Université de La Rochelle - UFR Sciences **École doctorale de La Rochelle**

Exécution adaptative par observation et analyse de comportements Application à des logiciels interactifs pour des enfants autistes

THÈSE

présentée et soutenue publiquement le 7 décembre 2005

pour l'obtention du

Doctorat de l'université de La Rochelle

(spécialité informatique)

par

Karim SEHABA

Composition du jury

Rapporteurs :	Sylvie DESPRÉS, Maître de Conférences - HDR	Université Paris V
	Stéphane NATKIN, Professeur des Universités	ENJMIN & CEDRIC-CNAM
Examinateurs :	Didier ARQUÈS, Professeur des Universités	Université de Marne-la-Vallée
	Pascal ESTRAILLIER, Professeur des Universités	Université de La Rochelle
	Didier LAMBERT, Psychiatre des Hôpitaux	U.P.E.A La Rochelle
	Jean-Claude MARTIN, Maître de Conférences	CNRS-LIMSI Université Paris 8

Laboratoire Informatique – Image – Interaction — UPRES EA 2118

Remerciements

Je tiens à remercier tous les membres du jury qui se sont intéressés à ce travail, malgré l'ampleur de leurs responsabilités :

- Monsieur Pascal ESTRAILLIER, Professeur des universités à l'Université de La Rochelle, pour m'avoir accueilli au sein du laboratoire L3i qu'il dirige et m'avoir proposé et encadré ce sujet pendant ces années de travail.

- Madame Sylvie DESPRÉS, Maître de conférences-Habilité à diriger des recherches à l'Université Paris V, et Monsieur Stéphane NATKIN Professeur des universités et directeur de l'École Nationale de Jeux et Media Interactifs Numériques, qui ont accepté la lourde tâche de rapporter sur mon travail de thèse.

- Monsieur Didier ARQUES, Professeur à l'Université de Marne-la-Vallée et directeur de l'É cole Doctorale ICMS (Information Communication Modélisation Simulation), et Jean-Claude MARTIN, Chercheur CNRS/LIMSI - LINC Université Paris 8, qui ont accepté d'examiner ce travail.

- Monsieur Didier LAMBERT, Docteur-Psychiatre des Hôpitaux et directeur du service de pédopsychiatrie du Centre Hospitalier Marius Lacroix à La Rochelle. J'ai apprécie l'échange d'idées que nous avons eu pendant les nombreuses réunions de travail.

- Merci à tous mes amis et collègues du laboratoire L3i, ainsi qu'à tous les autres doctorants que j'ai eu le plaisir de côtoyer à travers la vie associative.

- J'ai une pensée pour tous mes amis du parcours universitaire, en particulière, pour Mounir qui m'a apporté une aide très précieuse dans la correction du manuscrit.

- Un grand merci à ma famille, en particulier mes parents, qui m'ont encouragé tout au long de mon parcours scolaire et universitaire.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Dans ce chapitre, nous présentons la problématique de l'exécution adaptative par observation et analyse de comportements. Notre propos, illustré par des exemples d'application, décrit l'intérêt de prendre en compte le comportement de l'utilisateur dans les applications interactives qui incluent l'utilisateur humain. Par la suite, nous présentons le cahier des charges ainsi que le plan du mémoire que nous avons adopté.

1.1 Problématique de l'exécution adaptative par observation et analyse de comportements

Dans un spectacle, un artiste joue devant son public, mais aussi *avec* son public, *selon* son public. Dans le monde de l'interactif, raconter une histoire, c'est bien sûr exprimer de manière créative une trame scénaristique prédéfinie, mais c'est aussi devoir prendre en considération, comme le fait l'artiste sur une scène, les réactions des spectateurs afin d'adapter son spectacle, sur la *forme* et parfois sur le *fond*.

Dans un contexte informatique, combien de fois nous sommes-nous énervés devant un logiciel qui réagit d'une manière ne correspondant pas à nos besoins, compétences ou préférences. D'où le besoin de systèmes intelligents capables de proposer des activités personalisées à chaque utilisateur.

Notre objectif général consiste à définir un système capable de « comprendre » le comportement de l'utilisateur et d'y répondre, de manière personnalisée et en temps réel, par des activités adaptées en tenant compte des consignes du concepteur de l'application[1]

Le comportement de l'utilisateur doit être interprété par le système en utilisant de deux approches complémentaires, dont il convient de confronter les résultats à chaque instant pour adapter l'exécution en temps-réel, il s'agit :

- des *actions explicites* de l'utilisateur exprimées au moyen des interfaces traditionnelles (clavier, souris, joystick, écran tactile, etc.). Un dispositif de capture de mouvement sans marqueur permettra aussi à l'utilisateur de « commander » des actions prédéfinies ;
- des situations comportementales de l'utilisateur relevant de l'*implicite* pourront être identifiées au moyen de caméras (gestes, expressions faciales, regard , etc.). De manière complémentaire, une étude sémantique de certaines réalisations commandées par l'application et effectuées par l'utilisateur pourront être interprétées en terme de comportement.

Dans tous les cas, il est nécessaire de se référer à un corpus préalablement établi par un expert du domaine applicatif concerné. Nous pensons que le rôle de ce dernier ne doit pas se limiter à une prescription simple comme tout utilisateur (adaptation, configuration, etc.) mais aussi et surtout en tant qu'auteur dès la conception de l'application (choix et définition des éléments des activités de l'application à présenter, des directives associés à l'application, etc.). Pour cela, les fonctionnalités du système reposent sur une architecture et des mécanismes permettant à l'expert :

- d'utiliser un ensemble d'outils de traitements et de gestion de ressources prédéfinies, pour caractériser et mettre en place une production interactive adaptée à la situation ;
- de définir, au moyen de directives contextualisées, des mécanismes liés au contrôle

[1] Le terme *Application* désigne toute application interactive qui inclut l'utilisateur humain.

adaptatif de l'interactivité et de l'exécution pour établir un cadre narratif ;
- d'observer et d'interpréter, en temps-réel, des comportements inventoriés de l'utilisateur afin de toujours lui proposer une activité adaptée ;
- de gérer, à partir du profil de l'utilisateur, une véritable personnalisation de la production en fonction des caractéristiques et du comportement de l'utilisateur ;
- d'améliorer la production interactive, après évaluation de sa pertinence par la production et l'exploitation de traces d'exécution. Des mécanismes contribuent à la gestion évolutive du profil de l'utilisateur et des directives contextualisées.

