companions." In *Proceedings of the 4th International Workshop on Human-Computer Conversation*.
- Castellano, Ginevra, Loic Kessous, and George Caridakis. 2008. "Emotion Recognition Through Multiple Modalities: Face, Body Gesture, Speech." In *Affect and Emotion in Human-Computer Interaction*, edited by Christian Peter and Russell Beale, 4868:92–103. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. http://dx.doi.org/10.1007/978-3-540-85099-1_8.
- Catenazzi, Nadia, Vanessa De Luca, Lorenzo Sommaruga, and Massimo Botta. 2012. "Guidelines to Design Inclusive Ambient Intelligence Solutions for Human Activity Sharing." In *CISIS 2012*,

- 496–501. Parleme, Italie: IEEE. doi:10.1109/CISIS.2012.130. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6245649>.
- Cerezo, E., S. Baldassarri, I. Hupont, and F.J. Seron. 2008. "Affective Embodied Conversational Agents for Natural Interaction." *Affective Computing*.
- Chatterjee, Mousumi. 2007. "Design Research: Building Human-Centered System." In *IPCC 2007*, 1–6. Etats-Unis: IEEE. doi:10.1109/IPCC.2007.4464097. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4464097>.
- Chen, P.L., and Tsai-Yen Li. 2006. "Realizing Emotional Autonomous Virtual Agents in a Multi-user Virtual Environment." In *Proceedings of the International Computer Symposium*. Taiwan.
- Chilukoti, Naveen, Kenneth Early, Sarvinder Sandhu, Cheryl Riley-Doucet, and Debatosh Debnath. 2007. "Assistive Technology for Promoting Physical and Mental Exercise to Delay Progression of Cognitive Degeneration in Patients with Dementia." In *BIOCAS 2007*, 235–238. Montreal, Quebec: IEEE. doi:10.1109/BIOCAS.2007.4463352. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4463352>.
- Chin, Jeannette, Vic Callaghan, and Graham Clarke. 2009. "Soft-appliances: A Vision for User Created Networked Appliances in Digital Homes." *Journal of Ambient Intelligence and Smart Environments* 1 (1) (January 1): 69–75. doi:10.3233/AIS-2009-0010.
- Clare, L., RT Woods, ED Moniz Cook, M. Orrell, and A. Spector. 2003. "Cognitive Rehabilitation and Cognitive Training for Early-stage Alzheimer's Disease and Vascular Dementia." *Cochrane Database Syst Rev* 4.
- Clodic, Aurelie, Maxime Ransan, Rachid Alami, and Vincent Montreuil. 2007. "A Management of Mutual Belief for Human-robot Interaction." In *International Conference on Systems, Man and Cybernetics*, 1551–1556. Canada: IEEE. doi:10.1109/ICSMC.2007.4414019. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4414019>.
- Cook, Diane J. 2009. "Multi-agent Smart Environments." *Journal of Ambient Intelligence and Smart Environments* 1 (1) (January 1): 51–55. doi:10.3233/AIS-2009-0007.
- Cook, Diane J., Juan C. Augusto, and Vikramaditya R. Jakkula. 2009. "Ambient Intelligence: Technologies, Applications, and Opportunities." *Pervasive and Mobile Computing* 5 (4) (August): 277–298. doi:10.1016/j.pmcj.2009.04.001.
- Coroama, V., J. Bohn, and F. Mattern. 2004. "Living in a Smart Environment Implications for the Coming Ubiquitous Information Society." In *International Conference on Systems, Man and Cybernetics*, 6:5633–5638. Pays-Bas: IEEE. doi:10.1109/ICSMC.2004.1401091. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1401091>.
- Curgeon, M. 2011. "Marc: Modèles Informatiques Des Émotions Et De Leurs Expressions Faciales Pour L'interaction Homme-machine Affective Temps Réel". Ecole doctorale EDIPS. Directeurs de thèse : Jean-Claude Martin et Christian Jacquemin.: Université Paris Sud-Paris XI.
- Curgeon, M., J.C. Martin, and C. Jacquemin. 2008. "MARC: Un Personnage Virtuel Réactif Expressif." In *Troisième Workshop Francophone Sur Les Agents Conversationnels Animés. WACA'08*.
- Crandall, Aaron S., and Diane J. Cook. 2009. "Coping with Multiple Residents in a Smart Environment." *Journal of Ambient Intelligence and Smart Environments* 1 (4) (January 1): 323–334. doi:10.3233/AIS-2009-0041.
- Darwin, C. 2002. *The Expression of the Emotions in Man and Animals*. Oxford University Press, USA.
- Dautenhahn, Kerstin, Chrystopher L. Nehaniv, Michael L. Walters, Ben Robins, Hatice Kose-Bagci, N. Assif Mirza, and Mike Blow. 2009. "KASPAR – a Minimally Expressive Humanoid Robot for Human–robot Interaction Research." *Applied Bionics and Biomechanics* 6 (3-4) (December 2): 369–397. doi:10.1080/11762320903123567.
- De Andrade, Norberto Nuno Gomes. 2011. "The 'A.I.vatar': Artificial Intelligent Agents in the Context of Ambient Intelligence." *Journal of Ambient Intelligence and Smart Environments* 3 (4) (January 1): 363–370. doi:10.3233/AIS-2011-0123.
- De Ruyter, Boris. 2011. "Social Interactions in Ambient Intelligent Environments." *Journal of Ambient Intelligence and Smart Environments* 3 (2) (January 1): 175–177. doi:10.3233/AIS-2011-0104.

- De Sevin, E., R. Niewiadomski, E. Bevacqua, A.M. Pez, M. Mancini, and C. Pelachaud. 2010. "Greta, Une Plateforme D'agent Conversationnel Expressif Et Interactif." *Technique Et Science Informatiques* 29 (7): 751.
- Déniz, O., M. Castrillón, J. Lorenzo, and M. Hernández. 2012. "CASIMIRO, The Sociable Robot." In *Computer Aided Systems Theory – EUROCAST 2007*, edited by Roberto Moreno Díaz, Franz Pichler, and Alexis Quesada Arencibia, 4739:1049–1056. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed June 27. http://www.springerlink.com/index/10.1007/978-3-540-75867-9_131.
- Derycke, A., V. Chevrin, J. Rouillard, and others. 2005. "Intermédiations Multicanales Et Multimodales Pour l'E-Formation: l'Architecture Du Projet Ubi-Learn."
- Dey, Anind K. 2009. "Modeling and Intelligibility in Ambient Environments." *Journal of Ambient Intelligence and Smart Environments* 1 (1) (January 1): 57–62. doi:10.3233/AIS-2009-0008.
- Dooley, James, Christian Wagner, Hani Hagra, and Gaetan Pruvost. 2011. "FollowMe: The Persistent GUI." In *Parallel Computing in Electrical Engineering (PARELEC)*, 123–126. Royaume-Uni: IEEE. doi:10.1109/PARELEC.2011.15. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5770413>.
- Ducatel, K., M. Bogdanowicz, F. Scapolo, J. Leijten, and J.C. Burgelman. 2001. *Scenarios for Ambient Intelligence in 2010*. Office for official publications of the European Communities.
- Dybala, Pawel, Michal Ptaszynski, Jacek Maciejewski, Mizuki Takahashi, Rafal Rzepka, and Kenji Araki. 2010. "Multiagent System for Joke Generation: Humor and Emotions Combined in Human-agent Conversation." *Journal of Ambient Intelligence and Smart Environments* 2 (1) (January 1): 31–48. doi:10.3233/AIS-2010-0053.
- Ekman, P., and W.V. Friesen. 2003. *Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues*. Ishk.
- Ekman, P., W.V. Friesen, M. O'Sullivan, and K. Scherer. 1980. "Relative Importance of Face, Body, and Speech in Judgments of Personality and Affect." *Journal of Personality and Social Psychology* 38 (2): 270.
- Elgammal, Ahmed. 2006. "Human-centered Multimedia: Representations and Challenges." In *HCM 2006*, 11–18. Etats-Unis: ACM Press. doi:10.1145/1178745.1178751. <http://portal.acm.org/citation.cfm?doid=1178745.1178751>.
- Francone, Jérémie, and Laurence Nigay. 2011. "Using the User's Point of View for Interaction on Mobile Devices." In *IHM 2011*, 1. France: ACM Press. doi:10.1145/2044354.2044360. <http://dl.acm.org/citation.cfm?doid=2044354.2044360>.
- Freeman, Dustin, Fanny Chevalier, Emma Westecott, Kyle Duffield, Kate Hartman, and Derek Reilly. 2012. "Tweetris: Play with Me." In *TEI'12*, 319–320. Canada: ACM Press. doi:10.1145/2148131.2148201. <http://dl.acm.org/citation.cfm?doid=2148131.2148201>.
- Fritsch, J., M. Kleinhagenbrock, A. Haasch, S. Wrede, and G. Sagerer. 2005. "A Flexible Infrastructure for the Development of a Robot Companion with Extensible HRI-Capabilities." In *ICRA 2005*, 3408–3414. Taiwan: IEEE. doi:10.1109/ROBOT.2005.1570637. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1570637>.
- Fujita, M., Y. Kuroki, T. Ishida, and T.T. Doi. 2003. "Autonomous Behavior Control Architecture of Entertainment Humanoid Robot SDR-4X." In *IROS 2003*, 1:960–967. Etats-Unis: IEEE. doi:10.1109/IROS.2003.1250752. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1250752>.
- García-Herranz, Manuel, Xavier Alamán, and Pablo A. Haya. 2010. "Easing the Smart Home: A Rule-based Language and Multi-agent Structure for End User Development in Intelligent Environments." *Journal of Ambient Intelligence and Smart Environments* 2 (4) (January 1): 437–438. doi:10.3233/AIS-2010-0085.
- Gardner, Michael, Jonathon Richter, and Aki Härmä. 2012. "Virtual and Mixed Reality Intelligent Environments." *Journal of Ambient Intelligence and Smart Environments* 4 (1) (January 1): 3–4. doi:10.3233/AIS-2011-0134.

- Gat, E., and others. 1998. "On Three-layer Architectures." *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems* 195.
- Gat, Erann. 1991. "Integrating Reaction and Planning in a Heterogeneous Asynchronous Architecture for Mobile Robot Navigation." *ACM SIGART Bulletin* 2 (4) (July 1): 70–74. doi:10.1145/122344.122357.
- Georgeff, M.P., and F.F. Ingrand. 1989. *Decision-making in an Embedded Reasoning System*. Australian Artificial Intelligence Institute.
- Gockley, R., A. Bruce, J. Forlizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. Sellner, et al. 2005. "Designing Robots for Long-term Social Interaction." In *IROS 2005*, 2199–2204. Canada: IEEE. doi:10.1109/IROS.2005.1545303. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1545303>.
- Gong, J., and P. Tarasewich. 2004. "Guidelines for Handheld Mobile Device Interface Design." In *Proceedings of DSI 2004 Annual Meeting*, 3751–3756.
- Goth, Gregory. 2011. "Brave NUI World." *Communications of the ACM* 54 (12) (December 1): 14. doi:10.1145/2043174.2043181.
- Goumopoulos, C., A. Kameas, H. Hagnas, V. Callaghan, M. Gardner, W. Minker, Weber, Y. Bellik, and A. Meliones. 2008. "ATRACO: Adaptive and Trusted Ambient Ecologies." In *Self-Adaptive and Self-Organizing Systems Workshops*, 96–101. Italie: IEEE. doi:10.1109/SASOW.2008.13. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4800660>.
- Graf, Birgit, Ulrich Reiser, Martin Hagele, Kathrin Mauz, and Peter Klein. 2009. "Robotic Home Assistant Care-O-bot 3 - Product Vision and Innovation Platform." In *Proceedings of the 13th International Conference on Human-Computer Interaction HCI International*, 139–144. Etats-Unis: IEEE. doi:10.1109/ARSO.2009.5587059. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5587059>.
- Gross, H.-M., H. Boehme, Ch. Schroeter, S. Mueller, A. Koenig, E. Einhorn, Ch. Martin, M. Merten, and A. Bley. 2009. "TOOMAS: Interactive Shopping Guide Robots in Everyday Use - Final Implementation and Experiences from Long-term Field Trials." In *RO-MAN 2009*, 2005–2012. Etats-Unis: IEEE. doi:10.1109/IROS.2009.5354497. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5354497>.
- Gross, H.-M., Ch. Schroeter, S. Mueller, M. Volkhardt, E. Einhorn, A. Bley, T. Langner, Ch. Martin, and M. Merten. 2011. "I'll Keep an Eye on You: Home Robot Companion for Elderly People with Cognitive Impairment." In *Systems, Man, and Cybernetics (SMC)*, 2481–2488. Anchorage, Alaska: IEEE. doi:10.1109/ICSMC.2011.6084050. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6084050>.
- Hans, M., B. Graf, and R.D. Schraft. 2002. "Robotic Home Assistant Care-O-bot: Past-present-future." In *RO-MAN 2002*, 380–385. Allemagne: IEEE. doi:10.1109/ROMAN.2002.1045652. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1045652>.
- Hanson, David, Stuart Baurmann, Thomas Riccio, Richard Margolin, Tim Dockins, Matthew Tavares, and Kevin Carpenter. 2009. "Zeno: a Cognitive Character." In *AI Magazine, and Special Proc. of AAAI National Conference, Chicago*.
- Hanson, David, Daniele Mazzei, Carolyn Garver, Arti Ahluwalia, Danilo De Rossi, Matt Stevenson, and Kellie Reynolds. 2012. "Realistic Humanlike Robots for Treatment of ASD, Social Training, and Research; Shown to Appeal to Youths with ASD, Cause Physiological Arousal, and Increase Human-to-Human Social Engagement." *PETRA (PErvasive Technologies Related to Assistive Environment)*.
- Heerink, M., B. Krose, V. Evers, and B. Wielinga. 2006. "The Influence of a Robot's Social Abilities on Acceptance by Elderly Users." In *RO-MAN 2006*, 521–526. Royaume-Uni: IEEE. doi:10.1109/ROMAN.2006.314442. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4107860>.
- Ho, Wan Ching, Kerstin Dautenhahn, Mei Yii Lim, Patricia A. Vargas, Ruth Aylett, and Sibylle Enz. 2009. "An Initial Memory Model for Virtual and Robot Companions Supporting Migration and Long-term Interaction." In *RO-MAN 2009*, 277–284. Japon: IEEE.

- doi:10.1109/ROMAN.2009.5326204.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5326204>.
- Hsu, Sheng-Hui, Chih-Yueh Chou, Fei-Ching Chen, Yuan-Kai, and Tak-Wai Chan. 2007. "An Investigation of the Differences Between Robot and Virtual Learning Companions' Influences on Students' Engagement." In *DIGITEL '07*, 41–48. Taiwan: IEEE. doi:10.1109/DIGITEL.2007.10.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4148830>.
- Huang, Jun-Da. 2011. "Kinerehab: a Kinect-based System for Physical Rehabilitation – a Pilot Study for Young Adults with Motor Disabilities." In *ASSETS '11*, 319. Ecosse: ACM Press. doi:10.1145/2049536.2049627. <http://dl.acm.org/citation.cfm?doid=2049536.2049627>.
- Hudlicka, Eva, Sabine Payr, Rodrigo Ventura, Christian Becker-Asano, Kerstin Fischer, Iolanda Leite, and Christian Von. 2009. "Social Interaction with Robots and Agents: Where Do We Stand, Where Do We Go?" In *ACII 2009*, 1–6. Hollande: IEEE. doi:10.1109/ACII.2009.5349472. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5349472>.
- Hussaan, Aarij Mahmood, Karim Sehaba, and Alain Mille. 2011. "Tailoring Serious Games with Adaptive Pedagogical Scenarios: A Serious Game for Persons with Cognitive Disabilities." In *ICALT '11*, 486–490. Etats-Unis: IEEE. doi:10.1109/ICALT.2011.152. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5992250>.
- Imbeault, Frederick, Bruno Bouchard, and Abdenour Bouzouane. 2011. "Serious Games in Cognitive Training for Alzheimer's Patients." In *SeGAH 2011*, 1–8. Portugal: IEEE. doi:10.1109/SeGAH.2011.6165447. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6165447>.
- Issarny, Valérie, Daniele Sacchetti, Ferda Tartanoglu, Françoise Sailhan, Rafik Chibout, Nicole Levy, and Angel Talamona. 2005. "Developing Ambient Intelligence Systems: A Solution Based on Web Services." *Automated Software Engineering* 12 (1) (January): 101–137. doi:10.1023/B:AUSE.0000049210.42738.00.
- Jacquet, C. 2006. "Présentation Opportuniste Et Multimodale D'informations Dans Le Cadre De L'intelligence Ambiante". PhD thesis, Université d'Orsay Paris-Sud.
- Jacquet, C., Y. Bellik, and Y. Bourda. 2007. "KUP: a Model for the Multimodal Presentation of Information in Ambient Intelligence." In *Intelligent Environments, 2007. IE 07. 3rd IET International Conference On*, 432–439.
- Jain, Jhilmil, Arnold Lund, and Dennis Wixon. 2011. "The Future of Natural User Interfaces." In *CHI EA '11*, 211. Canada: ACM Press. doi:10.1145/1979742.1979527. <http://portal.acm.org/citation.cfm?doid=1979742.1979527>.
- James, W. 1884. "What Is an Emotion ?" *Oxford University Press on Behalf on the Minde Association* 9 (34): 188–205.
- Kanda, T., and H. Ishiguro. 2005. "Communication Robots for Elementary Schools." In *Proc. AISB'05 Symposium Robot Companions: Hard Problems and Open Challenges in Robot-Human Interaction*, 54–63.
- KangWoo Lee, Hyoung-Rock Kim, Wan Chul Yoon, Young-Sik Yoon, and Dong-Soo Kwon. 2005. "Designing a Human-robot Interaction Framework for Home Service Robot." In *RO-MAN 2005*, 286–293. IEEE. doi:10.1109/ROMAN.2005.1513793. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1513793>.
- Kawano, R. 1997. "Steps Toward the Realization of 'Human-centered Systems'-an Overview of the Human Factors Activities at TEPCO." In *Human Factors and Power Plants*, 13/27–13/32. Etats-Unis: IEEE. doi:10.1109/HFPP.1997.624880. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=624880>.
- Kędzierski, Jan, Robert Muszyński, Carsten Zoll, Adam Oleksy, and Mirela Frontkiewicz. 2013. "EMYS—Emotive Head of a Social Robot." *International Journal of Social Robotics* 5 (2) (March 16): 237–249. doi:10.1007/s12369-013-0183-1.
- Kessens, Judith M., Mark A. Neerincx, Rosemarijn Looije, Melanie Kroes, and Gerrit Bloothoof. 2009. "Facial and Vocal Emotion Expression of a Personal Computer Assistant to Engage, Educate

- and Motivate Children.” In *ACII 2009*, 1–7. Hollande: IEEE. doi:10.1109/ACII.2009.5349582. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5349582>.
- Kidd, C.D. 2003. “Sociable Robots: The Role of Presence and Task in Human-robot Interaction”. Master’s thesis, Massachusetts Institute of Technology.
- Kidd, CD, and C. Breazeal. 2006. *Designing a Sociable Robot System for Weight Maintenance. Submitted to IEEE Consumer Communications and Networking Conference (IEEE CCNC 2006). Las Vegas, NV, USA, 7–10 January 2006*.
- Kidd, Cory D., and Cynthia Breazeal. 2008. “Robots at Home: Understanding Long-term Human-robot Interaction.” In *IROS 2008*, 3230–3235. France: IEEE. doi:10.1109/IROS.2008.4651113. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4651113>.
- Kim, Song-Gook, Jang-Woon Kim, and Chil-Woo Lee. 2012. “Implementation of Multi-touch Tabletop Display for HCI (Human Computer Interaction).” In *Human-Computer Interaction. Interaction Platforms and Techniques*, edited by Julie A. Jacko, 4551:854–863. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed September 22. http://www.springerlink.com/index/10.1007/978-3-540-73107-8_94.
- Kim, Youjeong, and S. Shyam Sundar. 2012. “Anthropomorphism of Computers: Is It Mindful or Mindless?” *Computers in Human Behavior* 28 (1) (January): 241–250. doi:10.1016/j.chb.2011.09.006.
- Kirby, Rachel, Jodi Forlizzi, and Reid Simmons. 2010. “Affective Social Robots.” *Robotics and Autonomous Systems* 58 (3): 322 – 332. doi:10.1016/j.robot.2009.09.015.
- Kuhn, R., J.C. Junqua, D. Kryze, A. Nickolov, and K. Sloimenov. 2004. “A Multimodal, Networked Dialogue System for Access to Entertainment.” In *CCNC 2004*, 325–330. Etats-Unis: IEEE. doi:10.1109/CCNC.2004.1286881. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1286881>.
- Kulk, JA, and JS Welsh. 2008. “A Low Power Walk for the NAO Robot.” In *Proceedings of the 2008 Australasian Conference on Robotics & Automation (ACRA-2008)*, J. Kim and R. Mahony, Eds.
- Laperrousaz, C., and P. Teutsch. 2003. “Un Compagnon Logiciel Capable De Dialoguer. Clavie: Compagnon Logiciel D’aide Aux Apprenants Dans l’enVironnement CroisièrEs.” In *IEEE International Conference on Advanced Learning Technologies (ICALT’2003)*, 9–11.
- Law, Yee Wei, and Paul Havinga. 2011. “Security and Dependability for Ambient Intelligence: Informative but Busy.” *Journal of Ambient Intelligence and Smart Environments* 3 (4) (January 1): 373–374. doi:10.3233/AIS-2011-0119.
- Lazar, A. 2007. “Engaged Learning in a Computer Science Course.” *Journal of Computing Sciences in Colleges* 23 (1): 38–44.
- Lee, Kwan Min, Younbo Jung, Jaywoo Kim, and Sang Ryong Kim. 2006. “Are Physically Embodied Social Agents Better Than Disembodied Social Agents?: The Effects of Physical Embodiment, Tactile Interaction, and People’s Loneliness in Human–robot Interaction.” *International Journal of Human-Computer Studies* 64 (10) (October): 962–973. doi:10.1016/j.ijhcs.2006.05.002.
- Lester, James C., Sharolyn A. Converse, Susan E. Kahler, S. Todd Barlow, Brian A. Stone, and Ravinder S. Bhogal. 1997. “The Persona Effect: Affective Impact of Animated Pedagogical Agents.” In *CHI’97*, 359–366. Etats-Unis: ACM Press. doi:10.1145/258549.258797. <http://portal.acm.org/citation.cfm?doid=258549.258797>.
- Levin, Golan, and Paul Yarin. 1999. “Bringing Sketching Tools to Keychain Computers with an Acceleration-based Interface.” In *CHI EA ’99*, 268. Etats-Unis: ACM Press. doi:10.1145/632716.632881. <http://portal.acm.org/citation.cfm?doid=632716.632881>.
- Lew, Michael, Erwin M. Bakker, Nicu Sebe, and Thomas S. Huang. 2012. “Human-Computer Intelligent Interaction: A Survey.” In *Human–Computer Interaction*, edited by Michael Lew, Nicu Sebe, Thomas S. Huang, and Erwin M. Bakker, 4796:1–5. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed September 22. http://www.springerlink.com/index/10.1007/978-3-540-75773-3_1.

- Libin, A. 2003. "Persons and Their Artificial Partners: Robototherapy as an Alternative Non-pharmacological Treatment." *Annual Review of CyberTherapy and Telemedicine*: 45.
- Lindstrom, M., A. Oreback, and H.I. Christensen. 2000. "BERRA: a Research Architecture for Service Robots." In *ICRA 2000*, 4:3278–3283. Etats-Unis: IEEE. doi:10.1109/ROBOT.2000.845168. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=845168>.
- Lockerd, A.D., A.J. Brisben, and C.E. Lathan. 2004. "Robotic Toolkit for Pediatric Rehabilitation, Assessment and Monitoring." *Annual Review of CyberTherapy and Telemedicine*: 59.
- Lohse, Manja, Katharina J. Rohlfing, Britta Wrede, and Gerhard Sagerer. 2008. "'Try Something Else!' When Users Change Their Discursive Behavior in Human-robot Interaction." In *ICRA 2008*, 3481–3486. Etats-Unis: IEEE. doi:10.1109/ROBOT.2008.4543743. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543743>.
- Long To, B Thompson, J R Blum, G Maehara, R F Hess, and J R Cooperstock. 2011. "A Game Platform for Treatment of Amblyopia." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 19 (3) (June): 280–289. doi:10.1109/TNSRE.2011.2115255.
- Lopez-Martinez, Alvaro, Sandra Santiago-Ramajo, Alfonso Caracuel, Carlos Valls-Serrano, Miguel J. Hornos, and Maria J. Rodriguez-Fortiz. 2011. "Game of Gifts Purchase: Computer-based Training of Executive Functions for the Elderly." In *SeGAH 2011*, 1–8. Portugal: IEEE. doi:10.1109/SeGAH.2011.6165448. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6165448>.
- MacDorman, KARL F, and HIROSHI Ishiguro. 2004. "The Study of Interaction Through the Development of Androids." In *Computer Vision and Image Processing Workshop, Information Processing Society of Japan, SIG Technical Reports*, 69–75.
- Makedon, Fillia, Rong Zhang, Georgios Alexandrakis, Charles B. Owen, Heng Huang, and Andrew J. Saykin. 2010. "An Interactive User Interface System for Alzheimer's Intervention." In *PETRA 2010*, 1. Grèce: ACM Press. doi:10.1145/1839294.1839336. <http://portal.acm.org/citation.cfm?doid=1839294.1839336>.
- Michaud, F., C. Côté, D. Létourneau, Y. Brosseau, J.-M. Valin, é. Beaudry, C. Raïevsky, et al. 2006. "Spartacus Attending the 2005 AAAI Conference." *Autonomous Robots* 22 (4) (December 27): 369–383. doi:10.1007/s10514-006-9014-7.
- Minker, Wolfgang, Ramón López-Cózar, and Michael McTear. 2009. "The Role of Spoken Language Dialogue Interaction in Intelligent Environments." *Journal of Ambient Intelligence and Smart Environments* 1 (1) (January 1): 31–36. doi:10.3233/AIS-2009-0004.
- Morelli, Tony, John Foley, Luis Columna, Lauren Lieberman, and Eelke Folmer. 2010. "VI-Tennis: a Vibrotactile/Audio Exergame for Players Who Are Visually Impaired." In *FDG 2010*, 147–154. Etats-Unis: ACM Press. doi:10.1145/1822348.1822368. <http://portal.acm.org/citation.cfm?doid=1822348.1822368>.
- Mori, Yoshikazu, Yosuke Ojima, Masayuki Ishida, and Naoyuki Kubota. 2007. "Autonomous Behavior Generator for a Companion Robot 'SELF' with Which Humans Do Not Get Bored." In *CME 2007*, 1216–1220. Chine: IEEE. doi:10.1109/ICCM.2007.4381937. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4381937>.
- Moshkina, L., and R.C. Arkin. 2003. "On TAMEing Robots." In *SMC 2003*, 4:3949–3959. Etats-Unis: IEEE. doi:10.1109/ICSMC.2003.1244505. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1244505>.
- Murray, John C., Lola Cañamero, Kim A. Bard, Marina Davila Ross, and Kate Thorsteinsson. 2009. "The Influence of Social Interaction on the Perception of Emotional Expression: A Case Study with a Robot Head." In *Advances in Robotics*, edited by Jong-Hwan Kim, Shuzhi Sam Ge, Prahlad Vadakkepat, Norbert Jesse, Abdullah Al Manum, Sadasivan Puthusserypady K, Ulrich Rückert, et al., 5744:63–72. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/index/10.1007/978-3-642-03983-6_10.
- Nani, M., P. Caleb-Solly, S. Dogramadzi, T. Fear, and H. van den Heuvel. 2010. "MOBISERV: An Integrated Intelligent Home Environment for the Provision of Health, Nutrition and Mobility Services to the Elderly."

- Ndiwalana, Ali, C. M. Chewar, Jacob Somervell, and D. Scott McCrickard. 2003. "Ubiquitous Computing: By the People, for the People." In *CHI 2003*, 968. Etats-Unis: ACM Press. doi:10.1145/765891.766099. <http://portal.acm.org/citation.cfm?doid=765891.766099>.
- Nishio, Shuichi, Hiroshi Ishiguro, and Norihiro Hagita. 2007. "Geminoid: Teleoperated Android of an Existing Person." *Humanoid Robots-new Developments. I-Tech* 14.
- Nomura, Tatsuya, and Miyuki Sasa. 2009. "Investigation of Differences on Impressions of and Behaviors Toward Real and Virtual Robots Between Elder People and University Students." In *ICORR 2009*, 934–939. Japon: IEEE. doi:10.1109/ICORR.2009.5209626. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5209626>.
- Norrie, Lauren, and Roderick Murray-Smith. 2011. "Virtual Sensors: a Rapid Prototyping of Ubiquitous Interaction with a Mobile Phone and a Kinect." In *MobileHCI '11*, 25. Suède: ACM Press. doi:10.1145/2037373.2037378. <http://dl.acm.org/citation.cfm?doid=2037373.2037378>.
- Oh, Jun-ho, David Hanson, Won-sup Kim, Young Han, Jung-yup Kim, and Ill-woo Park. 2006. "Design of Android Type Humanoid Robot Albert HUBO." In *IROS 2006*, 1428–1433. IEEE. doi:10.1109/IROS.2006.281935. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4058572>.
- Ohara, Kenichi, Shunsuke Negi, Tomohito Takubo, Yasushi Mae, and Tatsuo Arai. 2009. "Evaluation of Virtual and Real Robot Based on Human Impression." In *Ro-man 2009*, 873–878. IEEE. doi:10.1109/ROMAN.2009.5326336. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5326336>.
- Patrizia, Marti, Moderini Claudio, Giusti Leonardo, and Pollini Alessandro. 2009. "A Robotic Toy for Children with Special Needs: From Requirements to Design." In *ICORR 2009*, 918–923. Japon: IEEE. doi:10.1109/ICORR.2009.5209500. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5209500>.
- Pesty, S., and D. Duhaut. 2011. "Artificial Companion: Building a Impacting Relation." In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference On*, 2902–2907.
- Pesty, S., C. Webber, and N. Balacheff. 2001. "Baghera: Une Architecture Multi-agents Pour L'apprentissage Humain." *Agents Logiciels, Cooperation, Apprentissage Et Activité Humaine, ALCAA*: 204–214.
- Peters, RA, DM Wilkes, DM Gaines, and K. Kawamura. 1999. "A Software Agent Based Control System for Human-robot Interaction." In *Proceedings of the Second International Symposium on Humanoid Robot*.
- Pollack, M.E., L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, S. Engberg, et al. 2002. "Pearl: A Mobile Robotic Assistant for the Elderly." In *AAAI Workshop on Automation as Eldercare*. Vol. 2002.
- Poupyrev, I., S. Maruyama, and J. Rekimoto. 2002. "Ambient Touch: Designing Tactile Interfaces for Handheld Devices." In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, 51–60.
- Preuveneers, Davy, and Paulo Novais. 2012. "A Survey of Software Engineering Best Practices for the Development of Smart Applications in Ambient Intelligence." *Journal of Ambient Intelligence and Smart Environments* 4 (3) (January 1): 149–162. doi:10.3233/AIS-2012-0150.
- Pruvost, G., and Y. Bellik. 2009. "Ambient Multimodal Human-Computer Interaction." In *Proceedings of the Poster Session at The European Future Technologies Conference*, 1–2.
- Qin, Weijun, Yue Suo, and Yuanchun Shi. 2006. "CAMPS: A Middleware for Providing Context-Aware Services for Smart Space." In *Advances in Grid and Pervasive Computing*, edited by Yeh-Ching Chung and José E. Moreira, 3947:644–653. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/index/10.1007/11745693_63.
- Rico, Julie, and Stephen Brewster. 2010. "Usable Gestures for Mobile Interfaces." In *CHI'10*, 887. Etats-Unis: ACM Press. doi:10.1145/1753326.1753458. <http://portal.acm.org/citation.cfm?doid=1753326.1753458>.

- Riviere, J., C. Adam, S. Pesty, C. Pelachaud, N. Guiraud, D. Longin, and E. Lorini. 2011. "Expressive Multimodal Conversational Acts for SAIBA Agents." In *Intelligent Virtual Agents*, 316–323.
- Ruyter, Boris de, and Elly Pelgrim. 2007. "Ambient Assisted-living Research in Carelab." *Interactions* 14 (4) (July 1): 30. doi:10.1145/1273961.1273981.
- Sadri, Fariba. 2011. "Ambient Intelligence: A Survey." *ACM Comput. Surv.* 43 (4) (October): 36:1–36:66. doi:10.1145/1978802.1978815.
- Saint-Aimé, S., B. Le Pévédic, D. Duhaut, and others. 2009. "iGrace—Emotional Computational Model for Eml Companion Robot." *Advances in Human-Robot Interaction*.
- Saint-Aime, Sebastien, Brigitte Le Pevedic, Sabine Letellier-Zarshenas, and Dominique Duhaut. 2009. "Eml - My Emotional Cuddly Companion." In *RO-MAN 2009*, 705–710. Japon: IEEE. doi:10.1109/ROMAN.2009.5326294.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5326294>.
- Sakagami, Y., R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. 2002. "The Intelligent ASIMO: System Overview and Integration." In *International Conference on Intelligent Robots and Systems*, 3:2478–2483. Japon: IEEE. doi:10.1109/IRDS.2002.1041641.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1041641>.
- Saldien, J., K. Goris, B. Vanderborght, B. Verrelst, R. Van Ham, and D. Lefeber. 2006. "ANTY: The Development of an Intelligent Huggable Robot for Hospitalized Children." In *CLAWAR Conference*, 123–128.
- Saldien, J., K. Goris, S. Yilmazyildiz, W. Verhelst, and D. Lefeber. 2008. "On the Design of the Huggable Robot Probo." *Journal of Physical Agents* 2 (2): 3–12.
- Salichs, Miguel, Ramon Barber, Alaa Khamis, Maria Malfaz, Javier Gorostiza, Rakel Pacheco, Rafael Rivas, Ana Corrales, Elena Delgado, and David Garcia. 2006. "Maggie: A Robotic Platform for Human-Robot Social Interaction." In *Conference on Robotics, Automation and Mechatronics*, 1–7. Espagne: IEEE. doi:10.1109/RAMECH.2006.252754.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4018870>.
- San Martín, Luis Ángel, Víctor M. Peláez, Roberto González, Antonio Campos, and Vanesa Lobato. 2010. "Environmental User-preference Learning for Smart Homes: An Autonomous Approach." *Journal of Ambient Intelligence and Smart Environments* 2 (3) (January 1): 327–342. doi:10.3233/AIS-2010-0075.
- Sansonnet, Jean-Paul, David Leray, and Jean-Claude Martin. 2006. "Architecture of a Framework for Generic Assisting Conversational Agents." In *Intelligent Virtual Agents*, edited by Jonathan Gratch, Michael Young, Ruth Aylett, Daniel Ballin, and Patrick Olivier, 4133:145–156. Berlin, Heidelberg: Springer Berlin Heidelberg.
http://www.springerlink.com/index/10.1007/11821830_12.
- Scheible, J., V.H. Tuulos, and T. Ojala. 2007. "Story Mashup: Design and Evaluation of Novel Interactive Storytelling Game for Mobile and Web Users." In *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, 139–148.
- Scherer, K.R. 1984. "Les Émotions: Fonctions Et Composantes." *Cahiers De Psychologie Cognitive* 4 (1): 9–39.
- Schröder, Marc. 2010. "The SEMAINE API: Towards a Standards-Based Framework for Building Emotion-Oriented Systems." *Advances in Human-Computer Interaction* 2010: 1–21. doi:10.1155/2010/319406.
- Scoditti, Adriano, Thomas Vincent, Joelle Coutaz, Renaud Blanch, and Nadine Mandran. 2011. "TouchOver: Decoupling Positioning from Selection on Touch-based Handheld Devices." In *IHM 2011*, 1. France: ACM Press. doi:10.1145/2044354.2044362.
<http://dl.acm.org/citation.cfm?doid=2044354.2044362>.
- Shae, Zon-Yin, Belle Tseng, and WingHo Leung. 2001. "Immersive Whiteboard Collaborative System." *Annals of Software Engineering* 12 (1): 193–212.
- Shibata, T. 2004. "An Overview of Human Interactive Robots for Psychological Enrichment." *Proceedings of the IEEE* 92 (11) (November): 1749–1758. doi:10.1109/JPROC.2004.835383.

- Shinozawa, Kazuhiko, Futoshi Naya, Junji Yamato, and Kiyoshi Kogure. 2005. "Differences in Effect of Robot and Screen Agent Recommendations on Human Decision-making." *International Journal of Human-Computer Studies* 62 (2) (February): 267–279. doi:10.1016/j.ijhcs.2004.11.003.
- Shuyin Li, M. Kleinhagenbrock, J. Fritsch, B. Wrede, and C. Sagerer. 2004. "'BIRON, Let Me Show You Something': Evaluating the Interaction with a Robot Companion." In *SMC 2004*, 3:2827–2834. Allemagne: IEEE. doi:10.1109/ICSMC.2004.1400761. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1400761>.
- Sidner, C.L., C. Lee, and N. Lesh. 2003. "The Role of Dialog in Human Robot Interaction." In *International Workshop on Language Understanding and Agents for Real World Interaction*.
- Soueina, SO. 2003. "An Enneagram Based Model for Personality Based Adaptive Systems." *Proceedings of AAMASS 3*.
- Stahl, O., B. Gambäck, P. Hansen, M. Turunen, and J. Hakulinen. 2008. "A Mobile Fitness Companion."
- Stiehl, W.D., J. Lieberman, C. Breazeal, L. Basel, L. Lalla, and M. Wolf. 2005a. "Design of a Therapeutic Robotic Companion for Relational, Affective Touch." In *RO-MAN 2005*, 408–415. Etats-Unis: IEEE. doi:10.1109/ROMAN.2005.1513813. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1513813>.
- . 2005b. "The Design of the Huggable: A Therapeutic Robotic Companion for Relational, Affective Touch." In *AAAI Fall Symposium on Caring Machines: AI in Eldercare, Washington, DC*.
- Streitz, Norbert A. 2012. "From Human–Computer Interaction to Human–Environment Interaction: Ambient Intelligence and the Disappearing Computer." In *Universal Access in Ambient Intelligence Environments*, edited by Constantine Stephanidis and Michael Pieper, 4397:3–13. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed September 23. http://www.springerlink.com/index/10.1007/978-3-540-71025-7_1.
- Stückler, J., K. Gräve, J. Kläyß, S. Muszynski, M. Schreiber, O. Tischler, R. Waldukat, and S. Behnke. 2009. "Dynamaid: Towards a Personal Robot That Helps with Household Chores." In *Proceedings of RSS 2009 Workshop on Mobile Manipulation in Human Environments, Seattle (June 2009)*.
- Syrdal, D.S., K.L. Koay, M.L. Walters, and K. Dautenhahn. 2009. "The Boy-robot Should Bark!—children's Impressions of Agent Migration into Diverse Embodiments." In *Proceedings of the New Frontiers in Human-Robot Interaction, a Symposium at the AISB2009 Convention*.
- Takanori Shibata, Kazuyoshi Wada, Yousuke Ikeda, and Selma Sabanovic. 2008. "Tabulation and Analysis of Questionnaire Results of Subjective Evaluation of Seal Robot in Seven Countries." In *RO-MAN 2008*, 689–694. Allemagne: IEEE. doi:10.1109/ROMAN.2008.4600747. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4600747>.
- Talbert, N. 1997. "Toward Human-centered Systems." *IEEE Computer Graphics and Applications* 17 (4) (August): 21–28. doi:10.1109/38.595262.
- Tapus, Adriana, Cristian Tapus, and Maja Mataric. 2009. "The Role of Physical Embodiment of a Therapist Robot for Individuals with Cognitive Impairments." In *Ro-man 2009*, 103–107. Los Angeles, USA: IEEE. doi:10.1109/ROMAN.2009.5326211. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5326211>.
- Tedre, Matti. 2008. "What Should Be Automated?" *Interactions* 15 (5) (September 1): 47. doi:10.1145/1390085.1390096.
- Tojo, T., Y. Matsusaka, T. Ishii, and T. Kobayashi. 2000. "A Conversational Robot Utilizing Facial and Body Expressions." In *SMC 2000*, 2:858–863. Japon: IEEE. doi:10.1109/ICSMC.2000.885957. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=885957>.
- Toma, Daniel Mihai, Tom O'Reilly, Joaquin del Rio, Kent Headley, Antoni Manuel, Arne Broring, and Duane Edgington. 2011. "Smart Sensors for Interoperable Smart Ocean Environment." In , 1–4. Espagne: IEEE. doi:10.1109/Oceans-Spain.2011.6003654. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6003654>.

- Torres-Jara, E., L. Natale, and P. Fitzpatrick. 2005. "Tapping into Touch": 22–24.
- Turner, Kenneth J. 2011. "Flexible Management of Smart Homes." *Journal of Ambient Intelligence and Smart Environments* 3 (2) (January 1): 83–109. doi:10.3233/AIS-2011-0100.
- Van Breemen, A.J.N. 2005. "iCat: Experimenting with Animabotics." In *Proceedings, Aisb 2005 Creative Robotics Symposium*.
- Van Breemen, AJN. 2004. "Bringing Robots to Life: Applying Principles of Animation to Robots." In *Proceedings of the International Conference for Human-computer Interaction, CHI2004, Vienna, Austria*.
- Vanden Abeele, Veronika A., and Veerle Van Rompaey. 2006. "Introducing Human-centered Research to Game Design : Designing Game Concepts for and with Senior Citizens." In *CHI EA'06*, 1469. Etats-Unis: ACM Press. doi:10.1145/1125451.1125721. <http://dl.acm.org/citation.cfm?doid=1125451.1125721>.
- Velásquez, J.D. 1998. "When Robots Weep: Emotional Memories and Decision-making." In *AAAI 98*, 70–75. Etats-Unis.
- Verstockt, S., D. Decoo, D. Van Nieuwenhuyse, F. De Pauw, and R. Van de Walle. 2009. "Assistive Smartphone for People with Special Needs : The Personal Social Assistant." In *HSI'09*, 331–337. Italie: IEEE. doi:10.1109/HSI.2009.5091001. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5091001>.
- Villaroman, Norman, Dale Rowe, and Bret Swan. 2011. "Teaching Natural User Interaction Using OpenNI and the Microsoft Kinect Sensor." In *SIGITE'11*, 227. Etats-Unis: ACM Press. doi:10.1145/2047594.2047654. <http://dl.acm.org/citation.cfm?doid=2047594.2047654>.
- Vo, Dong-Bach, Gilles Bailly, Eric Lecolinet, and Yves Guiard. 2011. "Un Espace De Caractérisation De La Télécommande Dans Le Contexte De La Télévision Interactive." In *IHM*, 1. France: ACM Press. doi:10.1145/2044354.2044375. <http://dl.acm.org/citation.cfm?doid=2044354.2044375>.
- Wada, K., T. Shibata, T. Saito, and K. Tanie. 2004. "Effects of Robot-assisted Activity for Elderly People and Nurses at a Day Service Center." *Proceedings of the IEEE* 92 (11) (November): 1780–1788. doi:10.1109/JPROC.2004.835378.
- Wei, Zhaozhi, Luo Li, Jinmao Zhu, and Zhou Chi. 2010. "Games for Service Science Education." In *ICSS*, 237–241. Chine: IEEE. doi:10.1109/ICSS.2010.20. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5494282>.
- Weingarten, Florian, Marco Blumendorf, and Sahin Albayrak. 2010. "Towards Multimodal Interaction in Smart Home Environments: The Home Operating System." In *DIS'10*, 430. Etats-Unis: ACM Press. doi:10.1145/1858171.1858255. <http://portal.acm.org/citation.cfm?doid=1858171.1858255>.
- Weiser, M. 1991. "The Computer for the 21st Century." *Scientific American* 265 (3): 94–104.
- Wistort, Ryan, and Cynthia Breazeal. 2009. "TOFU: a Socially Expressive Robot Character for Child Interaction." In *The 8th International Conference on Interaction Design and Children*, 292–293. Italie: ACM Press. doi:10.1145/1551788.1551862. <http://portal.acm.org/citation.cfm?doid=1551788.1551862>.
- Wu, Ya-Huei, Christine Fassett, and Anne-Sophie Rigaud. 2012. "Designing Robots for the Elderly: Appearance Issue and Beyond." *Archives of Gerontology and Geriatrics* 54 (1) (January): 121–126. doi:10.1016/j.archger.2011.02.003.
- Wu1, Ya-Huei, Mohamed Chetouani, Victoria Cristancho-Lacroix, Jade Le Maître, Céline Jost, Brigitte Le Pevedic, Dominique Duhaut, Consuelo Granata, and Anne-Sophie Rigaud. 2011. "ROBADOM: The Impact of a Domestic Robot on Psychological and Cognitive State of the Elderly with Mild Cognitive Impairment." In -, -. France. <http://hal.archives-ouvertes.fr/hal-00661797>.
- Yamato, J., K. Shinozawa, F. Naya, and K. Kogure. 2001. "Evaluation of Communication with Robot and Agent: Are Robots Better Social Actors Than Agents." In *INTERACT'01*, 9–13. Japon.
- Yoshiike, Yuta, P. Ravindra De Silva, and Michio Okada. 2010. "Cues for Sociable PC: Coordinate and Synchronize Its Cues Based on User Attention and Activities on Display." In *HRI*, 135–136.

- Japon: IEEE. doi:10.1109/HRI.2010.5453229.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5453229>.
- Yoshikawa, H. 1999. "A Ruminant on Human-centered Automation World: Multiagent System, Human Interface and People's Lifestyle." In *SMC*, 3:669–674. Japon: IEEE. doi:10.1109/ICSMC.1999.823308.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=823308>.
- Zapirain, B G, A M Zorrilla, and S Larra. 2010. "Psycho-stimulation for Elderly People Using Puzzle Game." In *Games Innovations Conference*, 1–8. Espagne: IEEE. doi:10.1109/ICEGIC.2010.5716903.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5716903>.

Publications

2013

C. Jost, B. Le Pévédic, D. Duhaut, Etude de l'impact du couplage geste et parole sur un robot, Intercompréhensions comparées dans trois types d'interactions : Homme-Homme, Animal-Homme-Machine et Homme-Machine E.M.E. Editions 2013

Y-H. Wu, J. Wrobel, V. Cristancho-Lacroix, H. Kerhervé, M. Chetouani, D. Duhaut, B. Le Pevedic, C. Jost, V. Dupourque, M. Ghrissi, A-S. Rigaud, Le projet Robadom : conception d'un robot d'assistance pour les personnes âgées, Revue de gériatrie (article accepté et en cours de publication)

J. Wrobel, Ya-Huei Wu, Hélène Kerhervé, Laila Kamali, Anne-Sophie Rigaud, Céline Jost, Brigitte Le Pévédic, Dominique Duhaut, Effect of agent embodiment on the elder user enjoyment of a game, in ACHI 2013, The Sixth International Conference on Advances in Computer-Human Interactions

2012

C. Jost, B. Le Pévédic, D. Duhaut. Une interaction la plus riche possible et à moindre coût. WACAI 2012 - Workshop Affect, Compagnon Artificiel, Interaction Grenoble - 15 et 16 Novembre 2012

C. Jost, V. André, A. Lemasson, B. Le Pévédic, M. Hausberger, D. Duhaut, Ethological evaluation of Human-Robot Interaction: are children more efficient and motivated with computer, virtual agent or robots?, IEEE ROBIO International Conference on Robotics and Biomimetics December 11-14, 2012, in Guangzhou, China

C. Jost, B. Le Pévédic, D. Duhaut, Robot is best to play with human!, In IEEE RO-MAN 2012, 21th International Symposium on Robot and Human Interactive Communication, Versailles, september 2012

C. Jost, B. Le Pévédic, D. Duhaut, ArCo : an architecture for children to control a set of robots, In IEEE RO-MAN 2012, 21th International Symposium on Robot and Human Interactive Communication, Versailles, september 2012

C. Jost, B. Le Pévédic, D. Duhaut : Study of the Importance of Adequacy to Robot Verbal and Non Verbal Communication in Human-Robot Interaction, In ISR 2012 International Symposium on Robotics, Taipei, Taiwan, August 2012.

C. Jost, B. Le Pévédic, D. Duhaut : Creating Interaction Scenarios With a New Graphical User Interface, in IIHCI 2012, International Workshop on Intelligent Interfaces for Human-Computer Interaction. The proceedings of the workshop will be published by IEEE CSP and are part of CISIS-2012 conference proceedings. Palermo, Italy, July 2012.

Y-H. Wu, V. Cristancho-Lacroix, E. Gabillet, J. Le Maitre, M. Chetouani C. Jost, B. Le Pevedic, D. Duhaut, A.S. Rigaud : Perception of affects from non-facial expressions of the robot Nabaztag, in ISG*ISARC2012, ISG, International Society for Gerontechnology/ISARC, International Symposium of Automation and Robotics in Construction, the Netherlands, june 2012.

2011

Y-H. Wu, M. Chetouani, V. Cristancho-Lacroix, J. Le Maitre, C. Jost, B. Le Pevedic, D. Duhaut, C. Granata, A.S. Rigaud ROBAdOM: The Impact of a Domestic Robot on Psychological and Cognitive State of the Elderly with Mild Cognitive Impairment in 5th CRI (Companion Robotics Institute) Workshop AAL User-Centric Companion Robotics Experimentoria, Supporting Socio-ethically Intelligent Assistive Technologies Adoption, 2011.

Y-H. Wu, V. Cristancho-Lacroix, M. Chetouani, J. Le Maitre, C. Jost, B. Le Pevedic, D. Duhaut, C. Granata, A.S. Rigaud Robadom : un robot d'assistance pour les personnes âgées présentant des troubles cognitifs légers : perspectives d'utilisateurs et conception du système (poster + oral) SFTAG 2011, Congrès de la Société Française des Technologies pour l'Autonomie et Geronotechnologies, juin 2011.

2010

M. Chetouani, Y-H. Wu, C. Jost, B. Le Pévédic, C. Fassert, V. Cristancho-Lacroix, S. Lassaille, C. Granata, A. Tapus, D. Duhaut, A.S. Rigaud Cognitive Services for Elderly People: The ROBAdOM project in ECCE workshop Robots that care 2010, The European Conference on Cognitive Ergonomics, the Netherlands, Delft, August 2010

Résumé

La progression rapide de la technologie a donné lieu à un panorama riche et varié de dispositifs numériques : caméra, téléphones mobiles, GPS, tablettes tactiles, liseuses numériques, robots, télévisions, éléments de domotique... La majorité de ces appareils sont aujourd'hui connectés à Internet. Et en plus de leurs fonctionnalités principales, ils permettent à leur propriétaire de rester en contact avec « le monde » à l'aide de logiciels de communication, de personnages virtuels ou de robots. Tous ces dispositifs numériques fonctionnent indépendamment les uns des autres.

La question qui se pose est de savoir si ces dispositifs numériques doivent être coordonnés afin de partager certaines informations et effectuer certaines actions ensembles. Cette collaboration entre les dispositifs numériques est gérée par le Compagnon Artificiel qui est en contact permanent avec un utilisateur par les biais des divers dispositifs numériques.

Une architecture modulaire ArCo permettant de mettre en place un Compagnon Artificiel a été réalisée dans le cadre de ce travail. Les dispositifs numériques sont gérés par des modules spécifiques, créés grâce à un cadre MICE (Machines Interaction Control in their Environment). L'utilisateur final du système peut programmer des scénarii d'interaction, qui indiquent les actions que doivent effectuer les dispositifs numériques, grâce à une interface de programmation visuelle AmbiProg. Chaque scénario est interprété par un module AmbiLive. Les conflits d'accès aux dispositifs numériques sont gérés par un module AmbiCop.

Un ensemble d'évaluations a permis de valider expérimentalement l'architecture ArCo et de répondre à des problématiques d'interaction homme-machine.

Abstract

The rapid advance in technology allowed manufacturers to offer a varied panorama of digital devices: camera, mobile phones, GPS, tactile tablets, digital readers, robots, televisions, home automation elements... Nowadays, most of these devices are connected to the Internet. Thus, their users can not only take benefit from their functionalities but also be in touch with “everyone” thanks to communicative software, virtual characters or robots.

It raises a main question: do these digital devices have to be coordinated in order to share information and achieve tasks together? This collaboration between digital devices could be managed by an Artificial Companion which could permanently be in touch with a user throughout these devices.

We realized a modular architecture called ArCo which allows the creation of an Artificial Companion. Some specialized modules can manage the digital devices. These modules are implemented with a framework called MICE (Machines Interaction Control in their Environment). The end-user can create interaction scenarios with a visual programming interface called AmbiProg. The scenarios contain actions that digital devices have to achieve. Each scenario is interpreted by the AmbiLive module. Digital devices access conflicts are managed by the AmbiCop module.

Some experiments validated the ArCo architecture from an experimental point of view. They also allowed studying some Human-Machine Interaction problems.



THESE / UNIVERSITE DE BRETAGNE SUD
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITE DE BRETAGNE SUD

Mention : Science et Technologies de l'information et de la Communication
Ecole doctorale SICMA

présentée par

Céline Jost

Laboratoire des Sciences Techniques de l'Information, de la
Communication et de la Connaissance UMR3192

ArCo : une **A**rchitecture informatique pour un **C**ompagnon
Artificiel en interaction avec un utilisateur

ANNEXES

Table des annexes

A. ASSURER UNE UTILISATION A LONG TERME D'UNE TECHNOLOGIE	9
A. RECOMMANDATIONS DE L'ERGONOMIE DES LOGICIELS.....	9
B. NORMES DE L'ERGONOMIE DES LOGICIELS.....	9
C. CRITERES ERGONOMIQUES POUR LA PERSUASION TECHNOLOGIQUE.....	9
D. TRIPTYQUE : UTILITE, UTILISABILITE, ACCEPTABILITE.....	10
E. SYMBIOSE HUMAIN-TECHNOLOGIE.....	13
B. FRISE CHRONOLOGIQUE DE QUELQUES ROBOTS PRINCIPAUX.....	15
C. DESCRIPTION DES MESSAGES RESERVES.....	17
A. MICE_1 ET MICE_6 : CONNEXION, DECONNEXION ET SUPPRESSION D'UN MODULE A MIIME.....	17
B. MICE_2 ET MICE_3 : NOTIFICATION DES IDENTIFICATIONS.....	19
C. MICE_4 ET MICE_5 : ROUTAGE DES MESSAGES.....	20
D. MICE_7 ET MICE_8 : ACCES AUX DESCRIPTIONS DES ENTITES AGISSANTES.....	21
E. MICE_9 ET MICE_10 : LISTE DES MODULES CONNECTES.....	23
F. MICE_13 : LES ACTIONS.....	24
G. MICE_14 : FIN DES ACTIONS.....	25
H. MICE_11 ET MICE_12 : NOTIFICATION DES ACTIONS.....	26
I. MICE_15 : LES PERCEPTIONS.....	27
D. AMBIPROG : OUTIL POUR LA CREATION DE SCENARIO.....	29
A. PRESENTATION GENERALE.....	29
B. PRESENTATION DU LANGAGE VISUEL.....	32
C. AMBIPROG : UNE INTERFACE COMPLETEMENT PERSONNALISABLE PAR FICHIERS XML.....	42
E. ANALYSE SYNTAXIQUE DU LANGAGE AMBIPROG.....	45
F. INTERPRETATION DE L'ARBRE SYNTAXIQUE DU LANGAGE AMBIPROG.....	51
G. BIBLIOTHEQUE AMBILIB.....	57
A. PAQUETAGE UBS.ROBADM.INTERPRETER.STRUCTURE.....	58
B. PAQUETAGE UBS.ROBADM.INTERPRETER.PARSING.....	58
C. PAQUETAGE UBS.ROBADM.INTERPRETER.INTERPRET.....	58
H. ECRITURE D'UN NOUVEAU MODULE POUR ARCO.....	61
A. BIBLIOTHEQUE MICE DISPONIBLE.....	61
B. ECRITURE D'UN MODULE DE PERCEPTION.....	69
C. ECRITURE D'UN MODULE D'ACTION.....	70
D. EXEMPLE : MODULE MICE ASSOCIE A GRETA.....	72
I. CONFIGURATION DE STIMCARDS.....	75
A. CONFIGURATION DES CARTES DE JEU.....	75
B. CONFIGURATION DE L'INTERFACE.....	77
J. EVALUATION SUR LA GESTUELLE D'UN ROBOT.....	79
A. IMPORTANCE DE L'ADEQUATION ENTRE LE DISCOURS ET LES GESTES.....	79
B. CREDIBILITE ET SINCERITE D'UN ROBOT MALGRE QUELQUES INCOHERENCES DANS LA COMMUNICATION.....	86
K. EVALUATION D'AMBIPROG.....	91
A. EVALUATION DE LA PRISE EN MAIN D'AMBIPROG.....	91
B. EVALUATION DE L'APPRENTISSAGE DE LA CREATION DE SCENARI.....	96
L. EVALUATION DE STIMCARDS (ENFANTS ET PERSONNAGES AGEES).....	105
A. EVALUATION DE STIMCARDS AUPRES D'ENFANTS.....	105
B. EVALUATION DE STIMCARDS AUPRES DE PERSONNES AGEES.....	107

M. EVALUATION DU MEILLEUR PARTENAIRE NUMERIQUE	111
A. PARTICIPANTS	111
B. METHODE	111
C. COLLECTE DE DONNEES ET ANALYSES	114
D. RESULTATS	115
E. DISCUSSION	118
N. EVALUATION D'AMBIPOP	119
A. LES SCENARIOS	119
B. DEROULEMENT DE L'EXPERIMENTATION	120
C. CONCLUSION	120
O. QUESTIONNAIRE DE L'ETUDE SUR LA GESTUELLE DU ROBOT	121

Table des Figures

FIGURE 1 : LE MODELE DE NIELSEN	10
FIGURE 2 : MODELE DE DILLON ET MORRIS	11
FIGURE 3 : MODELE DE PESTY ET DUHAUT	11
FIGURE 4 : MODELE DE L'ACCEPTATION DES TECHNOLOGIES DE DAVIS	12
FIGURE 5 : MODELE DE DISCONFIRMATION DES ATTENTES	13
FIGURE 6 : FRISE CHRONOLOGIQUE DE QUELQUES ROBOTS PRINCIPAUX	15
FIGURE 7 : ORGANISATION D'AMBIPOP	29
FIGURE 8 : LE MENU D'AMBIPOP	29
FIGURE 9 : MENU CONTEXTUEL D'AMBIPOP	30
FIGURE 10 : ZONE DE CONSTRUCTION DU SCENARIO	30
FIGURE 11 : ZONE D'ELEMENTS DE SCENARIO	31
FIGURE 12 : SELECTION D'UN ELEMENT DE SCENARIO	32
FIGURE 13 : GLISSER/DEPOSER UN ELEMENT DE SCENARIO	33
FIGURE 14 : SAISIR UN PARAMETRE D'UNE FONCTION	33
FIGURE 15 : UN ELEMENT DE SCENARIO DANS LA ZONE DE CONSTRUCTION	33
FIGURE 16 : AIDE VISUELLE INDIQUANT OU DEPOSER LES ELEMENTS DE SCENARIO	34
FIGURE 17 : L'ELEMENT "SI ALORS" ET "SI ALORS SINON" D'AMBIPOP	34
FIGURE 18 : L'ELEMENT "REPETER" D'AMBIPOP	35
FIGURE 19 : L'ELEMENT "TANT QUE" DE AMBIPOP	35
FIGURE 20 : L'ELEMENT PARALLELISME D'AMBIPOP	35
FIGURE 21 : L'ELEMENT BRANCHE D'AMBIPOP	36
FIGURE 22 : L'ELEMENT ACTIONS SYNCHRONISEES D'AMBIPOP	36
FIGURE 23 : L'ELEMENT EVENEMENT D'AMBIPOP	37
FIGURE 24 : L'ELEMENT ATTENDRE D'AMBIPOP	37
FIGURE 25 : L'ELEMENT BREAK D'AMBIPOP	37
FIGURE 26 : PROCEDURE DE CREATION DE MACRO - PROGRAMME A TRANSFORMER	38
FIGURE 27 : PROCEDURE DE CREATION DE MACRO – TRANSFORMATION	38
FIGURE 28 : PROCEDURE DE CREATION DE MACRO – ENREGISTREMENT	39
FIGURE 29 : PROCEDURE DE CREATION DE MACRO – RESULTAT	39
FIGURE 30 : CREATION D'UNE PREMIERE PERCEPTION SYMBOLIQUE	40
FIGURE 31 : CREATION D'UNE DEUXIEME PERCEPTION SYMBOLIQUE	41
FIGURE 32 : ENREGISTREMENT DES PERCEPTIONS SYMBOLIQUES	41
FIGURE 33 : AFFICHAGE DES PERCEPTIONS SYMBOLIQUES DANS AMBIPOP	42
FIGURE 34 : PERSONNALISATION D'UNE ACTION DANS AMBIPOP	43
FIGURE 35 : PREMIER ELEMENT DE L'ARBRE SYNTAXIQUE	45
FIGURE 36 : ARBRE SYNTAXIQUE DE L'ELEMENT ACTION	46
FIGURE 37 : ARBRE SYNTAXIQUE DE L'ELEMENT CONDITIONNEL	46
FIGURE 38 : ARBRE SYNTAXIQUE DE L'ELEMENT CONDITIONNEL AVEC ET ET OU	47
FIGURE 39 : ARBRE SYNTAXIQUE DE L'ELEMENT REPETER	47
FIGURE 40 : ARBRE SYNTAXIQUE DE L'ELEMENT TANTQUE	47
FIGURE 41 : ARBRE SYNTAXIQUE DE L'ELEMENT PARALLELISME	48

FIGURE 42 : ARBRE SYNTAXIQUE DE L'ELEMENT ACTIONS SYNCHRONISEES	48
FIGURE 43 : ARBRE SYNTAXIQUE DE L'ELEMENT EVENEMENT	49
FIGURE 44 : ARBRE SYNTAXIQUE DE L'ELEMENT ATTENDRE.....	49
FIGURE 45 : ARBRE SYNTAXIQUE DE L'ELEMENT BREAK.....	49
FIGURE 46 : PROCEDURE D'INTERPRETATION.....	51
FIGURE 47 : CHEMIN D'INTERPRETATION	52
FIGURE 48 : BIBLIOTHEQUE DE L'INTERPRETEUR : ORGANISATION GLOBALE.....	57
FIGURE 49 : BIBLIOTHEQUE DE L'INTERPRETEUR : INTERPRETEUR ABSTRAIT.....	59
FIGURE 50 : BIBLIOTHEQUE MICE - PACKAGE MICE	62
FIGURE 51 : BIBLIOTHEQUE MICE - PACKAGE MICE.CLIENT	63
FIGURE 52 : BIBLIOTHEQUE MICE - PACKAGE MICE.XML	64
FIGURE 53 : BIBLIOTHEQUE MICE - PACKAGE MICE.RUNTIME.....	65
FIGURE 54 : BIBLIOTHEQUE MICE - PACKAGE MICE.PERCEPTION.....	66
FIGURE 55 : BIBLIOTHEQUE MICE - PACKAGE MICE.ACTION	66
FIGURE 56 : BIBLIOTHEQUE MICE - PACKAGE MICE.XML.ACTUATOR	67
FIGURE 57 : BIBLIOTHEQUE MICE - PACKAGE MICE.XML.SENSOR	68
FIGURE 58 : BIBLIOTHEQUE MICE - PACKAGE MICE.XML.MODULES	68
FIGURE 59 : CADRE EXPERIMENTAL DE LA PREMIERE ETUDE	80
FIGURE 60 : MOUVEMENT 1 REPRESENTANT UNE GESTUELLE D'EXCUSE	81
FIGURE 61 : MOUVEMENT 2 REPRESENTANT UNE GESTUELLE NEUTRE	81
FIGURE 62 : MOUVEMENT 3 REPRESENTANT UNE GESTUELLE DE JOIE	82
FIGURE 63 : RESULTATS SUR LA CREDIBILITE ET LA SINCERITE DES MOUVEMENTS DU ROBOT EN FONCTION DES SCENARII A ET B	83
FIGURE 64 : CADRE EXPERIMENTAL DE LA DEUXIEME ETUDE	87
FIGURE 65 : RESULTAT SUR L'ACCEPTABILITE DU ROBOT AU COURS D'UNE INTERACTION MOYENNE.....	89
FIGURE 66 : ORGANISATION D'AMBIPROG PRESENTEE LORS DE LA TROISIEME ETUDE	92
FIGURE 67 : TEMPS POUR REUSSIR LA TACHE EN SECONDE EN RELATION AVEC L'AGE DES PARTICIPANTS.....	94
FIGURE 68 : REPONSES AUX QUESTIONS DE LA TROISIEME ETUDE.....	94
FIGURE 69 : AIDES DONNEES ET ERREURS OBSERVEES.....	95
FIGURE 70 : CADRE EXPERIMENTAL DE LA QUATRIEME ETUDE	97
FIGURE 71 : NABETTE ET NABI	98
FIGURE 72 : ETUDE 4 - EXPLICATIONS AVANT L'EXERCICE 01.....	99
FIGURE 73 : ETUDE 4 - EXERCICE 01.....	99
FIGURE 74 : ETUDE 4 - EXERCICE 02.....	99
FIGURE 75 : ETUDE 4 - EXPLICATIONS AVANT L'EXERCICE 03.....	99
FIGURE 76 : ETUDE 4 - EXERCICE 03.....	99
FIGURE 77 : ETUDE 4 - EXERCICE 04.....	100
FIGURE 78 : ETUDE 4 - EXERCICE 05.....	100
FIGURE 79 : ETUDE 4 - EXPLICATIONS AVANT L'EXERCICE 06.....	100
FIGURE 80 : ETUDE 4 - EXERCICE 06.....	100
FIGURE 81 : COMPARAISON DE L'EFFET DES INSTRUCTIONS ECRITES ET ORALES.	102
FIGURE 82 : COMPARAISON ENTRE LA REUSSITE AUX EXERCICES EN FONCTION DU TYPE D'INSTRUCTIONS	103
FIGURE 83 : REPONSES AUX QUESTIONS A, C, D, E ET F.....	103
FIGURE 84 : REPONSES AUX QUESTIONS G ET H.	104
FIGURE 85 : CADRE EXPERIMENTAL DE LA CINQUIEME ETUDE	106
FIGURE 86 : RESULTATS DE L'ETUDE DE L'ACCEPTABILITE DE STIMCARDS PAR LES ENFANTS.	107
FIGURE 87 : CADRE EXPERIMENTAL DE LA SIXIEME ETUDE	108
FIGURE 88 : RESULTATS DE L'ETUDE DE L'ACCEPTABILITE DE STIMCARDS PAR LES PERSONNES AGEES.....	110
FIGURE 89 : CADRE EXPERIMENTAL DE LA SEPTIEME ETUDE	111
FIGURE 90 : INTERFACE DE REPONSE PROPOSEE SUR LA TABLETTE TACTILE	112
FIGURE 91 : LES QUATRE CONDITIONS DE LA SEPTIEME ETUDE.....	112
FIGURE 92 : SCENARIO D'INTERACTION DE STIMCARDS LORS DE LA SEPTIEME ETUDE	114
FIGURE 93 : RESULTAT DU COMPAGNON CHOISI PAR LES ENFANTS	117
FIGURE 94 : RESULTAT DU COMPAGNON CHOISI EN FONCTION DE L'ORDRE DE PREFERENCE	118
FIGURE 95 : ORGANISATION DE L'EVALUATION D'AMBI COP	119

Table des Tableaux

TABLEAU 1 : CONTENU DES BRANCHES DE LA ZONE D'ELEMENTS DE SCENARIO.....	31
TABLEAU 2 : DECOMPOSITION D'UNE ACTION ET D'UNE PERCEPTION.....	45
TABLEAU 3 : DECOUPAGE D'UN SCENARIO EN BLOC.....	58
TABLEAU 4 : QUESTIONNAIRE DE LA PREMIERE ETUDE	82
TABLEAU 5 : RESULTATS OBTENUS POUR LE SCENARIO A DE LA PREMIERE ETUDE.....	84
TABLEAU 6 : RESULTATS OBTENUS POUR LE SCENARIO B DE LA PREMIERE ETUDE.....	84
TABLEAU 7 : COMPARAISON DE LA SINCERITE ET DE LA CREDIBILITE D'UN AGENT VIRTUEL ET DU ROBOT	85
TABLEAU 8 : LES QUATRE DIALOGUES DE LA DEUXIEME ETUDE.....	87
TABLEAU 9 : QUESTIONNAIRE DE L'ETUDE 5	106
TABLEAU 10 : QUESTIONNAIRE DE L'ETUDE 6	109
TABLEAU 11 : QUESTIONNAIRE DE L'ETUDE 7	114
TABLEAU 12 : SCORE OBTENU PAR LES COMPAGNONS AU QUESTIONNAIRE DE L'ETUDE 7	117

Introduction

Ce document contient les annexes du manuscrit « ArCo : une architecture informatique pour une interaction avec un Compagnon Artificiel ». Il contient 15 annexes réparties de la façon suivante :

- A. Cette annexe propose une introduction envers l'acceptabilité des systèmes informatiques pour compléter les contraintes générales du compagnon artificiel (cf. Chapitre I du document principal).
- B. Cette annexe présente une frise chronologique des principaux robots utilisés dans des évaluations d'interaction homme-machine depuis le tout premier robot compagnon : Paro de T. Shibata.
- C. Cette annexe présente les messages réservés du protocole de communication de MICE. C'est le détail du Tableau 2 du document principal.
- D. Cette annexe présente AmbiProg, l'interface de programmation des scénarii de perceptions symboliques et d'interaction d'ArCo. Cette annexe est un complément au Chapitre III.D du document principal.
- E. Cette annexe présente l'analyse syntaxique effectuée par l'interpréteur AmbiLive. Cette annexe est un complément au Chapitre III.E du document principal.
- F. Cette annexe présente le processus d'interprétation de l'arbre syntaxique généré par l'analyse syntaxique. Cette annexe est un complément au Chapitre III.D du document principal.
- G. Cette annexe présente la bibliothèque AmbiLib mise à disposition de tous les concepteurs pour créer un interpréteur, du langage visuel d'AmbiProg, personnalisé.
- H. Cette annexe présente la bibliothèque MICE mise à disposition de tous les concepteurs pour créer des modules. Il donne également un exemple de création d'un module d'action, d'un module de perception et du module utilisé pour associer Greta au compagnon artificiel. Cette annexe est un complément au Chapitre III du document principal.
- I. Cette annexe présente la façon dont StimCards peut être configuré.
- J. Cette annexe présente les détails de l'évaluation du « partenaire idéal » qui se trouve au Chapitre V.A du document principal.
- K. Cette annexe présente les détails de l'évaluation d'AmbiProg qui se trouve au Chapitre V.B du document principal.
- L. Cette annexe présente les détails de l'évaluation de StimCards (auprès des enfants et des personnes âgées) qui se trouve au Chapitre V.C du document principal.
- M. Cette annexe présente les détails de l'évaluation du « meilleur » partenaire numérique pour StimCards qui se trouve au Chapitre V.D du document principal.
- N. Cette annexe présente les détails de l'évaluation d'AmbiCop qui se trouve au Chapitre V.E du document principal.
- O. Cette annexe présente le questionnaire de la deuxième étude réalisée lors de l'évaluation de la gestuelle du robot

A. Assurer une utilisation à long terme d'une technologie

Lors de la commercialisation d'une nouvelle technologie, les personnes cibles sont au cœur des préoccupations. Si celles-ci jugent un nouveau produit négativement, alors il n'aura aucun succès. Cela peut avoir des conséquences catastrophiques pour l'entreprise qui l'a conçue, comme un énorme déficit. Pour éviter d'atteindre ces situations extrêmes, les concepteurs étudient les usages que pourront avoir les utilisateurs avec ces produits. Ils étudient également les caractères intrinsèques nécessaires à cette technologie pour être jugée positivement. Ce jugement positif est recherché pour favoriser une utilisation à long terme.

A. Recommandations de l'ergonomie des logiciels

Le domaine informatique a connu une grosse révolution lors de la création des interfaces graphiques. L'interaction homme-machine a pris une dimension plus vaste. Et l'ergonomie des logiciels a commencé à fournir un ensemble de recommandations pour la conception des interfaces. Un certain nombre de recommandations concernent l'interaction physique entre l'utilisateur et son système. Le choix des dispositifs d'interaction se fait en fonction des tâches à réaliser. D'autres recommandations s'attachent à présenter les aspects ergonomiques des périphériques de sorties d'informations en fonction de la tâche de l'utilisateur. Les périphériques peuvent être par exemple l'écran, l'imprimante ou encore la restitution en trois dimensions. Il existe un troisième type de recommandations qui s'intéresse à tous les mots et les phrases utilisées par l'ordinateur. Beaucoup d'autres recommandations décrivent les structures des logiciels pour permettre les meilleures interactions possibles. Le nombre important de recommandations pose problème car il est difficile de toutes les mettre en œuvre. Et pourtant, certaines tentatives de classification des recommandations ont permis de déterminer des critères de l'utilisabilité des logiciels.

B. Normes de l'ergonomie des logiciels

Les logiciels sont soumis à des contraintes encore plus fortes que les recommandations. Ce sont les normes ISO. La norme concernant l'ergonomie de l'interaction homme-machine est ISO 9241. Il existe un ensemble de normes décrivant comment doivent être les logiciels, dans les moindres détails : boîte de dialogue, menus, formulaires, présentation de l'information, cycle de conception centré utilisateur, évaluations, utilisabilité. L'utilisabilité, norme ISO 9241-11, est définie comme « le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ». L'efficacité indique que le produit permet à ses utilisateurs d'atteindre le résultat prévu. L'efficience suggère que l'utilisateur atteint le résultat avec un effort moindre ou avec un temps minimal. La satisfaction est le confort et l'évaluation subjective de l'interaction pour l'utilisateur. Toutes ces recommandations et toutes ces normes ont pour cible l'ordinateur.

C. Critères ergonomiques pour la persuasion technologique

Dans un autre registre, Nemery et Brangier [159] ont fourni une grille de critères ergonomiques pour l'appréciation de la Persuasion Technologique. Ces critères permettent de juger des logiciels qui tentent d'engager l'utilisateur à long terme. Cette grille est décomposée en deux types de critères : critères statiques et dynamiques. Les critères statiques sont :

- la crédibilité : désigne la capacité de l'interface à inspirer confiance et à montrer la véracité de ses informations.

- La **privac  ** : d  signe le respect des donn  es personnelles, la pr  servation de l'int  grit   de la personne et la s  r  t   de l'interaction.
- La **personnalisation** : d  signe la capacit   de l'interface    s'adapter    l'utilisateur avec lequel elle interagit.
- L'**attractivit  ** : d  signe l'utilisation de l'esth  tique pour capter l'attention de l'utilisateur et le soutenir dans l'interaction.

Les crit  res dynamiques sont :

- La **sollicitation** : repr  sente la capacit   du logiciel    faire accomplir une t  che    l'utilisateur.
- L'**accompagnement initial** : d  signe les   l  ments qui permettent la premi  re initiative de l'utilisateur, qui pense agir sans contrainte.
- L'**engagement** : d  signer le fait de continuer    impliquer l'utilisateur pour maintenir l'interaction.
- L'**emprise** : marque une volont   de contr  ler l'utilisateur.    ce stade, son implication est totale et il risque de d  velopper une d  pendance    l'interface.

D. Triptyque : utilit  , utilisabilit  , acceptabilit  

Ce concept d'utilisabilit   est commun    d'autres courants de pens  es qui d  finissent un triptyque ins  parable : utilit  , utilisabilit  , acceptabilit  . Selon Larousse, l'utilit   est l'aptitude d'un bien    satisfaire un besoin ou    cr  er les conditions favorables    cette satisfaction. Et l'acceptabilit   est le caract  re de quelque chose qui est plus ou moins tol  rable.

Le mod  le de Tricot et al (2003) [160] propose un mod  le relationnel indiquant que « toute relation formellement possible entre les trois dimensions (utilit  , utilisabilit   et acceptabilit  ) peut exister. Ainsi, ils montrent qu'il est possible de d  terminer logiquement quelle dimension doit   tre am  lior  e lorsqu'un utilisateur se d  tourne d'une technologie.

En 1993, Jakob Nielsen (expert en ergonomie informatique) d  finit l'acceptabilit   des syst  mes. Celle-ci se d  compose en acceptabilit   sociale et acceptabilit   pratique (voir Figure 1). Ce mod  le met en lumi  re utilit   et utilisabilit   en tant que composants de l'acceptabilit  . Il pr  sente plus l'acceptabilit   pratique que l'acceptabilit   sociale. Le r  le de l'homme dans l'acceptabilit   du syst  me n'est pas clairement d  fini.

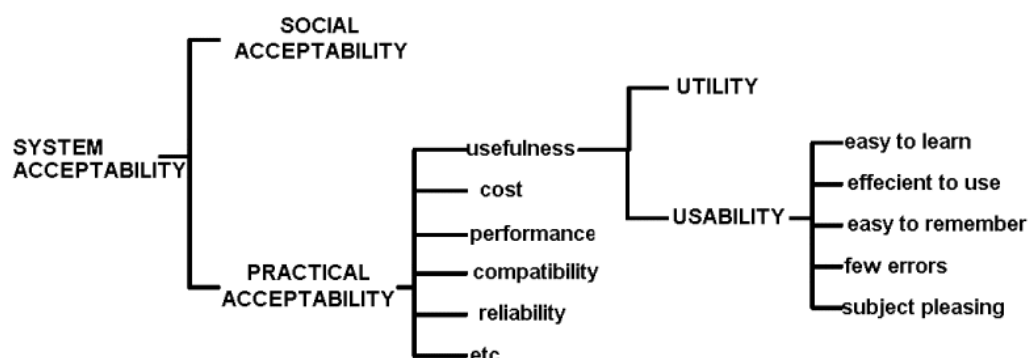


Figure 1 : Le mod  le de Nielsen

En 1996, Dillon et Morris (voir Figure 2) propose une approche d'avantage centr  e sur l'utilisateur. L'utilit   et l'utilisabilit   serait le r  sultat de la perception de l'utilisateur. Une personne peut penser qu'un outil sera facile    utiliser mais peu utile alors qu'en r  alit  , c'est le contraire. La

perception de l'utilisateur sur l'utilité et sur l'utilisabilité détermine l'acceptabilité de la technologie et ainsi l'usage qu'en fera l'utilisateur.

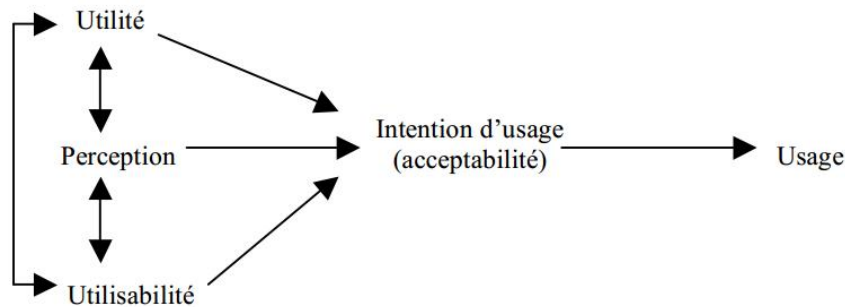


Figure 2 : Modèle de Dillon et Morris

En 2011, Pesty et Duhaut [161] propose un modèle d'acceptabilité basé sur trois approches dont le modèle d'acceptabilité de Nielsen (voir Figure 3). Ce modèle montre une description beaucoup plus détaillée de l'acceptabilité sociale, tout en conservant l'acceptabilité pratique de Nielsen. Ce modèle met l'accent sur la crédibilité du système. Et pour y parvenir, la personnalité, les émotions et l'historique de l'humain sont pris en compte.

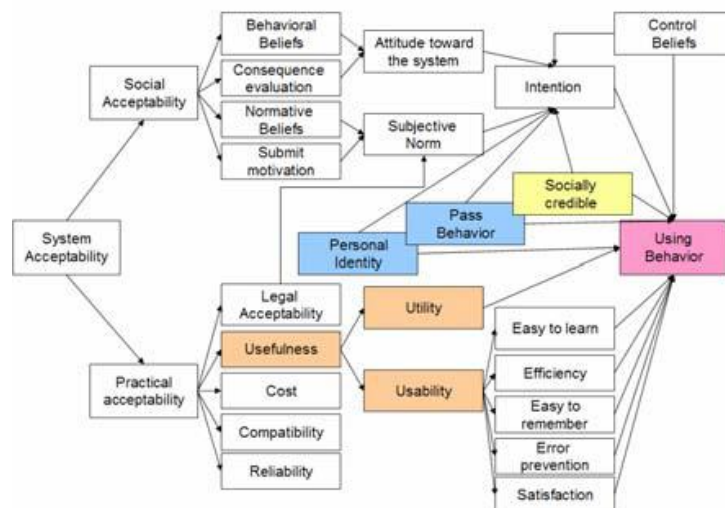


Figure 3 : Modèle de Pesty et Duhaut

Acceptation sociale

Un autre courant de pensées place l'humain au cœur des préoccupations : l'acceptation sociale. Ce courant de pensée indique que la technologie est un objet social avant d'être un objet technique, comme le suggéraient les courants de pensées cités ci-dessus. L'objectif ici est d'étudier la manière dont les utilisateurs réagissent face à des nouvelles technologies. Ce courant de pensées a fait naître quatre types de modèles.

Le premier modèle est celui de Davis (1986) : le modèle d'acceptation des technologies (voir Figure 4), venant du domaine de la psychologie sociale. Celui-ci se base sur l'utilité perçue et sur la facilité d'utilisation perçue. L'utilité perçue est l'impression que se fait l'utilisateur sur l'effet qu'a la technologie sur son rendement. La facilité d'utilisation perçue est l'impression que se fait l'utilisateur sur le faible niveau d'effort mental requis par cette technologie. Ces deux critères déterminent l'attitude de l'utilisateur envers cette technologie et déterminent son intention d'utiliser ou non cette

technologie. Le problème de cette approche est qu'elle n'est pas assez précise pour définir la réelle intention d'utilisation d'une technologie.

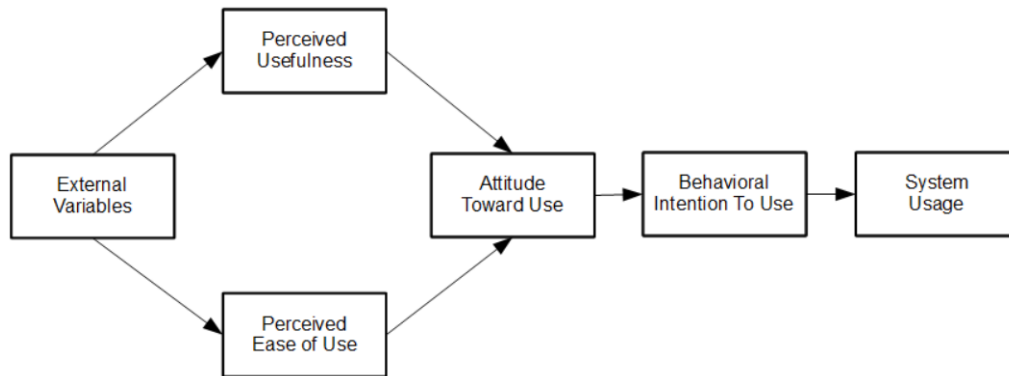


Figure 4 : Modèle de l'acceptation des technologies de Davis

Le deuxième type de modèle est basé sur la satisfaction de l'utilisateur et vient des sciences du management. Cette approche considère que la satisfaction de l'utilisateur est la condition sine qua none du succès d'une technologie. La mesure de la satisfaction se fait alors subjectivement à travers des questionnaires ou des interviews des utilisateurs. Plusieurs approches ont permis de déterminer que la satisfaction est plus importante que les impacts socio-organisationnels qu'elle requiert. D'autres approches indiquent que la qualité de service est un critère important de la satisfaction. Si le comportement de la technologie est celui qu'attend l'utilisateur alors il l'évaluera de bonne qualité. D'autres approches annoncent que l'acceptation d'une technologie aurait tendance à suivre un déroulement chronologique qui impacterait d'abord les aspects individuels puis organisationnels.

Le troisième type de modèle est basé sur la théorie de la disconfirmation des attentes. La disconfirmation spécifie la comparaison entre les performances attendues par un utilisateur et les performances réelles. Lorsque les performances réelles dépassent les performances attendues, il y a alors disconfirmation positive. Au contraire lorsque les performances attendues restent supérieures alors, il y a disconfirmation négative. Cette théorie s'intéresse à la raison pour laquelle un utilisateur utilise ou pas un produit et pourquoi il utilise certaines technologies à long terme. Oliver (1993) a fourni le modèle de la disconfirmation des attentes (voir Figure 5). Il indique que l'utilisation durable d'une technologie serait le fruit d'une expérience satisfaisante.

La satisfaction affecte directement le comportement de l'utilisateur et est directement affectée par la disconfirmation. La satisfaction serait donc l'évaluation de la comparaison entre les attentes de l'utilisateur et l'usage qu'il fait de la technologie. L'intention de continuer à utiliser une technologie repose sur cinq étapes: la formation d'attente à propos du produit, l'utilisation et la formation d'une image de sa performance à partir de l'expérience, la comparaison des attentes préformées et de la performance perçues, la formation d'un niveau de satisfaction selon la confirmation (performance attendue = performance réelle) ou disconfirmation (positive ou négative) des attentes et la construction de l'intention d'utiliser ou de continuer à utiliser selon le niveau de satisfaction ressentie. Ce type de modèle a deux failles. La première est qu'il est difficile de déterminer les attentes a priori. La deuxième est que ce modèle occulte les pratiques d'accompagnements socio-organisationnels de la mise en place d'une technologie.

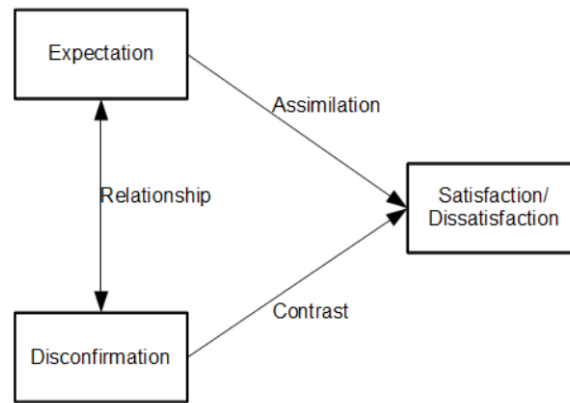


Figure 5 : Modèle de disconfirmation des attentes

Le quatrième type de modèle est au contraire basé sur l'agilité organisationnel. Ce terme renvoie à l'idée que la vitesse de transformation de l'environnement et les évolutions des marchés impliquent des changements incessants auxquels les entreprises doivent s'adapter. Ce modèle considère que l'acceptation des technologies dépend du succès des modalités organisationnelles de sa mise en œuvre. Le point important est la façon dont les technologies se conçoivent et sont introduites sur le marché.

E. Symbiose humain-technologie

Les différentes approches citées précédemment place l'humain comme un élément externe à la technologie. Brangier et al expliquent que ces approches positionnent l'humain en tant qu'agent d'acceptation qui jugerait de tous les éléments en jeu pour décider s'il souhaite utiliser une certaine technologie. Ils « utilisent la notion de symbiose pour restituer le fait que la relation entre l'humain et la technologie qui était peut-être au départ plus utilitaire, se mue au fur et à mesure en dépendance, en lien intime ou encore en mutuelle influence ». La technologie ferait donc partie de l'humain et il y aurait influence réciproque. La symbiose ne cherche pas à évaluer l'acceptation d'une technologie mais plutôt à identifier les conditions de couplage entre les humains, les technologies et leur contexte. Le modèle proposé par Brangier et al repose sur trois conditions : la symbiose au niveau des fonctionnalités de la technologie, la symbiose au niveau de l'utilisabilité et la symbiose au niveau des régulations socio-organisationnelles. La symbiose au niveau des fonctionnalités de la technologie signifie que les fonctionnalités proposées par la technologie sont conformes à ce que l'homme souhaite réaliser pour effectuer une tâche donnée. La symbiose au niveau de l'utilisabilité prend trois facteurs en compte : l'utilisabilité de la technologie, les connaissances de l'humain et la tâche à réaliser. Si la tâche à réaliser peut être facilement réalisée grâce à l'utilisabilité de la technologie et en accord avec la représentation que l'humain se fait des fonctionnalités de la technologie, alors il y a acceptabilité du système. Pour finir, la symbiose au niveau des régulations socio-organisationnelles indique qu'une technologie doit être utilisée dans un contexte social qui l'accepte. Pour apprendre à utiliser une nouvelle technologie, les humains doivent faire des compromis et des arrangements qui doivent être acceptables si la technologie n'est pas capable de s'adapter elle-même. La technologie évolue avec l'usage de l'humain, en même temps que l'humain évolue avec la technologie.

B. Frise chronologique de quelques robots principaux

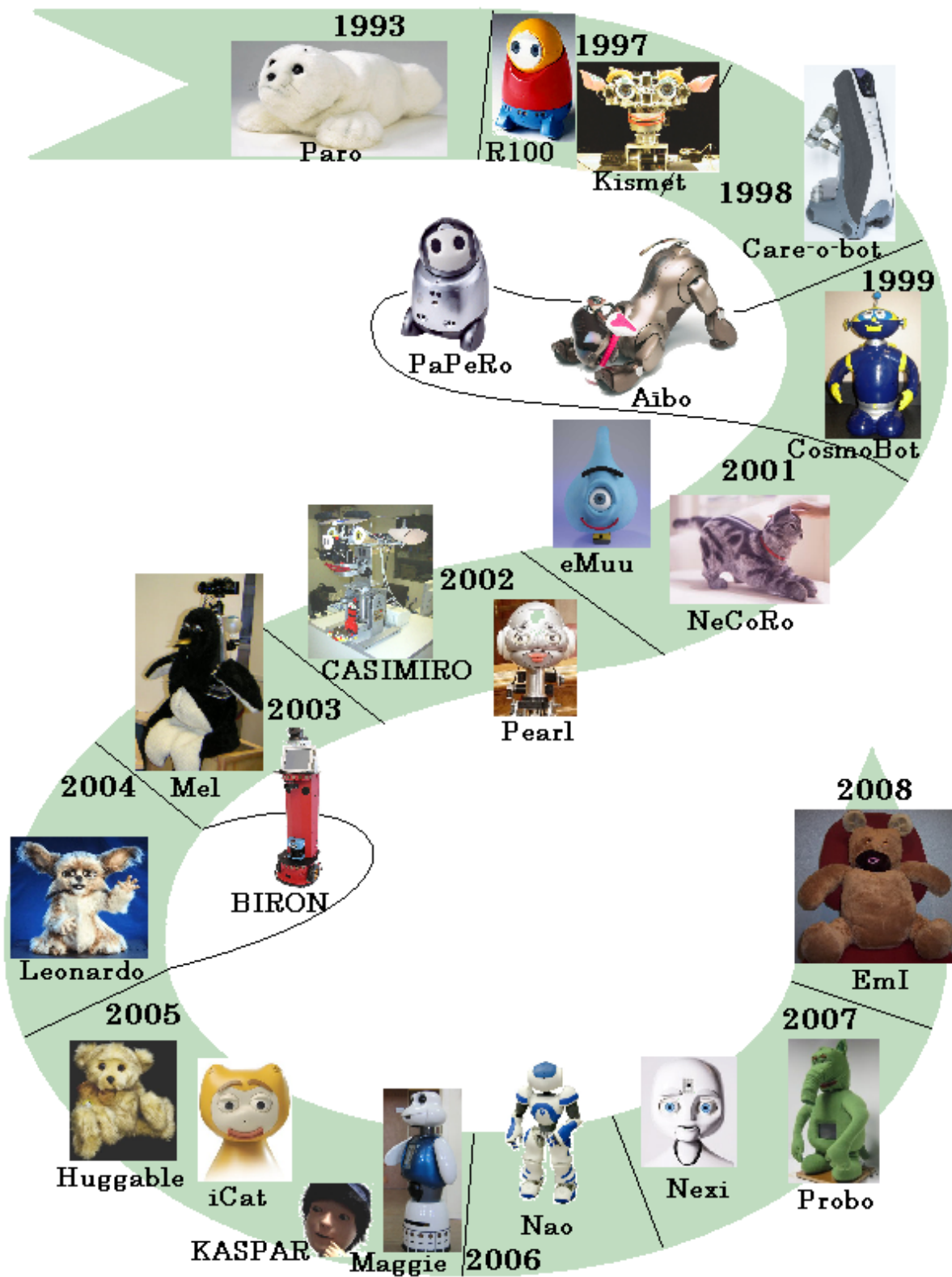


Figure 6 : Frise chronologique de quelques robots principaux

Frise chronologique de quelques robots principaux

Robot	Naissance		Constructeur
	Année	Lieu	
Paro	1993	Japon	AIST
Kismet	1997	USA (Massachusetts)	MIT
R100	1997	Japon	NEC
Care-o-bot	1998	Allemagne (Stuttgart)	Stuttgart Fraunhofer IPA
Aibo	1999	Japon (Tokyo)	Sony (Digital Creature Lab)
CosmoBot	1999	USA (Maryland)	AnthroTonix, Inc
PaPeRo	1999	Japon	Nec
eMuu	2001	Hollande (Eindhoven)	Eindhoven University of Technology Advanced Telecommunication Research
NeCoRo	2001	Japon	Omron
CASIMIRO	2002	Espagne	Universidad de Las Palmas de Gran Canaria
Pearl	2002	USA	Michigan, Pittsburgh, Carnegie Mellon
Mel	2003	USA (Massachusetts)	MIT & Mitsubishi Electric Research Laboratories
Leonardo	2004	USA (Massachusetts)	MIT
BIRON	2004	Allemagne (Bielefeld)	Bielefeld University
HOAP-3	2005	Japon	Fujitsu Automation
Huggable	2005	USA (Massachusetts)	MIT
iCat	2005	Hollande (Eindhoven)	Philips Research
KASPAR	2005	Royaume-Uni	University of Hertfordshire
Maggie	2005	Espagne	University Carlos III of Madrid
Nao	2006	France (Paris)	Aldebaran Robotics
Jido	2007	Allemagne (Gerlingen)	Neobotix
Nexi	2007	USA (Massachusetts)	MIT MediaLab
Probo	2007	Belgique (Bruxelles)	VUB - Robotics & Multibody Mechanics research groupe R&MM
EmI	2008	France (Vannes)	Université de Bretagne Sud – Laboratoire VALORIA

C. Description des messages réservés

A. MICE_1 et MICE_6 : Connexion, déconnexion et suppression d'un module à MIIME

Pour s'enregistrer auprès du serveur, un module doit connaître l'adresse et le port du serveur de MICE. Ces deux données permettent de créer un socket lié au serveur. Lorsque le socket est créé, le module doit envoyer le message d'identification suivant :

```
<mice type="configuration">
  <subject>identification</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><id type="type">nom module émetteur</id></message>
</mice>
```

Ce message MICE est un message de configuration du système destiné au serveur, d'où le type configuration. C'est ce message que le serveur fait suivre aux entités agissantes inscrites pour recevoir les notifications d'identification des modules. C'est un message d'identification, désigné par le sujet id. La balise from doit contenir le nom du module qui émet le message tandis que la balise to doit contenir server puisque ce n'est pas un message destiné à un autre module. La fréquence est 0 car ce message n'est pas répétitif, il n'existe en une seule occurrence. Le cœur du message est la chaîne :

```
<id type="type">nom module émetteur</id>
```

La balise racine de ce message est id, comme le sujet du message. Une identification est définie avec un type et avec un nom de module. Les modules peuvent avoir des types s'ils nécessitent un traitement différent en fonction de leur catégorie. Il y a actuellement trois types possibles :

- input : indique que ce module est un module d'entrée. Cela signifie qu'il ne reçoit aucune information sur son flux d'entrée et qu'il utilise uniquement son flux de sortie.
- deliberator : indique que ce module est un module intermédiaire de MIIME relayant des messages io. Son flux d'entrée et son flux de sortie sont activés. Il effectue des opérations en utilisant les informations des messages qu'il reçoit afin d'envoyer le résultat aux modules qui le suivent.
- output : indique que ce module est un module de sortie. Cela signifie qu'il n'envoie aucune information et utilise uniquement son flux d'entrée.

Lorsqu'un module se connecte, le serveur ajoute le nom de ce module dans une liste de modules connectés. Ainsi, il connaît à chaque instant la liste des modules disponibles associés à leurs

informations de connexion. Il peut ainsi envoyer des messages en ciblant précisément le destinataire. Lorsqu'un module se déconnecte, il est enlevé de la liste.

Les entités n'envoient pas de message lors de leur déconnexion car c'est l'arrêt du programme qui détermine la déconnexion du serveur. Lorsque le serveur détecte qu'une entité est déconnectée, il envoie le message suivant aux entités inscrites :

```
<mice type="configuration">
  <subject>identification</subject>
  <from> server </from>
  <to>nom module émetteur</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><id state="off">nom module émetteur</id></message>
</mice>
```

Le message que le serveur envoie aux entités est un message id qui contient un attribut **state** ayant pour valeur **off**. L'absence d'attribut **state** indique vaut **on** par défaut. Le message contient le nom du module concerné par l'identification.

Lorsqu'un module doit être supprimé du serveur MICE car le dispositif numérique associé a été jeté, par exemple, un module doit envoyer le message suivante :

```
<mice type="configuration">
  <subject>delete</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><delete type="module">nom module</delete></message>
</mice>
```

Le message que le module envoie au serveur est un message **delete** de type **module**. Le sujet du message MICE est **delete**. Le nom du module à supprimer du serveur MICE est le texte de la balise **delete**.

B. MICE_2 et MICE_3 : Notification des identifications

Un module peut s'inscrire à la réception des messages d'identification et de déconnexion au serveur. Le message de souscription est la suivante :

```
<mice type="configuration">
  <subject>subscribe</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><subscribe type="id">nom module</unsubscribe></message>
</mice>
```

Le message que le module envoie au serveur pour s'inscrire à la réception des identifications des entités agissantes est un message **subscribe** de type **id**. Le nom du module qui s'inscrit à la notification est représenté par **nom module** dans l'exemple ci-dessus.

Pour se désinscrire, un module doit envoyer le message suivant au serveur :

```
<mice type="configuration">
  <subject>unsubscribe</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><unsubscribe type="id">nom module</unsubscribe></message>
</mice>
```

Le message que le module envoie au serveur pour se désinscrire à la réception des identifications des entités agissantes est un message **unsubscribe** de type **id**. Il est composé de la même façon que le message **subscribe**.

C. MICE_4 et MICE_5 : Routage des messages

Chaque module ignore l'identité des expéditeurs des messages qu'il reçoit ainsi que l'identité des destinataires des messages qu'il envoie. Les messages suivent un chemin précis qui est enregistré auprès du serveur. Pour créer ce chemin, il existe deux messages :

- connect : ce message permet de créer un lien entre deux modules
- disconnect : ce message permet de supprimer un lien entre deux modules.

Ces messages permettent de créer le chemin de transit des messages entre les modules : les liens. Un lien entre le module A (module d'entrée) et le module B (module de sortie) indique que les informations envoyées par le flux de sortie de A seront transmises au flux d'entrée de B. C'est un lien unidirectionnel. Cela signifie que les informations de B ne sont pas envoyées à A.

Les deux messages sont de type info car ce sont des informations qui concernent uniquement le serveur.