Très schématiquement, dans notre approche, l'exécution de l'application peut être vue comme une progression d'états successifs à partir d'un état initial. Cette exécution doit être à tout moment adaptée à la *situation* caractérisée par une représentation du contexte associé à l'utilisateur (croyance sur son nouvel état en fonction du dernier comportement interprété, par exemple), l'état des ressources de l'application, et la situation par rapport à la trame scénaristique. Pour cela, différentes capacités complémentaires doivent être mises en œuvre dans le système :

- la *perception :* permet au système de connaître (au moins partiellement) l'utilisateur ;
- la *scénarisation :* regroupe l'ensemble des mécanismes lui permettant de raisonner à partir de son propre état, d'une modélisation de l'utilisateur et de ses moyens d'action afin de choisir et de planifier les activités à présenter à l'utilisateur ;
- l'*exécution :* des activités est rendue possible par des effecteurs qui lui permettent d'agir sur l'application.

L'adaptation, que nous considérons, dans ce travail n'est pas uniquement dans le processus de scénarisation du système qui consiste à produire des activités à présenter à l'utilisateur, mais également dans l'interaction utilisateur-application. C'est ce que nous appelons ***l'exécution adaptative***.

> ***L'exécution adaptative*** est la gestion de l'évolution de l'exécution de l'application par le système en fonction du profil, besoins et comportement de l'utilisateur, dans le but de déclencher la meilleure action possible permettant :
> - de réduire la "distance" entre l'état estimé concernant le comportement de l'utilisateur et l'état dans lequel l'expert du domaine voulait que soit l'utilisateur à ce stade ;
> - de préserver les ressources liées à l'activité ;
> - de progresser dans la trame scénaristique.

La figure 1.1 illustre notre propos. Une fois que le système détecte que les réactions du l'utilisateur ne sont pas conformes à l'activité en cours (par exemple, l'action 2 qui sort de la trame scénaristique qui mène à l'objectif souhaité), le système modifie l'activité de manière à ce qu'elle soit conforme au comportement perçu de l'utilisateur comme le fais l'enseignant dans son cours. Dans un premier temps, il définit une stratégie sous forme d'un plan de séquences d'éléments pédagogiques. Le plan peut être remis en cause en pratique lorsque l'enseignant détecte des malentendus de ses élèves sur certains éléments. Dans ce cas il change sa stratégie en approfondissant ces éléments de manière à pourvoir continuer le reste de son plan initial.

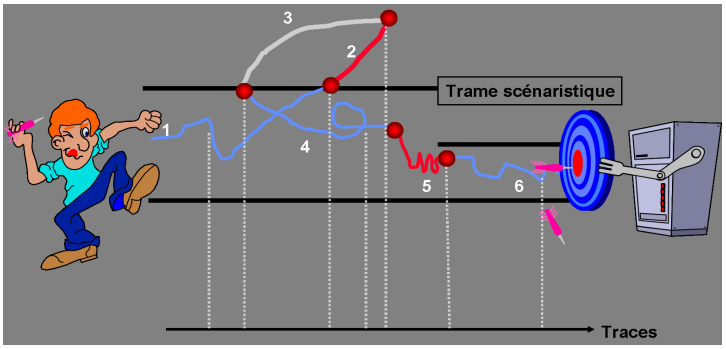

FIG. 1.1 – Principe de l'exécution adaptative

En ce sens, la prise en compte du comportement de l'utilisateur, dans les applications interactives qui incluent l'utilisateur humain, est nécessaire à l'exécution d'un mécanisme permettant de générer des activités adaptées à la situation. Par ce lien, l'observation du comportement de l'utilisateur devient un concept utile au développement des mécanismes de prise de décision des systèmes interactifs.

En conséquence, nous proposons de prendre en compte le comportement de l'utilisateur dans une architecture assurant un processus d'exécution adaptative. Il s'agit d'observer et d'analyser le comportement de l'utilisateur afin de détecter les cas où les activités proposées par le système ne sont plus adaptées à son comportement. Cette analyse peut servir à alimenter le système pour qu'il adapte ses scénarios.

La difficulté de ce processus d'exécution adaptative apparaît suivant le degré de prise en compte des réalités qu'on veut observer du comportement de l'utilisateur. En effet, il faut tenir compte de la nature dynamique et plus ou moins imprévisible de ce dernier. Les deux extrêmes sont les suivants :

- Une première technique consiste à planifier un scénario sans tenir compte de la nature dynamique et imprévisible des actions de l'utilisateur. Un problème survenant lors de l'exécution du scénario sera géré au moment où il se présente, et plusieurs techniques peuvent alors être utilisées : replanifier complètement, ou bien tenter de réparer le scénario suivant les critères considérés. On suppose en outre que le système, dans ce cas, possède une connaissance parfaite des effets de ses actions ; un défaut dans ces connaissances sera géré comme un événement d'exception ;
- Une seconde technique consiste à planifier un scénario en tenant compte de tout événement pouvant se produire et du fait que la connaissance des comportements observés est imparfaite. Il va alors falloir que le scénario d'activités proposées par le système soit capable de gérer tout événement imprévu, que celui-ci soit lié à l'état du comportement de l'utilisateur ou aux effets des actions. Un tel scénario est par nature beaucoup plus complexe que pour la première technique : le système doit effectuer des observations afin de contrôler l'exécution du scénario. Mais il est en revanche plus robuste, c'est-à-dire qu'il permettra de gérer un plus grand nombre de situations.

C'est dans la deuxième technique qu'on se place pour répondre à des problèmes liés aux applications interactives dont l'exécution évolue, en temps-réel et de manière personnalisée à chaque utilisateur en restant dans une trame scénaristique initialement prévue, mais en prenant en compte le comportement de l'utilisateur. Il s'agit donc *de contrôler l'exécution* par prise en considération du comportement de l'utilisateur.