Le message de connexion est le suivant :

```
<mice type="configuration">
  <subject>connect</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><connect><in>m1</in><out></out>m2</connect></message>
</mice>
```

Les caractéristiques du message io sont les mêmes que le message d'identification sauf pour le sujet qui est connect. Le contenu de la balise message indique qu'un lien est établi entre le module m1 et le module m2.

Le message de déconnexion est le suivant :

```
<mice type="configuration">
  <subject>disconnect</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><disconnect><in>m1</in><out></out>m2</disconnect></message>
</mice>
```

Les messages de connexion et de déconnexion sont quasiment similaires. La différence est le sujet du message et le contenu de la balise message qui sont disconnect ici. Le message ci-dessus indique que le lien entre le module m1 et le module m2 est rompu.

Les connexions et déconnexions des liens entre les modules peuvent être faites de façon dynamique sans que cela nuise au routage des informations de l'architecture. Le serveur stocke les

liens dans un tableau clef/valeur. La clef est le nom d'un module d'entrée et la valeur associée est l'ensemble de tous les modules de sortie associés à ce module d'entrée.

D. MICE_7 et MICE_8 : Accès aux descriptions des entités agissantes

La connaissance sur l'environnement est apportée par les descriptions des modules et par la liste des modules connectés. Ce paragraphe décrit les messages permettant de demander une description, de fournir une description, de demander la liste des modules connectés et de fournir cette liste.

Le serveur peut demander la description d'un module en envoyant le message suivant :

```
<mice type="informationnel">
  <subject>get</subject>
  <from>server</from>
  <to>nom module destinataire</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><get type="description" /></message>
</mice>
```

Le message que le serveur envoie aux modules pour demander les descriptions des entités est un message **get** de type **description**. Ce message MICE a un type **informationnel** car il permet la configuration de l'architecture et ne participe pas à l'interaction entre l'homme et les entités.

Le client fournit sa description en envoyant le message suivant :

```
<mice type="informationnel">
  <subject>set</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><set type="description">Description du module</set></message>
</mice>
```

Le message que les modules envoient au serveur pour envoyer les descriptions est un message **set** de type **description**, qui est le pendant du message **get** précédent. C'est également un message MICE de type **informationnel**. Le contenu du message **set** est l'ensemble des fichiers de description de l'entité agissante concernée.

Un module peut également envoyer ce message pour **mettre à jour** sa fiche de description si, l'entité agissante a été modifiée.

Un client peut également demander la liste de toutes les descriptions possibles. Pour cela il envoie le message suivant au serveur :

```
<mice type="informationnel">
  <subject>get</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><get type="description" /></message>
</mice>
```

Le message que le module envoie au serveur pour connaître la liste des descriptions des perceptions et des actions est également un message **get** de type **description**.

Le serveur donne la liste de toutes les descriptions en envoyant le message suivant :

```
<mice type="informationnel">
  <subject>set</subject>
  <from>server</from>
  <to>nom module destinataire</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><set type="description">Toutes les descriptions<set></message>
</mice>
```

Le message que le serveur envoie au module pour partager la description des entités est un message **set** de type **description**.

E. MICE_9 et MICE_10 : Liste des modules connectés

Un client peut demander la liste des modules en envoyant le message suivant :

```
<mice type="informationnel">
  <subject>get</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><get type="modules" /></message>
</mice>
```

Le message que le module envoie au serveur pour demander la liste des modules connectés est un message **get** de type **modules**.

Le serveur fournit la liste des modules connectés en envoyant le message suivant :

```
<mice type="informationnel">
  <subject>set</subject>
  <from>server</from>
  <to>nom module destinataire</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><set type="modules">Liste des modules<set></message>
</mice>
```

Le message que le serveur envoie au module pour fournir la liste des modules connectés est un message **set** de type **modules**. La liste des modules est envoyée sous le format présenté dans le chapitre III.D.2.d.

F. MICE_13 : les actions

Dans MICE, les perceptions sont envoyées à n'importe quel moment, ainsi que les actions. Il n'est pas possible de connaître en avance le temps que prend chaque action car l'architecture se heurte aux contraintes réelles. Si on demande à quelqu'un d'aller faire les courses, il n'est pas possible de connaître l'heure à laquelle cette personne rentrera. L'architecture doit pouvoir s'adapter à un environnement où les actions ne sont pas prévisibles. Pour cette raison, les entités agissantes actives doivent être capables d'indiquer la fin d'une action. Pour éviter le transit d'informations qui ne sont pas utiles pour tous les modules – c'est-à-dire des informations qui ne participent pas à l'interaction – les modules qui veulent connaître la fin des actions doivent s'inscrire auprès du serveur. Seuls ces modules recevront le message de fin d'action.

Une action $A(a,e,o)$ est un triplé composé de :

- **a** : l'identifiant d'une action. C'est une valeur unique au sein d'un scénario.
- **e** : le nom du scénario qui envoie l'action. Cela permet de définir un identifiant d'action unique au sein de toute l'architecture composé de **a** et **e**.
- **o** : l'action en elle-même qui est une chaîne XML qui peut être interprétée uniquement par l'entité agissante. C'est un ordre écrit dans son langage.

Il y a trois types d'actions :

- **Actions simples** : elles visent une entité précise et sont envoyées une par une. Elles ne peuvent pas être synchronisées avec d'autres actions.
- **Actions génériques** : elles ne visent pas une entité précise mais plutôt une capacité d'action. Le gestionnaire choisit l'entité qui peut effectuer l'action.
- **Actions synchronisées** : elles visent plusieurs entités précises et doivent être exécutées en même temps.

Les actions simples sont envoyées de la façon suivante :

```
<mice type="action">
  <subject>nom module destinataire</subject>
  <from>nom module émetteur</from>
  <to>nom module destinataire</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><action id="a" src="e">o</action></message>
</mice>
```

L'action est représentée par un identifiant **a** et par un nom **e** qui représentent la source de l'action, par exemple le nom du module émetteur. En effet, si deux modules envoient une action à la même entité et que les identifiants sont identiques, il sera impossible de savoir quel ordre a été réalisé lorsque l'entité indiquera avoir fini l'action. Et comme il n'est pas possible d'avoir deux modules portant le même nom dans l'architecture, en ajoutant le nom d'un possible émetteur, on assure d'identifier de façon unique une action au sein de l'architecture.

Les actions génériques sont envoyées par le même message que les actions simples sauf que le sujet du message est systématiquement **all**.

Les actions synchronisées sont gérées différemment. Toutes les actions sont envoyées dans un seul message action qui contient tous les ordres :

```
<mice type="action">
  <subject>synchronized actions</subject>
  <from>nom du module émetteur</from>
  <to>premier identificateur de l'action </to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message>
    <action id="a" src="e">
      <sync_act module="id module 1">o</sync_act>
      ...
      <sync_act module="id module X">o</sync_act>
    </action>
  </message>
</mice>
```

La chaîne XML, représentée par la lettre **o** dans l'exemple, est dépendante de l'entité agissante. MICE assure de conserver le langage propre à chaque entité en plus de permettre une interopérabilité entre toutes les entités numériques, MICE assure un protocole de communication similaire entre chaque module pour un bon fonctionnement global. Par exemple, il est possible d'intégrer Greta à MICE en écrivant une chaîne FML ou BML à la place du mot Ordre et en développant un module qui récupère le FML ou le BML reçu et qui le transmet à Greta. Grâce à la bibliothèque fournie, le nombre de lignes de code est minimal (voir l'exemple fourni dans H.D).

G. MICE_14 : fin des actions

Une entité agissante active indique la fin d'une action en envoyant le message suivant :

```
<mice type="action">
  <subject>action</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><action id="a" src="e" state="STOPPED" /></message>
</mice>
```

Le message indiquant la fin d'une action est constitué de la même balise que celui qui indique le début d'une action. La différence ici est l'ajout du paramètre state qui a pour valeur **STOPPED**. Un message action ne contenant pas de paramètre state indique que l'état est **RUNNING**, par défaut.

H. MICE_11 et MICE_12 : Notification des actions

Pour s'inscrire à la réception de la notification de fin d'action, un module doit envoyer le message suivant au serveur :

```
<mice type="action">
  <subject>subscribe</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><subscribe type="action">nom module</subscribe></message>
</mice>
```

Le message que le module envoie au serveur pour s'inscrire à la notification des actions est un message **subscribe** de type **action**. Le nom du module qui s'inscrit à la notification est représenté par **nom module** dans l'exemple ci-dessus.

Pour se désinscrire, un module doit envoyer le message suivant au serveur :

```
<mice type="action">
  <subject>unsubscribe</subject>
  <from>nom module émetteur</from>
  <to>server</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>0</frequency>
  <message><unsubscribe type="action">nom module</unsubscribe></message>
</mice>
```

Le message que le module envoie au serveur pour se désinscrire à la notification des actions est un message **unsubscribe** de type **action**. Il est composé de la même façon que le message **subscribe**.

I. MICE_15 : les perceptions

Les perceptions ne possèdent pas de durée d'exécution et il n'y a pas de message indiquant la fin d'exécution d'une perception. Une entité agissante perceptive analyse l'environnement, effectue des calculs, des traitements puis détermine qu'un évènement s'est passé. Cet évènement est envoyé sous forme de perception. Le destinataire de ce message dépend du chemin stocké par IOLink.

Une perception $P(m,p,v,c)$ est un quadruplet composé de :

- m : Le nom du module émetteur de la perception
- p : Le nom de la perception
- v : Une valeur associée à la perception
- c : Un coefficient de vraisemblance

Par exemple, la perception $P(\text{camera}, \text{personne_tombee}, 1, 100)$ indique que l'entité agissante perceptive « camera » a envoyé une perception nommée `personne_tombee`. La valeur 1 indique que la personne est tombée. C'est un booléen. Cela signifie que la personne n'est pas tombée si la perception vaut $P(\text{camera}, \text{personne_tombee}, 0, 100)$. Le coefficient 100 indique que l'entité estime à 100% la fiabilité de l'information.

Pour envoyer une perception, une entité doit envoyer le message suivant :

```
<mice type="perception">
  <subject>perception</subject>
  <from>nom module émetteur</from>
  <to>all</to>
  <size>nombre de caractères du contenu de la balise message</size>
  <time>yyyyMMddHHmmss</time>
  <frequency>une fréquence d'envoie</frequency>
  <message>
    <perception module="m" name="p" value="v" coefficient="c" />
  </message>
</mice>
```

La chaîne XML qui correspond à la perception contient les quatre informations décrites ci-dessus. C'est un message dont la balise racine est **perception** et qui contient quatre attributs : **module**, **name**, **value** et **coefficient**.

Dans l'exemple ci-dessus, le destinataire du message est « all ». Les modules peuvent utiliser ce mot clef lorsqu'ils n'adressent pas le message au serveur puisqu'ils ignorent le nom des modules qui composent l'architecture. Le serveur fera suivre ce message jusqu'au dernier module sauf si un des modules stoppent le relai.

Dans les exemples précédents, la fréquence était toujours à 0 car ces messages n'étaient envoyés qu'une seule fois, pour la configuration de l'architecture. Dans les messages de perception, cette valeur peut être importante pour les modules qui utilisent cette perception afin d'adapter la façon de traiter l'information en fonction du débit par exemple.

D. AmbiProg : Outil pour la création de scénario

A. Présentation générale

AmbiProg est composé de trois parties, comme le montre la Figure 7. La partie 1 est le **menu** de l'application. La partie 2 est la **zone de construction du scénario**. Enfin, la partie 3 est la **zone d'éléments de scénarii**. Ces trois parties sont détaillées ci-dessous.

La partie 1 est le menu de l'application. La partie 2 est la zone de construction du scénario. Enfin, la partie 3 est la zone de programmation. C'est à cet endroit que l'on choisit les éléments du scénario à assembler dans la partie 2.

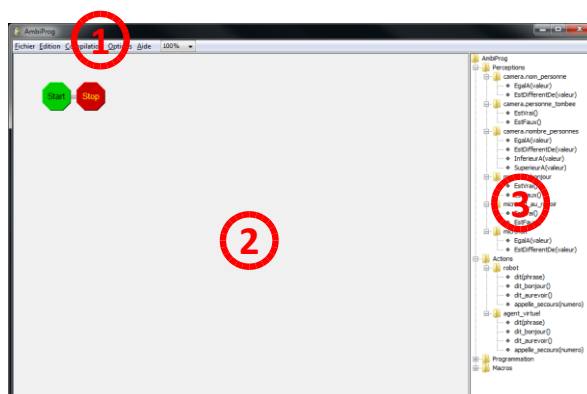


Figure 7 : Organisation d'AmbiProg

1. Le menu

Le menu, visualisé sur la Figure 8, permet de gérer les scénarii et la configuration visuelle de l'interface.

Le menu **Fichier** permet d'afficher une nouvelle page de construction de scénario, d'ouvrir un scénario existant, d'enregistrer le scénario. Il est également possible d'exporter le scénario actuelle sous forme d'une image jpg ou png.

Le menu **Edition** permet d'annuler l'action précédente ou d'effectuer à nouveau une action annulée.

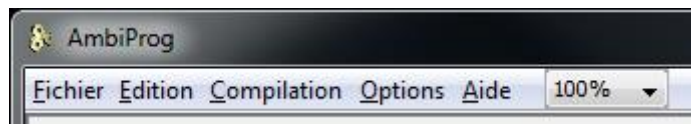


Figure 8 : Le menu d'AmbiProg

Le menu **Compilation** permet de vérifier si le scénario est syntaxiquement correct. Ce menu permet également d'enregistrer le scénario dans une forme compatible avec ArCo, qui n'est pas lisible par l'utilisateur mais uniquement par AmbiLive, le module d'interprétation d'ArCo.

Le menu **Options** permet de définir l'apparence de l'interface et de choisir si une aide visuelle doit être affichée pendant la construction du scénario.

Le menu **Aide** permet d'afficher une fenêtre d'aide sur l'utilisation d'AmbiProg et une fenêtre d'information sur cette interface.

La zone de droite permet de modifier la taille d'affichage de la zone de construction de scénario.

Un menu contextuel existe lorsqu'on clique, avec le bouton droit de la souris, sur une sélection de scénario. La Figure 9 montre un exemple de menu contextuel. Il permet d'effectuer des opérations classiques comme le couper, copier et coller. Il permet aussi de modifier les paramètres lors du clic droit sur une boîte d'actions ou d'éléments de scénario qui propose des paramètres. Il est également possible de supprimer la sélection. Le sous-menu Plier permet de réduire un ensemble de scénario. A la place, un  apparaîtra. Cela permet de gagner de la place dans la zone de construction de scénario. Le menu contextuel permet également de transformer la sélection en une macro (définie et détaillée dans la partie 3).

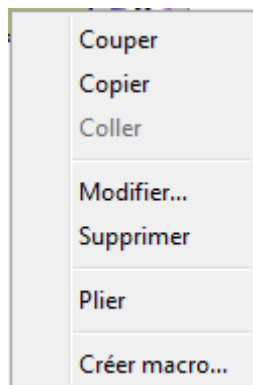


Figure 9 : Menu contextuel d'AmbiProg

2. La zone de construction de scénarii.

La zone de construction de scénario, présentée dans la Figure 10, commence par l'icône **Start** et se termine par l'icône **Stop**. Une barre grise relie ces deux icônes et représente le scénario actuel qui est vide. L'utilisateur doit ajouter des éléments sur la barre grise pour créer un scénario. La partie 3 détaille la construction des scénarii.

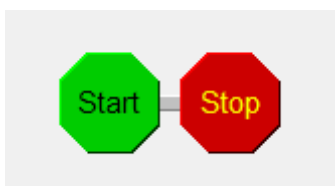


Figure 10 : Zone de construction du scénario

3. La zone d'éléments de scénarii.

La zone d'éléments de scénario, représentée par la Figure 11, contient un arbre qui permet de sélectionner rapidement les informations souhaitées. L'arbre est composé de quatre branches :

- Perceptions : cette branche regroupe toutes les perceptions qui se trouvent dans le fichier XML capteurs.desc.
- Actions : cette branche regroupe toutes les actions qui se trouvent dans le fichier XML entites.desc.

- **Programmation** : cette branche regroupe tous les éléments de programmation qui permettent de faire le lien entre les perceptions et les actions.
- **Macros** : cette branche regroupe toutes les macros qui sont des morceaux de scénarii préalablement construits pour pouvoir être utilisé dans un scénario à la manière d'un sous-scénario. Les macros permettent d'éviter des répétitions dans les scénarii.

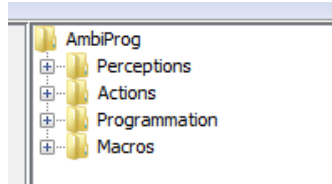


Figure 11 : Zone d'éléments de scénario

La branche Macros ne contient aucune information à l'initialisation d'AmbiProg. C'est l'utilisateur qui construit ses propres macros. Les trois autres branches contiennent les informations suivantes (avec les deux fichiers montrés dans la partie 1) du tableau 1.

Tableau 1 : Contenu des branches de la zone d'éléments de scénario

Perceptions	Actions	Programmation
- camera.nom_personne ◆ EgalA(valeur) ◆ EstDifferentDe(valeur) - camera.nom_personne.coefficient ◆ EgalA(valeur) ◆ EstDifferentDe(valeur) ◆ InferieurA(valeur) ◆ SuperieurA(valeur) - camera.personne_tombée ◆ EstVrai() ◆ EstFaux() - camera.personne_tombée.coefficient - camera.nombre_personnes ◆ EgalA(valeur) ◆ EstDifferentDe(valeur) ◆ InferieurA(valeur) ◆ SuperieurA(valeur) - camera.nombre_personnes.coefficient - micro.dit_bonjour - micro.dit_bonjour.coefficient - micro.dit_au_revoir - micro.dit_au_revoir.coefficient - micro.dit - micro.dit.coefficient	- AmbiProg ◆ creePerceptionSymbolique() ◆ affectePerception(Nom du i) - robot ◆ dit(phrase) ◆ dit_bonjour() ◆ dit_aurevoir() ◆ appelle_secours(numero) - agent_virtuel ◆ dit(phrase) ◆ dit_bonjour() ◆ dit_aurevoir() ◆ appelle_secours(numero) ◆ lit_informations() - all ◆ dit(phrase) ◆ dit_bonjour() ◆ dit_aurevoir() ◆ appelle_secours(numero)	◆ Branche ◆ Parallélisme ◆ Actions synchronisées ◆ Répéter ◆ Tant Que ◆ Si Alors ◆ Si Alors Sinon ◆ Événement ◆ Attendre ◆ Break ◆ Et ◆ Ou ◆ Non ◆ ()

Les branches Perceptions et Actions montrent les informations qui se trouvent dans les fichiers .desc.

Les perceptions sont accompagnées de plusieurs fonctions qui dépendent du type de la perception. Les types peuvent être *number*, *string* ou *boolean*. Si c'est un nombre, alors AmbiProg permet de vérifier si la valeur de la perception est égale, différente, inférieure ou supérieure à une valeur entrée par l'utilisateur. Si c'est une chaîne de caractère, AmbiProg permet de vérifier si la valeur de la perception est égale ou différente à la valeur entrée par l'utilisateur. Si c'est un booléen, alors AmbiProg permet de vérifier si la perception est vraie ou fausse. Chaque perception est

accompagnée de son coefficient de vraisemblance. Il est possible de faire des tests sur ce coefficient de la même façon que sur le type number.

La branche Actions contient autant de sous-branches qu'il existe d'entités agissantes dans le fichier de description. Pour chaque entité agissante, les actions disponibles sont affichées sous forme de fonctions qui peuvent avoir des paramètres si cela était spécifié dans le fichier de description. Dans l'exemple donné, le robot et l'agent virtuel peuvent dire une phrase écrite par l'utilisateur, dire bonjour, dire au revoir ou appeler les secours avec un numéro spécifié par l'utilisateur. L'ensemble des actions portant la même signature sont regroupées dans la sous branche all qui permet de gérer les actions génériques (cf. Document principal). La sous branche AmbiProg est fixe et contient deux fonctions :

- `creePerceptionSymbolique(nom du module, nom de la perception, valeur, coefficient, type)` : permet de créer une perception symbolique.
- `affectePerception(nom du module, nom de la perception, valeur, coefficient)` permet d'affecter une valeur et un coefficient à une perception désignée par le nom du module et le nom de la perception.

La branche de programmation contient un ensemble d'éléments qui permet de créer les scénarii en utilisant les perceptions en tant que conditions (dans un langage de programmation classique) et les actions en tant que fonctions. Chacun des éléments de programmation est décrit dans la partie 4.

B. Présentation du langage visuel

Cette partie explique la procédure pour créer des scénarii puis présente le langage visuel d'AmbiProg en détail.

1. Procédure de construction d'un scénario

Pour créer un scénario, les utilisateurs doivent sélectionner les éléments de scénario et les faire glisser jusqu'à la zone de construction de scénario et les déposer (drag & drop).

La première étape dans la construction d'un scénario est de sélectionner les éléments du scénario en cherchant dans l'arborescence de l'arbre, comme sur la Figure 12.

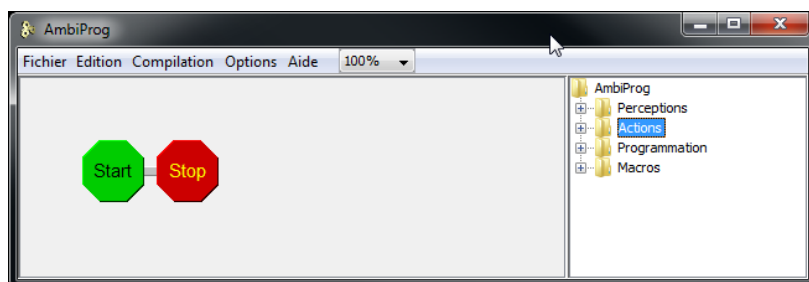


Figure 12 : Sélection d'un élément de scénario

Après avoir sélectionné un élément de scénario, l'utilisateur déplace l'élément sur une zone foncée de la barre grise de la zone de construction de scénario, comme illustré sur la Figure 13. La barre grise est le fil conducteur du scénario, une sorte de frise chronologique qui montre quelles sont les actions à effectuer les unes après les autres.

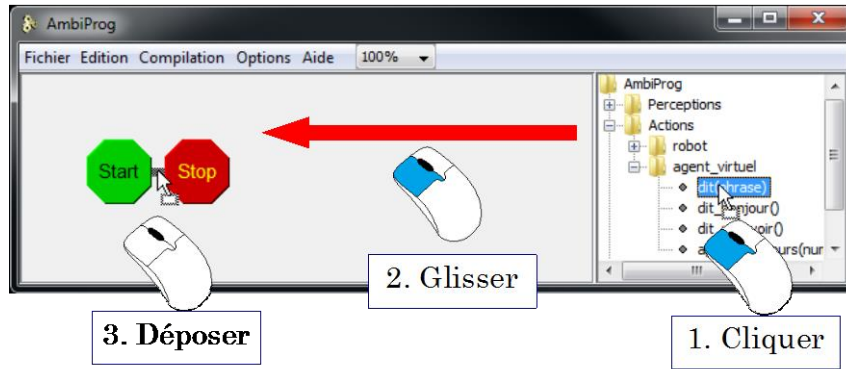


Figure 13 : Glisser/Déposer un élément de scénario

Lorsque l'élément déplacé est une fonction avec un ou des paramètres, une fenêtre apparaît pour saisir le paramètre. La Figure 14 montre la saisie d'une phrase qui est le paramètre de la fonction dit().

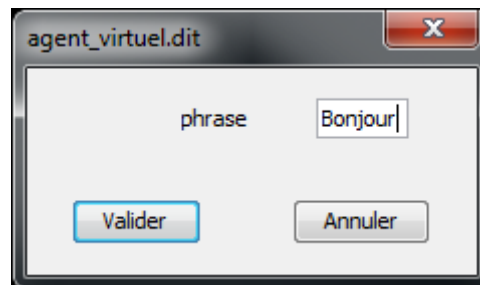


Figure 14 : Saisir un paramètre d'une fonction

Une fois que l'utilisateur a validé sa saisie, l'élément apparaît dans la zone de construction du scénario, comme le montre la Figure 15. Sur l'exemple, l'action est représentée par une boîte.

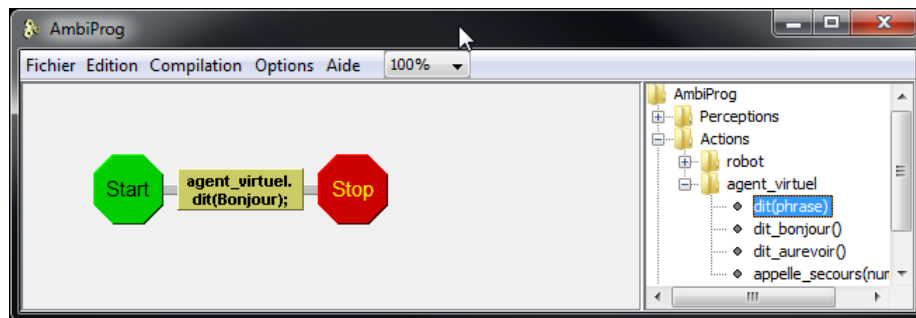


Figure 15 : Un élément de scénario dans la zone de construction

D'autres éléments peuvent être déposés sur la barre grise. Une aide visuelle permet de localiser les endroits où peuvent être placés les éléments. La Figure 16 montre un exemple d'aide. Il est possible de placer des éléments de scénario partout où une coloration foncée est visible.

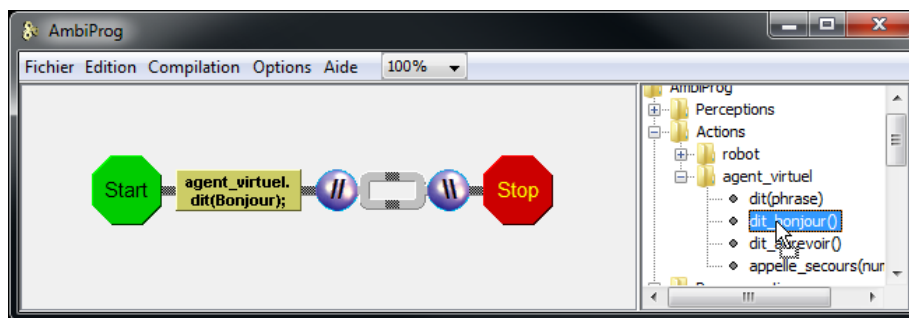


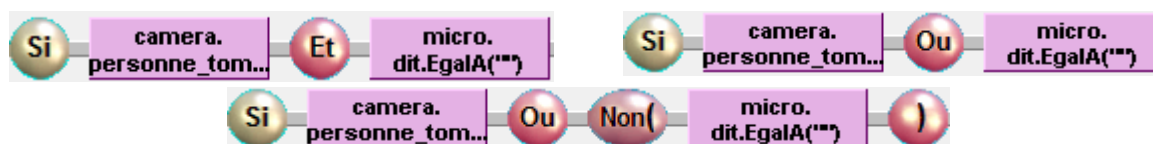
Figure 16 : Aide visuelle indiquant où déposer les éléments de scénario

Lorsque le scénario est fini, il reste à le compiler, via le menu **Compilation**, afin qu'il soit pris en compte par ArCo. L'intégration est détaillée dans la partie E.1.

2. Eléments du langage visuel

Le langage visuel propose de manipuler quatre concepts qui permettent de réaliser un nombre maximum de scénarii différents de façon simple. Ce n'est pas un langage de programmation traditionnel qui propose tous les concepts manipulés par les informaticiens. Ce langage est proche du langage naturel de façon à être le plus évident possible à comprendre pour les utilisateurs. Tous les éléments vus dans cette partie se trouvent dans la branche Programmation. Le premier concept abordé ici est la notion de conditionnelle. Dans un deuxième temps, les boucles seront présentées. Ensuite la construction de sous-scénarii en parallèle. Puis, pour finir, nous aborderons l'attente de perception.

La notion de conditionnelle est représentée par les deux feuilles « Si Alors » et « Si Alors Sinon » auxquelles s'ajoutent les « Et », « Ou », « Non », et « () ». Cette notion est la plus simple à comprendre car elle s'exprime de la même façon en langage naturel : « Si la personne tombe, alors le robot appelle les secours ». La Figure 17 montre la représentation graphique de l'élément « Si Alors » à gauche et de l'élément « Si Alors Sinon » à droite. La flèche bleue montre les endroits où l'on doit insérer des perceptions. Il n'est pas possible d'insérer un autre élément qu'une condition. Une condition peut être une perception. Mais il est également possible de faire des conditions plus complexes mêlant le ET, le OU, le NON et les parenthèses, par exemple :



La flèche verte montre les endroits où doivent être insérées le sous-scénario qui fait suite à cette condition. Ce sous-scénario peut être une simple action, une suite d'actions ou alors un ensemble plus complexe utilisant n'importe quel autre éléments de scénarii.

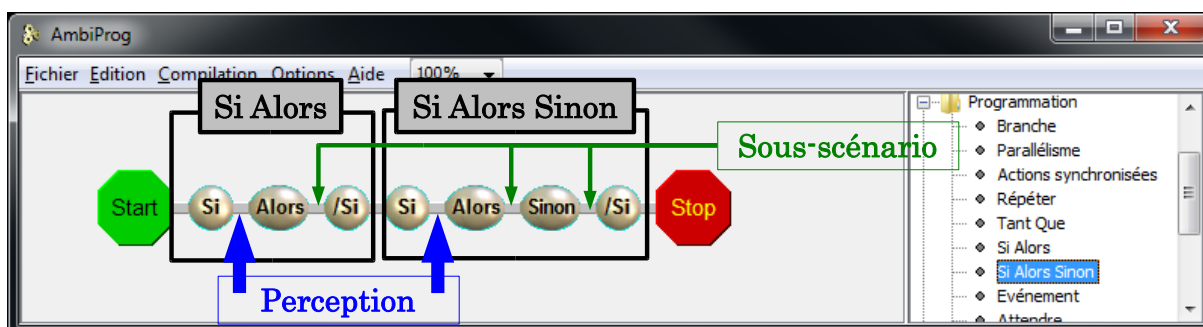


Figure 17 : L'élément "Si Alors" et "Si Alors Sinon" d'AmbiProg

AmbiProg offre également la possibilité de faire des boucles. La Figure 18 montre la représentation graphique de l'élément. C'est une boucle qui permet de répéter un sous-scénario un nombre de fois déterminé par l'utilisateur. La Figure 18 montre l'exemple de dix itérations.

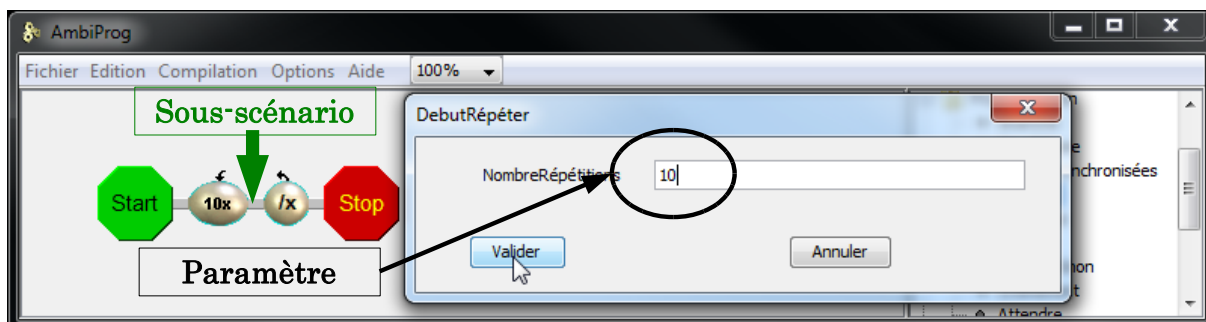


Figure 18 : L'élément "Répéter" d'AmbiProg

La Figure 19 montre un exemple d'utilisation de l'élément Tant Que, c'est une boucle bloquante. Tant que la condition – perception ou un ensemble de perceptions jointes par ET, OU, NON et les parenthèses – est vraie, alors le sous-scénario spécifié après Faire est effectué. La suite du scénario ne pourra être effectuée que lorsque la condition sera fausse.



Figure 19 : L'élément "Tant Que" de AmbiProg

Le troisième concept que permet de manipuler AmbiProg est la programmation de tâches en parallèle. La Figure 20 montre l'élément Parallélisme qui permet de doubler la barre grise et donc de doubler le nombre d'actions à effectuer dans un même temps. Il est possible de mettre tous les éléments de scénario sur chacune de ces barres et donc d'avoir deux scénarii en parallèle. Ces scénarii en parallèle sont notés Sous-scénario sur la figure.

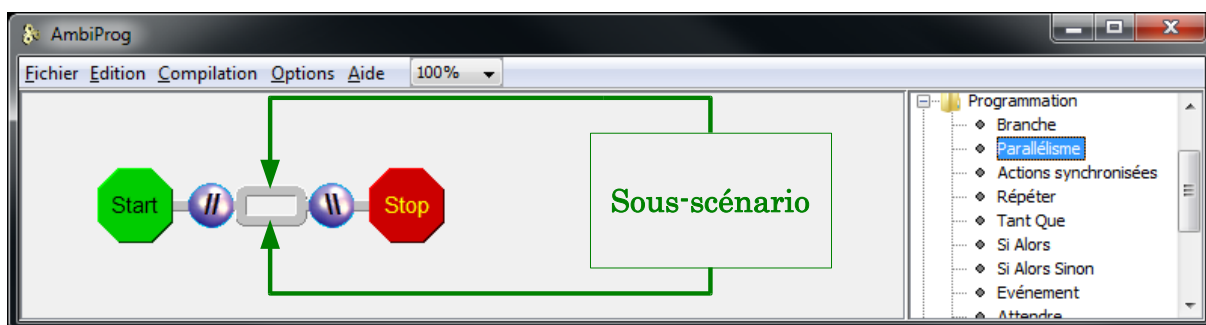


Figure 20 : L'élément Parallélisme d'AmbiProg

L'élément Parallélisme ne permet d'afficher que deux barres grises. Mais il peut être intéressant de définir d'avantage de sous-scénarii en parallèle. Pour cela, AmbiProg propose l'élément Branche qui permet d'augmenter le nombre de barre grise de l'objet Parallélisme, comme le montre la Figure 21.

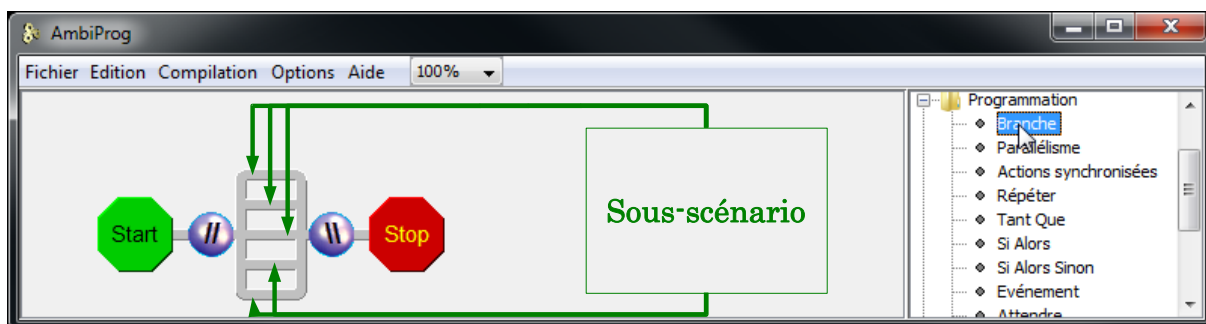


Figure 21 : L'élément Branche d'AmbiProg

L'élément Actions Synchronisées, illustrée sur la Figure 22, fonctionne de la même façon que l'élément Parallélisme. L'élément n'affiche que deux barres grises, dont le nombre peut augmenter grâce à l'élément Branche. Cet élément est très spécifique car il n'est possible d'y insérer que des actions. Chaque branche doit avoir le même nombre d'actions. Le fonctionnement est le suivant : les premières actions de chaque branche démarrent en même temps. Lorsque toutes les premières actions de chaque branche sont terminées, les deuxièmes actions peuvent alors démarrer. Ainsi de suite jusqu'à ce que toutes les actions ait été effectuées.

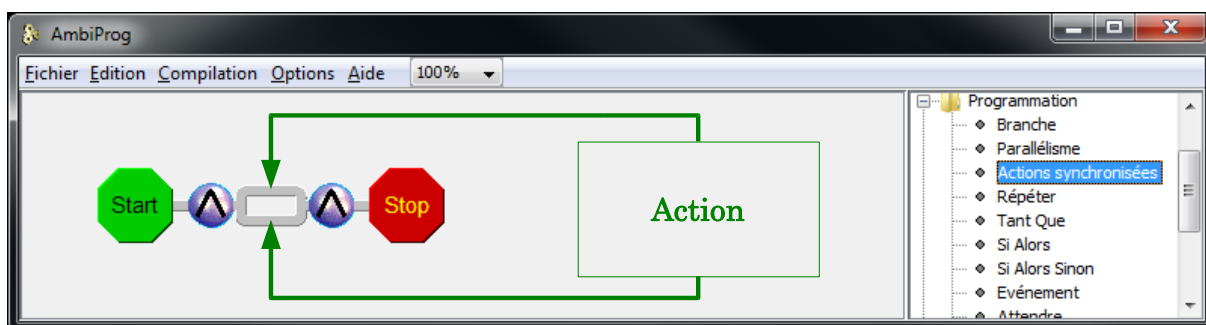


Figure 22 : L'élément Actions Synchronisées d'Ambiprog

Le quatrième concept proposé par AmbiProg est l'attente de perception. C'est de la programmation événementielle du point de vue de l'informatique. La Figure 23 montre l'élément Événement qui permet d'attendre la réception d'une perception spécifique avant d'effectuer le sous-scénario suivant. Cet élément est bloquant. Le scénario attendra que la perception soit envoyée. C'est la différence avec la conditionnelle qui est calculée directement pendant l'interprétation du scénario. L'élément Si Alors suggère que la perception a déjà été envoyée alors que l'élément Évènement attend une perception qui n'a pas encore été envoyée.

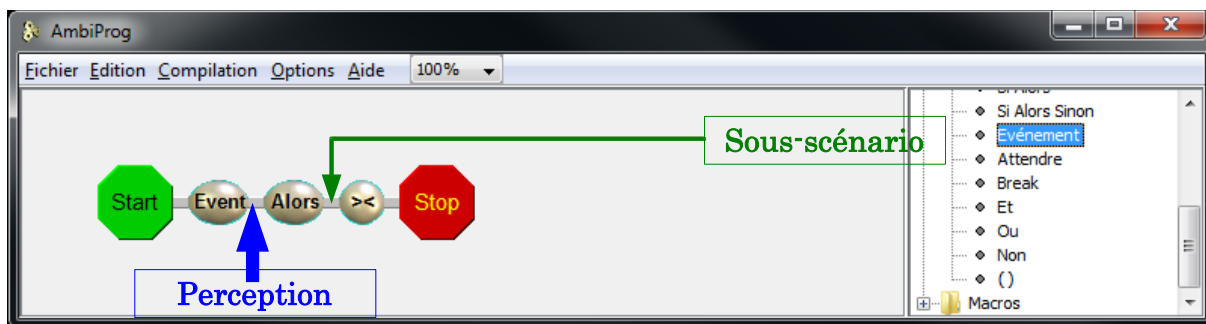


Figure 23 : L'élément Événement d'AmbiProg

Afin d'augmenter les possibilités dans la construction des scénarii, il existe deux autres éléments : Attendre et Break. L'élément Attendre, montré sur la Figure 24, permet de suspendre l'interprétation de la barre grisée qui contient l'élément pendant un laps de temps défini par l'utilisateur.

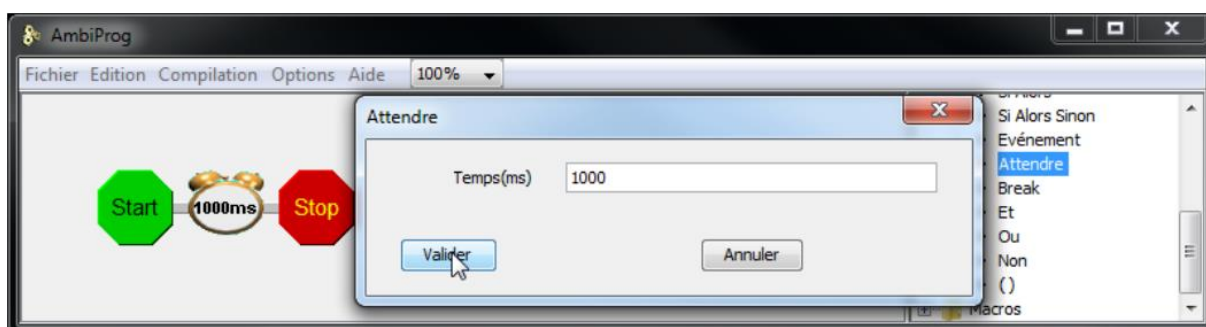


Figure 24 : L'élément Attendre d'AmbiProg

L'élément break, illustrée sur la Figure 25 (hors contexte), permet de sortir des boucles ou de l'élément parallélisme. Son fonctionnement est expliqué en détail dans le chapitre 0.



Figure 25 : L'élément Break d'AmbiProg

3. Création de Macro

Les macros sont des morceaux de scénarii qui ont été enregistrés par l'utilisateur pour être utilisés dans plusieurs scénarii. Ce paragraphe explique comment enregistrer une macro et comment l'utiliser. La Figure 26 montre le scénario qui est utilisé pour illustrer l'explication. Ce scénario est une attente de perception : si la caméra indique que la personne est tombée alors le robot appelle les secours.

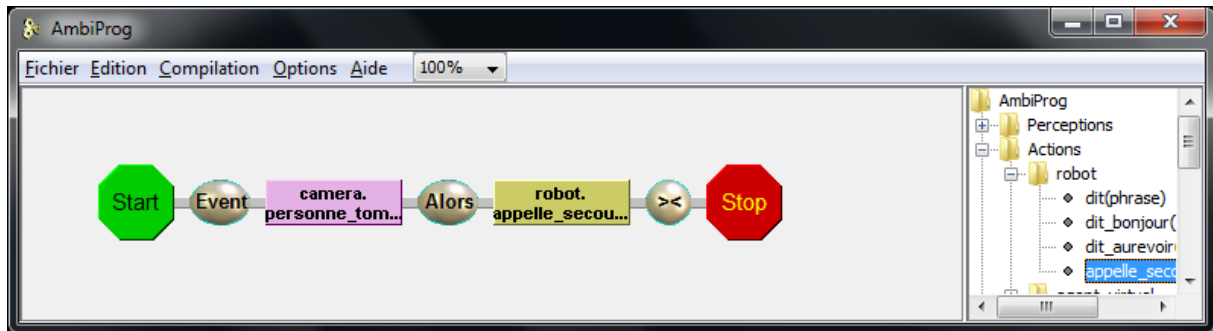


Figure 26 : Procédure de création de macro - programme à transformer

Pour transformer ce scénario en macro, il faut le sélectionner. Cela consiste à maintenir le bouton gauche de la souris en encadrant la zone du scénario à transformer. La zone sélectionnée devient bleu comme illustrée sur la Figure 27. La suite de la procédure consiste à effectuer un clic droit sur la zone sélectionnée afin de faire apparaître le menu contextuel et de cliquer sur Créer macro...

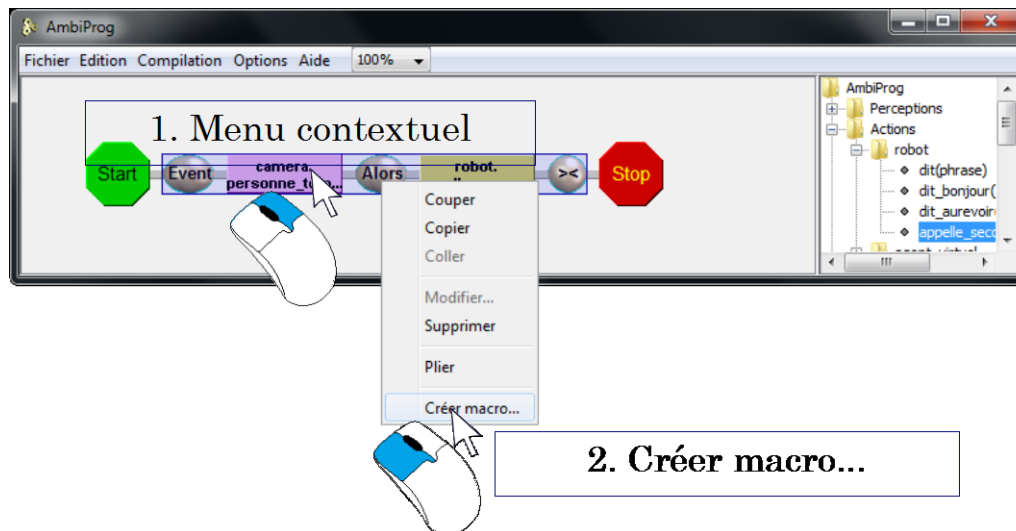


Figure 27 : Procédure de création de macro – transformation

La Figure 28 montre la fenêtre d'enregistrement de la macro, enregistrée sous le nom surveillance_chute.

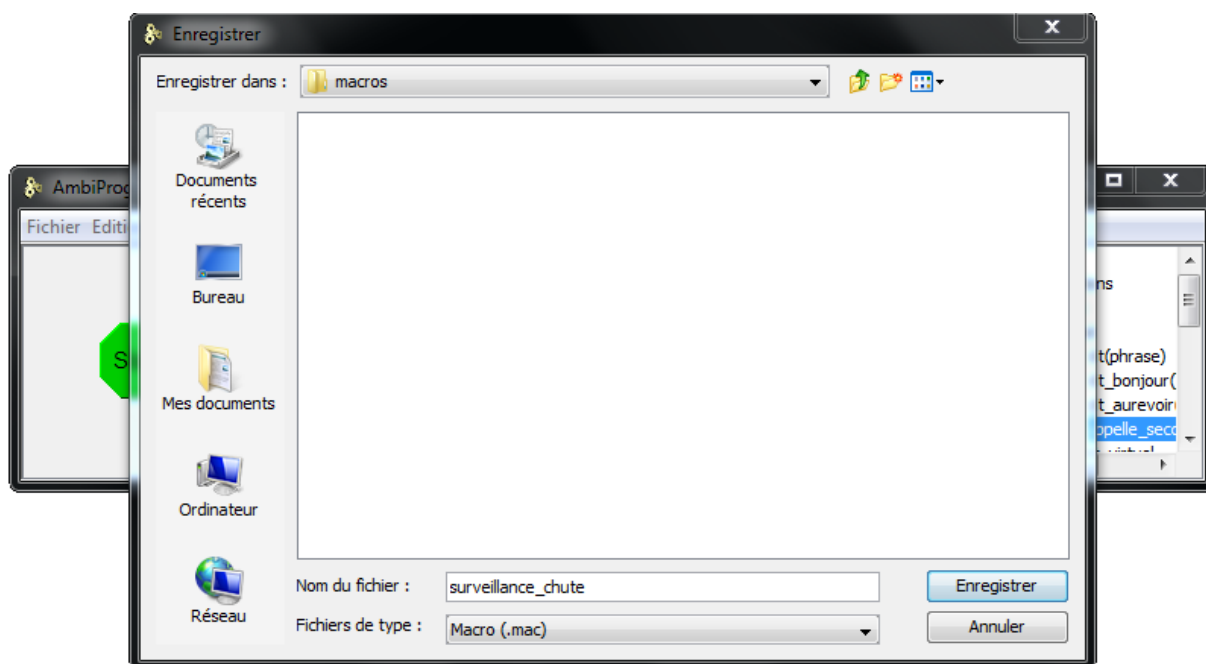


Figure 28 : Procédure de création de macro – enregistrement

La macro se trouve dans la branche Macros, identifiée par le nom enregistré précédemment, comme le montre la Figure 29. Pour utiliser la macro, il suffit de la glisser et déposer sur la barre grise, comme tout autre élément de scénario. Il est possible de voir le contenu de la macro, avec le menu contextuel, en cliquant sur Ouvrir macro.

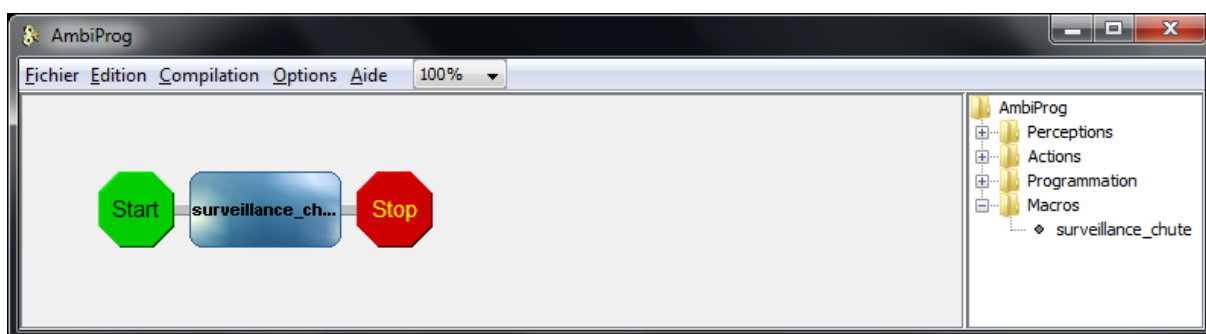
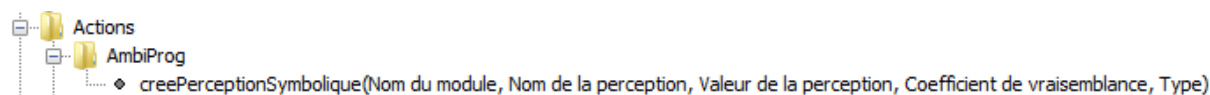


Figure 29 : Procédure de création de macro – résultat

4. Gestion des perceptions symboliques

Les perceptions symboliques sont des perceptions qui sont envoyées par des scénarii en utilisant la fonction `creePerceptionSymbolique` qui se trouve dans Actions.