Dans la section suivante, l'intérêt de l'exécution adaptative par observation et analyse de comportements est illustré à travers quelques exemples d'application.

1.2 Exemples d'application

Actuellement, les applications interactives qui incluent l'utilisateur humain sont nombreuses pour illustrer l'intérêt de la prise en compte du comportement de l'utilisateur. On se limite à trois exemples : les jeux, les logiciels éducatifs et à la navigation sur le web.

1.2.1 Les jeux

Parmi les champs d'application de l'informatique, les jeux ont occupé une place particulière comme le montre plusieurs études [NY05] [NV03]. Après le graphisme, il semble qu'actuellement le prochain défi est d'introduire plus d'intelligence dans les jeux comme le souligne [MRU01], ce qui peut être considéré comme un élément moteur pour les technologies informatiques dans la mesure où il concerne un secteur économique important et de nombreux capitaux.

L'exécution adaptative du comportement des personnages du jeu constitue un challenge pour l'informatique. Non seulement les personnages doivent planifier leurs actions pour atteindre leurs buts, mais également tenir compte des agissements des autres agents participants (humains ou informatiques) de manière continue. Pour cela il faut mettre à l'épreuve des réflexes et capacités de raisonnement instantanés afin que les personnages s'adaptent à la situation à laquelle ils sont confrontés.

Considérons un personnage dont le but est de prendre un objet dans une pièce derrière une porte fermée. Supposons qu'au moment où il cherche la clé qui convient, un autre agent change la serrure de cette porte. Si, au cours de sa recherche, le personnage est amené à voir la porte maintenant avec une nouvelle serrure, il semble raisonnable qu'il arrête cette recherche (de la première clef) pour en rechercher la deuxième afin d'atteindre l'objet visé. En conséquence, ce personnage doit être capable de construire un scénario pour atteindre son objectif ("prendre l'objet derrière la porte fermée") mais doit également être capable d'adapter la stratégie du scénario aux changements imprévus de son environnement ("arrêter de chercher l'ancienne clé et chercher la nouvelles").

Dans cette optique de l'exécution adaptative du comportement des personnages, des travaux de recherche proposent d'intégrer la notion de l'apprentissage guidé par des connais-

sances du joueur et sa maîtrise progressive du jeu[2]*. Au fil des parties, le jeu en apprend de plus en plus sur le joueur, afin d'adapter ainsi sa façon d'agir pour renouveler toujours l'intérêt du jeu et offrir plus de diversité dans le comportement du jeu. Plus le joueur progresse, plus le jeu est capable de soutenir son niveau. D'autres travaux (voir par exemple [MRU01]) considèrent l'interaction comme un moyen de représentation de connaissances et comme la base de la dynamique du moteur de raisonnement des agents. Ces derniers représentent les personnages du jeu. Cette représentation permet à l'agent d'adapter son comportement en fonction de l'état de sont environnement.*

Cependant, l'état actuel des techniques adaptatives auxquelles nous nous intéressons font que tous les problèmes supplémentaires que posent les jeux sont des obstacles difficiles à surmonter pour l'instant. En outre, les jeux avec des techniques adaptatives sont encore peu nombreux, comme le souligne [Sig04], malgré l'intérêt d'anticiper les comportements des personnages pilotés par les êtres humains, de manière à mettre en œuvre un ensemble de mécanismes adaptatifs permettant d'élaborer des comportements adaptés.

1.2.2 Les logiciels éducatifs

Les logiciels éducatifs désignent tout environnement informatique conçu pour favoriser un apprentissage humain. Le terme utilisé dans la littérature est EIAH (Environnement Informatique pour l'Apprentissage Humain). Il s'agit d'un environnement qui intègre des agents humains (apprenants ou tuteurs) et artificiels (informatiques) et leur offre des conditions d'interactions, localement ou à travers des réseaux informatiques.

Les domaines d'applications couverts par ses environnements concernent l'apprentissage de la preuve en mathématiques [Bal99] [Lue99], la géométrie descriptive [Pau99], l'apprentissage de la lecture [Clé02], etc.

L'exécution adaptative dans ce contexte applicatif est justifiée par l'imprévisibilité des événements qui peuvent avoir une issue non prévue par les scénarios d'activités prédéfinis. Il s'agit de détecter ces évènements, provoqués par les actions des apprenants, et de proposer des actions adéquates. Comme le fait le tuteur lorsqu'il détecte des erreurs ou des malentendus de ses élèves lors de la session, il attire habituellement l'attention des élèves sur un petit sous-ensemble de connaissances impliquées, de manière à corriger efficacement ces erreurs ou ces malentendus.

Les mécanismes de prise de décision par élaboration des scénarios d'activités pédagogiques ont toujours été considérés comme une question importante dans l'enseignement et l'apprentissage (voir à titre d'exemple [EFnC00]). Dans une perspective de l'exécution adaptative des scénarios pédagogiques, les auteurs [RMPB03] proposent de répartir les scénarios entre l'hypermédia et l'agent pédagogique pour des situations pédagogiques données (niveau de l'apprenant, contenus à présenter), d'autres considèrent *l'observation* comme un facteur important dans la qualité du scénario pédagogique [SCE05b] [SC05]

[2]Par exemple, le sujet de thèse proposé par Stéphane NATKIN (2005) "*Un modèle de l'apprentissage du joueur dans les jeux vidéo*", Laboratoire CEDRIC

[MHCF04] et qui doit même se trouver au centre même de l'acte éducatif. L'observation consiste ici à capter et évaluer la prestation et les compétences acquises par les apprenants au cours du processus éducatif.

En conséquence, l'exécution adaptative, consiste à recueillir des données relatives au comportement de l'apprenant vis-à-vis des ressources pédagogiques proposées par le système (cours, exercices,..) et d'agir sur celle-ci. L'observation concerne l'interaction entre l'apprenant et le système pour identifier les capacités de l'apprenant, de détecter les difficultés éventuelles, ou encore de noter les caractéristiques inattendues de la situation afin d'avoir des indications sur la compréhension ou la satisfaction de l'apprenant. Ces indications vont alimenter le système pour qu'il mette à jour, sur le fond et la forme, le contenu des activités qu'il propose afin de maintenir et satisfaire les besoins de l'apprenant.