Cette fonction prend 5 arguments : le nom du module, le nom de la perception, la valeur de la perception, le coefficient de vraisemblance et enfin le type de la perception (string, boolean et number). Les 4 premiers arguments sont les données envoyées en tant que perception brute. Le dernier argument sert à l'affichage des perceptions dans AmbiProg (chapitre III.E.4.b). Le dernier argument de la fonction permet à AmbiProg de proposer les bonnes fonctions de test. Ces

perceptions peuvent être utilisées dans tous les scénarii. L'interpréteur de perception symbolique permet d'interpréter l'ensemble des scénarii de perceptions symboliques (chapitre III.E.4.c).

Création de perceptions symboliques

La création de perceptions symboliques s'effectue avec AmbiProg comme le montre l'exemple de la Figure 30. La procédure consiste à écrire un scénario. Tous les éléments de scénario peuvent être utilisés. La seule condition est d'utiliser la fonction `creePerceptionSymbolique`. La figure montre les 5 paramètres saisis par l'utilisateur. Cette perception symbolique vaut $P(\text{micro}, \text{personne_presente}, 1, 90)$ et est un booléen.

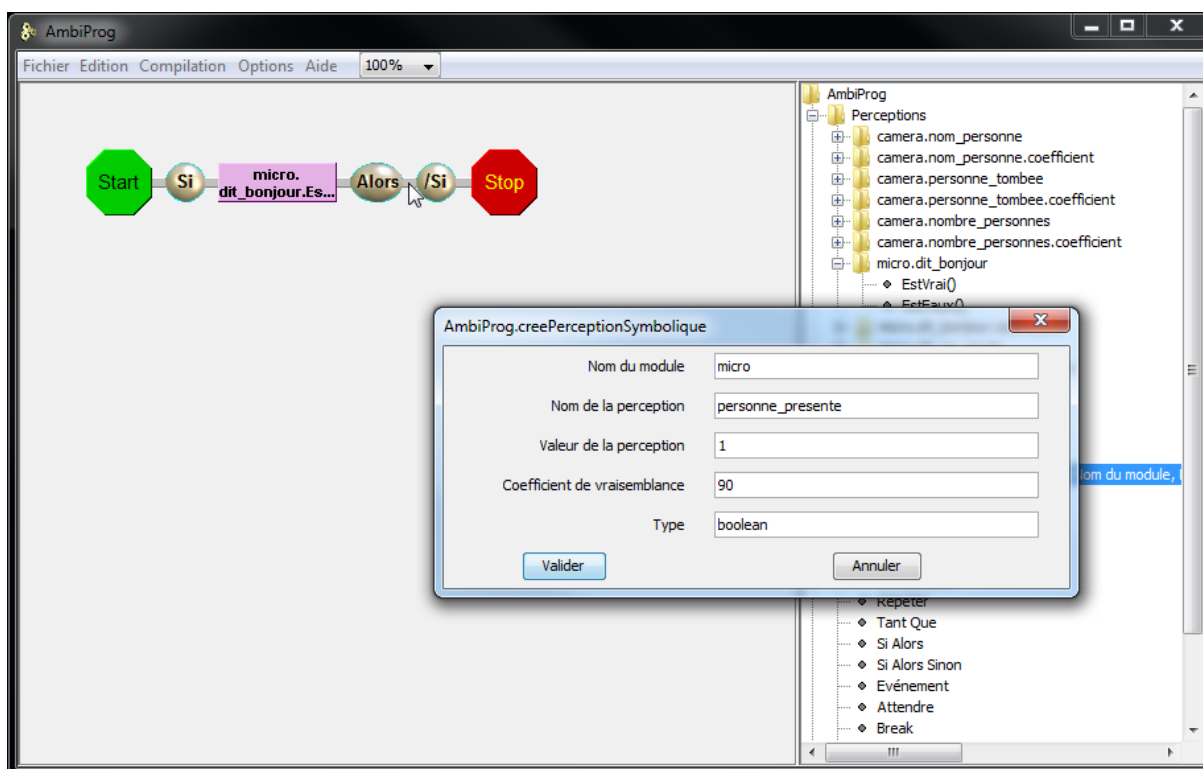


Figure 30 : Création d'une première perception symbolique

Le scénario indique que « si le micro a détecté que la personne a dit bonjour, alors on crée une perception qui s'appelle **personne_presente** et qui vaut VRAI » :



La Figure 31 montre la création d'une deuxième perception symbolique. L'image montre la saisie d'une perception $P(\text{camera}, \text{personne_absente}, 1, 100)$ qui est aussi un booléen. Le scénario indique que « lorsque la caméra indique que le nombre de personne est de 0, alors on crée une perception qui s'appelle **personne_absente** et qui vaut VRAI ».



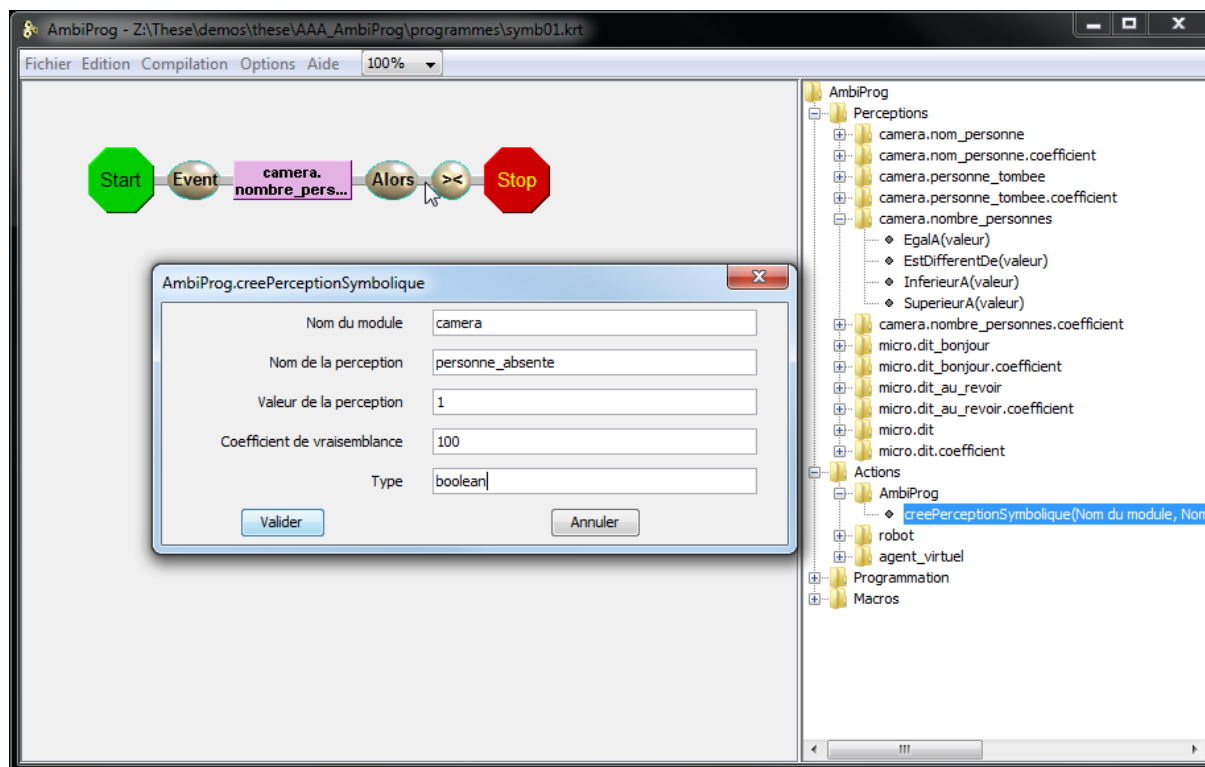


Figure 31 : Création d'une deuxième perception symbolique

Les scénarii de perceptions symboliques doivent être enregistrés dans un répertoire spécifique afin que l'interpréteur puisse les retrouver facilement et les interpréter. La Figure 32 montre l'enregistrement des scénarii de perception symbolique dans le répertoire **symboliques**.

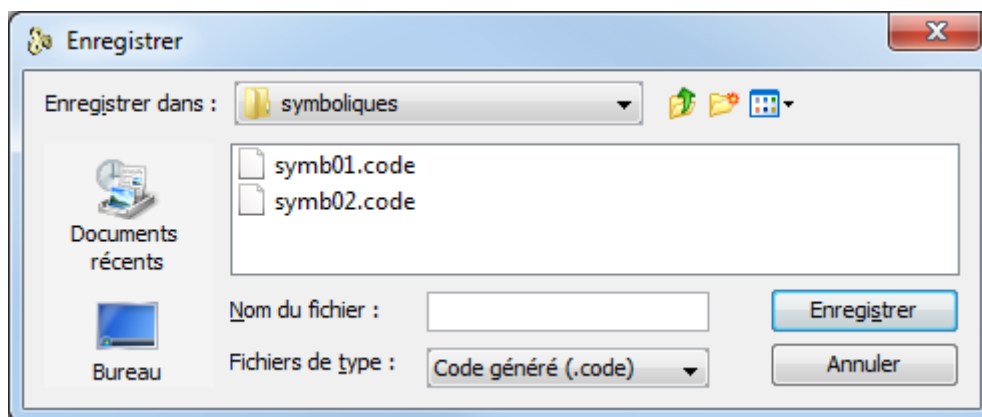


Figure 32 : Enregistrement des perceptions symboliques

Affichage des perceptions symboliques dans AmbiProg

Dans la version actuelle, le logiciel AmbiProg doit être redémarré pour prendre en compte les modifications car il charge le répertoire des perceptions symboliques passé en paramètre au démarrage. Par exemple :

```
> java -jar AmbiProg.jar "../serveur/perceptions/" "../serveur/actions/" "../symboliques/"
```

La Figure 33 montre la liste des perceptions après le redémarrage d'AmbiProg qui montre la prise en compte des perceptions symbolique en tant que perception brute. Ces perceptions peuvent être utilisées dans des scénarii. Le type booléen de ces perceptions a permis à AmbiProg d'afficher les fonctions EstVrai et EstFaux pour ces deux nouvelles perceptions. AmbiProg a également ajouté les coefficients des perceptions.

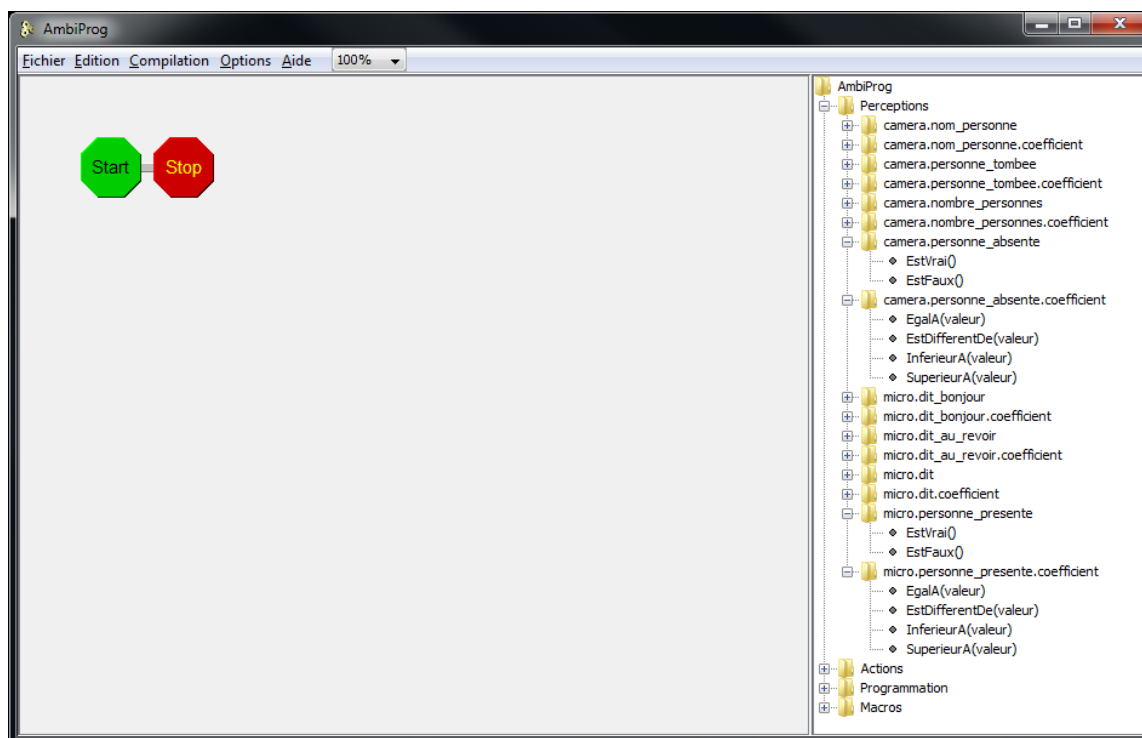


Figure 33 : Affichage des perceptions symboliques dans AmbiProg

C. AmbiProg : une interface complètement personnalisable par fichiers XML

AmbiProg peut être entièrement configuré par deux fichiers XML :

- Objets.xml : fichier qui contient l'ensemble des perceptions et des actions à afficher dans AmbiProg.
- Constructeurs.xml : fichier qui contient toutes les informations sur le langage de programmation visuelle.

1. La branche Perceptions en XML

Le fichier objets.xml contient la description de l'ensemble des perceptions affichées dans AmbiProg. Ces informations sont contenues dans une balise **ListeConditions**. Chaque perception est contenue dans une balise **Condition**. L'exemple ci-dessous concerne la perception camera.non_personne vue dans le tableau 1.

```
<ListeConditions>
  <Condition>
    <Nom>camera.non_personne</Nom>
```

```

<ListePrimitives>
  <Primitive>
    <Methode>EgalA</Methode>
    <Parametres>
      <Param>
        <NomParam>valeur</NomParam>
        <ValMin></ValMin>
        <ValMax></ValMax>
        <TypeParam>string</TypeParam>
      </Param>
    </Parametres>
    <Type>Condition</Type>
    <TypeAttendu>Opérateur</TypeAttendu>
    <Image>images/objets/RectangleCondition.gif</Image>
    <TextePresent />
  </Primitive>
</ListePrimitives>
</Condition>
</ListeConditions>

```

Chaque condition possède un nom et une liste de primitives associées. Les primitives sont les noms des fonctions disponibles pour la perception. Par exemple : EgalA, EstDifferentDe, InferieurA et SuperieurA. Chaque balise **Primitive** est déterminée par un nom (balise Methode), par une liste de paramètres, un type, un type attendu et une image. Le type attendu indique le type d'objet qui peut être placé sur la barre grisée de cet élément. Ici, il n'est pas possible de mettre une autre perception dans cet élément. De plus, il est possible de personnaliser chaque perception à un utilisateur en lui associant une image. La balise TextePresent permet de définir si le texte doit être écrit sur l'image ou non. Pour ne pas afficher le texte, il suffit d'enlever cette ligne.

La balise ListeConditions nourrit la branche Perceptions d'AmbiProg et est créé par la lecture du fichier de descriptions des perceptions des entités agissantes.

2. La branche Actions en XML

Le fichier objets.xml contient également la description de l'ensemble des actions affichées dans AmbiProg. Ces informations sont contenues dans une balise **ListeActionneurs**. Chaque action est contenue dans une balise Actionneur qui est définie par un **Nom** et une **ListePrimitives**. C'est la même organisation XML que pour les perceptions. Ainsi, chaque action possède un nom de fonction qui peut posséder un nombre indéterminée de paramètres (dont 0).

La personnalisation est également possible pour les actions. Par exemple, pour l'action robot.dit(phrased). Si l'on remplace la balise image par <Image>images/objets/bouche.png</Image> et qu'on enlève la balise TextePresent alors on obtient un nouvel affichage pour cette action. La Figure 34 montre la différence entre l'affichage par défaut et l'affichage personnalisé.

La balise ListeActionneurs nourrit la branche Actions d'AmbiProg et est créé par la lecture du fichier de descriptions des actions des entités agissantes.



Figure 34 : Personnalisation d'une action dans AmbiProg

3. La branche Programmation en XML

Le fichier constructeurs.xml contient la description de l'ensemble des éléments de scénarii affichés dans AmbiProg. Ces informations sont contenues dans une balise **ListeConstructeurs**. Chaque élément de scénario est contenu dans une balise Constructeur. Cette balise contient le nom de l'élément qui apparaît dans la branche Programmation et un ensemble de balises permettant de décrire l'élément de programmation. Ces balises décrivent la façon dont se construisent graphiquement les éléments de scénario, leur type, leur type attendu, le code qu'ils génèrent et l'image qui représente chacun des éléments. L'ajout de nouveaux éléments de scénario se fait alors très simplement.

La balise ListeConstructeurs nourrit la branche Programmation d'AmbiProg et n'est pas généré dynamiquement. Ce fichier est fixé.

E. Analyse syntaxique du langage AmbiProg

L'analyse syntaxique détermine la structure du scénario et crée un arbre syntaxique qui permet d'automatiser son interprétation.

Pour chaque scénario, le nœud parent PROG de l'arbre est automatiquement créé (voir Figure 35). L'arbre est construit de gauche à droite et de façon descendante. Le nœud PROG a autant de nœuds fils que de nombre d'instructions dans le scénario.



PROG

Figure 35 : Premier élément de l'arbre syntaxique

Chaque action a pour nœud parent ACTION qui est décomposé en un nœud IDENT et un nœud FUNCTION (voir Figure 36). Le nœud IDENT contient la liste des identificateurs d'une fonction. Le nœud FUNCTION est composé d'un nœud NAME qui contient le nom de la fonction et d'un nœud PARAM qui contient l'ensemble des valeurs des paramètres de la fonction. Le nœud ACTION est utilisé pour les actions et pour les perceptions car elles ont la même structure.

La tableau 3 montre un exemple de cette structure identique.

Tableau 2 : Décomposition d'une action et d'une perception

Scénario	IDENT	NAME	PARAM
camera.personne_tombee.EstVrai()	camera personne_tombee	EstVrai	
robot.appelle_secours(18)	robot	Appelle_secours	18

La Figure 36 montre l'arbre syntaxique généré du scénario suivant :

Lumière.Rouge.Allume(54, "test");

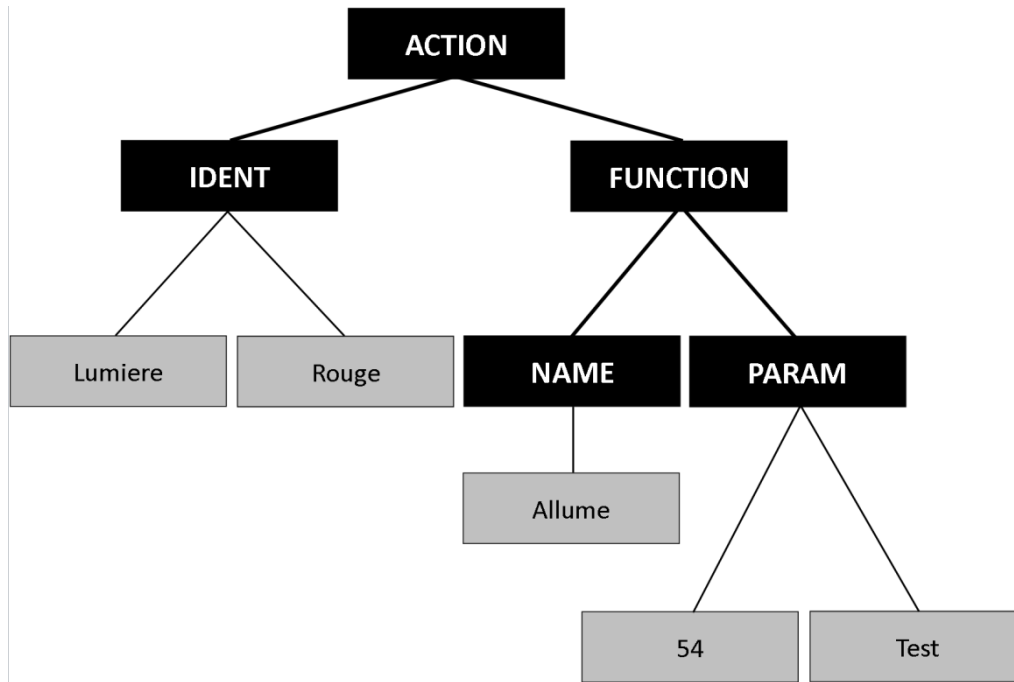


Figure 36 : Arbre syntaxique de l'élément Action

Chaque conditionnelle (élément **Si Alors** et **Si Alors Sinon**) est représentée par le nœud CONDITIONNEL, illustré sur la Figure 37. Ce nœud a 2 ou 3 fils. Le fils n°1 est toujours la condition qui suit le si. Le fils n°2 contient les instructions qui suivent le Alors. Le fils n°3 concerne les instructions qui suivent le Sinon.

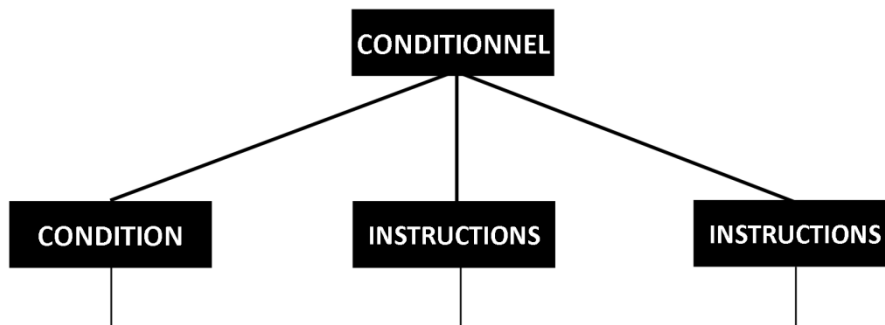


Figure 37 : Arbre syntaxique de l'élément Conditionnel

Le nœud fils du nœud CONDITION est soit un nœud ACTION, soit un nœud ET (&), soit un nœud OU (|). Prenons l'exemple suivant :



L'arbre de la Figure 38 est la représentation du sous-scénario précédent. Les conditions les plus proches des feuilles de l'arbre sont calculées en premières. D'abord A ET B sont calculés (résultat noté E ici). Ensuite C ET D sont calculés (résultat noté F ici). Ensuite, la condition E OU F est calculée.

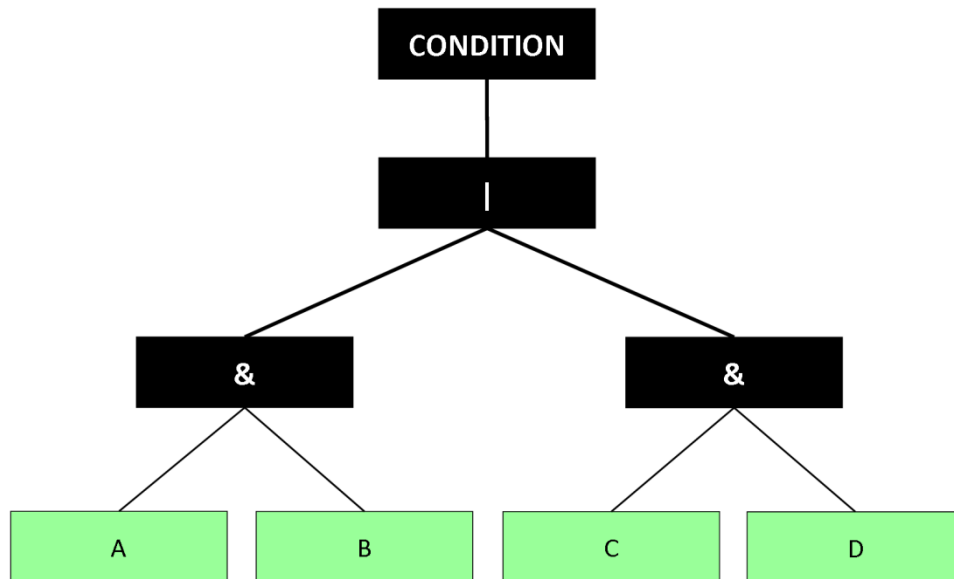


Figure 38 : Arbre syntaxique de l'élément Conditionnel avec ET et OU

La Figure 39 présente l'élément **répéter** est représenté par un nœud REPETER ayant deux fils : une feuille qui contient le nombre de répétitions à effectuer et le nœud INSTRUCTIONS qui contient le scénario à effectuer plusieurs fois.

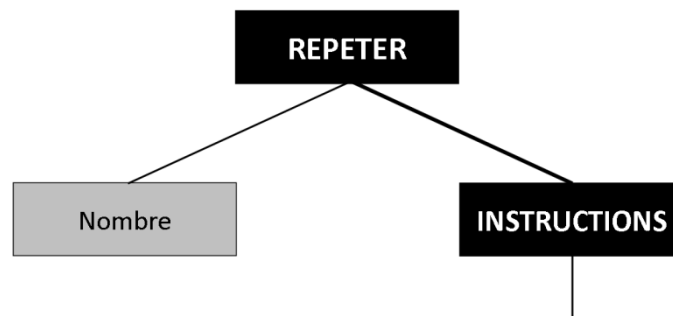


Figure 39 : Arbre syntaxique de l'élément Répéter

L'élément **tant que** est composé de la même façon que l'élément Si Alors avec deux fils qui représentent la condition et les instructions à effectuer, comme illustré sur la Figure 40.

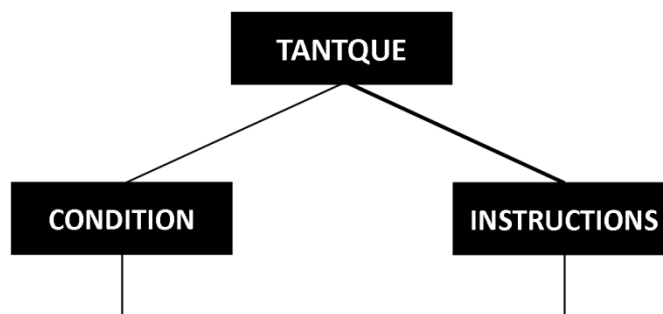


Figure 40 : Arbre syntaxique de l'élément TantQue

L'élément **parallèle** contient au minimum deux nœuds BRANCHE. L'exemple de la Figure 41 montre l'arbre syntaxique correspondant au scénario suivant :

```
//(code1;, code2;, code3;,);
```

Le nœud BRANCHE contient les mêmes nœuds que PROG et INSTRUCTIONS. Chaque branche contient un scénario entier.

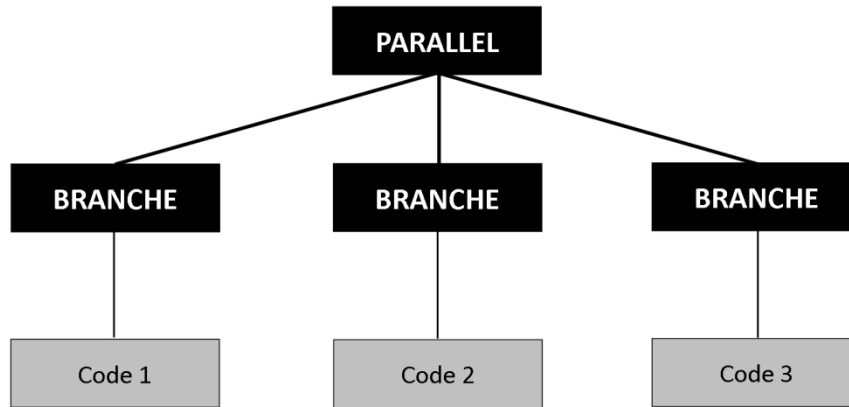


Figure 41 : Arbre syntaxique de l'élément Parallélisme

L'élément **actions_sync** contient au minimum deux nœuds BRANCHE. L'exemple de la Figure 42 montre l'arbre syntaxique correspondant au scénario suivant :

```
/\ (action1, action2, action3,);
```

Ici les nœuds branches ne contiennent qu'un nœud **action**.

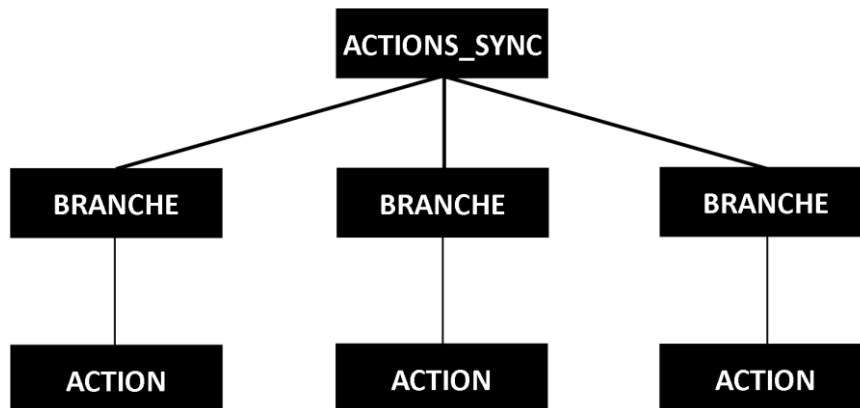


Figure 42 : Arbre syntaxique de l'élément Actions Synchronisées

L'élément **événement** est représenté par le nœud EVENT (voir la Figure 43). Les nœuds fils sont les mêmes que pour l'élément **tant que** à savoir la condition et les instructions.

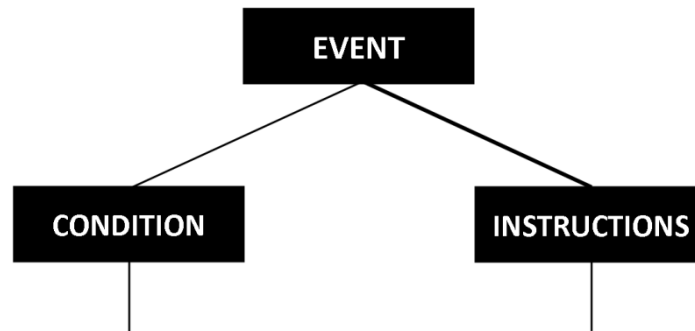


Figure 43 : Arbre syntaxique de l'élément Evènement

L'élément **wait** ne possède qu'un nœud **WAIT** et une feuille qui contient la durée d'attente, comme illustrée sur la Figure 44.

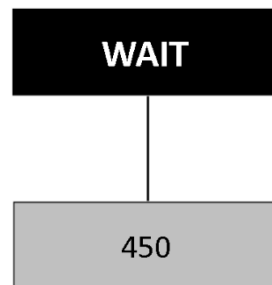


Figure 44 : Arbre syntaxique de l'élément Attendre

L'élément **break** est représenté par un nœud **BREAK** (voir Figure 45) et est toujours une feuille de l'arbre. Break ne peut avoir aucun nœud fils.



Figure 45 : Arbre syntaxique de l'élément Break

La construction de l'arbre syntaxique est l'étape nécessaire à l'interprétation des scénarii. Cela permet d'automatiser l'interprétation.

F. Interprétation de l'arbre syntaxique du langage AmbiProg

Composition de l'arbre : Chaque nœud de l'arbre contient les informations suivantes :

- Une **identification** définie par la position du nœud dans l'arbre. La Figure 46 montre un exemple d'identification. Le nœud parent a pour valeur 0. Les nœuds fils ont pour identification une valeur composée de la valeur du nœud parent suivi d'un nombre indiquant la position du fils : 01, 02, 03... Et ainsi de suite pour chaque niveau de l'arbre. Chaque nœud de l'arbre porte une identification unique.
- Un nom ayant pour valeur PROG, ACTION, IDENT, FUNCTION, NAME, PARAM, INSTRUCTIONS, CONDITION, TANTQUE, REPETER, EVENEMENT, CONDITIONNEL, PARALLEL, BRANCHE, WAIT, BREAK ou une valeur (nombre ou chaîne de caractère).
- La liste de ses nœuds fils.
- Le nœud père.

Ces informations permettent de récupérer les nœuds en fonction de leur identification, de connaître rapidement la façon d'interpréter chaque nœud et de parcourir l'arbre de façon descendante et ascendante.

Principe d'interprétation : L'interprétation commence par le nœud père. S'il a plusieurs fils alors ils sont interprétés les uns après les autres. Il y a donc toujours un seul nœud à interpréter. Mais dans le cas d'un nœud PARALLEL, chaque branche doit être interprétée en même temps. Il y a donc plusieurs nœuds à interpréter.

Le principe est donc de gérer une liste de nœuds de l'arbre. C'est cette liste qui est interprétée à chaque fois. La Figure 46 montre un exemple de répartition des nœuds à interpréter.

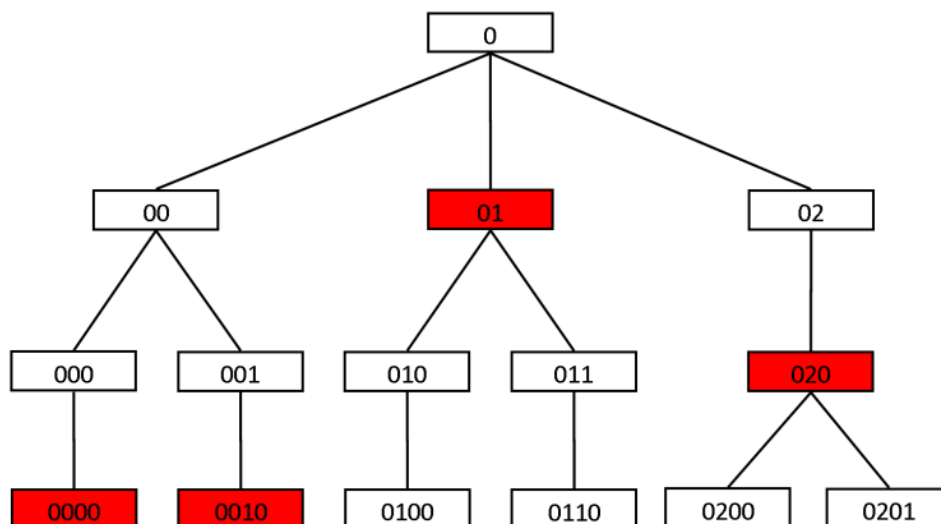


Figure 46 : Procédure d'interprétation

La Figure 47 montre le chemin parcouru par un pointeur de programme et le principe général de l'interprétation. Tout d'abord, le curseur est sur le nœud PROG. Ce nœud n'est pas interprété. Le curseur passe donc au nœud fils. Si ce nœud fils ne peut pas être interprété et qu'il possède plusieurs fils, le curseur passe au premier fils suivant. Le curseur continue son parcours descendant jusqu'à

pouvoir interpréter un nœud. Lorsque le nœud est interprété, le curseur remonte au père et coupe le lien entre le père et le fils. Si le père possède encore un fils alors le curseur descend et reproduit le même schéma. A la fin de l'interprétation, il ne reste plus que le nœud PROG qui finit par être supprimé également. L'interprétation s'effectue récursivement sur le fils.

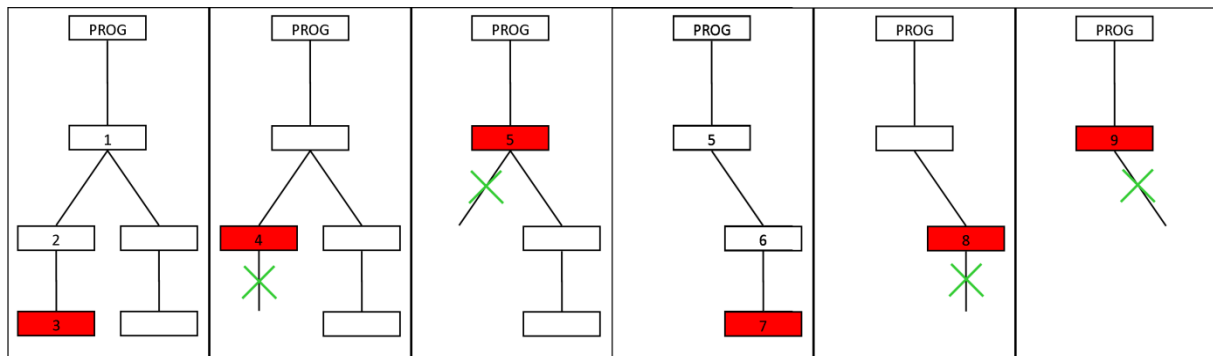


Figure 47 : Chemin d'interprétation

Objets utilisés : L'interprétation est effectuée avec l'aide de trois objets :

- TabActionsState[id_action, state] : liste de toutes les actions, identifiées par leur valeur id_action, associées à leur état state valant RUNNING ou STOPPED.
- TabIdentificationAction[identification, id_action] : liste de toutes les identifications des nœuds associées aux identifications des actions, identifiées par leur valeur id_action.
- TabIdentificationTimer[identification, temps] : liste de toutes les identifications des nœuds associées à une durée d'attente (dans le cas du nœud WAIT).

Initialisation de l'interprétation : L'interprétation est effectuée par la fonction interprete(N, L) qui prend en paramètre un nœud N et une liste de nœuds à interpréter L. Le premier nœud à interpréter est le nœud PROG. A l'initialisation, la liste L est vide.

Interprétation : La fonction interprete(N,L) est décomposée de la façon suivante :

Si N = PROG ou N = INSTRUCTIONS alors
 Si N a des enfants, ajouter dans L le premier enfant de N
 Si N n'a pas d'enfant et a un père P, enlever N à P et mettre P dans L

Les nœuds PROG et INSTRUCTIONS n'ont pas d'interprétation. C'est pour cela que si le nœud a un fils, cela veut dire que le curseur est en phase descendante, et le fils est ajouté à la liste des nœuds à interpréter. Si le nœud n'a pas de fils, cela signifie que le curseur est en phase ascendante et le père est mis dans la liste des nœuds à interpréter. Si le nœud n'a ni père ni fils, alors l'interprétation est terminée.

Si N = REPETER
 Si N a deux enfants
 Récupérer Nombre_repeter le premier enfant de N
 Si Nombre_repeter > 0
 Récupérer Instructions le deuxième enfant de N
 Mettre Instructions dans L
 Faire une copie C de Instructions
 Ajouter C à N
 Diminuer la valeur de Nombre_repeter de N
 Si Nombre_repeter = 0
 Si N a un père P

Enlever N de P et mettre P dans L

Le nœud REPETER est interprété en fonction du nombre de fois où les instructions doivent être répétées. Si ce nombre est supérieur à 0, alors le nœud Instructions est mis dans la liste des nœuds à interpréter. De plus, ce nœud est copié et ajouté aux enfants du nœud REPETER et le nombre de répétition est décrémenté. Il est nécessaire de copier le nœud Instructions car lorsque le curseur est en phase ascendante, le nœud Instructions d'origine est supprimé alors que le nombre de répétition est supérieur à 0. S'il n'y a plus de répétition à exécuter alors si le nœud REPETER a un père, celui-ci est mis dans la liste des nœuds à interpréter et le nœud TANTQUE est supprimé.

Si N = TANTQUE

Si N a deux enfants

Récupérer Condition le premier enfant de N

Récupérer Resultat le résultat de Condition

Si Resultat est vrai

Récupérer Instructions le deuxième enfant de N

Mettre Instructions dans L

Faire une copie C de Instructions

Ajouter C à N

Si Resultat est faux

Si N a un père P

Enlever N de P et mettre P dans L

Le fils Instructions du nœud TANTQUE est interprété seulement si le résultat du fils Condition est vrai. Ainsi, si le résultat est vrai, le nœud Instructions est mis dans la liste des nœuds à interpréter et une copie est créée et ajoutée aux enfants du nœud TANTQUE. En effet, tant que la condition est vraie, les instructions doivent être interprétées. Or, une fois que l'interprétation est terminée, le curseur supprime les instructions dans sa phase ascendante. La copie permet de pouvoir refaire une phase descendante. Si le résultat est faux et que le nœud TANTQUE a un père, celui-ci est mis dans la liste des nœuds à interpréter et le nœud TANTQUE est supprimé.

Si N = EVENEMENT

Si N a deux enfants

Récupérer Condition le premier enfant de N

Récupérer Resultat le résultat de Condition

Si Resultat est vrai

Mettre dans L le deuxième enfant de N

Si Resultat est faux

Si N a un père P

Mettre P dans L

Si N n'a pas deux enfants

Si N a un père P

Enlever N de P et mettre P dans L

Le nœud EVENEMENT est bloquant jusqu'à ce que la condition soit vraie. Donc tant que le résultat renvoyé est faux, les instructions ne sont pas interprétées et le père est mis dans la liste des nœuds à interpréter. Si le résultat renvoyé est vrai, le fils Instructions est mis dans la liste des nœuds à interpréter. Lorsque l'interprétation est terminée, le nœud Instructions est supprimé. Ainsi, le nœud EVENEMENT ne possède plus qu'un fils et la dernière condition de l'algorithme est vérifiée. Si le nœud EVENEMENT n'a pas deux enfants et s'il a un père, alors le nœud est retiré des enfants du père et le père est mis dans la liste des nœuds à interpréter.

Si N = WAIT

Si TabIdentificationTimer contient l'identification de N

Récupérer DateM la date de maintenant

Récupérer DateT la date associée à l'identification de N

Si DateM < DateT

Mettre N dans L

```

Si DateM >= DateT
    Retirer l'identification de N dans TabIdentificationTimer
    Si N a un père
        Enlever N de P et mettre P dans L
Si TabIdentificationTimer ne contient pas l'identification de N
    Récupérer DateM la date de maintenant
    Si N a un enfant
        Récupérer D la durée d'attente
        Créer DateT = DateM+D
        Ajouter dans TabIdentificationTimer DateT associé à l'identification de N
    Mettre N dans L
    
```

L'interprétation du nœud WAIT nécessite l'utilisation de l'objet TabIdentificationTimer. Si le curseur interprète le nœud WAIT pour la première fois, la date et l'heure actuelle sont récupérées dans DateM. Si le nœud WAIT a un enfant, c'est-à-dire une valeur d'attente, alors la date limite est calculée en ajoutant cette durée à DateM. La nouvelle date associée à l'identifiant du nœud interprété est ajoutée à l'objet TabIdentificationTimer. Le nœud WAIT est ajouté à la liste des nœuds à interpréter. Si le curseur interprète un nœud WAIT déjà interprété préalablement alors l'interpréteur compare l'heure actuelle avec l'heure de fin d'attente. Si l'heure n'est pas passée, alors le nœud WAIT est ajouté à la liste des nœuds à interpréter. Si la l'heure est passée alors la date et l'identification de nœud sont enlevés de l'objet TabIdentificationTimer. Et si le nœud a un père, celui-ci est mis dans la liste des nœuds à interpréter et le nœud WAIT est supprimé.

```

SI N = BREAK
    Si N a un père P
        Tant que P n'est pas REPETER ou TANTQUE ou PARALLEL
            Si P a un père GP
                P = GP
            //Arrivé ici, P = REPETER, TANTQUE ou PARALLEL
            Break_enfant(P)
        Si P a un père GP
            Enlever P de GP et mettre GP dans L
    
```

L'interprétation du nœud BREAK nécessite de vérifier qu'il n'y a pas de timer en route, ni d'action en attente de notification de fin. Pour cela, si le nœud a un père, le curseur remonte au père tant que le nœud courant n'est ni REPETER, NI TANTQUE, NI PARALLEL. En effet, le break ne peut être utilisé que dans un de ces trois nœuds. Si le nœud courant est un de ces trois nœuds, alors l'interpréteur soustraite l'interprétation à la fonction Break_enfant(N), chargé de supprimer tous les nœuds et toutes les dépendances. Ensuite, si le nœud courant a un père, celui-ci est mis dans la liste des nœuds à interpréter et le nœud courant est supprimé.

```

Fonction Break_enfant(N) :
    Si N = WAIT
        Créer DateP une date quelconque dans le passé
        Mettre DateP dans TabIdentificationTimer associé à l'identification de N
    Si N = ACTION
        Stopper l'action associée dans TabActionsState
    Sinon
        Récupérer liste_E les enfants de N
        Si le nombre d'éléments de liste_E supérieur à 0
            Pour chaque enfant E de liste_E
                Si E a un père
                    E supprime son père
                    Break_enfant(E)
        Supprimer les enfants de N
    
```

La fonction Break_enfant permet de supprimer les dépendances des nœuds qui ont été supprimés suite au BREAK. Si le nœud N passé en paramètre est un WAIT alors l'interpréteur modifie

l'heure associée à l'identification de N dans TabIdentificationTimer en mettant une heure passée. Ainsi le timer est annulé. Si le nœud N est un ACTION, alors l'action associée est stoppée dans TabActionState. Si c'est un autre nœud et que le nœud a des enfants alors l'interpréteur récupère la liste liste_E des enfants du nœud. Si les enfants ont un père, alors celui-ci est supprimé de façon à ce que le curseur ne puisse pas effectuer de phase ascendante et que l'interprétation s'arrête. Ensuite, la fonction Break_enfant est appelée pour chaque nœud de liste_E. Ensuite, les enfants de N sont supprimés pour le curseur ne puisse pas être en phase descendante.

```
Si N = PARALLEL
  Si N a des enfants liste_E
    Pour chaque enfant E de liste_E
      Mettre E dans L
  Si N n'a pas d'enfant
    Enlever N de P et Mettre P dans L
```

Si le nœud à interpréter est de type PARALLEL, chaque enfant est mis dans la liste des nœuds à interpréter. Lorsque tous les fils ont été interprétés, le nœud n'a plus enfant et le père est mis dans la liste des nœuds à interpréter et le nœud courant est supprimé.

```
Si N = ACTIONS_SYNC
  Si N a des enfants liste_E
    Pour chaque enfant E de liste_E
      Mettre E dans L
  Si N n'a pas d'enfant
    Enlever N de P et Mettre P dans L
```

Si le nœud à interpréter est de type ACTIONS_SYNC, chaque enfant est mis dans la liste des nœuds à interpréter. Lorsque tous les fils ont été interprétés, le nœud n'a plus enfant et le père est mis dans la liste des nœuds à interpréter et le nœud courant est supprimé.

```
Si N = BRANCHE
  Si N a des enfants
    Mettre le premier enfant dans L
  Si N n'a pas d'enfant
    Si N a un père
      Enlever N de P
      Si P n'a pas d'enfant
        Mettre P dans L
```

L'interprétation du nœud BRANCHE est similaire au nœud PROG et au nœud INSTRUCTIONS. Si le nœud a des enfants, alors le premier enfant est mis dans la liste des nœuds à interpréter. Si le nœud n'a pas d'enfant, cela signifie que l'interprétation des enfants est terminée, alors le père est mis dans la liste des nœuds à interpréter et le nœud BRANCHE est supprimé.

```
Si N = CONDITIONNEL
  Si N a au moins un enfant
    Récupérer Condition le premier enfant de N
    Récupérer Resultat le résultat de Condition
    Si Resultat est vrai
      Récupérer Instructions1 le deuxième enfant de N
      Mettre Instructions1 dans L
      Supprimer les enfants de N
      Ajouter Instructions1 aux enfants de N
    Si Resultat est faux
      Si N a deux enfants ET N a un père P
        Enlever N de P et Mettre P dans L
      Si N a trois enfants
        Récupérer Instructions2 le troisième enfant de N
        Mettre Instructions2 dans L
        Supprimer les enfants de N
```

```
Ajouter Instructions2 aux enfants de N
Si N n'a pas d'enfant
  Si N a un père
    Enlever N de P et Mettre P dans L
```

L'interprétation du nœud CONDITIONNEL nécessite le test du premier fils CONDITION comme les nœuds EVENEMENT et TANTQUE. Si le résultat de la condition est vrai alors, que ce soit un **Si Alors** ou un **Si Alors Sinon**, c'est le premier fils Instructions qui est mis dans la liste des nœuds à interpréter. Les autres enfants du nœud courant sont supprimés. Si le résultat de la condition est vrai, alors si le nœud est un **Si Alors**, le père est mis dans la liste des nœuds à interpréter et le nœud CONDITIONNEL est supprimé du point de vue du père. Si le nœud est un **Si Alors Sinon**, le deuxième fils instructions est mis dans la liste des nœuds à interpréter. Les autres enfants du nœud courant sont supprimés. Si le nœud n'a pas d'enfant, cela signifie que les enfants ont été interprétés alors le père est mis dans la liste des nœuds à interpréter et le nœud CONDITIONNEL est supprimé du point de vue du père.

```
Si N = ACTION
  Si TabIdentificationAction contient l'identification de N
    Récupérer id_action l'action associée à l'identification de N
    Récupérer state l'état associée à l'action id_action
    Si state = RUNNING
      Mettre N dans L
    Si state = STOPPED
      Supprimer l'identification de N dans TabIdentificationAction
      Supprimer l'action id_action de TabActionsState
      Si N a un père
        Enlever N de P et Mettre P dans L
  Si TabIdentificationAction ne contient pas l'identification de N
    Créer id_action avec setOutput(action)
    Ajouter N à L
    Ajouter identification de N et id_action dans TabIdentificationAction
    Ajouter id_action dans TabActionsState
```

L'interprétation du nœud ACTION nécessite l'utilisation des objets TabIdentificationAction et TabActionsState. Si le curseur passe pour la première fois sur le nœud ACTION, alors l'action est lancée avec la fonction setOutput(action). Cette fonction retourne une chaîne de caractère qui est un identifiant de cette action : id_action. Le nœud ACTION est ajouté à la liste des nœuds à interpréter car l'interpréteur doit attendre que l'action soit terminée pour poursuivre l'interprétation. La valeur id_action et l'identification de N sont ajoutés dans TabIdentificationAction. De plus, la valeur id_action et l'état RUNNING sont ajoutés dans TabActionsState.

Si le curseur interprète à nouveau le nœud ACTION, alors TabIdentificationAction contient un identifiant d'action associé à l'identification du nœud courant. L'interpréteur récupère id_action et l'état de l'action. Si l'action est RUNNING, alors elle n'est pas encore terminée et le nœud courant est ajouté à la liste des nœuds à interpréter. Au contraire, si l'action est STOPPED, alors l'identification du nœud est supprimée dans TabIdentificationAction et TabActionsState. Si le nœud courant a un père, alors le père est mis dans la liste des nœuds à interpréter et le nœud ACTION est supprimé du point de vue du père.

G. Bibliothèque AmbiLib

Le module d'interprétation utilise une bibliothèque JAVA¹ : AmbiLib. Cette bibliothèque a pour objectif de fournir un ensemble de classes capables d'interpréter le langage AmbiProg. Elle est générale de façon à pouvoir être utilisée dans des cas différents.

La Figure 48 montre l'organisation globale de la bibliothèque. Elle se compose de trois paquets :

- `ubs.robodom.interpreter.interpret` : ce paquetage est le point d'entrée de la bibliothèque et contient les classes qui permettent d'interpréter un programme,
- `ubs.robodom.interpreter.parsing` : ce paquetage contient les classes qui permettent d'analyser les programmes pour créer l'arbre syntaxique nécessaire pour l'interprétation,
- `ubs.robodom.interpreter.structure` : ce paquetage regroupe les différentes structures utilisées pendant l'interprétation.

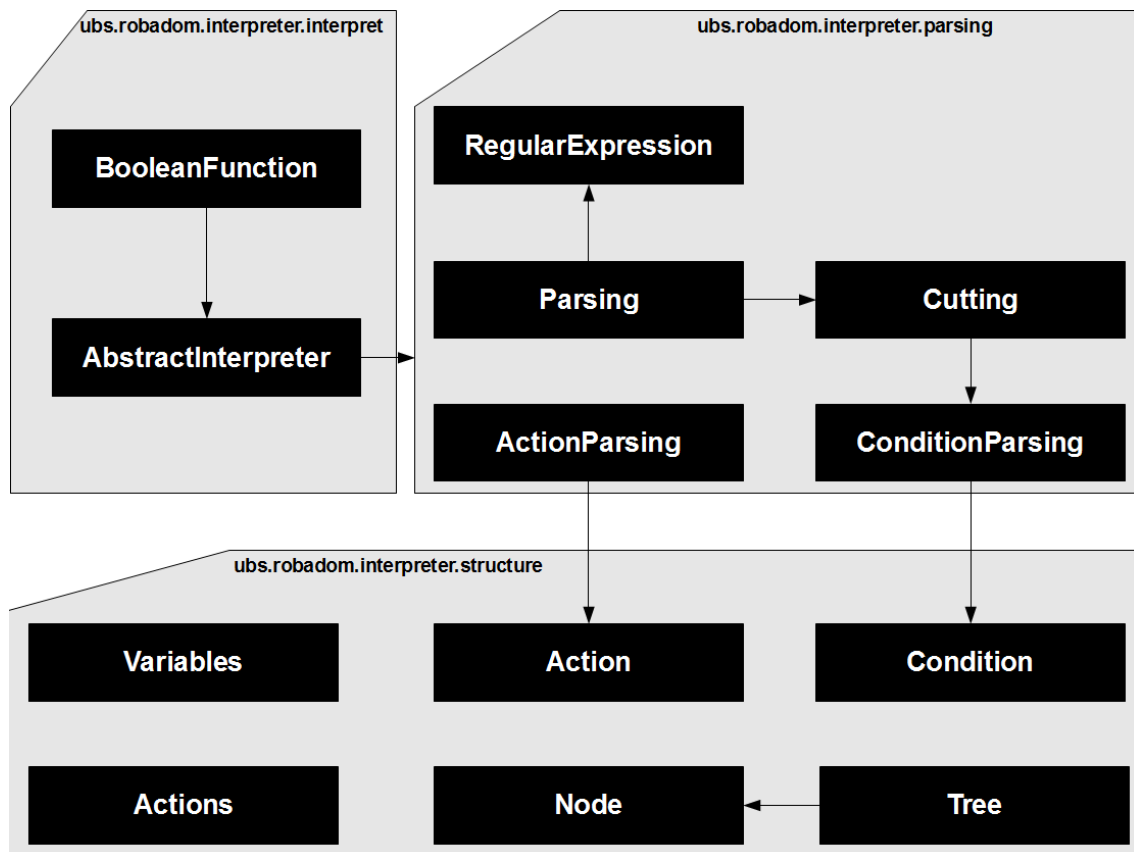


Figure 48 : Bibliothèque de l'interpréteur : organisation globale

¹ Le JAVA est un langage informatique orienté objet.

A. Paquetage `ubs.robodom.interpreter.structure`

Nous avons réutilisé le vocabulaire et la structure utilisés pour les arbres en informatique :

- La classe `Tree` représente un arbre syntaxique et contient un `Node` père,
- La classe `Node` représente, classiquement, un nœud de l'arbre. Le détail d'implémentation de ces nœuds est donné en Annexe F.

Plus spécifiquement au langage AmbiProg, ce paquetage contient les classes suivantes :

- La classe `Actions` stocke l'ensemble des actions que l'interpréteur a rencontré et qui doivent être exécutées. Cet ensemble contient la liste des identifiants des actions associés à leur état : `RUNNING` ou `STOPPED`,
- La classe `Variables` stocke l'ensemble des perceptions qui ont été reçues par l'interpréteur,
- La classe `Action` permet de gérer une fonction. Elle stocke l'ensemble des identificateurs, le nom et l'ensemble des paramètres d'une fonction,
- La classe `condition` stocke une conditionnelle composée d'un élément `SI`, d'un élément `ALORS` et d'un élément `SINON` s'il existe (cf. Annexe E).

B. Paquetage `ubs.robodom.interpreter.parsing`

La classe `Parsing` est la classe principale du paquetage **parsing**. Elle utilise la classe `Cutting` et `RegularExpression` pour créer l'arbre syntaxique. La classe `Cutting` est chargée de découper un scénario en bloc. Le tableau 4 montre un exemple de découpage en blocs (zone verte). La classe `RegularExpression` permet de déterminer si une chaîne de caractère est un des éléments du langage AmbiProg. Ainsi, à l'initialisation du tableau 4, aucun élément n'est détecté et la classe `Cutting` effectue une première itération de découpage. Au bout de la première itération, la classe `Cutting` détecte qu'il y a trois blocs : une action, un élément `EVENEMENT` et un élément `PARALLEL`. L'action peut être interprétée directement, mais pas les deux autres. Ainsi, la classe `Cutting` est utilisée pour chaque élément jusqu'à ce que les blocs contiennent des actions ou des perceptions afin de pouvoir les interpréter.

Tableau 3 : Découpage d'un scénario en bloc

Itération	Découpage du scénario : act1;<perception>(act2;act3) ;// (act4,act5,act6) ;							
1	act1	<perception>(act2,act3)			//(act4,act5,act6)			
2		perception	act2,act3		//	act4,act5,act6		
3			act2	act3		act4	act5	act6

C. Paquetage `ubs.robodom.interpreter.interpret`

La classe `AbstractInterpreter` est le point d'entrée de la bibliothèque. C'est une classe abstraite qui doit être étendue pour pouvoir être utilisée. Elle se sert de la classe `BooleanFunction` pour tester les conditions. C'est elle qui permet de renvoyer le résultat vrai ou faux dans les algorithmes présentés dans l'annexe F. La Figure 49 montre le diagramme de la classe `AbstractInterpreter`. Les propriétés de la classe sont présentées sans l'annexe F : `TabActionsState`, `TabIdentificationAction` et `TabIdentificationTimer`. L'objet `parsing` permet de récupérer l'arbre syntaxique correspondant à un scénario passé en paramètre.

AbstractInterpreter
<pre># parsing : Parsing # tabActionsStates : Actions - tabIdentificationAction : HashMap<String, String> - tabIdentificationTimer : HashMap<String, String></pre>
<pre>+ AbstractInterpreter(Actions) + interprete(String) # getPerceptionType(String, String) : String # getInput(String, String) : Perception # removeInput(String, String) : String # setOutput(Action) : String - interprete(ArrayList<Node>) : ArrayList<Node> - interpreteNode() : ArrayList<Node> - interpreteProgOrInstructions(Node, ArrayList<Node>) - interpreteRepeter(Node, ArrayList<Node>) - interpreteTantQue(Node, ArrayList<Node>) - interpreteEvenement(Node, ArrayList<Node>) - interpreteWait(Node, ArrayList<Node>) - interpreteBreak(Node, ArrayList<Node>) - breakChildren(Node) - interpreteParallel(Node, ArrayList<Node>) - interpreteActionsSync(Node, ArrayList<Node>) - interpreteBranche(Node, ArrayList<Node>) - interpreteConditionnel(Node, ArrayList<Node>) - interpreteAction(Node, ArrayList<Node>) - analyseComplexCondition(Node) : boolean - analyseCondition(Node) : boolean</pre>

Figure 49 : Bibliothèque de l'interpréteur : interpréteur abstrait

Les fonctions privées permettent d'interpréter chacun des éléments de scénario. L'interprétation des éléments tels que PROG, INSTRUCTIONS REPETER, WAIT, BREAK, PARALLEL, ACTIONS_SYNC et BRANCHE ne nécessitent aucune information venant de l'extérieur. Ainsi, leur interprétation est toujours la même, peu importe la spécialisation de l'interpréteur. Cependant, les éléments tels que TANTQUE, EVENEMENT, CONDITIONNEL et ACTION nécessitent des informations venant de l'extérieur. Ces informations peuvent être de deux natures :

- Soient ce sont des entrées et elles apportent des informations aux programmes,
- Soient ce sont des sorties et le programme peut faire effectuer des actions.

Quatre fonctions permettent de gérer ces entrées et ces sorties :

- `getPerceptionType` : cette fonction prend le nom d'un module et d'une perception en paramètre et renvoie le type de la perception (string, number ou boolean). Cela permet à l'interpréteur de savoir comment évaluer cette perception,
- `getInput` : cette fonction prend le nom d'un module et d'une perception en paramètre et renvoie un objet Perception. Cela permet de connaître les valeurs de la perception et de son coefficient,
- `removeInput` : cette fonction permet de supprimer une perception reçue en prenant en paramètre le nom d'un module et d'une perception. Cela peut être utile si la perception n'est plus valable,
- `setOutput` : cette fonction permet d'appeler un objet Action passé en paramètre afin d'exécuter l'action définie.

Dans le cadre d'ArCo, les perceptions sont gérées par la classe Variables. Ainsi lorsqu'une perception est envoyée par une entité agissante perceptive, la perception est ajoutée dans **Variables**. Lors de l'interprétation des scénarii, l'interpréteur peut interroger l'objet de la classe Variables. Les actions sont gérées par la classe Action. Une action se décompose en trois parties : une liste d'identificateurs, un nom de fonction, une liste de paramètres. Lorsqu'une action doit être effectuée, l'interpréteur cherche dans le fichier de description des actions, l'entité agissante qui doit envoyer la perception. Ensuite, il fait le lien entre l'action et le message à envoyer à l'entité agissante.

Les fonctions précédentes sont gérées de la façon suivante :

- `getPerceptionType` : cette fonction lit le fichier de descriptions des perceptions et récupère dans le fichier le type de la perception passée en paramètre,
- `getInput` : cette fonction lit l'objet **variables** et récupère l'objet Perception qui correspond aux noms du module et de la perception passés en paramètre,
- `removeInput` : cette fonction supprime l'entrée dans l'objet **variables** qui correspond aux noms du module et de la perception passés en paramètre,
- `setOutput` : cette fonction lance l'action passée en paramètre.

H. Ecriture d'un nouveau module pour ArCo

Une bibliothèque MICE a été implémentée en JAVA et en C++ pour faciliter la création de modules MICE. En quelques lignes de code, un développeur peut créer un module de perception ou d'action par ArCo, par exemple. Ce chapitre donne une description simple de la bibliothèque MICE (pour les détails, veuillez-vous référer à la Javadoc). Cette introduction à la bibliothèque est un prérequis aux parties 2 et 3 qui expliquent comment écrire un module de perception et d'action pour MICE. Enfin, à titre d'exemple, la partie 4 montre le module développé pour rendre compatible ArCo à Greta.

A. Bibliothèque MICE disponible

La bibliothèque est composée des neuf paquetages suivants :

- **mice** : c'est le paquetage parent qui contient l'ensemble des classes de la bibliothèque et qui définit l'organisation d'un module de MICE.
- **mice.client** : gère l'ensemble du protocole de communication exigé d'un module client de MICE.
- **mice.runtime** : gère le serveur et le protocole de communication exigé d'un module serveur de MICE. Ce paquetage contient l'exécutable du serveur de MICE.
- **mice.perception** : gère les messages de type perception
- **mice.action** : gère les messages de type action
- **mice.xml** : gère l'ensemble des parseurs XML utiles pour la bibliothèque MICE. Ce paquetage définit quelques méthodes STATIC utilisées par tous les parseurs.
- **mice.xml.actuator** : permet de créer, de lire et d'interpréter le fichier XML qui contient toutes les actions disponibles pour une entité agissante active.
- **mice.xml.sensor** : permet de créer, de lire et d'interpréter le fichier XML qui contient toutes les perceptions disponibles pour une entité agissante perceptive.
- **mice.xml.modules** : permet de créer, de lire et d'interpréter le fichier XML qui contient la liste de tous les modules connus par le serveur.

1. Le paquetage mice

Ce paquetage, présenté par la Figure 50, contient une interface **MICEModuleInterface** et une classe **MICEModule**, implémentant cette interface, qui décrivent la base du fonctionnement des modules de MICE. Le principe est le suivant :

- Un module est défini par un id et un type.
- Pour démarrer un module, il faut utiliser la fonction `startModule()` qui lance le thread qui se met en attente de réception de message. La fonction `treatmentBeforeRun()` est appelé avant le lancement du thread.
- Pour stopper le thread, il faut utiliser la fonction `stopModule()`.
- La communication s'effectue avec des objets de type `IOMessage` (paquetage runtime).

- Les informations envoyées sur le flux d'entrée du module sont récupérées par la fonction `treatAnswer(IOMessage)` qui prend un paramètre un objet `IOMessage` et retourne un objet `IOMessage`. Le message retourné peut être :
 - Le même si le module ne l'a pas transformé (soit il l'a simplement ignoré, soit il l'a consulté sans le modifier).
 - Un nouveau message généré par la transformation du précédent.
 - Un message vide (fonction `makesEmpty()` de la classe `IOMessage`). Dans ce cas, cela indique au serveur qu'il n'y a aucun message à relayer. Le module stoppe la transmission de l'information.
- Les informations sont envoyées sur le flux de sortie du module avec la fonction `sendMessage(IOMessage)`. Pour créer un objet de type `IOMessage` spécifique au protocole de communication MICE, il faut utiliser la fonction `buildMessage(...)`.

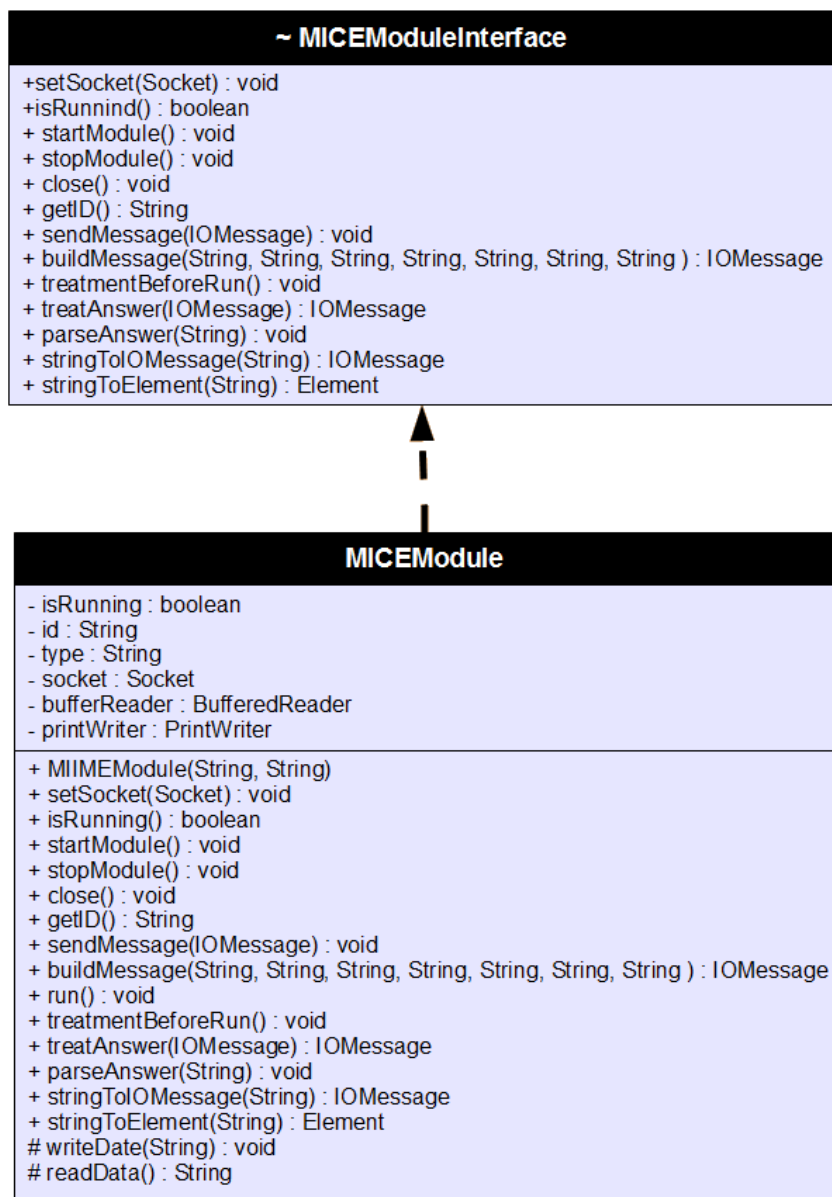


Figure 50 : Bibliothèque MICE - package mice

2. Le paquetage mice.client

Ce paquetage, présenté par la Figure 51, contient une seule classe **MICEClient**. C'est la classe à étendre pour créer un module de MICE. La fonction `treatAnswer(IOMessage)` traite la réception de tous les messages MICE_1 à MICE_15 (cf. Annexe C). Tous les messages qui ne sont pas des messages réservés sont traités par la fonction `treatOtherAnswer()` à étendre lors de la création d'un module de MICE si besoin. Les messages de perception sont traités par la fonction `treatPerceptionMessage` et les messages d'action sont traités par la fonction `treatActionMessage`.

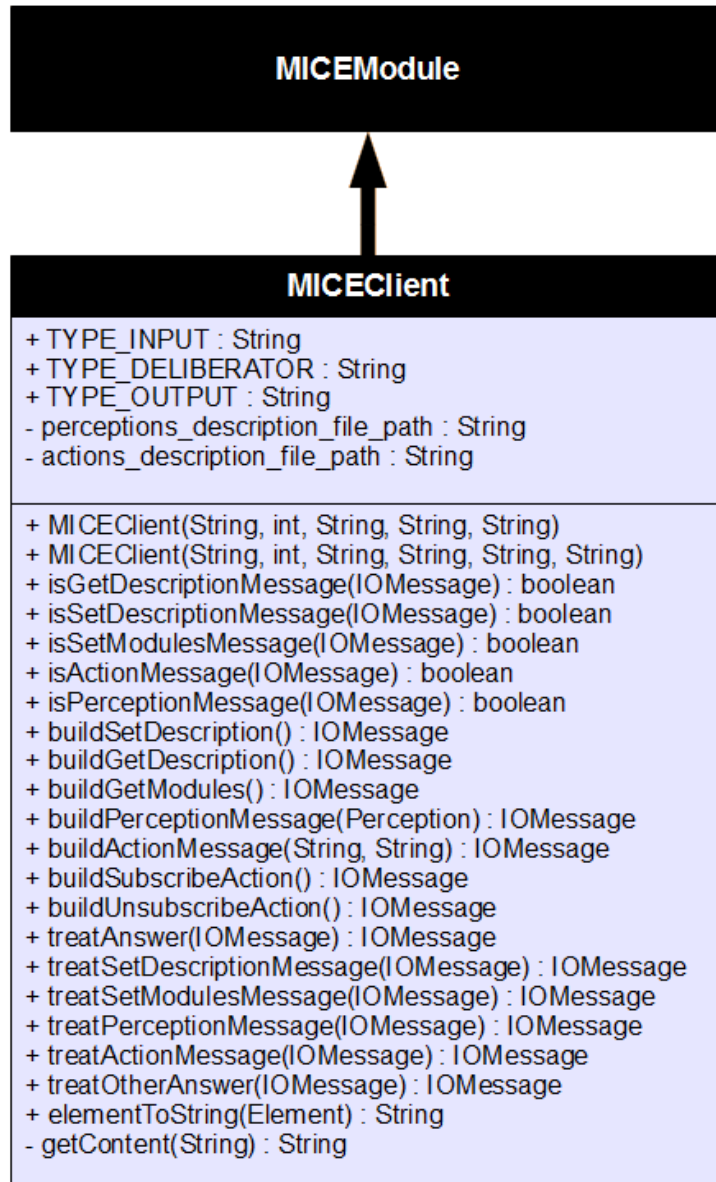


Figure 51 : Bibliothèque MICE - package mice.client

3. Le paquetage `mice.runtime`

Cette bibliothèque, présentée par la Figure 53, met à disposition un serveur fonctionnel dont le point d'entrée est la fonction *main* de la classe `MICERuntime`. Le constructeur prend 4 paramètres :

- Le numéro de port du serveur.
- Le nom du fichier qui contient la liste des modules connus du serveur.
- Le nom du fichier contenant la description de toutes les entités agissantes perceptives.
- Le nom du fichier contenant la description de toutes les entités agissantes actives.

Le serveur de MICE gère le chemin logique de l'information avec l'objet `IOLink` et l'ensemble des clients connectés avec la variable `lesClients`. A chaque fois qu'un client se connecte, le serveur crée un gestionnaire de clients (une instance de la classe `MICEClientManager`) qu'il stocke dans un vecteur. Cet objet est un sous-traitant du serveur qui bénéficie d'un lien direct avec le client. Il s'occupe de la communication entre ce client et le serveur.

Lorsque le gestionnaire de clients reçoit un message du client, si c'est un message d'identification, l'objet `MICEClientManager` affecte sa variable `id`. Ainsi lorsque le serveur appellera la fonction `getID()` du `MICEClientManager`, il aura accès au nom du module associé. Si c'est un message `connect` ou `disconnect`, le serveur ajoute ou retire de l'objet `IOLink`, les modules concernés. Si le message reçu n'est pas un message de `MICE_1` à `MICE_14` (excepté `MICE_13`), alors le serveur relaye le message à tous les modules liés à la sortie du module qui a émis le message. Le serveur agit comme un routeur, il n'effectue jamais aucune modification sur le contenu des messages. Seuls les modules ont un rôle sur le contenu des messages.

4. Le paquetage `mice.xml`

Le paquetage, représenté par la Figure 52, contient une classe `Parser`. Cette classe contient des fonctions statiques utiles pour chaque parseur de la bibliothèque MICE.

Parser
+ <u><code>stringToElement(String) : Element</code></u> + <u><code>elementToString(Element) : String</code></u> + <u><code>getContent(String) : Element</code></u> + <u><code>save(String, String) : void</code></u>

Figure 52 : Bibliothèque MICE - package `mice.xml`

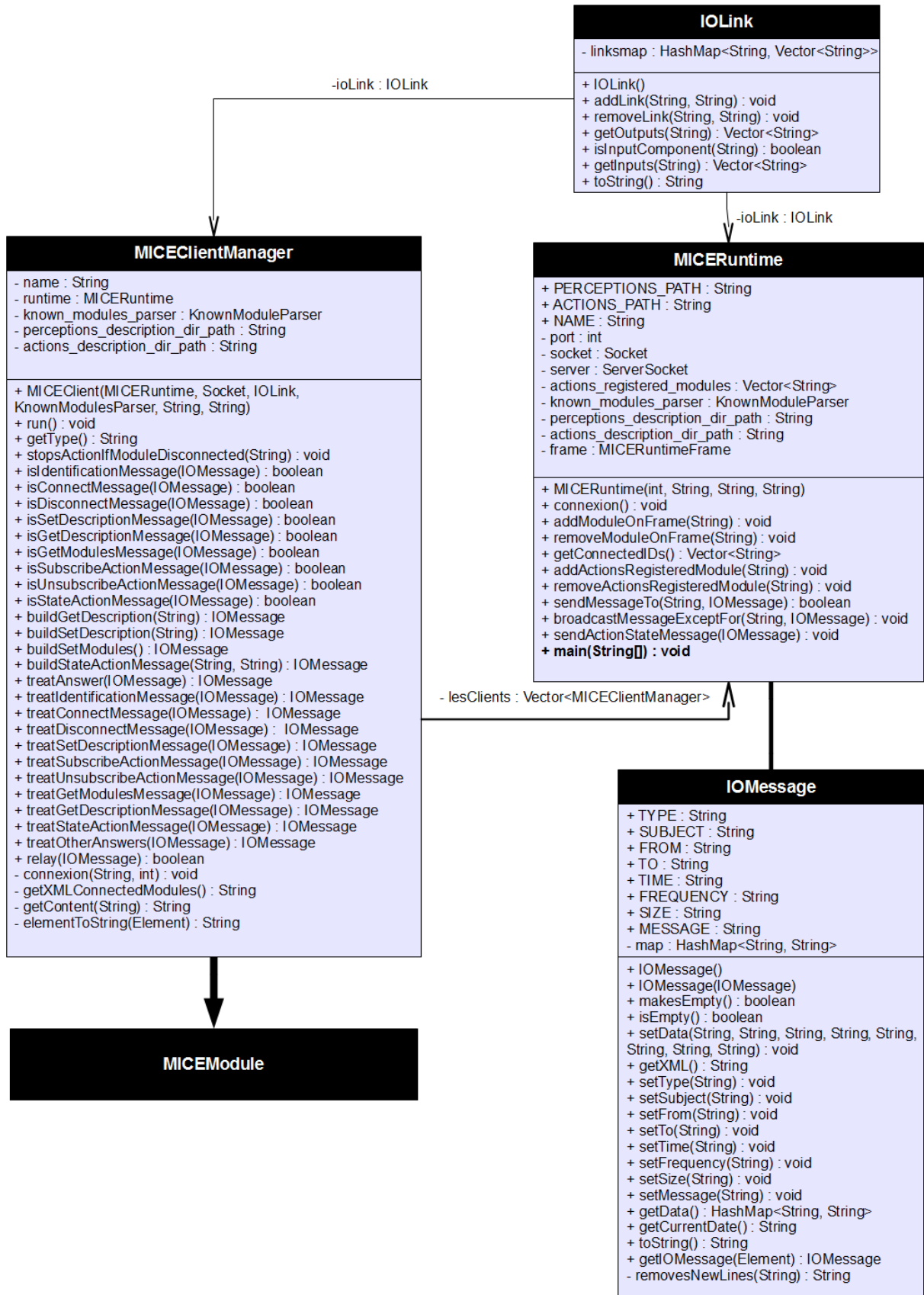


Figure 53 : Bibliothèque MICE - package mice.runtime

5. Le paquetage mice.perception

Cette bibliothèque, présentée par la Figure 54, gère la sérialisation des messages de type Perception. A partir d'une chaîne de caractère, représentant le message XML d'une perception, le parser crée un objet de Perception ou inversement.

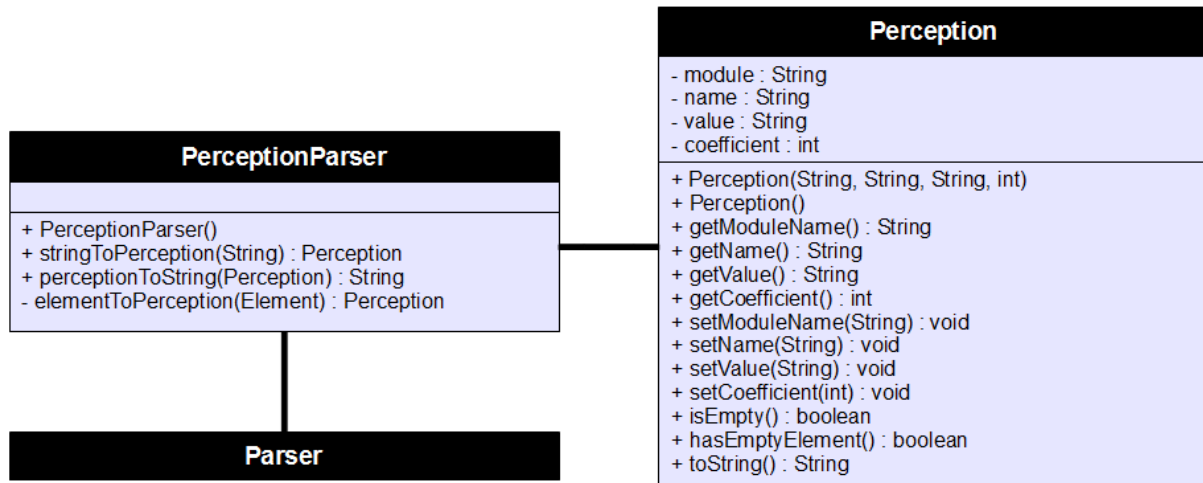


Figure 54 : Bibliothèque MICE - package mice.perception

6. Le paquetage mice.action

Cette bibliothèque, présentée par la Figure 55, gère la sérialisation des messages de type Action. A partir d'une chaîne de caractère, représentant le message XML d'une action, le parser crée un objet de Action ou inversement.

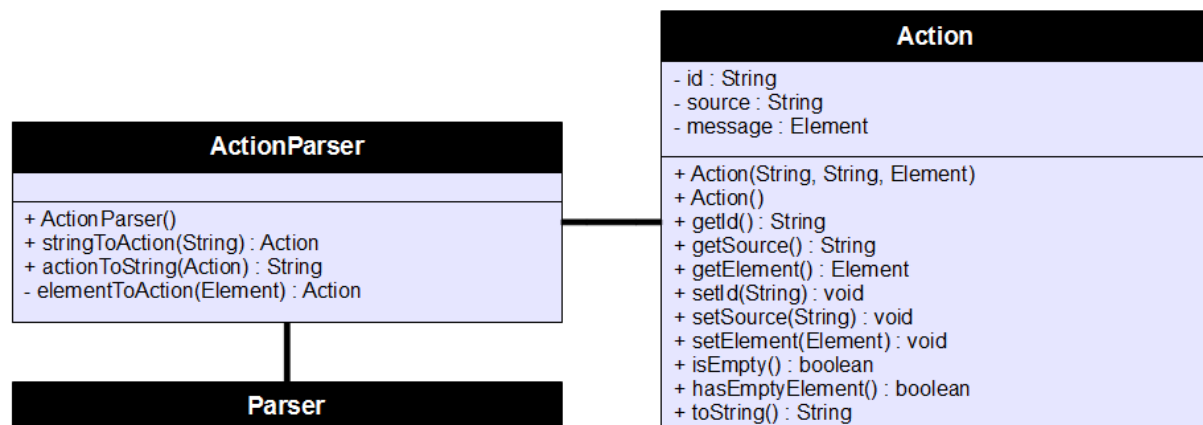


Figure 55 : Bibliothèque MICE - package mice.action

7. Le paquetage mice.xml.actuator

Le paquetage, représenté par la Figure 56, contient les classes qui permettent de gérer les fichiers de description XML des entités agissantes actives.

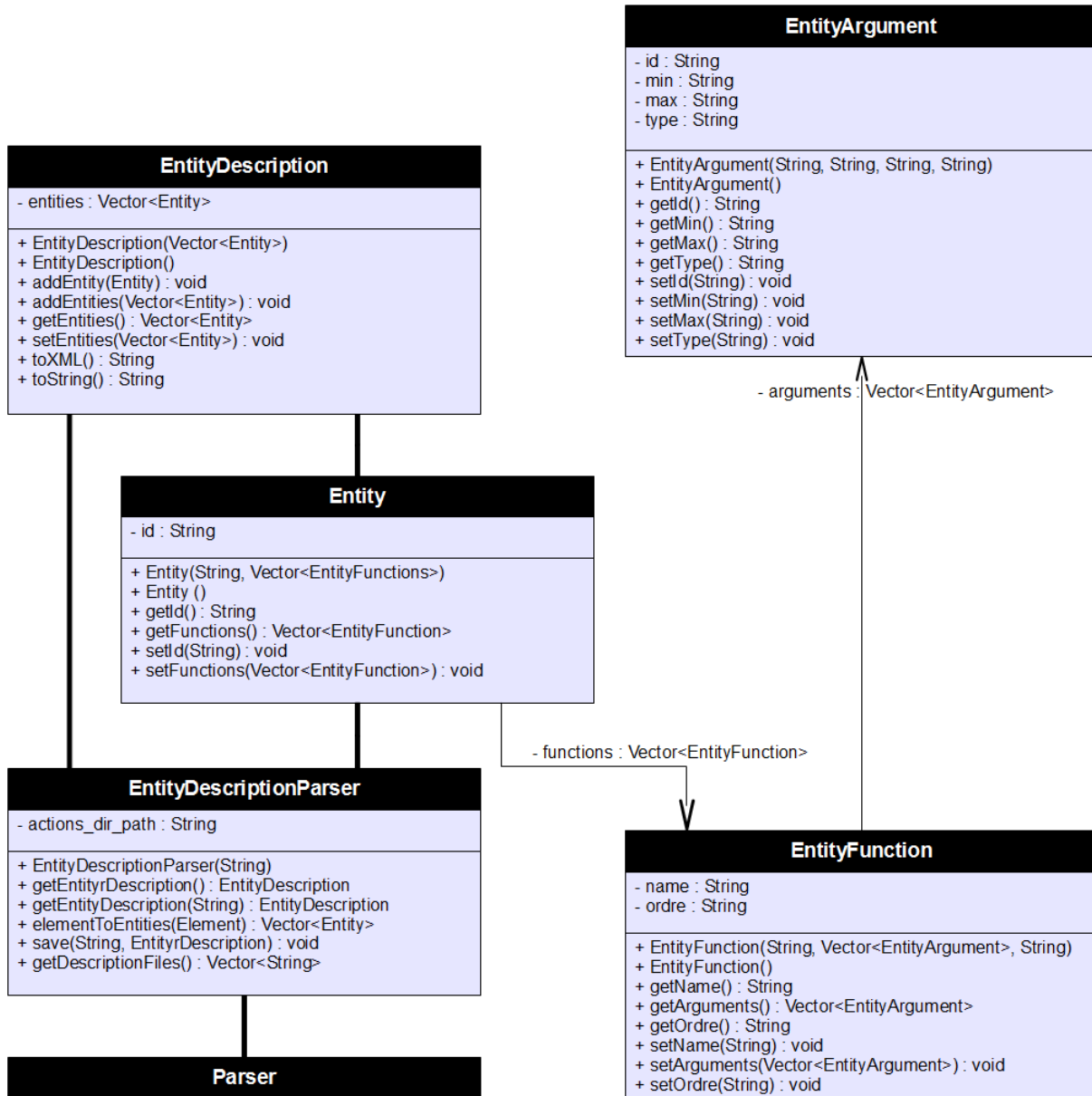


Figure 56 : Bibliothèque MICE - package mice.xml.actuator

8. Le paquetage mice.xml.sensor

Le paquetage, représenté par la Figure 56, contient les classes qui permettent de gérer les fichiers de description XML des entités agissantes perceptives.

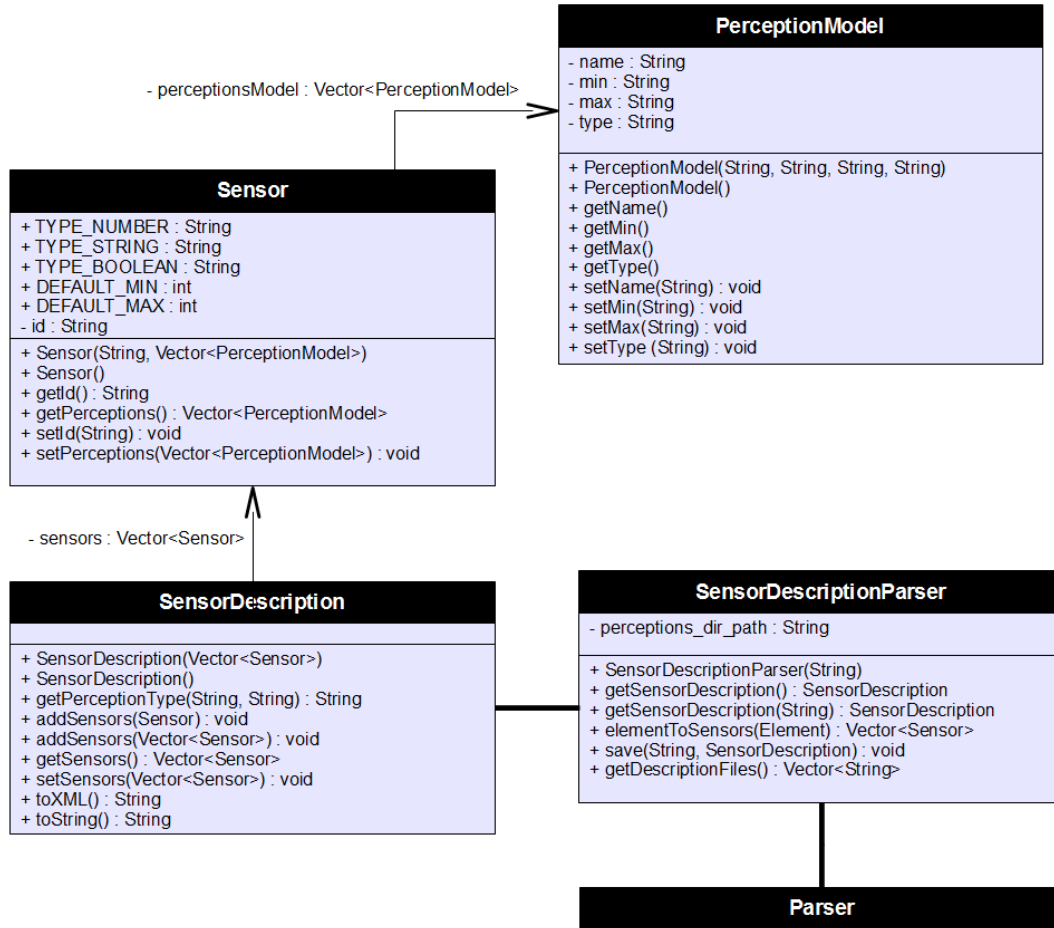


Figure 57 : Bibliothèque MICE - package mice.xml.sensor

9. Le paquetage mice.xml.modules

Le paquetage, représenté par la Figure 58, contient les classes qui permettent de gérer le fichier XML des modules connus du serveur MICE.

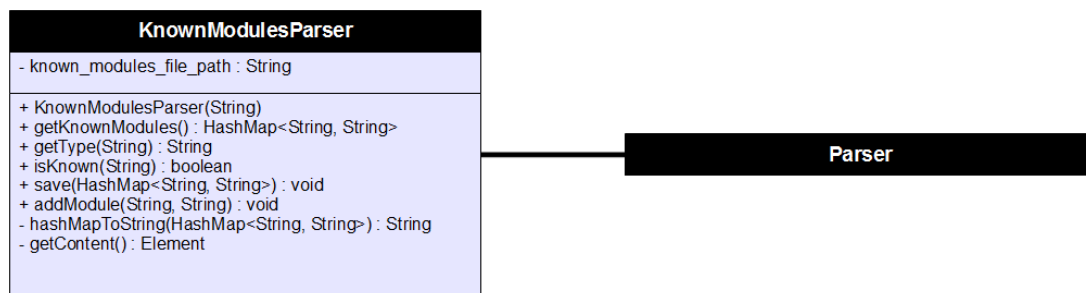


Figure 58 : Bibliothèque MICE - package mice.xml.modules

B. Ecriture d'un module de perception

L'exemple ci-dessous montre le code JAVA qui permet de créer un module de perception. La classe contient un constructeur, une fonction `analyseEnvironnement` et une fonction `main` qui est le point d'entrée du programme.

```
package mice.client;

import mice.client.MICEClient;
import mice.perception.Perception;
import mice.runtime.IOMessage;

public class ModulePerceptif extends MICEClient {

    public ModulePerceptif(String address, int port, String id, String
description_file_path) throws Exception {
        super(address, port, id, MICEClient.TYPE_INPUT, description_file_path);
    }

    public void analyseEnvironnement() {
        //traitement
        Perception p = new Perception(); //à remplir
        IOMessage ioMessage = this.buildPerceptionMessage(p);
        this.sendMessage(ioMessage);
        //traitement
    }

    public static void main(String[] args) {
        ModulePerceptif module;
        try{
            if(args.length==4){
                String adresse = args[0];
                int port = Integer.valueOf(args[1]);
                String id = args[2];
                String perceptions_file_path = args[3];
                module = new ModulePerceptif(adresse, port, id, perceptions_file_path);
                module.startModule();
            }
            else{
                System.out.println("Usage: java -jar ModulePerceptif [adresse] [port] [id]
[perceptions_file_path]");
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Pour qu'un module envoie une perception, il doit créer une perception :

```
Perception p = new Perception();
```

Une fois que la perception contient les informations à envoyer, il faut la transformer en message `IOMessage` :

```
IOMessage ioMessage = this.buildPerceptionMessage(p);
```

Ensuite, le message doit être envoyé au serveur de MICE :

```
this.sendMessage(ioMessage);
```

C. Ecriture d'un module d'action

L'exemple ci-dessous montre le code JAVA qui permet de créer un module d'action. La classe contient

- un constructeur,
- la fonction `treatPerceptionMessage` utilisée si le module doit analyser des perceptions,
- la fonction `treatActionMessage` utilisée si le module doit effectuer des actions,
- la fonction `treatOtherAnswer` utilisée si le module doit analyser d'autres sortes de message,
- une fonction `main` qui est le point d'entrée du programme.

```
package mice.client;

import java.io.IOException;

import org.jdom.JDOMException;

import mice.action.Action;
import mice.client.MICEClient;
import mice.perception.Perception;
import mice.perception.PerceptionParser;
import mice.runtime.IOMessage;

public class ModuleActif extends MICEClient {

    public ModuleActif(String address, int port, String id, String
description_file_path) throws Exception {
        super(address, port, id, MICEClient.TYPE_OUTPUT, description_file_path);
    }

    public IOMessage treatPerceptionMessage(IOMessage ioMessage){
        try {
            String message = ioMessage.getData().get(IOMessage.MESSAGE);
            PerceptionParser parser = new PerceptionParser();
            Perception perception = parser.stringToPerception(message);
            System.out.println(perception);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        }
        //ioMessage.makeEmpty();
        return ioMessage;
    }

    public IOMessage treatActionMessage(IOMessage ioMessage){
        String message = ioMessage.getData().get(IOMessage.MESSAGE);
        ActionParser parser = new ActionParser();
        Action action = parser.stringToAction(message);
        String id_action = action.getId();
        String message = action.getMessage();

        //traitement
        ioMessage.makeEmpty();
        return ioMessage;
    }

    public IOMessage treatOtherAnswer(IOMessage iom){
        String message = iom.getData().get(IOMessage.MESSAGE);
        System.out.println("Message reçu : "+message);
    }
}
```

```
//on relaie le message
return iom;
}

public static void main(String[] args) {
    ModuleActif module;
    try{
        if(args.length==4){
            String adresse = args[0];
            int port = Integer.valueOf(args[1]);
            String id = args[2];
            String actions_file_path = args[3];
            module = new ModuleActif(adresse, port, id, actions_file_path);
            module.startModule();
        }
        else{
            System.out.println("Usage: java -jar ModuleActif [adresse] [port] [id] [actions_file_path]");
        }
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

1. Récupérer une perception

Pour qu'un module traite une perception, il doit d'abord récupérer le message :

```
String message = ioMessage.getData().get(IOMessage.MESSAGE);
```

Il faut ensuite instancier un objet PerceptionParser :

```
PerceptionParser parser = new PerceptionParser();
```

L'objet Perception est créé en appelant la fonction stringToPerception :

```
Perception perception = parser.stringToPerception(message);
```

2. Récupérer une action

Pour qu'un module traite une action, il doit d'abord récupérer le message :

```
String message = ioMessage.getData().get(IOMessage.MESSAGE);
```

Il faut ensuite instancier un objet ActionParser :

```
ActionParser parser = new ActionParser();
```

L'objet Action est créé en appelant la fonction stringToAction :

```
Action action = parser.stringToAction(message);
```

3. Récupérer tout type de messages

Pour qu'un module traite n'importe quel message, il doit récupérer le message, comme vu précédemment, puis effectuer le traitement du message.

D. Exemple : Module MICE associé à Greta

A titre d'exemple, voici le code JAVA du module qui permet de faire effectuer des actions à Greta (version psychone). Le module reçoit des actions dont le message est une chaîne XML au format FML.

Ce module utilise la fonction `treatActionMessage` pour ne recevoir que les messages d'actions. Ensuite, il vérifie que ce message lui est adressé :

```
if (iom.getData().get(IOMessage.SUBJECT).equals(this.id)) {
```

Ensuite, le module vérifie que le message d'action commence par la balise FML [134] puis fait suivre ce message à Greta.

```
package mice.client;

import java.io.IOException;
import java.util.List;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

import mice.action.Action;
import mice.action.ActionParser;
import mice.client.MICEClient;
import mice.runtime.IOMessage;
import com_greta.FMLSender;

public class GretaModule extends MICEClient{

    FMLSender fmlSender;

    public GretaModule(String address, int port, String id, String type, String
filePath) throws Exception {
        super(address, port, id, type, filePath);
        fmlSender = new FMLSender();
        fmlSender.connect("127.0.0.1", 10000);
    }

    public IOMessage treatActionMessage(IOMessage ioMessage){
        String message = ioMessage.getData().get(IOMessage.MESSAGE);
        ActionParser parser = new ActionParser();
        Action action = parser.stringToAction(message);
        String id_action = action.getId();
        String message = action.getMessage();

        if ((iom.getData().get(IOMessage.TYPE).equals("output")) &&
            (iom.getData().get(IOMessage.SUBJECT).equals(this.id))) {

            if (message.startsWith("<fml")) {
                this.sendFML(message);
            }

            this.sendMessage(this.buildActionMessage(id_action, "STOPPED"));
        }

        ioMessage.makesEmpty();
        return ioMessage;
    }
}
```

```
public static void main(String[] args) {
    GretaModule module;
    try {
        if(args.length==5){
            String adresse = args[0];
            int port = Integer.valueOf(args[1]);
            String id = args[2];
            String type = args[3];
            String filePath = args[4];
            module = new GretaModule(adresse, port, id, type, filePath);
            module.startModule();
        }
        else{
            System.out.println("Usage: java -jar GretaModule.jar [adresse] [port] [id]
[type] [filePath]");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void sendFML(String fml){
    fmlSender.sendFML(fml);
}
}
```


I. Configuration de StimCards

A. Configuration des cartes de jeu

Chaque carte est donc associée à un fichier XML. Par exemple, le fichier XML de la Figure 56 du document principal est le suivant :

```
<question label="Qui a chanté l'Aigle noir en 1970 ?"
url="images_cartes/musique.png" type="mcq" category="musique">

  <defs_color>
    <color name="red" red="255" green="0" blue="0" />
    <color name="green" red="0" green="255" blue="0" />
    <color name="blue" red="0" green="0" blue="255" />
    <color name="white" red="255" green="255" blue="255" />
    <color name="black" red="0" green="0" blue="0" />
    <color name="pink_answer" red="255" green="200" blue="200" />
    <color name="green_answer" red="111" green="255" blue="162" />
    <color name="blue_answer" red="30" green="183" blue="255" />
    <color name="yellow_answer" red="255" green="255" blue="128" />
    <color name="blue_background" red="50" green="130" blue="163" />
    <color name="orange_background" red="225" green="160" blue="63" />
  </defs_color>

  <color>
    <background_color color="blue_background" />
    <label_color color="black" />
    <font_color color="black" />
  </color>

  <clues>
    <clue>Il s'agit d'une chanteuse née en 1930</clue>
    <clue>Ce n'est pas Juliette Greco</clue>
  </clues>

  <suggested_answers>
    <answer code="A" url="" code_color="pink_answer"
answer_color="black">Barbara</answer>
    <answer code="B" url="" code_color="green_answer"
answer_color="black">Juliette Greco</answer>
    <answer code="C" url="" code_color="blue_answer" answer_color="black">Edith
Piaf</answer>
    <answer code="D" url="" code_color="yellow_answer"
answer_color="black">Petula Clark</answer>
  </suggested_answers>

  <>true_answers>
    <code>A</code>
  </true_answers>

</question>
```

Chaque question lue par StimCards contient une balise racine **question** qui groupe les informations principales :

- **label** : contient la question à poser à l'utilisateur
- **url** : contient l'adresse d'une image à associée à la question ou une chaîne vide s'il n'y a aucune image à afficher.
- **type** : contient le type de la question
- **category** : contient une catégorie de questions. Cela peut être divertissement, sciences, maths...

Le fichier contient également un ensemble de définition de couleurs, contenu dans la balise **defs_color**. Chaque couleur est définie par une balise **color** qui contient les attributs :

- **name** : contient le nom de la couleur qui pourra être utilisé dans le reste du fichier XML.
- **red** : contient la valeur, entre 0 et 255, du rouge.
- **green** : contient la valeur, entre 0 et 255, du vert.
- **blue** : contient la valeur, entre 0 et 255, du bleu.

Le fichier contient également la description des couleurs utilisés pour l’affichage de la question, contenue dans la balise **color**. La balise **color** contient trois balises :

- **background_color** : indique la couleur de fond de la zone d’affichage de la question.
- **label_color** : indique la couleur dans laquelle sont écrites les informations qui ne sont pas contenus dans le fichier XML. Sur la Figure 56 du document principal, il s’agit des mots : « Questions » et « Réponses » et des indices.
- **font_color** : indique la couleur dans laquelle sont écrites les informations contenues dans le fichier XML, hormis celles dont la couleur est précisée. Dans l’exemple, il s’agit de la couleur de police des indices.