1.2.3 La navigation sur le web

L'accès à l'immense quantité d'informations grâce au World Wide Web, est en train de modifier l'approche de recherche et d'accès aux documents qui contiennent cette information. Ces documents ne sont pas statiques ni consultés passivement mais ils sont souvent générés à la demande (documents virtuels), et dans lesquels la consultation implique une participation active de l'utilisateur internaute. Ce dernier point rend donc importante la notion d'adaptation et de personnalisation (voir par exemple [CGLSN05]) de ces documents virtuels par prise en compte du comportement de l'utilisateur, afin de faciliter leur consultation. L'adaptation nécessite une acquisition de connaissances sur les internautes et leurs manières de réagir, leurs préférences, compétences, etc.

L'analyse du comportement des utilisateurs des sites Web, également connue sous le nom de Web Usage Mining (WUM), consiste à adapter les techniques de fouille de données sur la trace issue de la navigation de l'utilisateur. Il s'agit de caractériser chaque document par des éléments tels que sa structure, son contenu, sa position dans l'ontologie, etc. Une fois cette caractérisation effectuée, des méthodes d'extraction de motif sont appliquées. Le principe de ces techniques consiste à rechercher des motifs (voir des connaissances sur les comportements des utilisateurs d'un site Web) qui prennent en compte tous les éléments qui caractérisent le document afin de fournir des résultats qui soient compréhensibles par l'utilisateur.

L'apport de ces travaux réside aussi dans la prédiction du comportement probable de l'utilisateur [CPM02] afin adapter de différentes façons le document en cours de consultation. Il s'agit bien d'une exécution adaptative dans laquelle l'observation et l'analyse de comportements sont appliqués sur certaines opérations effectuées sur la page web.

L'intérêt de l'exécution adaptative, dans ce contexte, consiste à faciliter à l'utilisateur l'accès à un certain nombre d'opérations. Par exemple : changement de l'ordre des éléments d'un menu, mise en évidence des liens hypertextes par un changement de typographie ou de mise en page, etc. Cette démarche relève de l'instrumentation du document virtuel, ou de l'Adaptive navigation support selon la taxonomie proposée par [Bru01].

1.2.4 Synthèse

À travers ces trois exemples, nous pouvons constater que l'analyse du comportement de l'utilisateur devient un aspect important dans le déroulement des applications interactives incluant l'utilisateur humain. Cette analyse porte sur les actions de l'utilisateur appliquées sur les objets de l'application qui présentent un intérêt particulier, comme nous allons démontrer dans cette thèse en se basant sur des fondements théoriques issus de la psychologie.

Nous avons présenté trois exemples dans lesquelles l'utilisateur est en interaction. Dans chaque exemple, nous avons montré l'intérêt de prendre en compte le comportement de l'utilisateur dans le but d'assurer une exécution adaptative :

- l'intérêt de l'exécution adaptative du comportement des personnages du jeu réside dans le renouvellement de l'intérêt du jeu afin d'offrir plus de diversités concernant son déroulement ;
- l'imprévisibilité des actions des apprenants en interactions avec les logiciels éducatifs, nécessite un mécanisme d'exécution adaptative permettant aux systèmes de proposer des activités adéquates à la situation ;
- concernant la navigation sur le web, l'exécution adaptative permet d'assister l'utilisateur et d'adapter la façon de la structuration du document en consultation.

Bien évidement les trois exemples que nous avons cités, ne constituent pas l'ensemble des domaines applicatifs uniques dans lesquels l'exécution adaptative par observation et analyse de comportements peut être utile. D'autres domaines applicatifs peuvent être cités comme par exemple, l'interaction homme robot. Il s'agit dans ce cas d'adapter le comportement du robot en fonction de sa perception de son environnement en particulier de la présence humaine.

1.3 Cahier des charges et le plan adopté

La détermination de la stratégie de l'analyse de comportements de l'utilisateur s'avère nécessaire afin de garantir une exécution adaptative. Il s'agit de définir un modèle permettant la représentation et l'identification du comportement. Cet objectif constitue notre première problématique. Sur la base de l'identification de comportement, qu'il convient de déterminer, notre deuxième problématique concerne le processus de décision qui consiste à adapter le scénario, même au cours de son exécution, de manière à ce qu'il soit cohérent par rapport aux profil, besoins et comportement de l'utilisateur. Cette problématique nécessite de pouvoir spécifier des mécanismes dont l'interactivité avec l'utilisateur prend en compte l'analyse de son comportement.

Sur la base de ces deux problèmes, nous spécifions les principaux résultats attendus de notre travail dans cette thèse, il s'agit :

1. définir ce qu'est un comportement et établir un formalisme permettant la représentation et l'analyse de comportements ;

2. établir une représentation du profil de l'utilisateur et des directives de l'expert, ainsi définir le rôle de ce dernier ;
3. définir le processus d'adaptation du scénario même au cours de son exécution ;
4. concevoir une architecture comportant des mécanismes assurant le contrôle de l'exécution, en temps réel, dans un cadre applicatif permettant d'évaluer la pertinence et l'efficacité de l'approche que nous proposons à travers des expérimentations sur le terrain.

Après ce chapitre d'introduction, nous présentons nos travaux en quatre parties (voir la figure 1.2). Le deuxième chapitre, présente le contexte applicatif dans lequel notre travail se situe. Il s'agit du PROJET AUTISME en partenariat avec le service pédopsychiatrie de l'hôpital de La Rochelle[3]. Nous justifions le choix de ce projet pour traiter la problématique de l'exécution adaptative par observation et analyse de comportements. Par la suite, nous donnons une brève description de l'architecture fonctionnelle de l'application. La dernière section est consacrée à la présentation du jeu *Coucou Caché* développé dans le cadre du projet. Ce jeu sera l'exemple illustratif de nos propositions dans cette thèse.