Chacune de ces trois balises contiennent un attribut **color** dont la valeur est le nom d’une des couleurs définies dans la balise **defs_color**. Cette séparation des couleurs permet de rendre plus simple la saisie de fichiers XML de questions par des non-experts en informatique.

Le fichier contient un ensemble d’indices contenus dans la balise **clues**. Lorsque l’interface affiche la question, les indices ne sont pas visibles. L’utilisateur peut afficher les indices les uns après les autres en cliquant sur le bouton d’aide s’il joue seul avec l’ordinateur (ou avec le dispositif prévu lorsqu’il joue avec ArCo).

Chaque indice est contenu dans des balises **clue** et est une chaîne de caractère

Dans le cas d’une question à choix multiple, les réponses proposées à l’utilisateur sont contenues dans la balise **suggested_answers**. Chaque réponse proposée est contenu dans la balise **answer** dont les attributs sont :

- **code** : le code représente une lettre, un symbole qui permet de distinguer la réponse. Par exemple, A, B, C ou D dans l’exemple.
- **url** : contient l’adresse d’une image à associée à la question ou une chaîne vide s’il n’y a aucune image à afficher.
- **code_color** : indique la couleur dans laquelle il faut écrire le code spécifié par la balise **code**. La couleur est une représentée par le nom d’une des couleurs contenue dans la balise **defs_color**.
- **answer_color** : indique la couleur dans laquelle la réponse sera écrite. La couleur est également représentée par le nom d’une des couleurs contenue dans la balise **defs_color**.

La réponse proposée à l’utilisateur est le texte de la balise **answer**.

Le fichier contient la réponse à la question dans la balise **true_answers**.

Chaque bonne réponse est contenue dans une balise **code** et fait référence à l’attribut **code** d’une des balises **answer**. Il est possible qu’une question ait plusieurs réponses correctes.

B. Configuration de l'interface

Chaque utilisateur peut configurer l'interface de StimCards, par exemple si une charte graphique doit être respectée ou pour maintenir le logiciel si le logo ou le nom change. Cette configuration se fait avec un fichier XML que l'interface charge au démarrage. Le fichier, chargé par l'interface de la Figure 56 du document principal, est le suivant :

```
<stimcards>
  <title>StimCards </title>
  <logo>images/logo_petit.png</logo>

  <color>
    <background red="0" green="0" blue="255" />
    <foreground red="0" green="255" blue="0" />
  </color>

  <font>
    <title_font>
      <name>Comic sans ms</name>
      <size>70</size>
    </title_font>
    <label_font>
      <name>Comic sans ms</name>
      <size>30</size>
    </label_font>
    <default_font>
      <name>Comic sans ms</name>
      <size>24</size>
    </default_font>
  </font>

  <label_question url="" >Question : </label_question>
  <label_answer url="">Réponses : </label_answer>
  <label_help url="images/aide.png"></label_help>

</stimcards>
```

Le fichier de configuration de l'interface est défini par la balise racine **stimcards**. Cette balise contient deux balises qui permettent de renseigner le nom et le logo de l'interface :

- **title** : contient le titre de l'interface qui est aussi utilisé en tant que nom de l'application.
- **logo** : adresse de l'image à afficher à côté du titre

Le fichier de configuration permet de déterminer la couleur de fond et la couleur de police du bandeau de titre de l'interface. Ces informations sont contenues dans la balise **color** et contient deux balises :

- **background** : permet de spécifier les valeurs rouge, vert et bleu de la couleur de fond du bandeau de titre.
- **foreground** : permet de spécifier les valeurs rouge, vert et bleu de la couleur de la police de titre.

Le fichier de configuration permet également de fixer le nom de la police utilisée dans l'interface ainsi que sa taille. Cela permet d'adapter la fenêtre aux utilisateurs qui l'utilisent. Ces informations sont contenues dans la balise **font**. Cette balise contient trois balises :

- **title_font** : cette balise permet de spécifier la taille et le nom de la police utilisée dans le bandeau de titre.
- **label_font** : cette balise permet de spécifier la taille et le nom de la police utilisée par les informations non contenues dans le fichier XML des questions (Questions, Réponses et indices dans

la Figure 56 du document principal).

- **default_font** : cette balise permet de spécifier la taille et le nom de la police utilisée pour afficher les informations du fichier XML des questions, comme la question et les propositions de réponses.

Chacune de ces balises contient deux balises :

- **name** : contient le nom de la police à utiliser.
- **size** : contient la taille en pt de la police à utiliser.

Il est également possible de configurer les chaînes de caractères ou les images utilisées par l'interface avec trois balises :

- **label_question** : cette balise concerne la chaîne de caractère ou image affichée avant l'intitulé de la question.
- **label_answer** : cette balise concerne la chaîne de caractère ou image affichée avant les propositions de réponses.
- **label_help** : cette balise concerne la chaîne de caractère ou image affichée pour demander un indice.

Ainsi, l'utilisateur choisit si une chaîne de caractère est affichée en la saisissant en tant que texte de la balise ou si c'est une image qui doit être affichée en saisissant l'adresse dans l'attribut **url** de la balise.

J. Evaluation sur la gestuelle d'un robot

Ces deux expérimentations ont fait l'objet de l'article [162].

Le Chapitre II du document principal a montré qu'il existe beaucoup de robots, de différentes formes, de différentes couleurs, avec différents objectifs, qui interagissent avec l'homme dans son quotidien. Certains robots sont mieux acceptés que d'autres et nous nous intéressons aux conditions qui favorisent leur acceptation par l'homme. Pour apporter un élément de réponse, l'étude présentée dans ce chapitre se concentre sur l'importance de la gestuelle d'un robot en fonction de son discours. La première étude mesure la crédibilité et la sincérité d'un robot en fonction de sa gestuelle lors de l'expression d'une seule phrase dite par le robot. La deuxième étude mesure la crédibilité et la sincérité d'un robot lors de quatre dialogues. Ces deux études ont pour objectif de vérifier si la gestuelle du robot participe à son acceptation.

A. Importance de l'adéquation entre le discours et les gestes

Cette étude a été réalisée en reprenant le protocole expérimental de Jérémy Rivière et Sylvie Pesty [163] dans leur étude sur les agents virtuels. L'objectif était de pouvoir évaluer le robot d'une part afin de comparer l'acceptation des agents virtuels et des robots d'autre part. Dans cette étude, le robot ou l'agent virtuel effectuait trois séries de gestes différents pour une même phrase. Il était mis en présence d'un observateur qui donnait ensuite son ressenti. L'expérimentation s'est déroulée pendant une journée, lors d'un concours de robotique, Robofesta.

1. Participants

Le concours Robofesta est un évènement qui amène des participants venant de collèges et de lycées de Bretagne. C'est un public qui a quelques notions des robots, ainsi il n'était pas nécessaire de faire une explication préalable sur la robotique et il était possible de faire des courtes évaluations pour libérer rapidement les participants. Ceux-ci ont été sélectionnés sur la base du volontariat parmi les concourants et les spectateurs de l'évènement. Soixante personnes ont participé à l'expérimentation (17 femmes et 43 hommes) âgées de 10 à 57 ans (moyenne de 18,1 ans).

2. Méthode

L'expérimentation s'est déroulée dans un stand isolé du concours, où les participants pouvaient se rendre en dehors de leurs épreuves. Le robot était positionné debout sur la table. La première partie de l'étude consistait à expliquer la consigne aux participants et à leur distribuer un questionnaire (cf. Annexe O). Ensuite, le robot exécutait des mouvements en prononçant des phrases. Individuellement, chaque participant observait le robot et donnait son impression en remplissant un questionnaire.

Equipement : La Figure 59 présente le cadre expérimental de l'expérimentation. Le robot se trouvait face aux participants. Il était relié à un ordinateur qui permettait de le piloter. L'expérimentateur était assis face à l'ordinateur pour lancer les mouvements. Le participant se tenait debout, face au robot pour lui assurer une bonne observation.

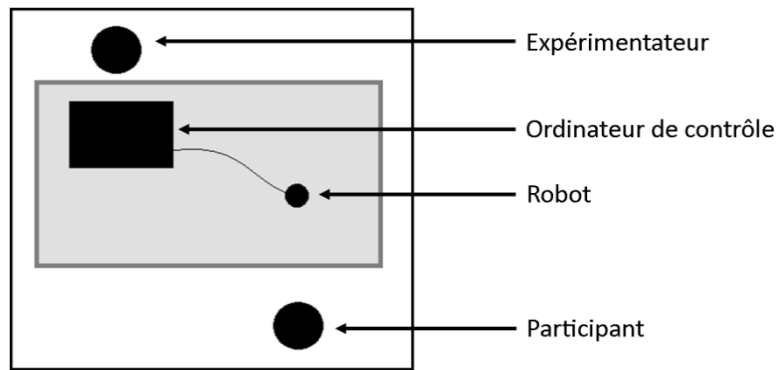


Figure 59 : Cadre expérimental de la première étude

Déroulement de l'expérimentation : la première partie de l'expérimentation consistait à expliquer aux participants le contexte de l'évaluation. Il s'agissait d'une sorte de jeu de rôle présenté de la façon suivante :

Ce soir vous avez prévu d'aller au cinéma, sans avoir décidé quel film aller voir. Vous décidez de demander son avis au robot qui vous conseille d'aller voir un film car il a de bonnes critiques. Vous écoutez son conseil et vous partez au cinéma.

Au retour du cinéma, il y a deux possibilités :

- *Scénario A : Vous n'avez pas du tout aimé le film proposé.*
- *Scénario B : Vous avez beaucoup aimé le film proposé.*

Les consignes indiquaient que les participants devaient observer six mouvements du robot. Les trois premiers mouvements jouaient le scénario A et les participants devaient observer la réaction du robot lorsqu'il apprenait que le participant n'avait pas du tout aimé le film. Les trois mouvements suivants jouaient le scénario B et les participants devaient observer la réaction du robot lorsqu'il apprenait que le participant avait bien aimé le film.

Lors du scénario A, le robot disait : «*Je suis vraiment désolé. Ce film avait de bonnes critiques. Je pensais qu'il te plairait.* ». Et lors du scénario B, le robot disait : «*Alors tu as aimé le film. Je suis content de t'avoir si bien conseillé.* »

Les mouvements proposés pour les deux scénarii étaient identiques.

Le mouvement 1, illustré par la Figure 60, représentait le comportement que nous proposons pour symboliser l'excuse. Le robot débutait chacun de ses mouvements par une expression neutre (vignette 1). Dans le cas du mouvement 1, il commençait par baisser la tête (vignette 2) puis il joignait ses bras et effectuait un gauche-droite du bassin (vignette 3). A la fin du mouvement, le robot retournait en position neutre (vignette 4).

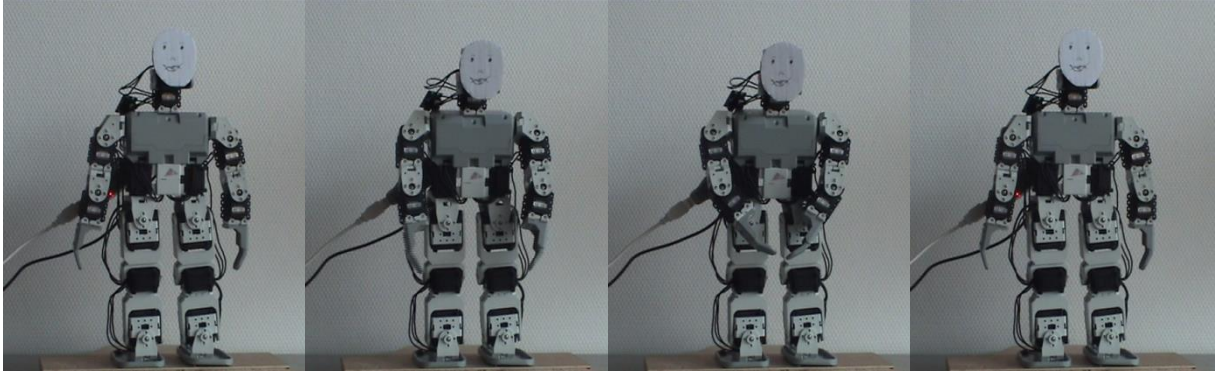


Figure 60 : Mouvement 1 représentant une gestuelle d'excuse

Le mouvement 2, illustré sur la Figure 61, était une expression neutre très légèrement négative. En partant de la position neutre (vignette 1), le robot baissait très légèrement la tête et effectuait une petite vague avec ses bras (vignette 2). Il remontait ensuite la tête (vignette3) pour retourner en position neutre (vignette 4).

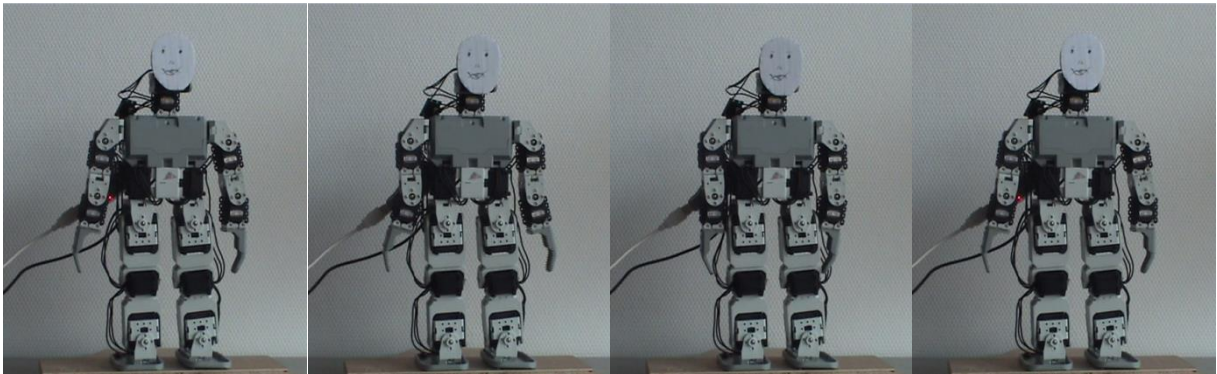


Figure 61 : Mouvement 2 représentant une gestuelle neutre

Le mouvement 3, illustré sur la Figure 62, représentait le comportement que nous proposons pour symboliser une joie intense. Le robot effectuait de grands mouvements de bras tout en bougeant le bassin, le faisant s'appuyer sur une jambe, puis sur l'autre, comme s'il dansait un peu.

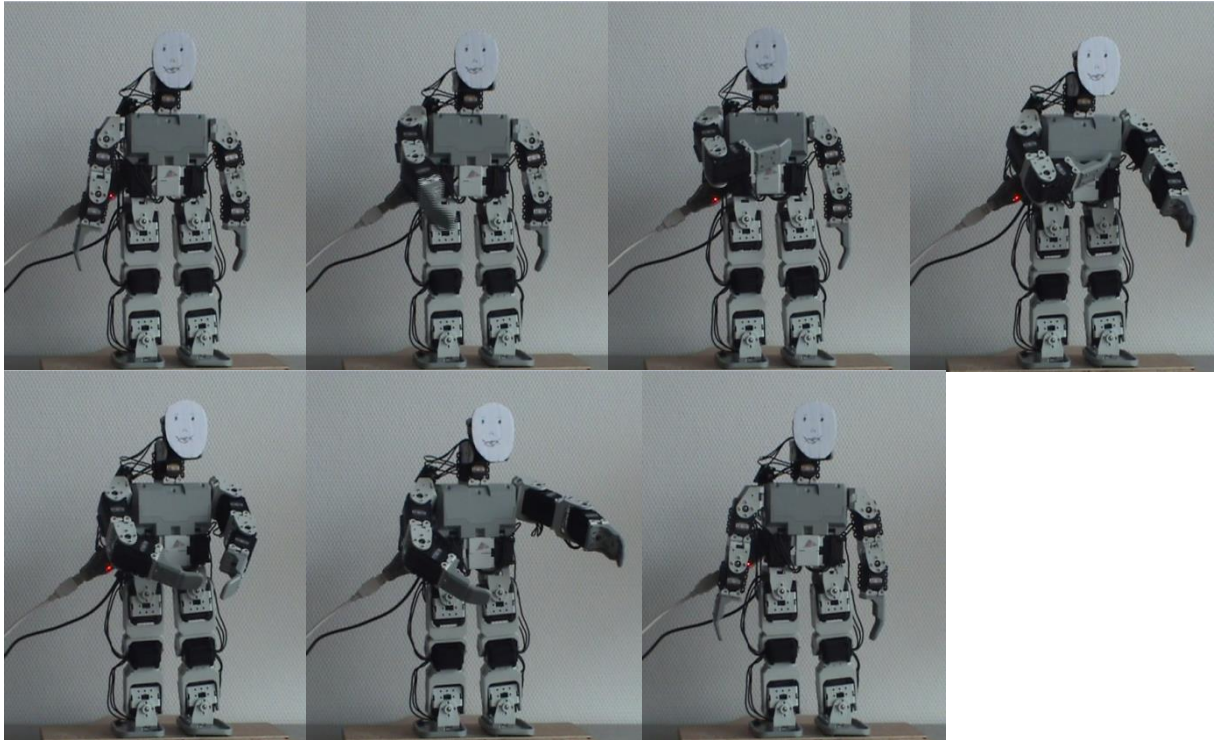


Figure 62 : Mouvement 3 représentant une gestuelle de joie

La situation congruente pour le scénario A était le mouvement 1 et pour le scénario B le mouvement 3.

Après avoir vu chacun des mouvements, les participants devaient dire dans quelle mesure le robot était sincère et crédible :

- Sincère : Le robot semble-t-il exprimer ce qu'il pense ?
- Crédible : Son expression vous paraît-elle plausible ?

3. Collecte de données et analyses

Après chaque mouvement, les observateurs devaient répondre à un questionnaire, représenté par le Tableau 4. Pour chacun des mouvements, ils devaient indiquer si le robot leur semblait sincère et crédible en utilisant l'échelle de Likert [164] suivantes : Pas du tout, Plutôt pas, Plutôt, Tout à fait. Le neutre avait volontairement été retiré du questionnaire pour forcer les participants à trancher. En effet, nous voulions éviter la tentation du neutre qui n'aurait pas permis de répondre à la question. Dans le tableau chaque mouvement est codé par la lettre du scénario et par le numéro du mouvement. Par exemple, A1 indique que c'est le mouvement 1 du scénario A qui a été joué.

Tableau 4 : Questionnaire de la première étude

Garçon/Fille

Age : _____

A1	Sincère ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
	Crédible ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
A2	Sincère ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
	Crédible ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐

A3	Sincère ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
	Crédible ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
B1	Sincère ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
	Crédible ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
B2	Sincère ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
	Crédible ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
B3	Sincère ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐
	Crédible ?	Pas du tout ☐	Plutôt pas ☐	Plutôt ☐	Tout à fait ☐

Les analyses statistiques ont été réalisées avec le logiciel Minitab 15®. Le test du Chi Deux a permis de vérifier les réponses significatives. Le seuil de significativité (p) a été fixé à 0,05.

4. Résultats

La Figure 63 : Résultats sur la crédibilité et la sincérité des mouvements du robot en fonction des scénarii A et B présente les résultats qui sont détaillés dans cette partie.

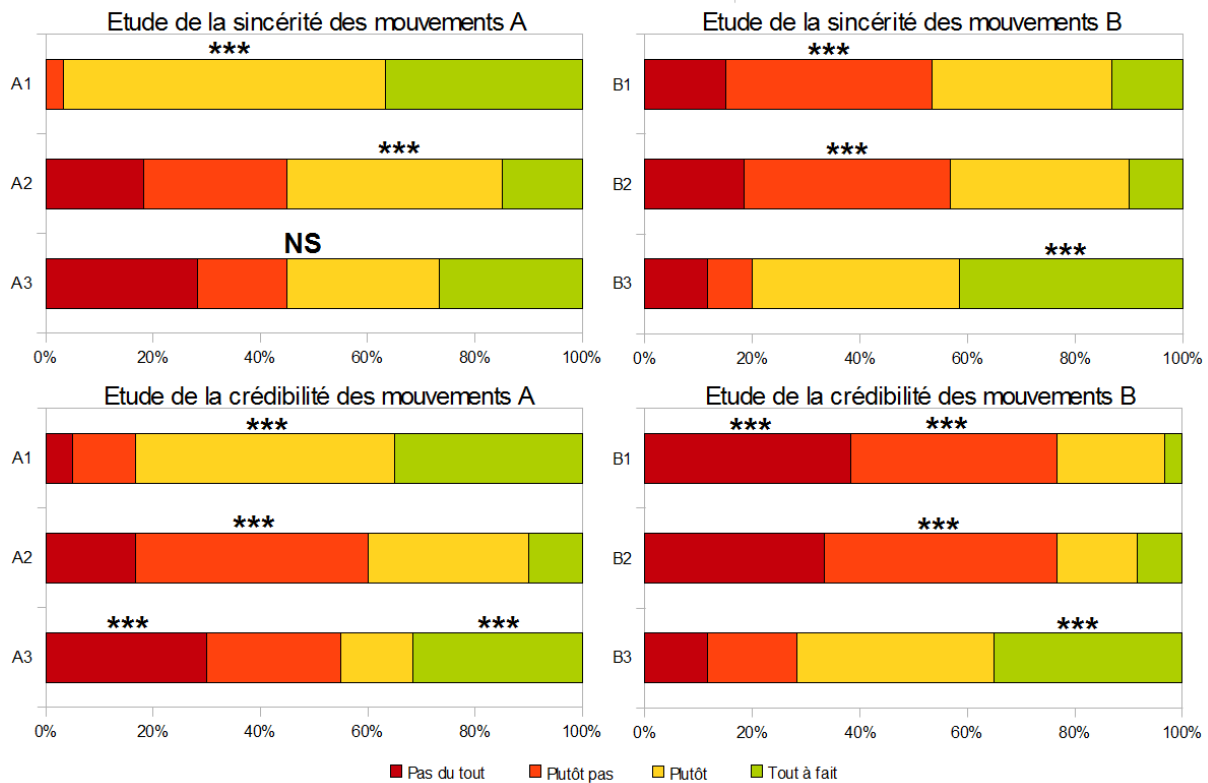


Figure 63 : Résultats sur la crédibilité et la sincérité des mouvements du robot en fonction des scénarii A et B

Niveau de significativité : * $p < 0,05$, ** $p < 0,01$, *** $p < 0,001$

NS : non significatif (Test du Chi-deux).

Concernant le scénario A, les participants n'ont pas répondu au hasard ($X^2=189,1$, $dl=33$, $p<0.001$). Le Tableau 5 : Résultats obtenus pour le scénario A de la première étude. présente les résultats significatifs obtenus pour chacun des trois mouvements.

Le mouvement 1 (Excuse) est celui qui a été jugé comme étant le plus sincère et le plus crédible. C'est ce mouvement qui a fait l'unanimité contrairement aux deux autres mouvements dont les réponses ont été plus contrastées.

Le mouvement 2 a été le plus souvent trouvé plutôt sincère. Néanmoins, il faut noter des réponses contrastées puisque si 18.3% l'ont estimé pas du tout sincère, 15% l'ont tout à fait estimé sincère. Ce mouvement a été plus fréquemment estimé pas du tout crédible.

Le mouvement 3 n'a pas donné de résultats concernant la sincérité du robot. Cependant, il est à noter une différence selon le genre : les femmes ont significativement plus fréquemment estimé le robot "pas du tout sincère" par rapport aux hommes (41,2% des femmes et 23,3% des hommes; $X^2=10.5$, $dl=3$, $p=0.015$). La crédibilité a été très contrastées, 61,67% des réponses se partagent les réponses « pas du tout » et « tout à fait », alors que les situations intermédiaires sont moins représentées (25% et 13,3%)

Tableau 5 : Résultats obtenus pour le scénario A de la première étude.

	Sincère	Crédible
A1	60% plutôt 36,7% tout à fait $X^2=132.7$, $dl=3$, $p<0.001$	48,3% plutôt 35% tout à fait $X^2=63.0$, $dl=3$, $p<0.001$
A2	18,3% pas du tout 40% plutôt 15% tout à fait $X^2=20.1$, $dl=3$, $p<0.001$	43% pas du tout $X^2=39.7$, $dl=3$, $p<0.001$
A3	Aucune différence statistique $X^2=5.9$, $dl=3$, $p=0.117$	30% pas du tout 31,67% tout à fait $X^2=12.4$, $dl=3$, $p=0.006$

Concernant le scénario B, les participants n'ont pas répondu au hasard ($X^2=88.4$, $dl=15$, $p<0.001$). Le Tableau 6 présente les résultats significatifs obtenus pour chacun des trois mouvements.

Le mouvement 1 est contrastée puisque un tiers des participants ont trouvé le robot plutôt sincère alors qu'un autre tiers le trouvait plutôt pas sincère. La réponse à la question sur la crédibilité est plus marquée puisque très peu de participants ont trouvé le robot tout à fait crédible.

Le mouvement 2 a donné à peu près le même ressenti que le mouvement 1 puisqu'un tiers des participants a trouvé le robot plutôt sincère et un autre tiers plutôt pas sincère. Et comme le mouvement 1, la crédibilité est d'avantage marquée puisque très peu de participants ont trouvé le robot tout à fait crédible.

Le mouvement 3 est celui qui a été jugé comme étant le plus sincère et le plus crédible.

Tableau 6 : Résultats obtenus pour le scénario B de la première étude

	Sincère	Crédible
B1	38,88% plutôt pas 33,33% plutôt $X^2=26$, $dl=3$, $p<0.001$	38,3% pas du tout 38,3% plutôt pas 3,3% tout à fait $X^2=46.7$, $dl=3$, $p<0.001$
B2	38,3% plutôt pas 33,3% plutôt $X^2=29.9$, $dl=3$, $p<0.001$	33,3% pas du tout 43,3% plutôt pas 8,3% tout à fait $X^2=42.7$, $dl=3$, $p<0.001$

B3	38,3% plutôt	36,7% plutôt
	41,7% tout à fait	41,7% tout à fait
	$\chi^2=56.5$, $df=3$, $p<0.001$	$\chi^2=26.2$, $df=3$, $p<0.001$

5. Discussion

Concernant le scénario A, le robot s'excusait d'avoir mal conseillé le participant. Le robot a été clairement jugé crédible et sincère lorsqu'il a effectué le mouvement 1 (Excuse). Cette situation est congruente et comprise par un panel large de personne. Pour les mouvements 2 (Neutre) et 3 (Joie), les réponses sont contrastées. Les personnes ne semblent pas réussir à se positionner. On peut imaginer que les réponses mitigées reflètent un niveau d'hésitation due à une incompréhension de la situation. Par exemple, pour la crédibilité du mouvement 3, le choix semble être fait au hasard (répartition équivalente des réponses dans les 4 choix). Ainsi, les mouvements 2 et 3 semblent difficiles à évaluer par les participants. L'intention du robot semble peu claire.

Concernant le scénario B, le robot se « félicitait d'avoir si bien conseillé le film ». Les participants ont jugé le mouvement 3 crédible et sincère. C'est la situation congruente qui a donc été choisie par les participants. Les deux autres mouvements ont été jugés fortement non crédibles alors que la sincérité est plus mitigée. Il semble donc que la sincérité soit plus difficile à évaluer que la crédibilité dans des contextes plus mitigés.

De façon générale, les participants ont plus facilement pensé que le robot semblait exprimer ce qu'il pensait même si son expression ne leur paraissait pas plausible. Dans les deux scénarii, les situations congruentes ont été choisies sans équivoque. Concernant les mouvements non congruents, il ne semble pas y avoir d'hésitation dans le cas d'une émotion positive. Au contraire, dans le cas d'une émotion négative, le robot est tout de même jugé plutôt sincère, même si sa crédibilité est diminuée et même si sa gestuelle est positive. De plus, le mouvement 2, qui était un neutre négatif n'a pas été retenu. Pourtant, le robot baissait la tête et effectuait un léger mouvement des mains contre ses jambes. Le mouvement 3 était exagéré. Ainsi puisque c'est le mouvement exagéré qui a été choisi, et puisque la sincérité est la même pour un mouvement neutre ou pour un mouvement positif, il semble important d'exagérer la gestuelle, dans le cas d'une émotion négative, pour rendre le robot d'avantage crédible et sincère. Rien ne peut être conclu sur la gestuelle d'une émotion positive.

6. Comparaison entre agent virtuel et robot

Le Tableau 7 présente les résultats obtenus par l'agent virtuel [163] et par le robot à propos de leur sincérité et de leur crédibilité en fonction des mouvements d'excuse et de joie. En comparant ces résultats, on remarque que les résultats du robot sont au moins aussi bons que ceux de l'agent virtuel. En moyenne, le robot semble plus crédible et sincère qu'un agent virtuel. Cela amène à la question suivante : est-ce qu'un robot est un meilleur partenaire social qu'un agent virtuel ? Pour répondre à cette question, une étude a été réalisée pour comparer l'impact d'un robot et d'un agent virtuel sur l'interaction homme-machine (voir le chapitre V.D).

Tableau 7 : Comparaison de la sincérité et de la crédibilité d'un agent virtuel et du robot

		Agent Virtuel (GRETA)	ROBOT
EXCUSE	SINCERITE	65,00%	96,67%
		65,00%	96,67%
	CREDIBILITE	70,00%	83,33%
		70,00%	83,33%

JOIE	SINCERITE	74,00%	80,00%
		74,00%	80,00%
	CREDIBILITE	78,00%	71,67%
		78,00%	71,67%

B. Crédibilité et sincérité d'un robot malgré quelques incohérences dans la communication

Cette étude s'est également inspirée d'un protocole expérimental de Jérémy Rivière et Sylvie Pesty. L'objectif de leur étude était d'évaluer les capacités expressives de leur agent virtuel en fonction d'un certain nombre d'actes conversationnels au cours d'une série de dialogue entre l'homme et l'agent virtuel. L'étude que nous avons réalisée est un peu différente. Notre travail n'est pas de déterminer le meilleur comportement non verbal en fonction d'un contexte mais d'étudier les conditions d'acceptation d'un robot. Les résultats de l'étude précédente ont montré que la gestuelle du robot doit être cohérente au discours prononcé pour que le robot paraisse sincère et crédible. La question qui se pose à la suite de ce résultat est la suivante : est-ce qu'un robot est rejeté si sa gestuelle est, de temps en temps, incohérente avec son discours ? Est-ce que des incohérences ont un impact sur la perception de l'humain ?

Pour répondre à cette question, une expérimentation a été mise en place. Elle consistait à demander à un participant d'observer un robot et un expérimentateur dialoguer ensemble, dans 4 contextes différents. De façon aléatoire le robot exprimait une gestuelle non adaptée à son discours.

1. Participants

Cette étude a été réalisée par 26 personnes (9 femmes et 17 hommes) âgées de 18 à 28 ans (moyenne de 22,5 ans). Les participants n'avaient pas de connaissance spécifique en robotique. Ils étaient étudiants en informatique, mathématiques et statistiques et étaient de niveau bac+2 et bac+4. Le choix de ces différences entre les cursus des étudiants avait pour objectif d'éviter d'obtenir des réponses biaisées par un environnement commun.

2. Méthode

L'expérimentation s'est déroulée dans une pièce dédiée. Chaque participant était isolé dans cette pièce avec l'expérimentateur et le robot. Les participants ont été choisis sur la base du volontariat parmi les étudiants de deuxième année d'IUT et de première année de Master. Le robot était positionné debout sur la table comme dans l'étude précédente. La première partie de l'étude consistait à expliquer la consigne aux participants et à leur faire prendre connaissance du questionnaire à remplir. Ensuite, les dialogues s'enchaînaient et les participants devaient remplir le questionnaire pendant le déroulement de l'expérimentation. Il était très difficile pour les participants de suivre et remplir le questionnaire en même temps. De cette façon, l'étude reproduisait au maximum des conditions réelles où l'on ne peut pas mettre pause lorsque quelqu'un parle et où l'on se fait une perception globale d'une personne avec qui l'on interagit.

Equipement : La Figure 64 présente le cadre expérimental de l'étude. Le robot était positionné debout sur la table, face à l'expérimentateur. Le robot de contrôle était positionné sur sa gauche et servait à lancer l'interaction. Le participant était un peu à l'écart de façon à pouvoir observer le robot et l'expérimentateur.

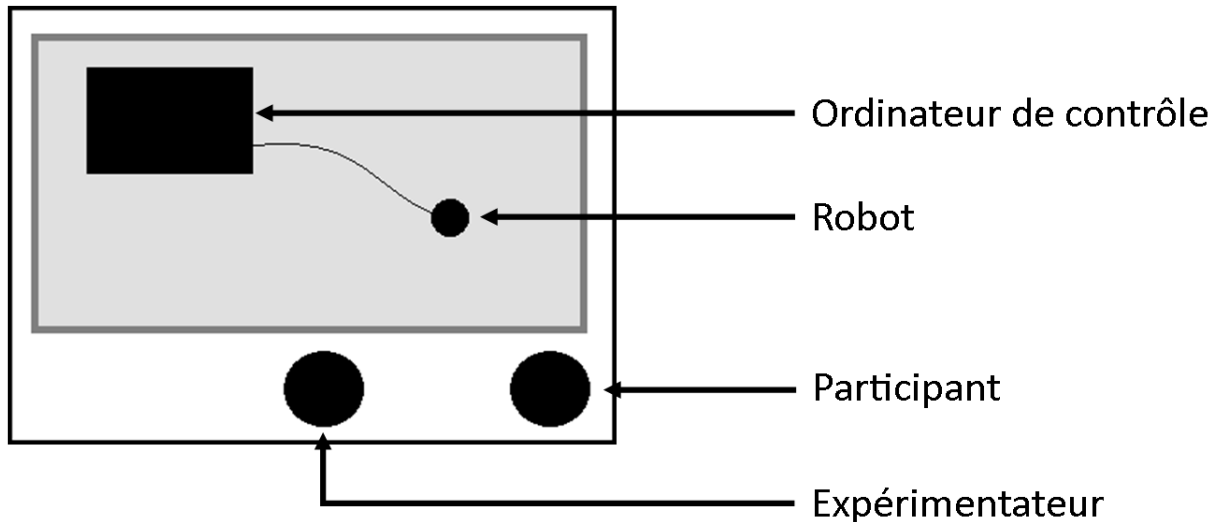


Figure 64 : Cadre expérimental de la deuxième étude

Déroulement de l'expérimentation : la première partie de l'expérimentation consistait à expliquer au participant qu'il devait observer une interaction entre l'humain (H) et le robot (R). Le Tableau 8 présente les quatre dialogues observés.

Tableau 8 : Les quatre dialogues de la deuxième étude

Dialogue 1	Dialogue 2
H : Bonjour Bioloïd R : [Bioloïd salue l'humain] H : Aides moi à organiser mes vacances, s'il te plaît. R : D'accord, j'ai besoin d'un certain nombre d'informations. H : Je voudrais me changer les idées et prendre l'air. R : [Bioloïd remercie l'humain] R : Je m'occupe de tout H : Merci Bioloïd, au revoir !	R : Je t'ai trouvé un camping, regardes sur l'écran. H : Merci, il a l'air super ! R : [Bioloïd se rejouit] R : Puis-je te proposer des randonnées ? H : Oui. R : Quels genres ? H : Je n'ai pas le temps pour ça, fait le toi-même. R : [Bioloïd se plaint]
Dialogue 3	Dialogue 4
H : Aujourd'hui, j'ai fait une super rando ! R : [Bioloïd se rejouit] H : Je suis déçue, le sentier était plein de détritus. R : [Bioloïd désapprouve] H : Mais j'ai tout ramassé dans un sac poubelle. R : [Bioloïd adhère et complimente l'humain]	H : Il pleuvait aujourd'hui pour ma randonnée R : [Bioloïd pleure] H : Tu aurais pu regarder la météo ! R : [Bioloïd s'excuse] H : C'est de ta faute, c'est bien beau de s'excuser ! R : Mais tu ne m'as pas demandé. H : Du coup, je suis partie du camping sans payer.

3. Collecte de données et analyses

Pendant chaque interaction, les participants devaient répondre à un questionnaire. Les questions portaient sur chaque phrase prononcée par le robot, puis sur chaque dialogue dans son ensemble puis sur le ressenti de l'interaction en entière.

Pour chacune des phrases, le participant devait indiquer s'il était d'accord avec les affirmations suivantes :

- Le geste s'accorde-t-il avec le contenu de la phrase ?
- La vitesse du geste est correcte ?
- Le geste semble-t-il souple et naturel ?
- Le robot est-il crédible quand il parle ?

Pour chaque dialogue dans son ensemble, le participant devait indiquer s'il était d'accord avec les affirmations suivantes :

- L'enchaînement des répliques est-elle correct ?
- La vitesse de réponse du robot et de l'homme est-elle correcte ?
- Les expressions du robot quand il écoute l'homme sont-elles correctes ?
- Est-ce que le robot est attentif ?

A la fin de l'interaction, le participant devait donner son point de vue sur l'affirmation suivante : Est-ce que le robot est agréable ?

Les réponses possibles étaient : Pas du tout, Un peu, Beaucoup, Sans idée ; sauf pour la deuxième affirmation où l'échelle de Likert a été utilisée : Trop lent, Lent, Normal, Rapide, Trop rapide. Le questionnaire peut être vu en annexe O.

Les analyses statistiques ont été réalisées avec le logiciel Minitab 15®. Le test du Chi Deux a permis de vérifier les réponses significatives. Le seuil de significativité (p) a été fixé à 0,05.

4. Résultats

La figure 5 présente les résultats qui sont détaillés dans cette partie.

Significativement et de façon globale, le geste est considéré comme un peu souple et naturel (50.51% un peu, $X^2=318.29$, $dl=3$, $p<0,001$), alors qu'il s'accorde beaucoup avec le contenu de la phrase (62.31% beaucoup, $X^2=365.03$, $dl=3$, $p<0,001$). De plus, la vitesse du geste est normale (66.67%, $X^2=623.32$, $dl=4$, $p<0,001$). L'enchaînement des répliques est correcte (67.31% beaucoup, $X^2=69.57$, $dl=2$, $p<0,001$).

Seulement 4.81% des participants estiment que la vitesse de réponse du robot et de l'homme n'est pas du tout correcte ($X^2=50.21$, $dl=2$, $p<0,001$). Et seulement 6.73% des participants estiment que les expressions du robot quand il écoute l'homme ne sont pas correctes ($X^2=31.27$, $dl=2$, $p<0,001$).

Concernant le robot lui-même, il est jugé crédible (60% beaucoup, $X^2=353.26$, $dl=3$, $p<0,001$) et attentif (67.31% beaucoup, $X^2=64.18$, $dl=2$, $p<0,001$). Pour finir, le robot n'a jamais été considéré comme désagréable ($X^2=13.04$, $dl=2$, $p=0,001$).

5. Discussion

La Figure 65 montre que l'interaction entre l'homme et le robot a été positivement perçue par l'observateur. Quelques affirmations n'ont pas été bien notées. Cela illustre la perception des erreurs d'adéquation placées aléatoirement dans la communication du robot. Par exemple, les expressions du robot quand il écoute l'homme n'ont pas toujours été jugées correctes. Les gestes du robot ne semblent qu'un peu souple et naturel. C'est normal étant donné que le robot utilisé n'a pas d'énormes capacités de mouvements. Malgré les imperfections des actions du robot, il est largement jugé attentif au cours de l'interaction et personne n'a estimé qu'il fût désagréable.

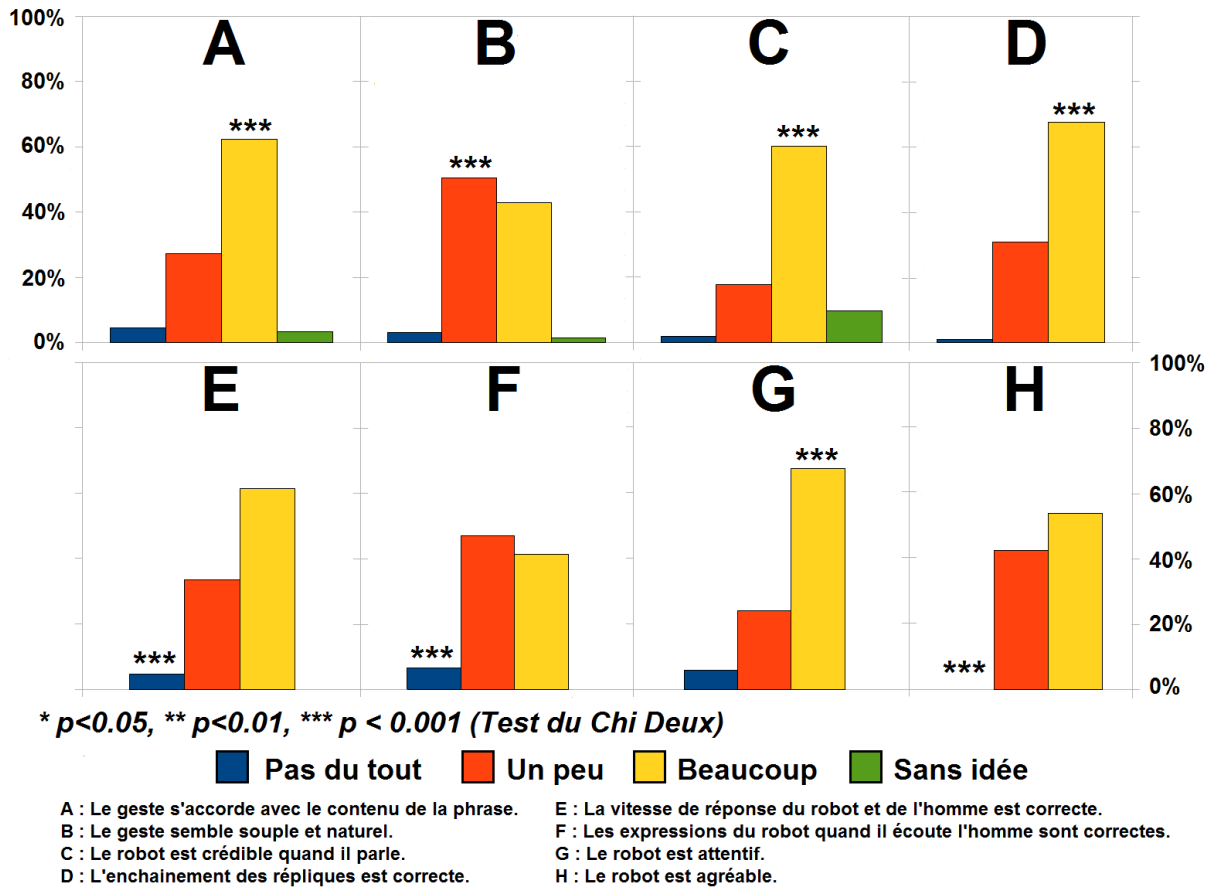


Figure 65 : Résultat sur l'acceptabilité du robot au cours d'une interaction moyenne

Niveau de significativité : * $p < 0,05$, ** $p < 0,01$, *** $p < 0,001$

NS : non significatif (Test du Chi-deux).

Cette étude montre que les humains ne perçoivent pas de manière négative un robot, même si de temps en temps son comportement non verbal n'est pas en adéquation avec son discours. Le robot semble avoir dépassé le statut de simple objet. En effet, si un objet ne remplit pas correctement sa fonction, l'homme ne s'y intéresse plus. Alors que dans le cas du robot, le jugement est moins sévère. Puisque l'homme semble avoir accepté les « défauts » du robot, il est possible qu'il ait dépassé le stade de simple objet et cela permet de penser, qu'un jour, le robot pourra être un partenaire social pour l'homme.

K. Evaluation d'AmbiProg

AmbiProg est une interface graphique permettant de créer des scénarii pour ArCo. En d'autres termes, AmbiProg est une interface de programmation visuelle. Généralement, la programmation est accessible à un cercle fermé de personnes qui ont un bon degré d'expertise de l'informatique. Mais l'objectif d'AmbiProg est de fournir un langage de programmation visuelle qui offre suffisamment de possibilités pour écrire des scénarii de tout type et qui soit suffisamment simple pour être accessible aux non experts. Deux études ont été réalisées pour se rendre compte du niveau de difficulté d'AmbiProg. La première étude avait pour objectif d'évaluer la façon dont les données sont présentées dans AmbiProg. Est-ce qu'un non expert comprend la philosophie de l'interface et retrouve facilement les informations qui l'intéressent ? La deuxième étude avait pour objectif d'étudier le niveau de difficulté de la conception de scénario avec AmbiProg. Est-ce qu'un non expert est capable d'apprendre à écrire des scénarii pour une utilisation future personnelle ?

A. Evaluation de la prise en main d'AmbiProg

Cette expérimentation a fait l'objet de l'article [165].

Cette étude avait pour objectif d'évaluer la facilité de compréhension du fonctionnement d'AmbiProg auprès de personnes ne l'ayant jamais utilisé avant. De plus, cette étude avait pour objectif de vérifier si des connaissances en informatique étaient nécessaires pour comprendre le fonctionnement d'AmbiProg.

1. Participants

Cette étude a été réalisée par 14 personnes (5 femmes et 9 hommes) âgées de 23 à 58 ans (moyenne de 31,7 ans). La moitié d'entre eux étaient considérés comme des experts en informatique (groupe nommé *experts*). Les autres avaient des connaissances en informatique limitées à une utilisation bureautique (groupe nommé *non experts*). Les participants ont été choisis sur la base du volontariat parmi la population pour obtenir un échantillon diversifié.

2. Méthode

L'expérimentation a été réalisée au domicile des personnes, dans un environnement connu. Chaque participant était isolé dans une pièce du logement avec l'expérimentateur et son ordinateur, sans aucun son pour perturber les personnes et faciliter leur concentration. Chaque participant a utilisé le même matériel pour effectuer le test.

Equipement : la Figure 66 montre l'interface graphique qui était affichée sur l'écran de l'ordinateur portable qui était prêté aux participants. Ils pouvaient utiliser l'interface avec une souris pour faciliter la manipulation, plutôt qu'avec un touchpad que tout le monde ne sait pas utiliser. Le participant était assis face à l'ordinateur alors que l'expérimentateur se trouvait debout, en retrait. Un chronomètre était utilisé pour calculer le temps qu'il fallait aux participants pour réussir ou pour échouer.

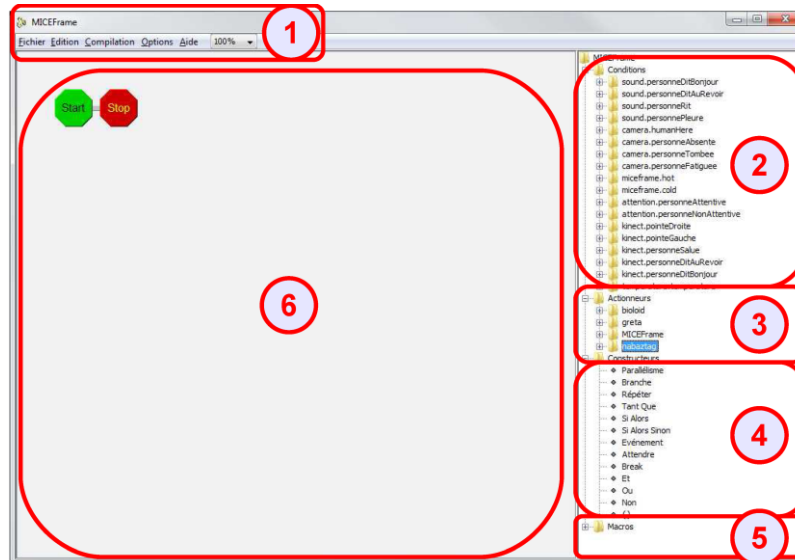


Figure 66 : Organisation d'AmbiProg présentée lors de la troisième étude

Déroulement de l'expérimentation : la première étape de l'expérimentation était d'expliquer le contexte de l'étude. L'expérimentateur expliquait qu'avec les progrès de la technologie, l'environnement numérique des personnes était de plus en plus compliqué à contrôler. La solution consistait à proposer une interface graphique pour permettre de programmer facilement un environnement numérique. Ensuite, l'expérimentateur indiquait que l'expérimentation commençait par une présentation d'AmbiProg puis par un exercice à réaliser. Ensuite, l'expérimentateur demandait au participant s'il souhaitait continuer l'expérimentation.

La deuxième étape consistait à expliquer l'organisation d'AmbiProg, comme le suggère la figure 6. L'expérimentateur présentait la zone d'éléments de scénario, les branches perceptions et actions, en insistant sur les concepts. Le contenu d'AmbiProg était abstrait. Les perceptions et les actions avaient été choisies pour ne pas être comprises par les participants de façon à ce que les exemples n'aident pas pendant la réalisation de l'exercice.

Après avoir donné les instructions l'expérimentateur vérifiait que le participant avait compris les consignes et proposait de répondre aux éventuelles questions. Lorsque le participant indiquait qu'il était prêt, alors il lui était demandé de réaliser le scénario suivant : « Si la personne est tombée alors le bioloïd appelle la personne pendant que la nabaztag appelle les secours ». Cette phrase était énoncée et écrite sur un papier à côté de l'ordinateur en tant qu'aide-mémoire.

Chaque participant pouvait prendre le temps de lire la phrase et commençait lorsqu'il était prêt. Le chronomètre était démarré lorsque le participant touchait la souris pour la première fois. L'expérimentateur stoppait le chronomètre soit lorsque le participant avait fini le scénario, soit lorsque celui-ci indiquait avoir fini.

3. Collecte de données et analyses

Le nombre d'aides données et le nombre d'erreurs observées sans demande d'aide étaient consignés par l'expérimentateur. A la fin de la session d'expérimentation, les participants répondaient à un court questionnaire.

Le questionnaire interrogeait les participants sur leur sexe, leur âge, leur niveau d'études et sur leur connaissance en informatique. Pour répondre aux sept questions posées, les participants utilisaient l'échelle de Likert avec les propositions suivantes : Pas du tout d'accord (1), pas d'accord (2), ni en désaccord ni d'accord (3), d'accord (4) et tout à fait d'accord (5).

Les questions posées étaient les suivantes :

1. Avez-vous trouvé l'exercice facile ?
2. Etes-vous à l'aise avec cette programmation ?
3. Est-il facile de comprendre le fonctionnement de l'interface graphique ?
4. Est-il facile d'apprendre à programmer des scénarii ?
5. Vous sentez-vous capable de programmer d'autres scénarii de votre choix ?
6. Si vous pouviez programmer les dispositifs numériques vous entourant, le feriez-vous avec cette interface graphique ?
7. Etes-vous favorable à ce type de pilotage des dispositifs vous entourant ?

Pour chaque question, le score minimum était de 1 et le score maximum de 5. Le score global de chaque question était calculé en fonction des réponses données. Plus le score était élevé, plus AmbiProg était perçu positivement. Tous les commentaires étaient enregistrés pour sauvegarder les commentaires spontanés et les explications dans le contexte.

Les observations ont permis de calculer trois données supplémentaires pour chaque session expérimentale :

- Le temps utilisé pour réussir ou échouer dans la création du scénario (en seconde) : si le participant réussissait le scénario, l'expérimentateur stoppait le chronomètre au moment où le participant pensait avoir fini sa tâche. Si le participant abandonnait, le chronomètre était stoppé au moment de l'abandon.
- Le nombre d'aides données : soit le participant posait des questions pour l'aider à créer le scénario, soit l'expérimentateur notait que le participant avait faux et allait être bloqué dans la réalisation du scénario et dans ce cas, l'expérimentateur corrigeait l'erreur pour que le participant puisse continuer le scénario.
- Le nombre d'erreurs observées : l'expérimentateur notait lorsque le participant avait faux. Généralement c'était suivi par une aide donnée si le participant ne corrigeait pas de lui-même l'erreur.

Les analyses statistiques ont été réalisées avec le logiciel Minitab 15©. Le test du Chi Deux a permis de vérifier les réponses significatives. Le seuil de significativité (p) a été fixé à 0,05. Les données collectées étaient le score des questions et le temps (en seconde). Comme les données n'étaient pas normalement distribuées, nous avons utilisé un test statistique non paramétrique appelé le test U de Mann-Whitney et un test de corrélation de Pearson [166]. Ces tests étaient utilisés pour étudier si les connaissances en informatique ou l'âge des participants avaient une influence sur les données collectées.

4. Résultats

Tous les participants ont réussi la tâche proposée avec un temps moyen de 131 secondes. Le temps minimum était de 51 secondes et le temps maximum de 284 secondes. La Figure 67 montre que plus la personne était âgée, plus le temps de succès de la tâche était long (Pearson : $r_s=0,81$, $p<0,001$). Même si les experts étaient un peu plus rapide dans la résolution de la tâche, la différence avec le temps de résolution des non experts n'étaient pas significatif ($(X \pm SD: 101,0 \pm 18,8$ versus $161,3 \pm 44,3$; tests U de Mann-Whitney; $n_1=7$ $n_2=7$, $U=42,0$ $p=0,201$).

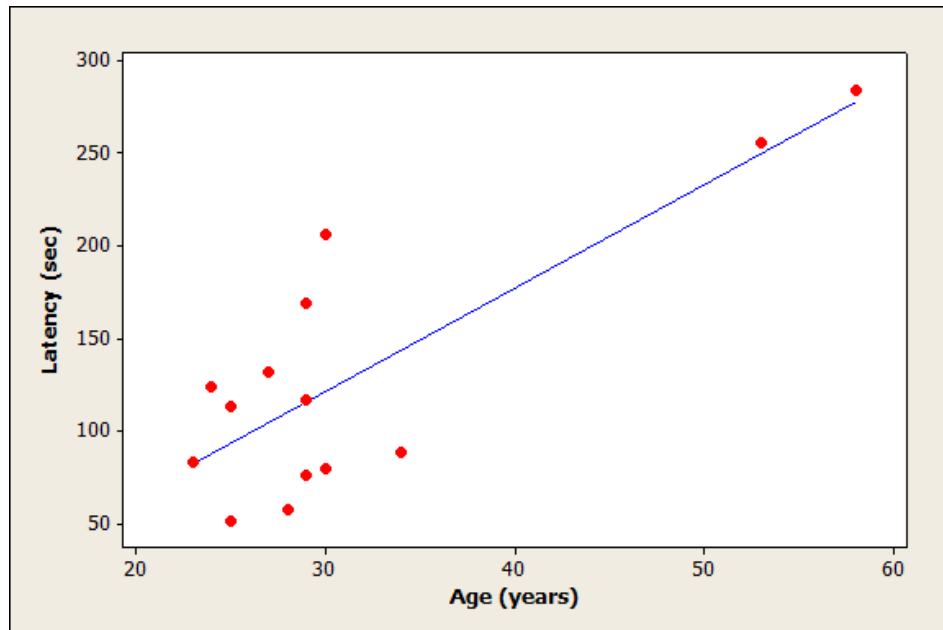


Figure 67 : Temps pour réussir la tâche en seconde en relation avec l'âge des participants

Corrélation de Pearson. Niveau de significativité : $p < 0,05$

La Figure 68 montre les scores obtenus par toutes les questions en fonction du niveau de connaissance en informatique des participants. Le score total moyen du questionnaire était de $32,1 \pm 1,9$. Le score maximum était de 35. Aucune différence statistique n'a été noté entre les *experts* et les *non experts* ($X \pm SD$: $33,6 \pm 0,9$ contre $30,6 \pm 2,4$; Tests U de Mann-Whitney; $n_1=7$ $n_2=7$, $U=60$ $p=0,360$).

Le score moyen de toutes les questions était supérieur à 4, ce qui montre qu'AmbiProg était perçue positivement. Là non plus, il n'y avait aucune différence statistique entre les *experts* et les *non experts* (tests U de Mann-Whitney, $p > 0,05$) sauf pour la question 4 (facilité d'apprendre à programmer les scénarii) où les *non experts* semblaient moins à l'aise que les *experts* ($X \pm SD$: $5,0 \pm 0,0$ contre $4,36 \pm 0,4$; Mann-Whitney U tests; $n_1=7$ $n_2=7$, $U=65$ $p=0,072$).

Plus le participant était âgé, moins il se sentait à l'aise avec la programmation (question 2; Pearson: $r_s = -0,626$, $p = 0,017$) et moins il trouvait qu'il était facile de créer des scénarii (question 4; Pearson: $r_s = -0,767$, $p = 0,001$).

Il n'y avait aucune autre différence statistique concernant l'âge des participants (Pearson, $p > 0,05$).

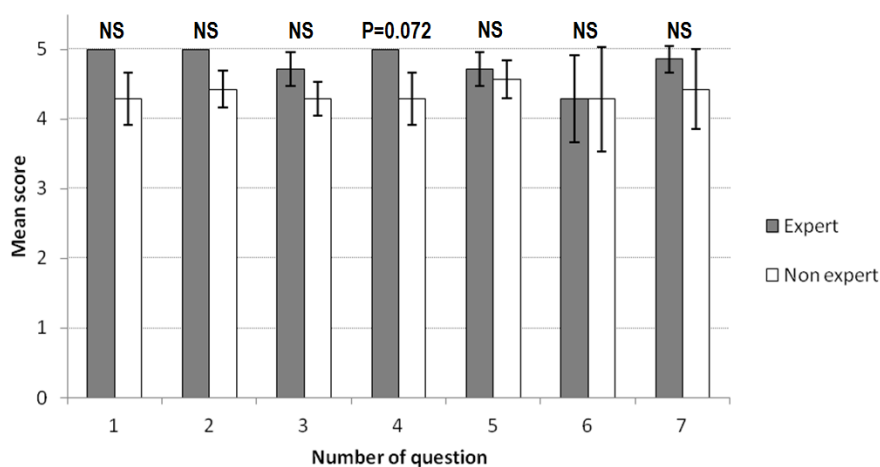


Figure 68 : Réponses aux questions de la troisième étude.

Niveau de significativité $p < 0,05$; NS : Non significatif ; test U de Mann Whitney.

La Figure 69 montre les résultats concernant les aides données et les erreurs observées en fonction de la connaissance en informatique des participants. Six personnes ont reçu de l'aide de l'expérimentateur ($X \pm SD$: $0,71 \pm 0,46$) avec un nombre d'aide maximum de deux aides. Neuf participants ont fait des erreurs ($X \pm SD$: $0,93 \pm 0,41$) avec un maximum de deux erreurs. Plus la personne était âgée, plus elle a reçu d'aide (Pearson: $r_s=0,687$, $p=0,007$). Il n'y avait aucune différence statistique entre les *experts* et les *non experts* (pour les deux, test U de Mann Whitney, $p>0,05$) et en fonction de l'âge pour les erreurs observées (Pearson: $r_s=0,183$, $p=0,532$).

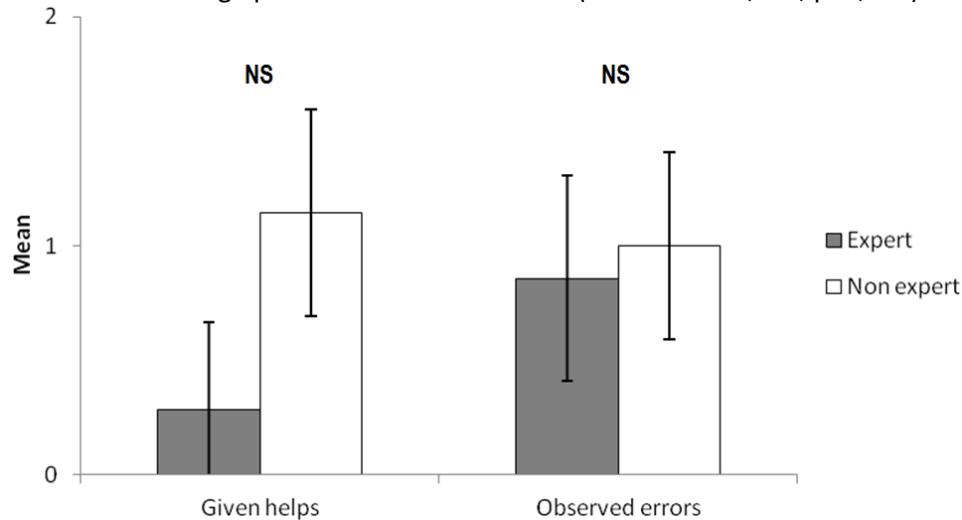


Figure 69 : Aides données et erreurs observées

Niveau de significativité : $p<0,05$; NS : Non significatif ; test U de Mann Whitney.

Les participants étaient libres de donner des commentaires concernant l'expérimentation et concernant AmbiProg. Deux types de commentaires ont été relevés : des commentaires techniques et des commentaires d'expériences personnels. L'avantage principe d'AmbiProg est que les participants l'ont trouvé facile à utiliser et marrant. Ils avaient envie de continuer à l'utiliser, ce qui pourrait permettre une intégration à domicile du système. L'inconvénient principal concerne la partie technique de l'interface graphique. Deux participants ont indiqué que la zone avec les fonctions, les perceptions et les éléments de scénario n'étaient pas assez intuitifs. Et il est possible que ce n'est pas accessible à tout le monde. Un participant a indiqué que les aides visuelles manquaient de visibilité. Cependant ce point peut être débattu car deux participants pensaient qu'AmbiProg était intuitifs et qu'il était possible de s'adapter rapidement. Ils ont aimé le fait qu'« une fois que le fonctionnement est compris, la création de scénario est vraiment rapide. »

5. Discussion

Les résultats ont montré qu'AmbiProg était positivement perçu et que les connaissances en informatique n'étaient pas requises pour la création de scénarii.

Pas de différence significative entre les *experts* et les *non experts* : les *experts* et les *non experts* ont tous réussi la tâche et n'ont pas abandonné ce qui était cohérent avec les réponses données par première question. De plus, ils ont mis le même temps moyen pour réussir la tâche. Tous les participants ont été capables de créer des scénarii, après une courte explication, en moins de cinq minutes. Ainsi, AmbiProg semble facile à comprendre et à utiliser et requière un temps court d'apprentissage (ici 5 minutes).

Concernant les questionnaires, les *experts* et *non experts* ont répondu différemment à la quatrième question. L'hypothèse était que les *non experts* n'étaient pas à l'aise avec la programmation et ne connaissaient pas leur compétence dans le domaine, au contraire des experts qui ont l'habitude de manipuler les concepts informatiques. Mais, une différence existe entre la représentation mentale

des participants (explorée par le questionnaire) et la réalité (succès dans la réalisation du scénario) parce que les non experts étaient capables de créer des scénarii sans montrer de réelles difficultés.

Des besoins qui augmentent avec l'âge ? Les résultats ont montré que, plus les participants étaient âgés, et plus le temps de succès était long. Il est possible que c'est dû au fait que l'informatique est une science jeune (les ordinateurs personnels sont apparus il y a une quarantaine d'années) et que les participants les plus âgés sont moins à l'aise avec l'informatique que les participants plus jeunes. De plus, les participants plus âgés ont reçu plus d'aide que les participants plus jeunes. Soit, cela confirme l'hypothèse que les personnes plus âgées ne sont pas à l'aise avec l'informatique et n'ont pas été capable d'être autonome avec l'ordinateur, soit c'était un biais apporté par l'expérimentateur qui a peut-être été plus tenté d'aider les participants plus âgés. Les prochaines expérimentations doivent lever ce genre de doutes.

Une réticence des experts ? Pendant l'expérimentation, l'attitude des *experts* étaient totalement différente de l'attitude des *non experts*. Les *experts* ont montré un manque d'engagement et ont évalué AmbiProg au nom des *non experts*. Trois d'entre eux n'étaient pas favorable à AmbiProg parce qu'ils préféraient l'idée de programmer eux même l'environnement numérique. Cependant, ils ont bien noté la question car ils étaient favorables à l'utilisation d'AmbiProg par des non informaticiens.

AmbiProg : un enthousiasme global : Les résultats ont montré que l'expérimentation était positive, même si les *non experts* ont eu tendance à limiter leur notation à 4, contrairement aux *non experts* qui ont facilement donné 5. Tous les *experts* ont eu un jugement similaire, plus contrasté que celui des *non experts*. Par exemple, les réponses des questions six et sept étaient variables. Soit, les personnes ont clairement accroché avec AmbiProg ou pas indiquant que l'interface provoque un avis tranché, soit cela signifie que les *non experts* n'étaient pas habitués à utiliser ce genre de système et qu'il leur était difficile d'imaginer des applications possibles. Cette seconde hypothèse a été confirmée par des commentaires. Trois femmes étaient très intéressées, voulaient continuer les exercices et trouvaient que la programmation pourraient être marrante avec un peu plus d'expériences. Deux participants ont imaginé comment AmbiProg pourrait leur apporter une solution dans leur environnement de travail.

B. Evaluation de l'apprentissage de la création de scénarii

Cette expérimentation a fait l'objet de l'article [167].

Cette étude avait pour objectif d'évaluer le langage de programmation visuelle de façon à savoir s'il est accessible à un grand nombre de personnes et si les gens peuvent facilement apprendre comment contrôler leur environnement avec ArCo. Chaque participant devait créer des scénarii et lancer l'interpréteur pour voir les robots effectuer les actions demandées.

1. Participants

L'étude a été réalisée avec 16 enfants (8 filles et 8 garçons), âgés de 10 à 11 ans (moyenne de 10,44 ans). Les participants étaient des élèves de CM2 d'une école primaire. Les participants ont été choisis parmi un ensemble d'enfants qui avaient obtenus l'autorisation de leurs parents pour participer à une évaluation.

Nous avons choisi de faire tester ArCo par des enfants parce que nous voulions vérifier si le langage de programmation visuelle était réellement facile à manipuler. Les jeunes personnes ne sont pas des experts en informatique, ainsi ils représentent un échantillon intéressant de la population.

De plus, les enfants n'ont pas de préjugé envers la technologie. Il était donc intéressant de tester ArCo d'abord sur des enfants pour éviter le biais dû à un rejet de la technologie. En effet, si les enfants ne réussissent pas les tâches, cela ne sert à rien de tester avec des personnes âgées.

2. Méthode

L'expérimentation a été réalisée dans une pièce de l'école. Deux participants y étaient isolés avec un expérimentateur. La présence de deux participants permettait de maintenir un bon niveau de concentration car les enfants se sentaient en compétition. Mais pour éviter la triche ou les influences réciproques, ils étaient positionnés presque dos à dos.

Equipement : La Figure 70 montre le cadre expérimental de l'évaluation. Les participants devaient utiliser un ordinateur qui affichait AmbiProg. Ils manipulaient le glisser/déposer avec une souris pour faciliter la réalisation des exercices. Ils avaient la possibilité de contrôler deux Nabaztag et une tablette numérique. Les robots étaient des entités agissantes actives capables d'effectuer des actions : chorégraphies (mouvements des oreilles, coloration de quatre leds placées sur le ventre) et prononciation de paroles. La tablette affichait une interface graphique avec quatre boutons et était une entité agissante perceptive capable d'envoyer quatre perceptions, une pour chaque bouton. Chaque participant pouvait contrôler les deux robots et recevoir les perceptions de la tablette. Un point d'accès permettait la communication entre les cinq dispositifs numériques.

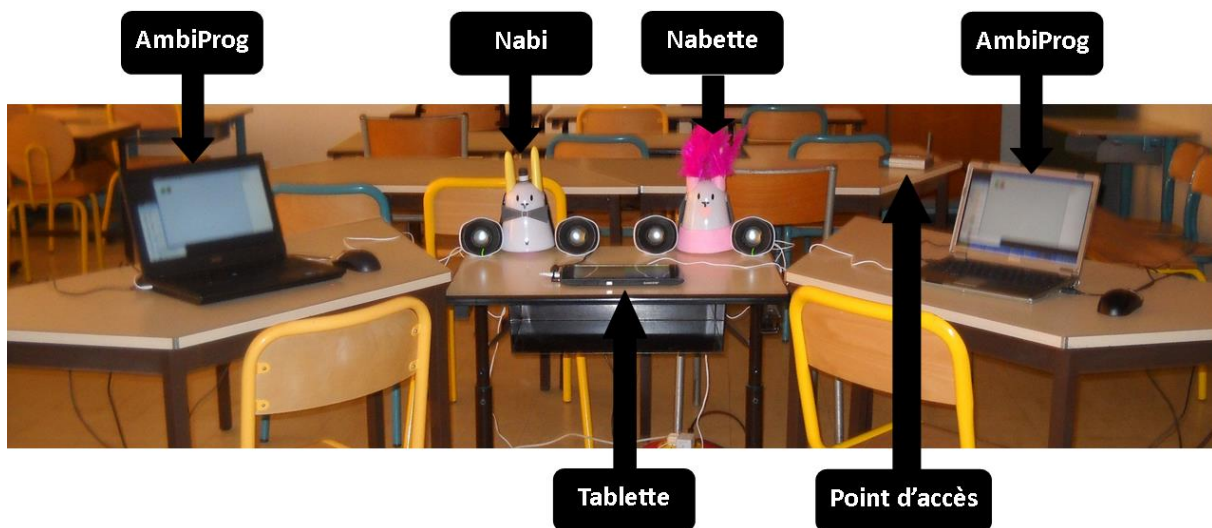


Figure 70 : Cadre expérimental de la quatrième étude

Déroulement de l'expérimentation :

Explication : La première partie consistait à expliquer le fonctionnement de l'interface graphique aux enfants. D'abord l'expérimentateur présentait les deux robots : Nabette et Nabi, représentés sur la Figure 71. Ensuite, l'expérimentateur expliquait que la tablette était une entité capable d'envoyer des perceptions (une donnée de l'environnement) et que dans cette expérimentation, il y avait quatre perceptions : « clic sur le bouton bleu », « clic sur le bouton rose », « clic sur le bouton vert » et « clic sur le bouton noir ». L'expérimentateur expliquait ensuite qu'il était possible de contrôler les robots en écrivant des scénarii, par exemple : « si l'utilisateur a cliqué sur le bouton rouge, alors Nabette dit bonjour ».



Figure 71 : Nabette et Nabi

L'expérimentateur expliquait ensuite qu'il était possible de le faire avec AmbiProg. Les enfants recevaient une explication de cinq minutes sur l'organisation d'AmbiProg. C'était une explication plus courte que dans la troisième étude.

Ensuite, l'expérimentateur expliquait le processus utilisé pendant l'expérimentation :

1. Les participants devaient créer un scénario en respectant une consigne donnée par l'expérimentateur.
2. Les participants devaient compiler le scénario, c'est-à-dire cliquer sur le menu « compiler » et sauvegarder le scénario sous un nom prédéfini.
3. Les participants devaient double-cliquer sur un fichier exécutable pour lancer l'interprétation.

Exercices : La deuxième partie de l'évaluation consistait à réaliser six exercices classés selon les trois concepts suivants :

- Séquentialité : les actions sont effectuées les unes après les autres
- Parallélisme : plusieurs actions peuvent être effectuées en même temps
- Évènement : les actions dépendent de la réception d'une perception. Le système attend de recevoir la perception avant d'effectuer l'action demandée.

Nous considérons que les trois concepts avaient presque le même niveau de difficulté. Ainsi, l'ordre donné par les exercices ne représente pas une progression dans le niveau de difficulté. Cependant, au sein de chaque concept, il y a plusieurs niveaux de difficulté.

Explications données avant les exercices de séquentialité : L'expérimentateur expliquait le langage de programmation par un exemple montrant la séquentialité, représenté sur la Figure 72. Le scénario signifiait : « Nabi joue la chorégraphie 01 et ensuite joue la chorégraphie 02 ».



Figure 72 : Etude 4 - explications avant l'exercice 01

Exercices sur la séquentialité : La première explication fit la transition avec le premier exercice. Les participants devaient utiliser le même concept et choisir d'autres actions. La réalisation de cet exercice permettait de vérifier si les premières explications étaient comprises. Le premier exercice (Figure 73) consistait à écrire le scénario : « Nabi dit bonjour, puis Nabi dit au revoir ».



Figure 73 : Etude 4 - exercice 01

Le deuxième exercice complétait le premier (Figure 74). Le scénario était : « Nabi joue la chorégraphie 01, puis dit bonjour, ensuite Nabi joue la chorégraphie 02 puis dit au revoir ». Cet évènement était le dernier du concept « séquentialité ».



Figure 74 : Etude 4 - Exercice 02

Explications données avant les exercices de parallélisme : L'expérimentateur présentait l'élément parallélisme. Cela augmentait un peu le niveau de difficulté parce que les participants devaient utiliser un élément de scénario pour la première fois. Celui-ci fait apparaître une seconde barre grisée qui permet de glisser/déposer deux actions qui seront exécutées en même temps. La Figure 75 montre le scénario qui était présenté aux participants : « Nabi joue la chorégraphie 01 pendant que Nabette joue la chorégraphie 01 ».



Figure 75 : Etude 4 - explications avant l'exercice 03

Exercices sur le parallélisme : L'explication de l'élément parallélisme permettait de faire la transition avec le troisième exercice. Les enfants devaient utiliser à nouveau de concept. La Figure 76 montre le troisième exercice qu'ils devaient réaliser : « Nabette dit bonjour et joue la chorégraphie 01 en même temps ».



Figure 76 : Etude 4 - exercice 03

Le quatrième exercice complétait le précédent. Il était demandé aux enfants d'écrire le scénario (Figure 77) : « Nabette dit bonjour et joue la chorégraphie 01 en même temps. Ensuite Nabi dit

bonjour et joue la chorégraphie 01 en même temps. ». Cet exercice utilisait à la fois le concept de séquentialité et de parallélisme ce qui augmentait le niveau de difficulté.

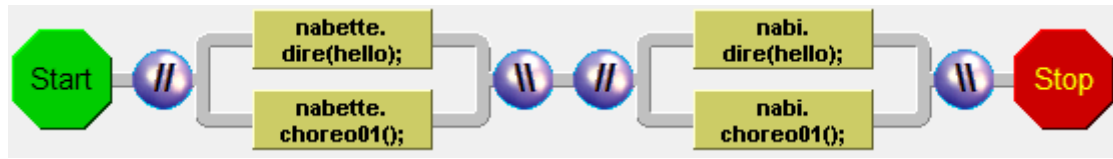


Figure 77 : Etude 4 - exercice 04

Le cinquième exercice complétait l'exercice précédent (cf. Figure 78). Les enfants devaient écrire le scénario : « Nabette dit bonjour et joue la chorégraphie 01 en même temps. Ensuite Nabi dit bonjour et joue la chorégraphie 01 en même temps. Ensuite, Nabi dit enchanté et joue la chorégraphie 02 pendant que Nabette dit enchantée et joue la chorégraphie 02. Ces quatre actions sont effectuées en même temps. ». Le niveau de difficulté était un peu augmenté car les participants devaient utiliser un nouvel élément de scénario : l'élément Branche qui permet d'ajouter une barre grisée à l'élément Parallélisme.

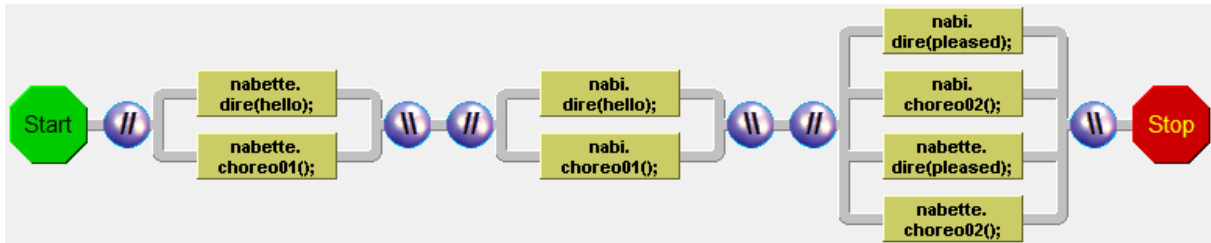


Figure 78 : Etude 4 - exercice 05

Explications données avant l'exercice d'évènement : L'expérimentateur présentait le concept d'évènement. Les participants devaient utiliser un nouvel élément de scénario. Cet élément requière une perception et une action. L'expérimentateur montrait l'exemple de la Figure 79 : « Si l'utilisateur a cliqué sur le bouton bleu alors Nabi joue la chorégraphie bleue ».



Figure 79 : Etude 4 - explications avant l'exercice 06

Exercice sur les événements : L'explication donnée faisait transition pour le dernier exercice, illustré sur la Figure 80. Les enfants devaient utiliser les éléments Evènement et Parallélisme. Ils devaient écrire le scénario : « Si l'utilisateur a cliqué sur le bouton noir, alors Nabi dit bonjour et joue la chorégraphie 03 en même temps ».

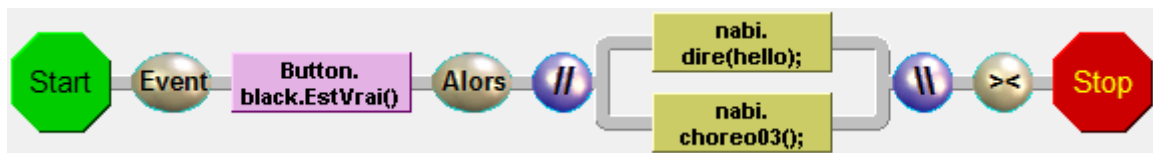


Figure 80 : Etude 4 - exercice 06

3. Collecte de données et analyses

Cette étude s'intéressait à deux valeurs : les observations faites par l'expérimentateur pendant l'évaluation et l'auto-évaluation des enfants effectuée par un questionnaire à la fin de la session expérimentale.

Observations : pendant l'expérimentation, l'expérimentateur observait les facilités et les difficultés concernant les concepts et les exercices. Les concepts suivants étaient surveillés : glisser/déposer, compilation, instructions, séquentialité, parallélisme et évènement. L'expérimentateur notait si chaque participant comprenait (noté 1) ou non (noté 0) ces concepts. Si un participant demandait de l'aide à propos d'un concept, celui-ci était noté comme étant incompris (noté 0). Dans ce cas, l'expérimentateur aidait le participant pour qu'il puisse continuer l'évaluation. L'aide donnée n'était pas un biais de l'évaluation car elle était prise en compte dans la notation. Ensuite, l'expérimentateur notait le succès ou l'échec pour chaque exercice indépendamment et pour chaque concept. La durée de l'expérimentation était également notée.

Questionnaire : A la fin de l'évaluation, les participants devaient remplir un questionnaire qui contenait les questions suivantes :

- A – Je n'ai eu aucun problème pour faire les exercices
- B – A partir de quel exercice tu as pensé pouvoir programmer tout seul ?
- C – J'ai trouvé qu'il était facile de faire des programmes avec l'interface graphique.
- D – Je me sens capable de faire d'autres programmes.
- E – L'interface graphique me plaît.
- F – Je pense qu'avec cette interface graphique, je peux programmer d'autres robots.
- G – Est-ce que tu souhaites continuer à utiliser l'interface graphique ?
- H – Est-ce que tu changerais quelque chose dans l'interface graphique ?

Ces questions permettaient de faire la représentation mentale des enfants et l'évaluation d'AmbiProg. Les questions de A à F, excepté la question B, étaient évaluées avec une échelle de Likert comprenant les valeurs suivantes : pas du tout d'accord (noté 0), Pas d'accord (noté 1), Ni d'accord ni en désaccord (noté 2), d'accord (noté 3), tout à fait d'accord (noté 4). La question B n'imposait pas de format spécifique de réponse. Les deux dernières questions pouvaient être répondu par oui (noté 1) ou non (noté 0).

4. Résultats

Quelques problèmes rencontrés lors de l'expérimentation sont à noter car il a fallu s'y adapter sans mettre en péril l'expérimentation. Les deux grands problèmes ont été la compréhension des instructions et le temps. Les résultats présentés sont de deux types : les observations faites par l'expérimentateur et les réponses du questionnaire.

Problèmes d'instructions : La moitié des participants ont reçu des instructions écrites. Nous avons remarqué qu'ils prenaient beaucoup de temps pour comprendre la nature des tâches à effectuer. Les observations et des discussions avec les enseignants des enfants ont indiqué que le problème venait de la compréhension des instructions. Il semblerait que les enfants étaient trop concentrés sur la lecture et que cela les empêchait de comprendre les instructions puisqu'ils utilisaient leurs ressources pour déchiffrer la consigne. Cela semblait être un biais pour l'expérimentation. Pour vérifier que le problème venait bien de l'écrit, la seconde moitié des participants ont reçu des instructions orales. Et nous avons comparé la compréhension des consignes dans les deux cas. Il était important de vérifier si la difficulté venait des consignes ou d'AmbiProg.

Problème de temps : En parallèle du premier problème, nous avons rencontré un problème de temps. Les enseignants n'ont pas pu mettre à disposition chaque enfant pour une même durée. Avant cette contrainte, notre idée était de laisser autant de temps que nécessaire aux enfants pour effectuer les exercices et d'observer combien de temps cela leur prenait pour réaliser les tâches. Le temps n'avait pas d'importance dans cette évaluation car son objectif était de tester si les enfants étaient capables d'apprendre à programmer avec AmbiProg. La conséquence de cette contrainte de

temps fut l'impossibilité de tester le troisième concept : les événements. Et le rythme de l'évaluation a dû être accéléré au dépend du temps d'explication d'AmbiProg.

Observation : La Figure 81 montre les différences entre la compréhension des concepts en fonction de la façon dont les instructions étaient données. Le graphique montre la valeur moyenne obtenue pour chaque participant (Compris=1, pas compris=0). Les résultats ont révélé que les participants avaient une meilleure compréhension lorsque les instructions étaient données oralement. Concernant le glisser/déposer, la totalité des participants l'ont correctement effectué avec les instructions orales contre 87,5% avec les instructions écrites. Il n'y a aucune différence concernant la compilation (75%). Les instructions écrites ont été comprises par 37,5% des participants tandis que 80,50% ont compris les instructions orales. La séquentialité a été comprise à 62,50% et le parallélisme à 75% dans le premier cas, alors que dans le deuxième cas, la totalité des participants a compris ces deux concepts. Enfin, 37,5% des participants ont compris le concept Événement quand les instructions étaient données à l'écrit alors que personne ne l'a compris lorsque les instructions étaient données à l'oral.

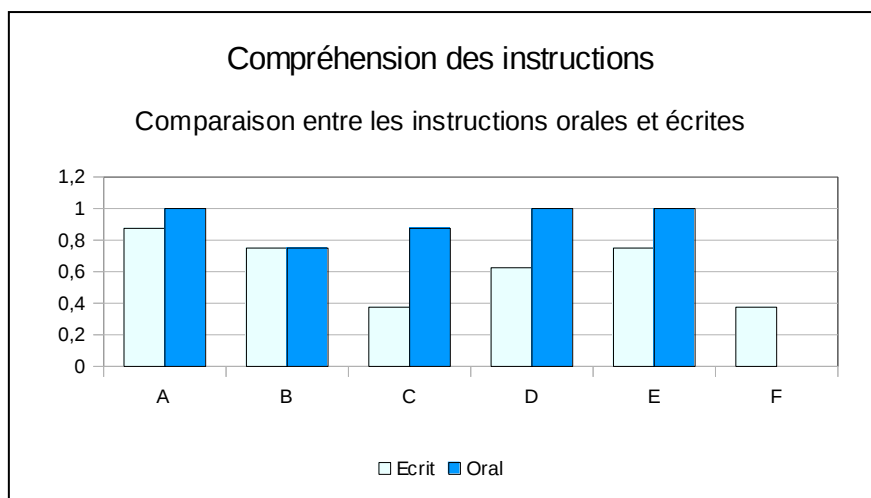


Figure 81 : Comparaison de l'effet des instructions écrites et orales.

Légende de la Figure 81 :

- A : Drag & drop
- B : Compilation
- C : Instructions
- D : Séquentialité
- E : Parallélisme
- F : Événement

La Figure 82 montre le pourcentage de résolution des exercices. Presque tous les participants ont réussi les exercices avec un maximum de deux aides. Les résultats sont vraiment prometteurs car les participants n'ont eu que peu de temps pour recevoir des explications et faire les exercices. Le graphique montre que les deux premiers exercices ont mieux été réussis lorsque les conditions étaient données à l'oral. Tous les autres exercices ont été mieux réussis avec les instructions écrites.

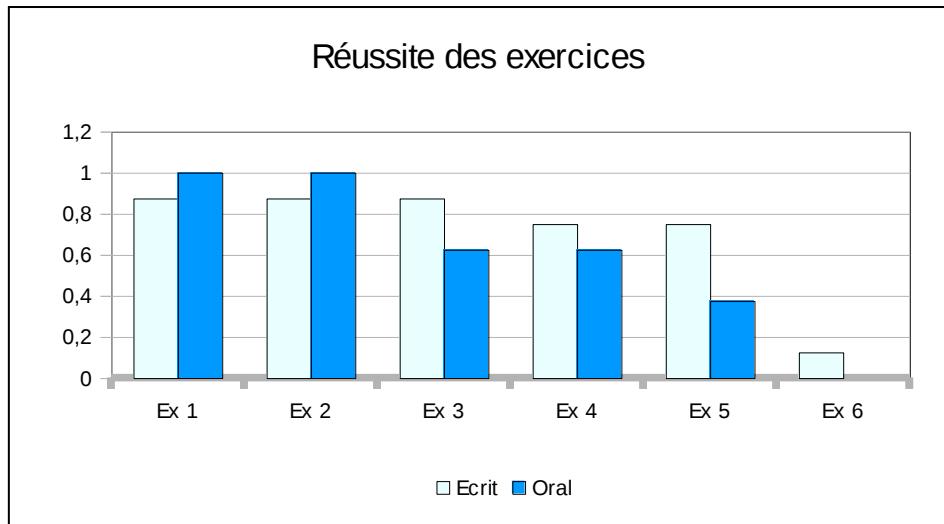


Figure 82 : comparaison entre la réussite aux exercices en fonction du type d'instructions

Questionnaire : la représentation mentale des enfants semblaient différentes des résultats de l'observation. La Figure 83 montre leur réponse aux questions A-F du questionnaire. Malgré le manque de contrôle complet d'AmbiProg, la majorité des enfants ont considéré n'avoir aucun problème pour faire les exercices (score de 4,44/5). La majorité d'entre eux ont jugé qu'il était facile de programmer (score de 4,25/5). Ils se sentaient capables de faire d'autres programmes seuls (score de 4,56/5). Leur jugement de l'interface était très positif. Ils ont aimé l'interface graphique (score de 4,88/5) et ils pensaient pouvoir contrôler d'autres robots avec AmbiProg (score de 4/5).

Concernant leur autonomie, ils pensaient être capables de créer des scénarii seuls à partir du second ou du troisième exercice (moyenne de 2,31).

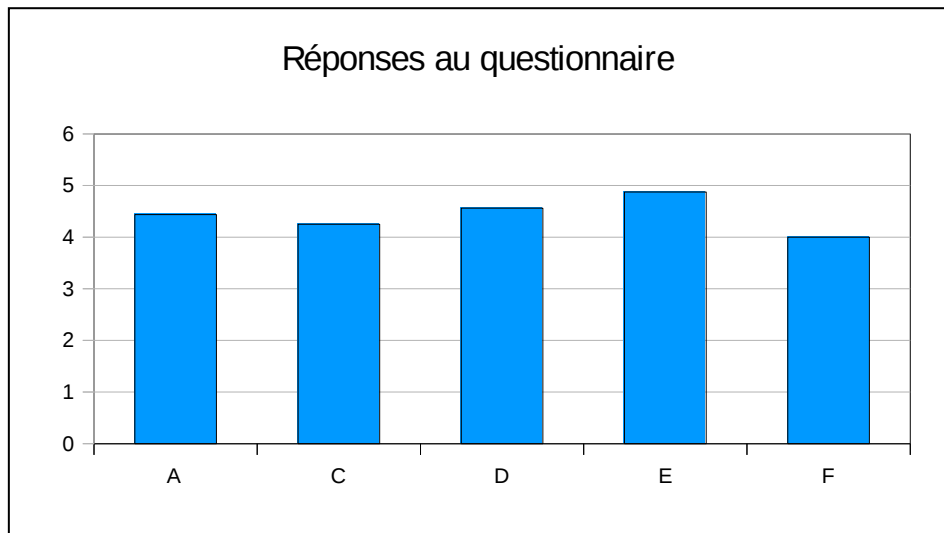


Figure 83 : réponses aux questions A, C, D, E et F.

Concernant l'acceptabilité d'AmbiProg, les résultats ont été très positifs. La Figure 84 montre les réponses aux questions G et H. Tous les participants ont aimé l'interface et ont indiqué vouloir continuer à l'utiliser. Seulement un participant voulait changer quelque chose à l'interface, sans donner d'autres précisions. Tous les autres ont indiqué que l'interface n'avait pas besoin d'être modifiée.

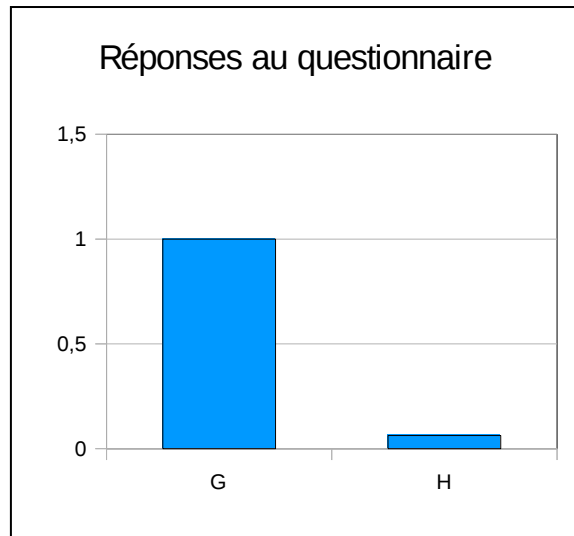


Figure 84 : réponses aux questions G et H.

5. Discussion

Différence entre les instructions écrites et orales : Les Figure 81 et Figure 82 comparent la compréhension des concepts et le succès des exercices en fonctions des instructions écrites et orales. Malgré l'amélioration de la compréhension des concepts dans la condition orale, le succès des exercices a diminué. En fait, les enfants ont mieux réussi les exercices dans la condition orale mais cela n'apparaît pas dans les statistiques à cause du manque de temps. En effet, les sessions expérimentales avec les premiers enfants n'étaient pas limitées dans le temps et duraient presque une heure. Mais après quelques passages, les enseignants ont expliqué que cela prenait trop de temps car chaque participant manquait une leçon complète. Les passages suivants étaient réalisés en fonction des activités des enfants et duraient entre 30 et 45 minutes. Ainsi, l'expérimentateur n'avait plus beaucoup de temps pour laisser les enfants réfléchir et chercher longtemps. Il fallait alors enchaîner les exercices rapidement et les noter comme étant incomplet même si les participants étaient sur le bon chemin. Cela explique pourquoi les exercices semblent avoir été mieux réussis avec les instructions écrites. Cette complication a ajouté un biais dans l'expérimentation et devrait être évité dans les futures expérimentations. Le manque de temps est également responsable du manque de compréhension de la condition orale. L'expérimentateur n'avait plus le temps de proposer le dernier exercice. Cela a faussé les résultats.

Cependant, cette expérience sur le temps a permis de comprendre qu'un enfant de 10 ans a besoin de 15 minutes pour comprendre un concept d'AmbiProg.

Cette hypothèse devrait être vérifiée lors d'une prochaine expérimentation. Pour éviter les biais et la perte de temps, il faudrait borner chaque session à 15 minutes et enseigner les concepts un par un.

Différences entre la représentation mentale des enfants et la réalité : la représentation mentale des enfants ne reflétait pas la réalité. Les enfants pensaient ne pas avoir de problème pour contrôler le robot ni pour effectuer les exercices. C'est extrêmement positif car cela signifie qu'ils étaient volontaires et motivés alors qu'ils devaient réaliser des tâches complexes. AmbiProg leur a permis de faire des scénarii, d'exécuter les scénarii et de contrôler les robots sans difficulté. Dans un processus d'apprentissage ce n'est pas un problème d'effectuer des erreurs au moment de créer des scénarii. Le plus important est d'assurer la motivation des apprenants. En effet, si des utilisateurs sont motivés pour utiliser AmbiProg, ils apprendront plus facilement comment créer des scénarii et contrôler leur environnement. AmbiProg est accepté par les enfants car il est facile à manipuler.

L. Evaluation de StimCards (enfants et personnages âgés)

Le Chapitre IV du document principal a présenté StimCards, une application, réalisée avec ArCo, qui propose un jeu de cartes pour la stimulation cognitive. StimCards possèdent deux particularités par rapport aux autres jeux qui existent :

- Le jeu est complètement ouvert et configurable. Il est ainsi possible de créer ses propres questions de jeu. Il n'y a donc pas de limite de questions et cela évite la redondance du jeu et *in fine* l'ennui des joueurs.
- StimCards fait partie des nouvelles générations d'Interface Homme-Machine : les NUI (Natural User Interface). Ces nouvelles interfaces permettent d'éviter la passivité des utilisateurs. Dans ce cas de StimCards, les joueurs manipulent des cartes de jeu et interagissent avec des robots ou des agents virtuels au moyen de caméra et de tablettes tactiles. La multiplication des modalités de communication permettent d'éviter l'ennui.

StimCards semble posséder des bonnes propriétés pour favoriser l'interaction entre l'homme et le jeu mais est-il vraiment accepté ? Deux études ont été réalisées pour tester son acceptabilité. Les deux études ont suivi le même protocole expérimental mais ont ciblé deux types de participants : des enfants de 10 ans et des personnes âgées.

A. Evaluation de StimCards auprès d'enfants

Cette expérimentation a fait l'objet de l'article [168].

1. Participants

Les participants de l'étude 5 étaient 52 élèves de CM2 (28 filles et 24 garçons) d'une moyenne d'âge de 10,27 ans. Pour cette première étude de StimCards, le choix s'est porté sur des enfants car l'objectif était de vérifier si le jeu était simple à comprendre et facile à utiliser. De plus, les enfants ne sont pas experts en informatique, ce qui représente un large échantillon de la population. Et comme ils n'ont pas de préjugés concernant la technologie, ils représentaient des bons indicateurs de l'acceptabilité de StimCards auprès des personnes âgées. Ce qui ne signifie pas que si StimCards plaît aux enfants alors il plaît aux personnes âgées. Mais cela signifie surtout que si StimCards ne plaît pas aux enfants, alors cela ne sert à rien de le proposer aux personnes âgées.

2. Méthode

L'expérimentation s'est déroulée dans une pièce spécifique où chaque participant était isolé avec StimCards. Les participants passaient les uns après les autres et jouaient avec StimCards pendant 4 sessions expérimentales.

Equipement : La Figure 85 présente la configuration minimale du cadre expérimental de l'expérimentation. Dans les quatre sessions, un ordinateur affiche l'interface de StimCards et la vue de la caméra. Cette vue permet de vérifier que la carte est bien scannée. La caméra est contenue dans un boîtier noir fendue. Chaque participant doit glisser la carte dans la fente pour que le système la détecte. Un jeu de carte est disposé, en cinq paquets devant l'ordinateur. Sur la gauche, se trouve une boîte en carton dans laquelle le participant dispose les cartes après les avoir utilisés. Enfin, une tablette tactile permet aux participants de saisir leur réponse. Dans les sessions 2, 3 et 4, un robot ou un agent virtuel accompagnait StimCards et donnait les consignes. (cf. chapitre V.D).



Figure 85 : Cadre expérimental de la cinquième étude.

Déroulement de l'expérimentation : Lors de la première partie de l'expérimentation, l'ordinateur, le robot ou l'agent virtuel expliquait les règles du jeu au participant. Ensuite, le participant suivait les instructions. Il devait présenter une carte à la caméra et répondre à la question. Après chaque réponse, le participant était félicité, s'il avait donné une bonne réponse, ou encouragé, si sa réponse était mauvaise. Ensuite, il était invité à prendre une nouvelle carte. Après neuf cartes, le participant était remercié et la session était terminée. (cf. chapitre V.D pour plus de détails).

Après les 4 sessions, le questionnaire du Tableau 9 était distribué et chaque participant devait y répondre.

3. Collecte de données et analyses

A la fin des 4 sessions, les participants devaient remplir un questionnaire qui évaluait StimCards. Le Tableau 9 montre les six questions posées aux participants. Les deux premières questions évaluaient l'intérêt de StimCards, la troisième question évaluait sa simplicité. Les 3 dernières questions s'intéressaient à la personnalisation de StimCards. Est-ce que chaque participant souhaite avoir un jeu totalement personnalisé ?

Tableau 9 : Questionnaire de l'étude 5

1. As-tu aimé ce jeu ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
2. Aimerais-tu avoir ce jeu à la maison ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
3. Est-ce que les règles du jeu étaient faciles ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
4. Est-ce que les interlocuteurs sont intervenus au bon moment ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
5. Est-ce que tu voudrais pouvoir faire intervenir ton interlocuteur quand tu veux et comme tu veux ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐

6. Est-ce que tu aimerais que ton interlocuteur soit personnel (spécialement adapté pour toi) ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐

4. Résultats

La Figure 86 présente les résultats du questionnaire. Les enfants ont énormément aimé jouer avec (4, $X^2=86,8462$, $p=0,000$). Ils aimeraient d'ailleurs l'avoir à la maison (4, $X^2=61,8462$, $p=0,000$). Les règles du jeu ont été jugées très faciles (4, $X^2=53,5769$). Concernant le scénario du jeu, les élèves ont estimé que les interlocuteurs sont intervenus au bon moment (4, $X^2=21,2549$, $p=0,000$). Les élèves ont majoritairement exprimé leur envie de pouvoir faire intervenir leur compagnon quand ils veulent et comme ils veulent (4, $X^2=31,6538$, $p=0,000$). Il faut cependant noter que cette question est contrastée car la deuxième réponse la plus formulée est le *Pas du tout*. Pour finir, les élèves aimeraient que leur compagnon soit personnalisé (4, $X^2=69,7308$, $p=0,000$).

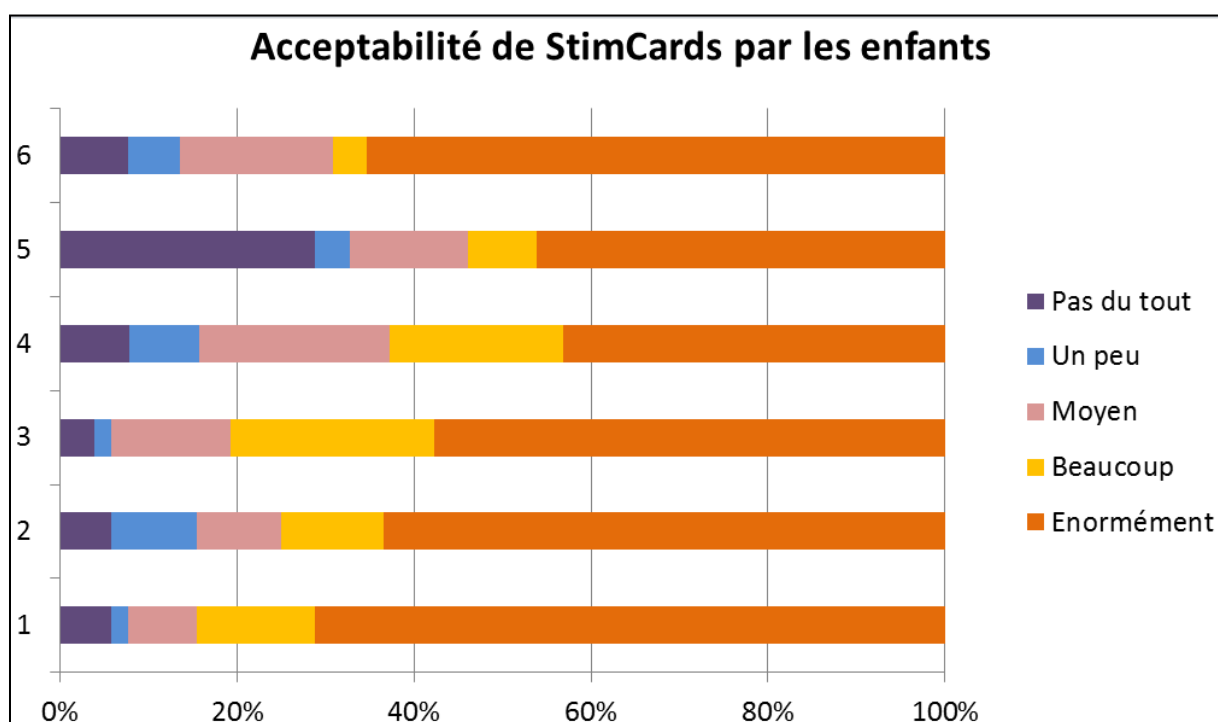


Figure 86 : Résultats de l'étude de l'acceptabilité de StimCards par les enfants.