Dans le troisième chapitre, nous traitons la problématique de l'analyse de comportements. Dans un premier temps, nous définissons cette problématique. Nous étudions, dans un seconde temps, la notion du terme *comportement*. Il s'agit d'étudier et analyser les différentes définitions issues de la littérature afin de proposer une définition dans le contexte de l'analyse de comportements de l'utilisateur en interaction avec des applications interactives. Par la suite, nous étudions les différentes approches de l'analyse de comportements, ainsi qu'une discussion comparative de ces approches. Il s'agit de déduire les principales caractéristiques et les problèmes de ces approches dans le cadre de notre problématique. Nous abordons, par la suite, « ***l'analyse de comportements par observation*** » qui constitue notre première contribution dans cette thèse. À partir d'une étude des fondements théoriques des systèmes cognitifs, nous proposons un formalisme de représentation et identification de comportement.

Le quatrième chapitre est consacré au contrôle de l'exécution des applications interactives. Il s'agit d'établir un formalisme permettant de suivre le déroulement des activités afin de répondre à la problématique de l'exécution adaptative telle que nous l'avons définie. Nous avons étudié différentes approches concernant les modèles de contrôle. Notre étude concerne, les systèmes à base de connaissance, les systèmes à base de procédures, les systèmes de classeurs et le storytelling interactif. Par la suite, nous déduisons quelques problèmes pouvant être rencontrés lors de la mise en place de ces approches dans le cadre de l'exécution adaptative afin d'aborder notre approche « ***contrôle d'exécution à partir de cas*** » basée sur le raisonnement à partir de cas. Cette dernière approche constitue notre deuxième contribution.

Dans le cinquième chapitre, nous présentons, avec plus de détails, le traitement de l'étude de cas liée au PROJET AUTISME. Il s'agit, dans une première étape, de resituer plus finement le contexte en référençant quelques travaux de recherches qui conjuguent autisme et technologie informatique. Par la suite, nous identifions quelques spécificités de notre

[3]U.P.E.A. Centre Hospitalier Marius Lacroix, La Rochelle.

application ainsi que l'approche que nous avons adoptée. Nous décrivons, par la suite, les différents composants intervenant dans la conception de notre système. Ce système constitue un cadre d'application des différentes contributions apportées aux chapitres 3 et 4. Afin de valider notre démarche dans ce contexte, nous analysons les résultats que nous avons obtenus avec nos partenaires du secteur médical dans le cadre de ce projet.

À la fin de ce mémoire, nous concluons ces travaux de recherche, et nous donnons quelques pistes pour les améliorations possibles et présentons des perspectives scientifiques dans la continuation de cette thèse.

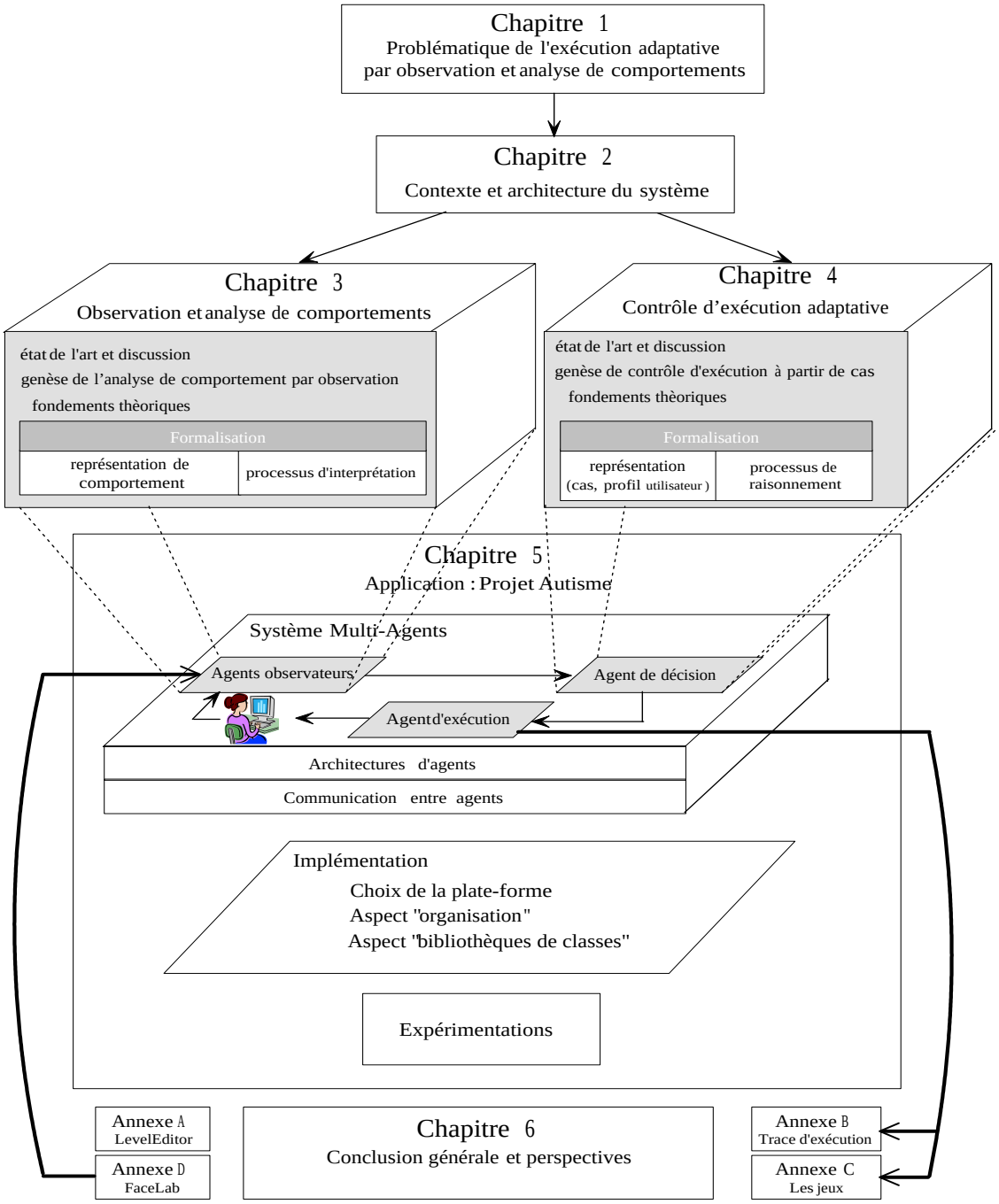

FIG. 1.2 – Plan général du mémoire de thèse

Chapitre 2

Contexte et architecture du système

Ce chapitre est consacré à la présentation du contexte applicatif dans lequel notre travail se situe. Il s'agit du PROJET AUTISME qui consiste à mettre en œuvre un système interactif qui propose des jeux éducatifs destinés à des enfants autistes dans un cadre thérapeutique. Dans un premier temps, nous donnons une brève description du projet. Par la suite, nous justifions le choix de ce projet pour traiter la problématique de l'exécution adaptative. Ainsi, nous évoquons dans ce qui suit le cadre pluridisciplinaire du projet. La dernière partie est consacrée à l'architecture générale du système.

2.1 Contexte applicatif : Le Projet Autisme

L'autisme est un trouble précoce de développement. Il se caractérise par un fonctionnement déviant et retardé dans le domaine des interactions sociales, de la communication et une indifférence de comportement. Souvent, les autistes préfèrent s'isolés, évitent le contacte du regard, ont un comportement répétitif anormal, etc. Le Projet Autisme, présenté dans ce chapitre, consiste à développer un environnement d'aide à la structuration des enfants autistes.

2.1.1 Brève description du projet

Le travail présenté dans cette thèse s'inscrit dans le cadre du Projet Autisme, mené par le laboratoire L3i[4] en partenariat avec le service de pédopsychiatrie[5] de l'hôpital de La Rochelle. Le projet vise à mettre en place un système matériel et logiciel d'aide à la structuration des enfants autistes. Il s'agit d'établir un dialogue multimodal et multimédia, entre l'enfant autiste et le système, dans un processus éducatif et thérapeutique. Le système en question doit permettre aux enfants souffrant d'autisme, de disposer d'un processus d'apprentissage par manipulations interactives des jeux éducatifs avec l'aide d'un tuteur. Les jeux considérés intègrent des éléments sensoriels et affectifs.

Une caractéristique des personnes autistes réside dans leurs différences interindividuelles très importantes. Ce constat conduit à la nécessité de repérer et comprendre les comportements habituels de chaque enfant, leurs sens cliniques et les interventions qui peuvent en réduire les conséquences désorganisées, telles que les comportements de ruptures, évitements, stéréotypés, etc. Une stratégie d'exécution adaptative est donc nécessaire pour la prise de décision du système. Ce dernier doit donc capter par différents moyens (caméras, écran tactile, clavier, etc.) la prestation de l'enfant, de l'analyser et finalement d'y répondre par production de séquences de jeux éducatifs adaptés selon le comportement observé de l'enfant.

2.1.2 Pourquoi ce projet

Nous considérons cette application comme une transposition métaphorique de l'exécution adaptative des interactions que l'on peut anticiper entre l'utilisateur humain et le système. Le système établit, en fonction du profil de l'enfant, un scénario adapté à ses besoins. Pendant l'exécution du scénario, le système détecte les cas où les activités du scénario proposées par le système ne sont pas cohérentes au comportement de l'enfant. De ces cas, l'exécution adaptative consiste à mettre à jour le scénario de manière à l'adapter au comportement observé tout en restant dans la trame scénaristique initialement prévue.

Le choix du contexte applicatif du projet autisme, dans cette thèse, se justifie par

[4]Laboratoire Informatique - Image - Interaction
[5]U.P.E.A. Centre Hospitalier Marius Lacroix, La Rochelle

plusieurs d'autres raisons :

- Premièrement, l'autisme est un domaine qui pose de nombreux problèmes d'ordre thérapeutique où l'outil informatique peut indéniablement apporter un plus comme le montrent plusieurs travaux (voir par exemple [Dau00] [FM05] [Par01]). Les principales difficultés sont liées aux différences interindividuelles très importantes entre individus autistes. Un environnement informatique permettant la libre exploration des jeux peut fournir une aide d'analyse et de compréhension des stratégies thérapeutiques et éducatives adaptées à chaque enfant ;
- Deuxièmement, les méthodes thérapeutiques et éducatives pour les personnes autistes s'inscrivent dans un domaine où il n'y a pas de consensus sur les connaissances à mettre en œuvre et sur leurs modes de présentation. Face aux difficultés inhérentes à ce domaine, il n'existe pas de solution unique et optimale à laquelle les experts puissent adhérer : chacun adapte selon ses propres besoins les méthodes et outils issus de son expérience. De ce fait, cela permet de disposer d'un terrain d'expérimentation idéal pour mettre en évidence le besoin d'une négociation des contenus et d'une paramétrisation des activités selon les besoins ;
- Troisièmement, les travaux antérieurs des étudiants stagiaires du L3i[6], au sein du service de psychiatrie infanto-juvenile de l'hôpital de La Rochelle, nous ont permis de disposer d'une expérience dans le domaine de la conception des jeux thérapeutiques pour des enfants autistes.

Nous voulons à travers ce projet développer un environnement intégrant diverses fonctionnalités fondamentales prêt à être « *rempli* » par des situations concrètes de la discipline choisie. Nous souhaitons même réaliser un « vrai produit », c'est-à-dire un système opérationnel sur le terrain.

2.1.3 Cadre de travail

Ce projet fait appel à différentes expertises théoriques et de terrain qui sont réunies dans une équipe. Ces expertises ne peuvent être recueillies et explicitées qu'au travers des questionnements réciproques. L'équipe est donc nécessairement pluridisciplinaire. La conduite d'un tel projet a mis en relation des participants experts dans des domaines relatifs aux développements d'outils informatiques et des techniques d'accompagnement et d'éducation spécialisées des personnes autistes.

Cette mise en relation est souvent difficile à réaliser : il faut pouvoir trouver des partenaires disponibles, établir des objectifs disciplinaires parfois ambigus et contradictoires, planifier et diriger la collaboration de manière à aboutir à un système qui satisfasse les différents participants.