Niveau de significativité : * $p<0,05$, ** $p<0,01$, *** $p<0,001$ (Test du Chi-deux)

StimCards a été accepté par les enfants. Malgré des exercices qu'ils ont parfois jugés difficiles, ils ont aimé le jeu. Ces résultats montrent que StimCards motive les enfants.

B. Evaluation de StimCards auprès de personnes âgées

Cette étude a été réalisée avec la collaboration de Ya-Huei Wu et Jérémy Wrobel de l'hôpital Broca et d'un ensemble de participants volontaires.

1. Participants

Les participants ont été contactés par téléphone parmi une base de données de participants potentiels. L'expérimentation a été effectuée par 18 personnes, âgées de 63 à 88 ans avec une moyenne d'âge de 75,71 ans. Chaque participant a signé un formulaire de consentement avant d'effectuer l'expérimentation.

2. Méthode

Tout comme l'étude 5, l'expérimentation s'est déroulée dans une pièce spécifique où chaque participant était isolé avec StimCards. Les participants passaient les uns après les autres et jouaient avec StimCards pendant 3 sessions expérimentales.

Equipement : La Figure 87 présente la configuration minimale du cadre expérimental de l'expérimentation. Dans les trois sessions, un ordinateur affiche l'interface de StimCards et la vue de la caméra. Cette vue permet de vérifier que la carte est bien scannée. La caméra n'est pas contenue dans un boîtier noir comme dans l'étude 5 car les enfants avaient parfois eu des difficultés à glisser la carte dans la fente. Ainsi, dans cette étude, les participants devaient présenter le code barre devant la caméra. Un jeu de carte est disposé devant l'ordinateur. Sur la gauche se trouve un boîtier de réponse que les participants devaient utiliser pour saisir la réponse aux questions. Dans les sessions 2 et 3, un robot ou un agent virtuel accompagnait StimCards et donnait les consignes.

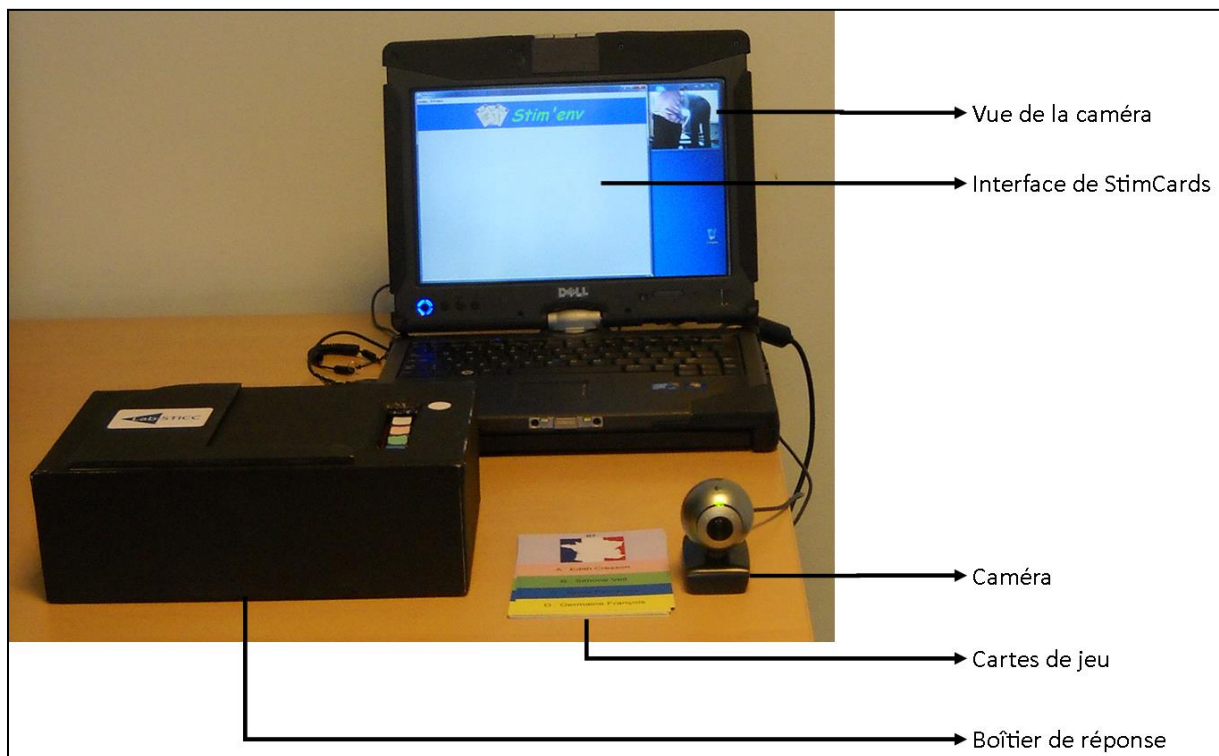


Figure 87 : cadre expérimental de la sixième étude

3. Collecte de données et analyses

A la fin des 3 sessions, les participants devaient remplir un questionnaire qui évaluait StimCards. Le Tableau 10 montre les six questions posées aux participants. C'étaient les mêmes questions que pour l'étude faite avec les enfants mais reformulées d'une autre façon, plus adaptée au langage et à la culture des personnes âgées. Les deux premières questions évaluaient l'intérêt de StimCards, la troisième question évaluait sa simplicité. Les 3 dernières questions s'intéressaient à la personnalisation de StimCards. Est-ce que chaque participant souhaite avoir un jeu totalement personnalisé ?

Tableau 10 : Questionnaire de l'étude 6

1. J'ai apprécié de jouer au jeu.				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
2. J'aimerais continuer à utiliser ce jeu dans l'avenir.				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
3. J'ai trouvé les règles du jeu facile.				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
4. L'ordinateur, l'agent virtuel ou le robot ont parlé aux bons moments pour dire les bonnes choses ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
5. Il serait intéressant de pouvoir modifier le scénario du jeu (décider des paroles prononcées, du moment où elles sont dites, du nombre de questions posées...)				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
6. Il serait intéressant que l'interlocuteur s'adresse à moi d'une façon particulière, avec de l'empathie, des émotions.				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐

4. Résultats

La Figure 88 présente les résultats du questionnaire. StimCards provoque un avis tranché puisque soit les participants ont beaucoup apprécié de jouer avec StimCards, soit pas du tout. Mais pour la majorité des personnes, l'avis est positif. En effet, 38,89% des personnes ont beaucoup apprécié le jeu et 55,56% l'ont énormément apprécié. La majorité des personnes ont souhaité continuer à utiliser le jeu dans l'avenir (27,78% Beaucoup, 38,89% Enormément). Seules 11,11% des personnes ne veulent plus utiliser ce jeu. Les questions 3 et 4 ont reçu des réponses positives puisque personne n'a répondu « Pas du tout », « Un peu » et « Moyen ». Tous les participants ont trouvé les règles du jeu faciles (11,11% Beaucoup, 88,89% Enormément). Tous les participants ont estimé que l'ordinateur, l'agent virtuel et le robot ont parlé aux bons moments pour dire les bonnes choses (44,44% Beaucoup, 55,56% Enormément). Les réponses aux deux dernières questions sont plus mitigées puisqu'autant de personnes ont répondu « Pas du tout » et « Enormément » pour les deux questions (27,78% pour la question 5 et 22,22% pour la question 6). La réponse la plus souvent donnée pour la question 6 est « Moyen » (27,78%).

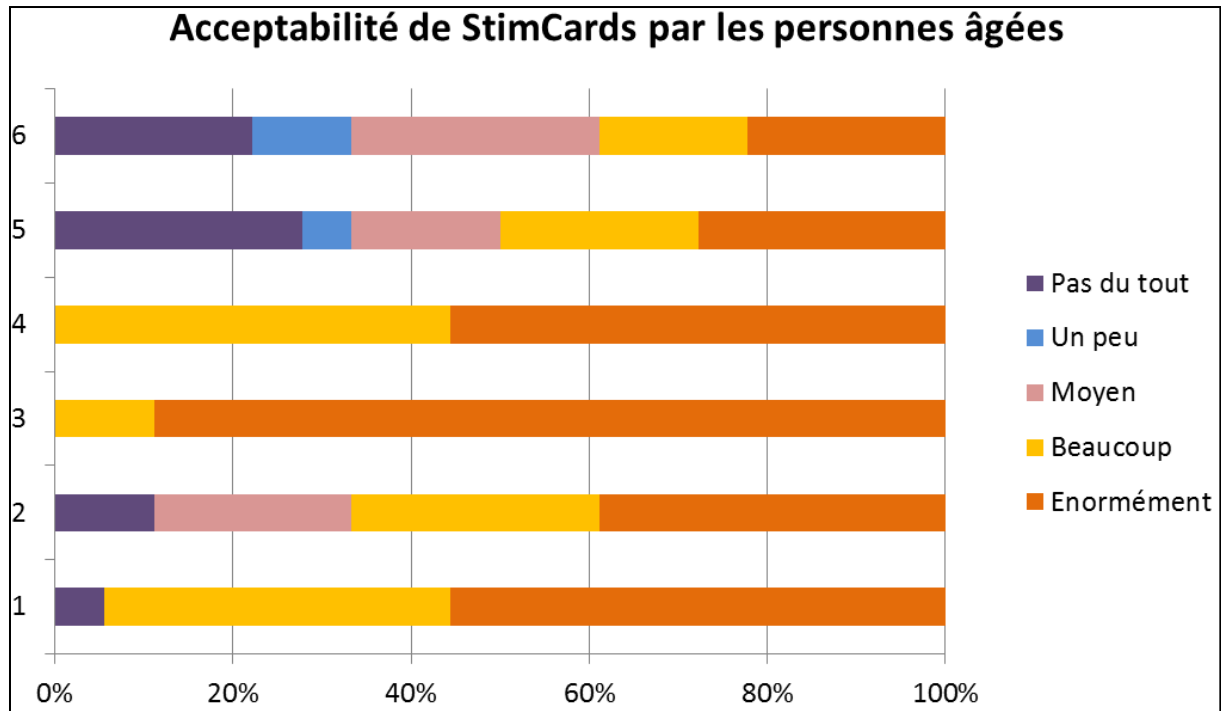


Figure 88 : Résultats de l'étude de l'acceptabilité de StimCards par les personnes âgées.

5. Discussion

StimCards a été accepté et apprécié par les personnes âgées. C'est un jeu qui semble être apprécié de tous, sans distinction d'âge, ni de sexe. Par contre, on note une différence en ce qui concerne les 4 dernières questions. Les personnes âgées ont exprimé beaucoup plus de voix positives concernant la facilité des règles du jeu et la participation correcte des ordinateur, agent virtuel et robot. Par contre, concernant les questions 5 et 6, les personnes âgées ont semblé moins intéressées que les enfants. Ceci est peut-être dû au fait qu'il est difficile de savoir ce qu'il est possible de faire avec StimCards. Lorsqu'on n'est pas initié à la programmation d'un jeu avec ArCo, il n'est pas évident d'imaginer les possibilités.

M. Evaluation du meilleur partenaire numérique

Cette expérimentation a fait l'objet de l'article [168].

Cette septième étude a été réalisée avec la collaboration du laboratoire Ethos de Rennes I, dans le cadre d'un stage de Master effectué par Vanessa André. L'objectif était de faire interagir un enfant avec un ordinateur, un robot ou un agent virtuel afin de mesurer lequel de ces trois interlocuteurs était préféré des enfants.

Cette étude a été mise en place avec le jeu StimCards et a permis d'effectuer l'évaluation du chapitre V.C.

A. Participants

Les participants étaient des élèves de CM2 (moyenne d'âge : 10,27 ans) de deux écoles françaises (28 filles et 24 garçons).

B. Méthode

Equipement : L'expérimentation se passait dans deux écoles différentes. Il n'était donc pas possible, à priori, d'obtenir le même cadre expérimental pour tous les élèves puisque les pièces fournies par les directeurs étaient différentes. Nous avons donc construit une structure, placée à l'intérieur de chaque pièce. La Figure 89 montre l'organisation de la structure. C'était une cabine cubique d'environ 1m60, fermée par des rideaux verts. Le vert avait été choisi non seulement pour augmenter la luminosité dans la cabine mais encore pour sa gaieté et sa nature apaisante.

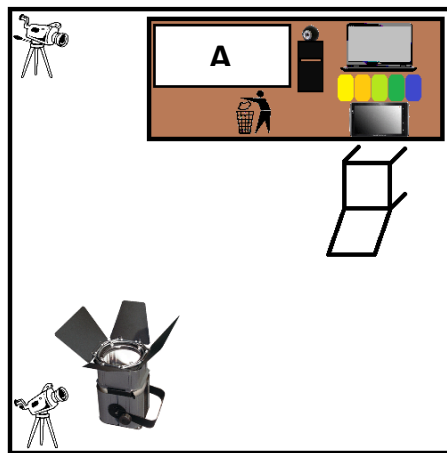


Figure 89 : Cadre expérimental de la septième étude

Deux miroirs sans tain se trouvaient sur les murs latéraux de la cabine, afin de contrôler le bon déroulement de l'expérimentation, puisque l'enfant était isolé à l'intérieur. Un bureau et une chaise se trouvaient au fond de la cabine, face à l'entrée. Deux caméras positionnées sur la gauche filmaient l'interaction. Une première caméra filmait le visage de l'enfant, tandis qu'une deuxième caméra filmait l'enfant en entier, sous un autre angle, pour apercevoir les mouvements des jambes et des mains et la posture générale. Un projecteur illuminait l'intérieur de la cabine. Le matériel sur le bureau était à disposition des enfants pendant la durée de l'expérimentation. Un ordinateur affichait l'interface de StimCards. Les cartes de jeu étaient posées devant l'ordinateur. Elles étaient séparées en cinq paquets : jaune clair, jaune foncé, vert clair, vert foncé et bleu. Chaque couleur représentait un niveau de difficulté, du plus facile au plus difficile. Une tablette tactile était posée devant l'enfant et affichait l'interface graphique de la Figure 90. Une caméra se trouvait dans un boîtier noir, à gauche de l'ordinateur. Les enfants devaient positionner une carte dans la fente, afin que

l'ordinateur puisse lire le code barre et traiter la question. Une boîte « poubelle » permettait de déposer chaque carte, une fois qu'elle avait été utilisée. La zone A était l'emplacement réservé au compagnon. Grâce à cette mise en place, le cadre expérimental était le même pour chacun des participants.



Figure 90 : Interface de réponse proposée sur la tablette tactile

Déroulement de l'expérimentation : L'expérimentation a duré 4 jours dans chacune des écoles. Chaque jour, les enfants passaient les uns après les autres pour une série d'exercices. La durée était d'environ 10 minutes, comprenant les explications et le jeu. La différence entre les 4 jours était le contenu de la zone A : l'ordinateur, un personnage animé projeté sur un écran, un robot et un animal (le robot déguisé en habit de peluche). La Figure 91 montre chacun des compagnons proposés. L'interlocuteur du jeu était soit l'ordinateur, le personnage, le robot ou l'animal et énonçait les consignes. Pour l'école 1, l'ordre de passage était le suivant : ordinateur, robot, personnage et animal. Pour l'école 2, l'autre de passage était le suivant : animal, personnage, robot et ordinateur.



Figure 91 : Les quatre conditions de la septième étude

Chaque session se déroulait de la façon suivante : Le compagnon se présente et explique les règles du jeu. C'est un jeu de calcul mental à 5 niveaux, du plus simple au plus complexe : jaune clair, jaune foncé, vert clair, vert foncé et bleu. L'utilisateur doit présenter une carte à la caméra. S'il ne fait rien pendant une minute, le compagnon lui demande à nouveau de présenter une carte à la caméra. S'il prend une mauvaise carte, le compagnon lui rappelle le niveau de la carte qu'il doit présenter. La question s'affiche dans l'interface de StimCards, par exemple : « Quel est le résultat de $28+31$? » L'utilisateur saisit sa réponse sur la tablette tactile à l'aide de l'interface illustrée sur la Figure 3. Si l'utilisateur donne la bonne réponse, il monte d'un niveau. S'il donne une mauvaise réponse ou s'il ne répond pas, il est invité à prendre une nouvelle carte du même niveau.

Il dispose en effet de 5 cartes par niveau. L'utilisateur est invité à refaire la même opération 9 fois afin de vider le paquet de cartes de plus haut niveau s'il ne se trompe jamais.

Le scénario du jeu est montré sur la Figure 92. Les phrases dites par le compagnon sont les suivantes :

- | | |
|------------|--|
| A | Bonjour je suis le compagnon. Tu te souviens ? Je me suis présenté la dernière fois. Comment t'appelles-tu ? (A' = Quel est ton nom ?) |
| B | Nous allons jouer à un jeu ensemble. Pour cela tu as besoin du jeu de cartes sur la table devant toi, de la caméra qui se trouve dans le boîtier noir, du clavier tactile devant toi et de la poubelle à gauche du clavier tactile. Est-ce que tu vois tout cela ? |
| C | Prends un instant pour repérer les touches sur le clavier tactile, il y a des touches jaunes avec des chiffres de 0 à 9 pour taper ta réponse. Ta réponse s'affichera dans la case blanche. Il y a aussi une touche rouge en haut à droite pour effacer ta réponse si tu te trompes. Il y a aussi une touche verte en bas à droite pour valider ta réponse. Est-ce que tu vois tout cela ? |
| D | Les cartes de jeu sont réparties dans des paquets de couleurs différentes : jaune clair, jaune foncé, vert clair, vert foncé et bleu. Tu vois ces paquets ? |
| E | Je vais t'expliquer les règles du jeu. Prends une carte et retourne là. Tu vois il y a un code barre derrière la carte. Quand je te dirais, "montre le code barre à la caméra", tu glisseras la carte dans le boîtier noir en faisant attention de mettre le code barre vers la caméra ensuite je te lirais la question. Tu as compris ? |
| F | Pour répondre à la question tu tapes la réponse sur le clavier tactile puis tu valides ta réponse. Tu as compris ? |
| G | Tu as une minute pour répondre. Quand tu as répondu à la question tu peux poser la carte dans la poubelle. Tu as tout compris ? |
| H | Maintenant nous allons jouer et je vais te guider pendant le jeu. |
| I | Prends une carte dans le paquet jaune clair et glisse la carte dans le boîtier. |
| I' | Je t'explique à nouveau. Prends une carte dans le paquet jaune clair et glisse la carte dans le boîtier. |
| I'' | Attention, tu t'es trompé de carte. Prends une carte dans le paquet jaune clair et glisse la carte dans le boîtier. |
| J | La question est la suivante : Quel est le résultat de [...] ? |
| K | Bravo c'est une bonne réponse |
| K' | Le temps est écoulé, mais ce n'est pas grave, nous allons essayer une nouvelle carte. |
| K'' | Ce n'est pas la bonne réponse mais ce n'est pas grave nous allons essayer une nouvelle carte |
| L | Le jeu est terminé. Merci beaucoup d'avoir joué avec moi. A bientôt !
Les phrases de B à G possèdent une phrase alternative, de B' à G', dans le cas où l'enfant n'a pas compris la consigne. Les phrases I, I' et I'' évoluent en fonction du niveau de la question : jaune clair, jaune foncé, vert clair, vert foncé et bleu. |
-

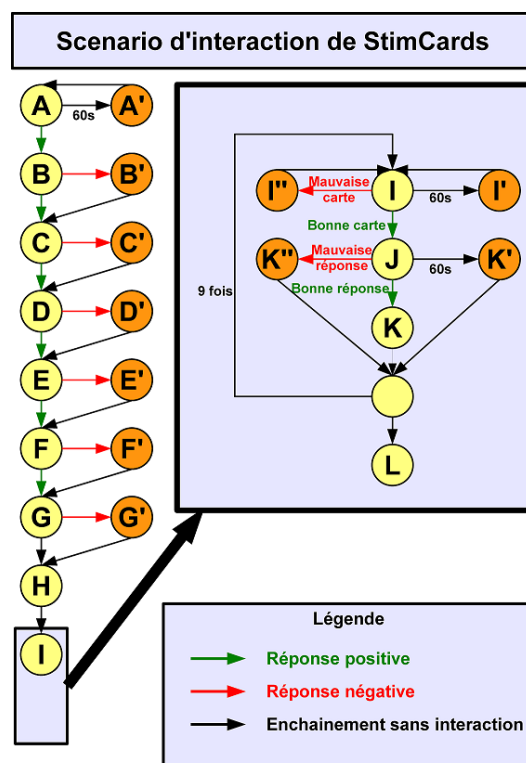


Figure 92 : Scénario d'interaction de StimCards lors de la septième étude

C. Collecte de données et analyses

Après chacune des sessions, les participants devaient remplir le questionnaire décrit par le Tableau 10. Les treize premières questions étaient évaluées avec une échelle de Likert. Les réponses de la question 14 étaient un classement des mots suivants : ordinateur, personnage, robot et animal. La réponse de la question 15 était un des mots de la liste précédente.

Tableau 11 : Questionnaire de l'étude 7

1. As-tu trouvé les exercices faciles ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
2. Penses-tu que la présence du compagnon t'ait aidé pour répondre aux questions ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
3. As-tu aimé jouer avec le compagnon ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
4. T'es-tu senti(e) encouragé(e) par le compagnon pendant le jeu ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
5. Trouves-tu le compagnon sympa ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
6. Trouves-tu le compagnon énervant ?				
Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐

Pas du tout ☐	Un peu ☐	Moyen ☐	Beaucoup ☐	Enormément ☐
7. Aimerais-tu avoir le compagnon pour t'aider à faire tes devoirs à la maison ?				
☐	☐	☐	☐	☐
8. Penses-tu que le compagnon sait faire des calculs mentaux ?				
☐	☐	☐	☐	☐
9. Penses-tu que le compagnon ait compris tes réponses ?				
☐	☐	☐	☐	☐
10. Penses-tu que le compagnon puisse te voir ?				
☐	☐	☐	☐	☐
11. Penses-tu que le compagnon puisse t'entendre ?				
☐	☐	☐	☐	☐
12. Penses-tu que le compagnon t'aime bien ?				
☐	☐	☐	☐	☐
13. Penses-tu que le compagnon soit content de jouer avec toi ?				
☐	☐	☐	☐	☐
14. Quel est ton ordre de préférence des compagnons ?				
15. Si tu pouvais choisir un compagnon pour t'aider dans la vie quotidienne, lequel choisirais-tu ?				

Pour les questions 1-13, l'échelle de Likert était utilisée pour les propositions de réponse : Pas du tout, Un peu, Moyen, Beaucoup, Énormément. Dans le reste du document, les notations sont 0 pour *Pas du tout*, 1 pour *Un peu*, 2 pour *Moyen*, 3 pour *Beaucoup* et 4 pour *Énormément*.

Les analyses statistiques ont été réalisées avec le logiciel Minitab 15©. Le test du Chi Deux a permis de vérifier les réponses significatives. Le seuil de significativité (p) a été fixé à 0,05. A chaque question de 1 à 13, les statistiques ont été calculées pour chaque compagnon (ordinateur, personnage, robot et animal) pour tous les élèves confondus, mais également en séparant les deux écoles. Les statistiques ont également été calculées pour l'ensemble des compagnons confondus et pour tous les élèves. En plus des statistiques, chaque question a reçu un score qui correspond à une somme pondérée (valeur de la réponse en fonction du nombre de votes obtenus). Ainsi, plus un score est élevé, plus la question a reçu de vote positif et inversement.

D. Résultats

Tout élève confondu, les exercices ne sont pas jugés difficiles que ce soit avec l'ordinateur (0, $X^2=23,3846$, $dl=4$, $p=0,000$), avec le personnage (0, $X^2=9,53846$, $dl=4$, $p=0,049$) ou avec le robot (0, $X^2=11,2692$, $dl=4$, $p=0,024$). Les résultats ne sont pas significatifs dans l'ensemble pour l'animal. La première école juge les exercices plus faciles avec l'animal (4, $X^2=22$, $dl=4$, $p=0,000$) et ni faciles, ni difficiles avec l'ordinateur (2, $X^2=18,5217$, $p=0,001$). Les autres résultats ne sont pas significatifs. La tendance est inversée pour la deuxième école qui juge les exercices plus faciles avec l'ordinateur (4, $X^2=16,6897$, $dl=4$, $p=0,002$) et ni faciles, ni difficiles avec l'animal (2, $X^2=16,3448$, $dl=4$, $p=0,003$). L'ordinateur reçoit le score le plus élevé et le personnage le score le moins élevé.

Les résultats ne sont pas significatifs concernant la question 2, sauf pour l'école 1 où la présence du personnage n'a pas aidé les élèves pour répondre aux questions (0, $X^2=11,1305$, $p=0,025$). Globalement, les réponses sont mitigées mais Pas du tout obtient le plus grand nombre de voix. Le score le plus élevé est obtenu par l'ordinateur et le plus faible par le personnage.

Les enfants ont aimé jouer avec les compagnons (4, $X^2=117,529$, $p=0,000$ pour l'ensemble des compagnons). Il n'y a aucune différence significative entre les différents compagnons. Le score le plus élevé est obtenu par l'animal et le score le plus faible est obtenu par le robot.

Les résultats ne sont pas significatifs pour la question 4, excepté pour l'animal où les élèves se sont sentis encouragés (les deux écoles : 3, $X^2=10,8846$, $p=0,028$, l'école 1 : NS, l'école 2 : 4, $X^2=34,2759$, $p=0,000$). Le score le plus élevé est obtenu par l'animal et le score le plus faible est obtenu par le robot.

Les élèves ont trouvé tous les compagnons sympas (Ordinateur : 4, $X^2=10,1154$, $p=0,039$, Personnage : 4, $X^2=18,9615$, $p=0,001$, Robot : 4, $X^2=18,5769$, $p=0,001$, Animal : 4, $X^2=39,1538$, $p=0,000$). Le score le plus élevé est obtenu par l'animal et le score le plus faible est obtenu par l'ordinateur.

Concernant la question 6, les élèves n'ont pas jugé les compagnons énervants (Ordinateur : 0, $X^2=73,5769$, $p=0,000$, Personnage : 0, $X^2=66,8462$, $p=0,000$, Robot : 0, $X^2=68,9615$, $p=0,000$, Animal : 0, $X^2=113,385$, $p=0,000$). Le meilleur score, qui est ici le score le plus faible est obtenu par l'animal, tandis que le moins bon score est obtenu par le robot.

De façon générale, les élèves aimeraient avoir le compagnon à la maison pour les aider à faire leurs devoirs (4, $X^2=56,4541$, $p=0,000$). Dans le détail, les résultats ne sont pas significatifs pour l'ordinateur. Le score maximal est obtenu par l'animal et le minimal par le personnage.

Les résultats ne sont pas significatifs pour la question 8 pour chacun des compagnons, mais en les prenant dans l'ensemble, il s'avère que les élèves pensent que leurs compagnons savent faire des calculs mentaux (4, $X^2=11,7596$, $p=0,019$). Une exception est faite pour l'école 1, qui considère de façon significative que le personnage ne sait pas faire de calcul mental (1, $X^2=11,5652$, $p=0,021$). Le score maximal est obtenu par l'ordinateur, tandis que le minimal est obtenu par l'animal.

Les élèves considèrent que les compagnons ont compris leurs réponses (dans l'ensemble : 4, $X^2=115,894$, $p=0,000$). Les résultats sont les mêmes pour tous les compagnons et pour les deux écoles indépendamment, sauf pour la deuxième école qui ne montre aucun résultat significatif pour l'ordinateur. Le résultat le plus fort est obtenu par l'animal, tandis que le résultat le plus faible est obtenu par le robot.

Concernant la question 10, dans l'ensemble, les élèves ne pensent pas que le compagnon peut voir (0, $X^2=31,2788$, $p=0,000$). Dans le détail, ces résultats ne sont significatifs que pour l'ordinateur et le robot. Le score le plus élevé est obtenu par le personnage tandis que le score le plus faible est obtenu par le robot.

Les élèves pensent dans l'ensemble que le compagnon peut les entendre (4, $X^2=22,5411$, $p=0,000$). Individuellement, les résultats ne sont pas significatifs, même si la réponse Énormément est choisie majoritairement pour tous les compagnons. Le score le plus élevé est obtenu par le robot et le plus faible par l'animal.

Concernant la question 12, il n'y a pas de résultats significatifs, ni dans l'ensemble, ni dans le détail. Les réponses sont très mitigées. Dans l'ensemble, les cinq réponses ont quasiment toutes obtenues le même nombre de votes, même si l'ordinateur et le personnage semble donner d'avantage l'impression d'aimer les élèves que le robot et l'animal. Le score maximal est obtenu par le personnage et le score minimal par le robot.

Les élèves pensent que le compagnon est content de jouer avec eux (Dans l'ensemble : 4, $X^2=56,2609$, $p=0,000$, Ordinateur : 3, $X^2=213,1923$, $p=0,010$, Personnage : 4, $X^2=17,8077$, $p=0,001$, Robot : 4, $X^2=13,4118$, $p=0,009$, Animal : 4, $X^2=20,1154$, $p=0,000$). Les statistiques les plus significatives sont celles de l'animal, suivies par le personnage, puis le robot et l'ordinateur. Le score le plus élevé est obtenu par l'animal et le score le plus faible est obtenu par l'ordinateur.

Pour toutes les questions, aucune différence significative entre les compagnons n'est apparue.

Le Tableau 12 répertorie le nombre de fois que chaque compagnon a obtenu le score maximal (m1) et le score minimal (m2). Le score total est la somme de tous scores obtenus auquel on a soustrait le score de la question 6 qui est négative.

Tableau 12 : Score obtenu par les compagnons au questionnaire de l'étude 7

Compagnon	m1	m2	Score total
Ordinateur	3	2	1450
Personnage	2	3	1432
Robot	1	6	1411
Animal	7	2	1483

Concernant le choix du compagnon préféré des enfants, le robot et l'animal arrive tous les deux en tête avec 39,22% des voix. Le système robotique totalise donc 78,43% des voix. Le troisième compagnon est le personnage avec 11,76% des voix, puis finalement l'ordinateur avec 9,80%. Significativement, l'ordinateur n'est pas le compagnon choisi par les enfants ($X^2=16,5294$, $p=0,001$, Figure 93).

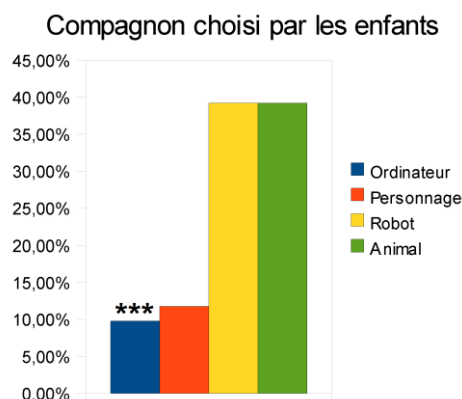


Figure 93 : Résultat du compagnon choisi par les enfants

Niveau de significativité : * $p<0,05$, ** $p<0,01$, *** $p<0,001$ (Test du Chi-deux)

La Figure 94 montre l'ordre de préférence donné par les enfants, qui indique que le robot a été le plus de fois cité en premier (46,15%, $X^2=21,2308$, $p=0,011$). Il n'a jamais été cité en dernier. Le deuxième compagnon le plus cité est également le robot avec 36,54% des voix, suivi de près de l'animal avec 30,77% des voix. Significativement, l'ordinateur est le moins cité en deuxième position (5,77%, $X^2=11,2308$, $p=0,011$). Le troisième compagnon est le personnage (42,31%, $X^2=10,3077$, $p=0,016$) et enfin le dernier compagnon est l'ordinateur (55,77%, $X^2=33,0769$, $p=0,000$). L'animal est plus cité en quatrième position qu'en troisième position, à hauteur de 21,15%.

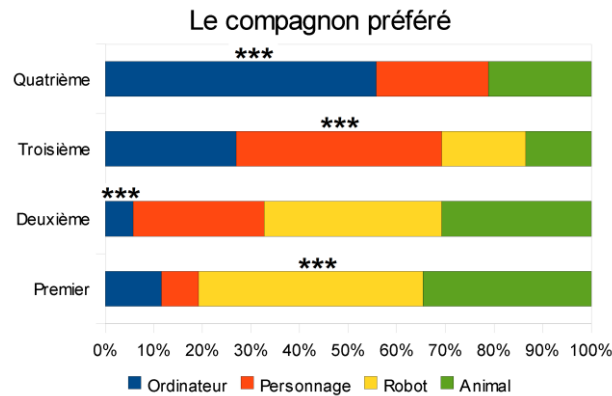


Figure 94 : Résultat du compagnon choisi en fonction de l'ordre de préférence

Niveau de significativité : * $p < 0,05$, ** $p < 0,01$, *** $p < 0,001$ (Test du Chi-deux)

E. Discussion

L'étude statistique des questions 1 à 13 n'ont pas déterminé l'interlocuteur préféré, qui a été jugé en tant que coordinateur du jeu. Il semblerait que l'apparence de l'interlocuteur n'avait pas d'importance sur la préférence des enfants. Nous pensons que les enfants auraient pu faire un choix si les interlocuteurs avaient eu un caractère et une personnalité. En effet, ils avaient tous un comportement neutre et cela confirme que l'apparence ne suffit pas à créer une relation avec un être humain. Les émotions et l'empathie sont nécessaires. Cependant, malgré cela, l'animal semble avoir provoqué le plus d'intérêt puisqu'il a obtenu le meilleur score. Mais ce résultat est contredit par l'ordre de préférence des enfants. Même la dernière question n'a pas pu départager le robot et l'animal car ils ont obtenu tous deux la première place. Pour conclure, il n'y a aucun doute que le système robotique est l'interlocuteur préféré des enfants puisqu'il a reçu environ 80% des votes.

Discussion: The statistical study of the questions 1-13 did not determine the preferred HGI. These questions judged it as game coordinator. The study showed that the HGI shape was not important to be liked. However, the children would surely be able to make a choice if the HGI had some character and personality. Nevertheless the animal seemed to be the most interesting for the children because it obtained the best questions score. This result was contradicted by the children order of preference. Indeed, the robot won this question. Even the last question was not able to decide between the robot and the animal, because they were tied as the first place. To conclude, there is no doubt that the robotic system is the preferred HGI because it received around 80% of the votes.

N. Evaluation d'AmbiCop

Cette évaluation n'est pas une étude comme les évaluations précédentes. Les participants n'ont pas eu de questionnaires à remplir ni de tâches à effectuer. Cette expérimentation consistait à tester AmbiCop en conditions réelles. Lors du concours Robofesta, évoqué au J.A.1, beaucoup de personnes peuvent avoir du temps libre entre chaque épreuve. Nous avons profité de cette occasion pour demander à des participants du concours de tester nos outils. La Figure 95 montre l'organisation de cette évaluation. Elle montre qu'il y avait :

- 3 entités agissantes perceptives : une manette de PS3, un smartphone android utilisé comme une manette et une caméra.
- 3 entités agissantes actives : un hexapode, un chien Aibo de Sony et Greta.
- 4 scénarii : permettant de gérer indépendamment les 3 entités agissantes actives et ayant des priorités différentes.

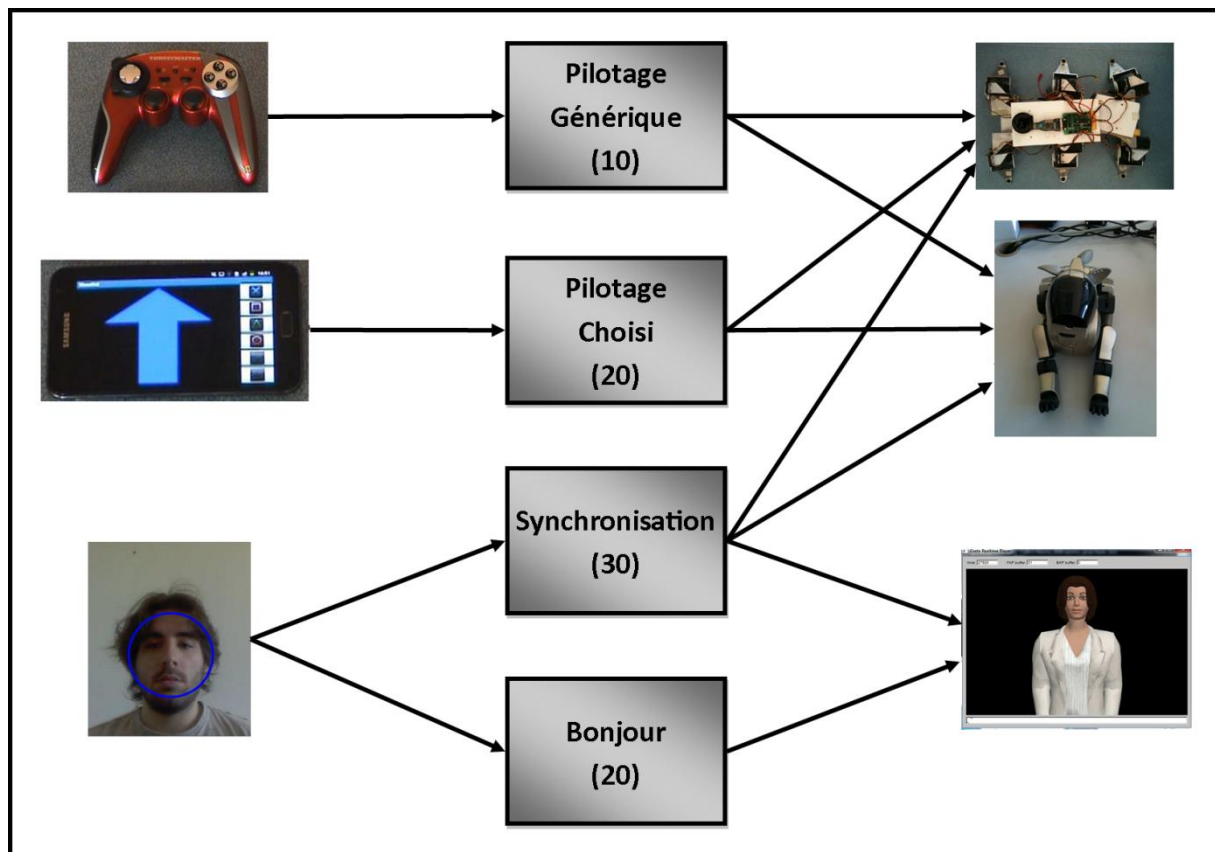


Figure 95 : Organisation de l'évaluation d'AmbiCop

A. Les scenarios

Scénario 1 : le premier scenario utilisait la manette de PS3, l'hexapode et l'Aibo. Il avait une priorité de 10, c'était donc le scénario le moins prioritaire. Il consistait à piloter un robot. Le scénario utilisait des actions génériques, donc la manette devait piloter le robot dont l'entité agissante était libre et de meilleure qualité. Par défaut, l'hexapode avait une meilleure qualité que l'Aibo.

Scénario 2 : le deuxième scénario permettait à l'utilisateur de choisir le robot qu'il voulait piloter en saisissant son choix sur l'interface présenté par le smartphone android. Il pouvait choisir l'hexapode ou l'Aibo. La priorité de ce scénario était supérieure à celui du premier scénario.

Scénario 3 : le troisième scénario utilisait la caméra et Greta. Il avait la même priorité que le deuxième scénario. Ceux-ci n'utilisant pas le même matériel, il n'était pas nécessaire de mettre une priorité différente. Lorsque la caméra détectait un visage, alors Greta saluait la personne qui se trouvait devant la caméra.

Scénario 4 : le quatrième scénario utilisait la caméra, l'hexapode, l'Aibo et Greta. Ce scénario avait la priorité la plus haute des quatre scénarii. Pour ce scénario, il fallait utiliser une carte de StimCards. Lorsque la caméra détectait le code barre prévu, l'hexapode, l'Aibo et Greta saluaient l'utilisateur de façon synchronisée. Aucun des trois ne pouvait participer à un nouveau scénario tant que les trois n'avaient pas complètement fini leur action.

B. Déroulement de l'expérimentation

La configuration par défaut était que l'hexapode avait une meilleure qualité que l'Aibo. Ainsi, si un premier utilisateur choisissait la manette PS3, il pilotait forcément l'hexapode. Si un deuxième utilisateur choisissait de piloter l'hexapode avec le smartphone, alors il devenait prioritaire et la manette PS3 contrôlait alors l'Aibo. Lors de cette expérimentation, tous les cas ont été testés. Par exemple, un premier utilisateur choisissait de piloter l'Aibo ou l'hexapode avec le smartphone. Lorsqu'un deuxième utilisateur commençait à manipuler la manette de PS3, le robot libre était choisi et le deuxième utilisateur pilotait le deuxième robot. En parallèle, Greta saluait toutes les personnes détectées par la caméra. De temps en temps, le carte de jeu était passée devant la caméra pour tester le bon fonctionnement de la synchronisation.

C. Conclusion

Cette expérimentation a permis de tester le bon fonctionnement d'AmbiCop en vérifiant :

- L'absence de conflits d'accès aux entités agissantes (et donc aux dispositifs numériques associés) lors de multi-scénarii.
- La synchronisation des actions.

Les actions génériques en fonction de la priorité des scénarii et de la qualité des entités agissantes.

O. Questionnaire de l'étude sur la gestuelle du robot

Garçon/Fille		Age : _____		
Dialogue 1				
Phrase	Le geste s'accorde-t-il avec le contenu de la phrase ?	La vitesse du geste est-elle correcte ?	Le geste semble-t-il souple et naturelle ?	Le robot est-il crédible quand il parle ?
P1	Pas du tout ☐	Trop lent ☐	Pas du tout ☐	Pas du tout ☐
	Un peu ☐	Lent ☐	Un peu ☐	Un peu ☐
	Beaucoup ☐	Normal ☐	Beaucoup ☐	Beaucoup ☐
	Sans idée ☐	Rapide ☐	Sans idée ☐	Sans idée ☐
		Trop rapide ☐		
P2	Pas du tout ☐	Trop lent ☐	Pas du tout ☐	Pas du tout ☐
	Un peu ☐	Lent ☐	Un peu ☐	Un peu ☐
	Beaucoup ☐	Normal ☐	Beaucoup ☐	Beaucoup ☐
	Sans idée ☐	Rapide ☐	Sans idée ☐	Sans idée ☐
		Trop rapide ☐		
P3	Pas du tout ☐	Trop lent ☐	Pas du tout ☐	Pas du tout ☐
	Un peu ☐	Lent ☐	Un peu ☐	Un peu ☐
	Beaucoup ☐	Normal ☐	Beaucoup ☐	Beaucoup ☐
	Sans idée ☐	Rapide ☐	Sans idée ☐	Sans idée ☐
		Trop rapide ☐		
P4	Pas du tout ☐	Trop lent ☐	Pas du tout ☐	Pas du tout ☐
	Un peu ☐	Lent ☐	Un peu ☐	Un peu ☐
	Beaucoup ☐	Normal ☐	Beaucoup ☐	Beaucoup ☐
	Sans idée ☐	Rapide ☐	Sans idée ☐	Sans idée ☐

		Trop rapide ☐		
L'enchaînement des répliques est-elle correcte ?		Pas du tout ☐ Un peu ☐ Beaucoup ☐		
La vitesse de réponse du robot et de l'homme est-elle correcte ?		Pas du tout ☐ Un peu ☐ Beaucoup ☐		
Les expressions du robot quand il écoute l'homme sont-elle correction ?		Pas du tout ☐ Un peu ☐ Beaucoup ☐		
Est-ce que le robot est attentif ?		Pas du tout ☐ Un peu ☐ Beaucoup ☐		
Dialogue 2				
Phrase	Le geste s'accorde-t-il avec le contenu de la phrase ?	La vitesse du geste est-elle correcte ?	Le geste semble-t-il souple et naturel ?	Le robot est-il crédible quand il parle ?
P1	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
P2	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
P3	Pas du tout ☐ Un peu ☐	Trop lent ☐ Lent ☐	Pas du tout ☐ Un peu ☐	Pas du tout ☐ Un peu ☐

	Beaucoup ☐ Sans idée ☐	Normal ☐ Rapide ☐ Trop rapide ☐	Beaucoup ☐ Sans idée ☐	Beaucoup ☐ Sans idée ☐
P4	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐

L'enchaînement des répliques est-elle correcte ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
--	--

La vitesse de réponse du robot et de l'homme est-elle correcte ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
--	--

Les expressions du robot quand il écoute l'homme sont-elles correctes ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
---	--

Est-ce que le robot est attentif ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
------------------------------------	--

Dialogue 3

Phrase	Le geste s'accorde-t-il avec le contenu de la phrase ?	La vitesse du geste est-elle correcte ?	Le geste semble-t-il souple et naturel ?	Le robot est-il crédible quand il parle ?
P1	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
P2	Pas du tout ☐	Trop lent ☐	Pas du tout ☐	Pas du tout ☐

Questionnaire de l'étude sur la gestuelle du robot

	Un peu ☐	Lent ☐	Un peu ☐	Un peu ☐
	Beaucoup ☐	Normal ☐	Beaucoup ☐	Beaucoup ☐
	Sans idée ☐	Rapide ☐	Sans idée ☐	Sans idée ☐
		Trop rapide ☐		
P3	Pas du tout ☐	Trop lent ☐	Pas du tout ☐	Pas du tout ☐
	Un peu ☐	Lent ☐	Un peu ☐	Un peu ☐
	Beaucoup ☐	Normal ☐	Beaucoup ☐	Beaucoup ☐
	Sans idée ☐	Rapide ☐	Sans idée ☐	Sans idée ☐
		Trop rapide ☐		

L'enchaînement des répliques est-elle correcte ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
--	--

La vitesse de réponse du robot et de l'homme est-elle correcte ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
--	--

Les expressions du robot quand il écoute l'homme sont-elles correctes ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
---	--

Est-ce que le robot est attentif ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
------------------------------------	--

Dialogue 4

Phrase	Le geste s'accorde-t-il	La vitesse du geste est-elle	Le geste semble-t-il	Le robot est-il crédible quand
---------------	--------------------------------	-------------------------------------	-----------------------------	---------------------------------------

	avec le contenu de la phrase ?	correcte ?	souple naturelle ?	et il parle ?
P1	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
P2	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
P3	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
P4	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Trop lent ☐ Lent ☐ Normal ☐ Rapide ☐ Trop rapide ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐	Pas du tout ☐ Un peu ☐ Beaucoup ☐ Sans idée ☐
L'enchaînement des répliques est-elle correcte ?		Pas du tout ☐ Un peu ☐ Beaucoup ☐		
La vitesse de réponse du robot et de l'homme		Pas du tout ☐ Un peu ☐ Beaucoup ☐		

Questionnaire de l'étude sur la gestuelle du robot

est-elle correcte ?	
Les expressions du robot quand il écoute l'homme sont-elle correction ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
Est-ce que le robot est attentif ?	Pas du tout ☐ Un peu ☐ Beaucoup ☐
Est-ce que le robot est agréable ?	

Références

- [1] A. NEMERY and E. BRANGIER, "Grille de critères ergonomiques pour l'appréciation de la Persuasion Technologique." avril-2011.
- [2] A. Tricot, F. Plégat-Soutjis, J. F. Camps, A. Amiel, G. Lutz, A. Morcillo, and others, "Utilité, utilisabilité, acceptabilité: interpréter les relations entre trois dimensions de l'évaluation des EIAH," in *Environnements Informatiques pour l'Apprentissage Humain 2003*, 2003, pp. 391–402.
- [3] S. Pesty, D. Duhaut, and others, "Acceptability in interaction-From robots to Embodied Conversational Agent," in *Computer graphics theory and applications*, 2011.
- [4] D. Heylen, S. Kopp, S. Marsella, C. Pelachaud, and H. Vilhjalmsson, "Why conversational agents do what they do? Functional representations for generating conversational agent behavior," in *The First Functional Markup Language Workshop. Estoril, Portugal*, 2008.
- [5] C. Jost, B. Le Pévédic, and D. Duhaut, "Study of the Importance of Adequacy to Robot Verbal and Non Verbal Communication in Human-Robot Interaction," presented at the ISR 2012, Taipei, 2012, p. 6.
- [6] J. Riviere, C. Adam, S. Pesty, C. Pelachaud, N. Guiraud, D. Longin, and E. Lorini, "Expressive multimodal conversational acts for SAIBA agents," in *Intelligent Virtual Agents*, 2011, pp. 316–323.
- [7] J. S. Uebersax, "Likert scales: dispelling the confusion," *Statistical methods for rater agreement*, vol. 31, 2006.
- [8] C. Jost, B. L. Pévédic, and D. Duhaut, "Creating Interaction Scenarios With a New Graphical User Interface," *arXiv:1206.3001*, Jun. 2012.
- [9] S. Siegel and N. J. Castellan, "Nonparametric statistics for the behavioral sciences," *New York: McGraw-Hill*, 1956.
- [10] C. Jost, B. Le Pévédic, and D. Duhaut, "ArCo: an architecture for children to control a set of robots," presented at the 21st IEEE International Symposium on Robot and Human Interactive Communication, Versailles, 2012, p. 7.
- [11] C. Jost, B. Le Pévédic, and D. Duhaut, "Robot is best to play with human!," presented at the 21st IEEE International Symposium on Robot and Human Interactive Communication, Versailles, 2012.