Notre travail, au sein du L3i, a permis de réunir une partie des conditions autour d'une équipe de chercheurs relevant de deux thématiques du laboratoire : ISI[7] spécialisée

[6]Laboratoire Informatique - Image - Interaction
[7]Image et Séquences d'Images

en traitement d'images et MoCA[8] spécialisée dans la modélisation du comportement et son contrôle dans des applications interactives. Un des verrous scientifiques de MoCA dans lequel nous nous situons, est la conception d'architectures logicielles de jeux centrées interaction. Des spécialistes du domaine de l'autisme ont été associés à cette équipe permanente d'informaticiens, de manière à bénéficier de leur savoir-faire, de leurs compétences et de leur expérience en terme d'accompagnement et d'éducation des enfants autistes. Il s'agit d'une équipe de pédopsychiatres spécialisés dans le domaine de l'autisme du service de psychiatrie infanto-juvénile de l'hôpital de La Rochelle.

2.2 Architecture générale du système

L'architecture du système vise à assurer une exécution adaptative des jeux afin d'apporter souplesse et modularité dans l'accompagnement individualisé de chaque enfant autiste. Elle permet, également, de contrôler l'exécution des jeux, en leur offrant un comportement autonome et intelligent lors de leur exécution. Pour cela, elle regroupe des boucles de contrôle plus ou moins longues, aussi bien pour les comportements réactifs déclenchés par des évènements provoqués par les actions de l'utilisateur, que pour les phases de scénarisation.

D'une manière générale, il s'agit de définir un modèle suffisamment souple pour s'adapter aux caractéristiques émotionnelles et comportementales de chaque utilisateur, et intégrer des données personnalisées de son monde familier et des croyances qui s'y rattachent. Une motivation fondamentale de cette démarche réside dans le lien étroit qui existe entre le comportement et la prise de décision chez les êtres humains. Le principe de notre modèle, présenté dans la figure 2.1, consiste à présenter un scénario d'activités répondant au besoins et profil de l'utilisateur. Au cours de l'interaction, évaluer la pertinence du scénario retenu par rapport au comportement de l'utilisateur et éventuellement le dériver vers d'autres scénarios mieux adaptés.

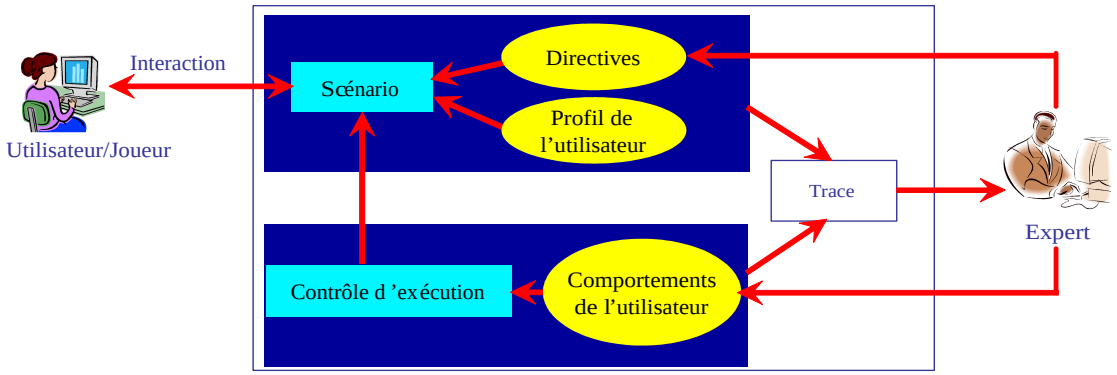

Fig. 2.1 – Architecture générale du système

[8]Modèles, Comportements, Architectures

Formalisation et précisions sur les termes employés
Avant de présenter notre approche, il nous semble nécessaire de préciser certains termes qui seront utilisés tout au long de cette thèse. Ainsi :
- Un ***Jeu*** est caractérisé par :
  - un *décor* statique ;
  - une *collection d'objets* (pictogrammes, musiques, images,. . .) munis de comportements qui définissent les formes d'interactions associées ;
  - des *règles de fonctionnement* ;
  - des *paramètres de configuration* et des *objectifs* à atteindre.
- Une ***Activité*** est une instance d'un jeu avec une configuration particulière et des objectifs qualifiés et quantifiés ;
- Un ***Scénario*** est une séquence d'activités ordonnées, déterminée en vue de permettre à l'enfant de réaliser des objectifs complexes confrontés à des situations complexes ;
- Une ***situation*** caractérise un état particulier du système (et en particulier son évolution liée au comportement de l'utilisateur) et lui associe des traitements qui s'adaptent au cours de l'activité.

Du point de vue de l'utilisateur, nous distinguons le *joueur* (au sens classique du terme) de *l'expert*[a]. Le joueur est défini par un profil le caractérisant dont la formalisation adoptée est présentée à la section 4.4.5. L'expert est caractérisé par son savoir-faire dans un domaine donné. C'est lui qui définit le *jeu*, détermine les *activités* associées. La section 5.3.2.1 décrit plus en détail le rôle de l'*expert*.

[a]Le terme *utilisateur* désigne le *joueur* s'il n'est pas précisé

Étant donnée l'importance de l'interaction dans les systèmes multi-agents, nous avons utilisé ce paradigme pour la mise en œuvre de l'architecture de notre système. Ce choix[9] n'est qu'un moyen de mise en oeuvre *fonctionnelle* de l'ensemble des modules de notre système.

L'architecture que nous proposons inclut les connaissances de l'*expert*, le profil du *joueur* et de la dynamique de leurs interactions. La figure 2.2 montre l'architecture générale du système. Cette architecture est constituée de trois types d'agents : l'agent observateur, l'agent de décision et l'agent d'exécution. Le rôle de l'agent d'observation consiste à identifier les comportements de l'utilisateur qui portent un intérêt particulier dans le déroulement de l'application. Le rôle de l'agent de décision consiste à générer et adapter les activités de l'application au comportement de l'utilisateur. Le rôle de l'agent d'exécution est de mettre en œuvre les actions délibérées par l'agent de décision. Ce qui explique la boucle de la figure 2.2. Les flèches représentent les messages qui circulent entre les agents. Chaque type d'agent est soumis à des contraintes différentes et exploite une représentation de données qui lui est propre. Nous présentons ci-après les fonctions et les principales caractéristiques de chaque agent. L'architecture des agents est présentée en

[9]La section 5.4 donne les éléments qui justifient notre choix de ce paradigme

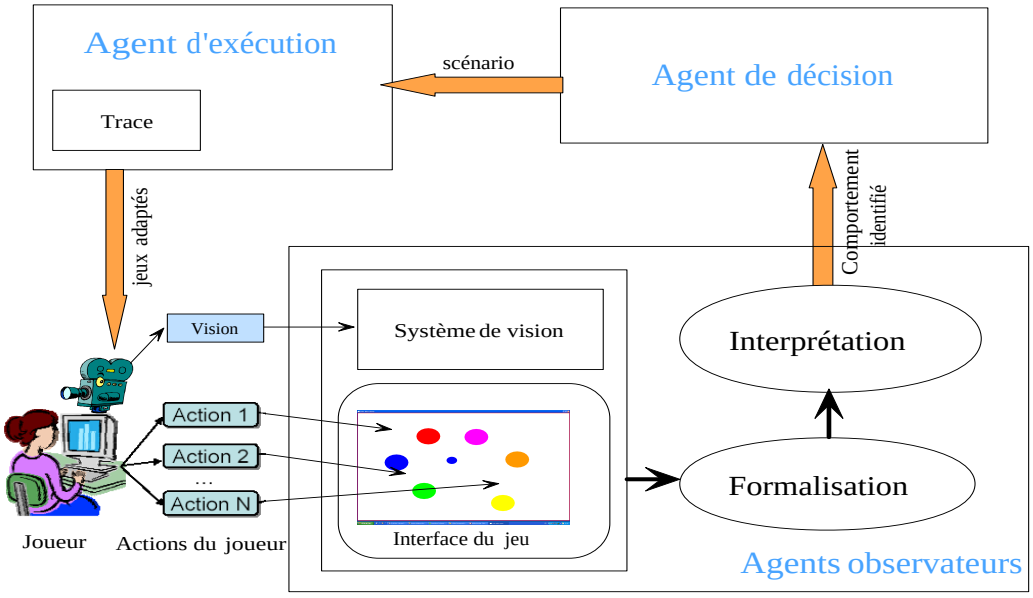

FIG. 2.2 – Architecture générale du système

détail au chapitre 5.

2.2.1 Agent observateur

Cet agent joue le rôle de médiateur entre le système et le joueur. Principalement, il observe les actions du joueur, informe d'autres agents du comportement du joueur si nécessaire et donne l'accès aux ressources du système.

À partir de leurs observations et en se basant sur *la théorie des affordances* [Gib77] [Gib79] et *la théorie de la sémantique procédurale* [Jl77] [Woo81], les agents observateurs détectent et interprètent les réactions du joueur en se référant à un corpus préalablement établi par l'expert. L'observation est faite selon deux approches [SCE06] [SCE05a] : *Action logicielle* et *Vision*.

- Dans l'approche *Action logicielle*, il s'agit de récupérer les actions du joueur sur les éléments de contrôle : souris, écran tactile, clavier, etc ;
- Dans l'approche *Vision*, il s'agit de mesurer les caractéristiques concernant la représentation 3D du visage et de l'orientation du regard. On trouve ici, des fonctions de calculs numériques, de traitement d'images, etc, ainsi que d'autres, plus complexes, telles que le calcul d'une trajectoire ou le tracking visuel. Ces fonctions sont décrites dans un module fonctionnel, il s'agit d'un composant logiciel réutilisable et accessible par des requêtes.

2.2.2 Agent de décision

Cet agent sélectionne et adapte le scénario à adopter en fonction des besoins et du profil du joueur. Il s'appuie sur les agents observateurs pour l'analyse et l'évaluation du comportement du joueur. Il peut donc interagir avec les agents observateurs, accéder au profil du joueur pour sélectionner et adapter les activités du scénario au joueur et mémoriser les nouvelles expériences pour enrichir ses connaissances. Il s'agit donc d'un traitement beaucoup plus long que ceux qui sont mis en œuvre dans les autres agents. En même temps, les scénarios peuvent être modifiés pendant la session s'ils sont incohérents avec le comportement du joueur.

2.2.3 Agent d'exécution

Cet agent, comme son nom l'indique, est chargé d'exécuter les activités fournies par l'agent de décision et de gérer la trace d'exécution. Cette dernière concerne les résultats des interactions joueur-jeu. L'annexe C donne quelques exemples de la trace d'exécution.

2.3 Le jeu *Coucou caché*

Nous venons de voir dans la section 1.2 quelques exemples d'application possibles dont l'exécution adaptative par observation et analyse de comportements peut être utile. Afin de mieux comprendre nos propos, décrivons un exemple d'un jeu développé dans le cadre du PROJET AUTISME. Il s'agit du jeu « *Coucou caché* ». Ce jeu, destiné aux enfants autistes, sera l'exemple illustratif de notre proposition de l'exécution adaptative dans cette thèse. D'autres jeux seront présentés en annexe D.

La figure 2.3 montre l'interface du jeu *Coucou caché*. Ce dernier permet la manipulation d'une ou plusieurs boules de différentes couleurs présentent à l'écran. Ces boules figurent dans un cadre de travail bien délimité et occupent la quasi-totalité de l'écran, laissant ainsi une grande marge de manœuvre à l'enfant. Deux sortes de boules sont présentes à l'écran :

- celles que l'enfant peut manipuler. Il s'agit de la petite boule, appelée *boule curseur*, qu'on peut manipuler en appliquant dessus une force ;
- celles qui restent immobiles. Il s'agit des grosses boules de différentes couleurs.

L'interaction entre la boule curseur et les grosses boules présente deux comportements possibles : soit la boule curseur disparaît à l'approche d'une grosse boule et réapparaît lorsqu'elle s'en éloigne (ce qui laisse à penser qu'elle « se cache derrière »), soit elle stoppe sa progression lorsqu'elle arrive à la périphérie d'une grosse boule et s'accole à elle. Ces interactions entre ces deux objets du jeu (boule curseur et grosses boules) sont accompagnées d'un signal sonore caractérisant une réussite. De même, si l'enfant tente de sortir la boule curseur hors des limites du cadre de travail, un signal sonore est émis lui indiquant qu'il s'éloigne du but recherché.

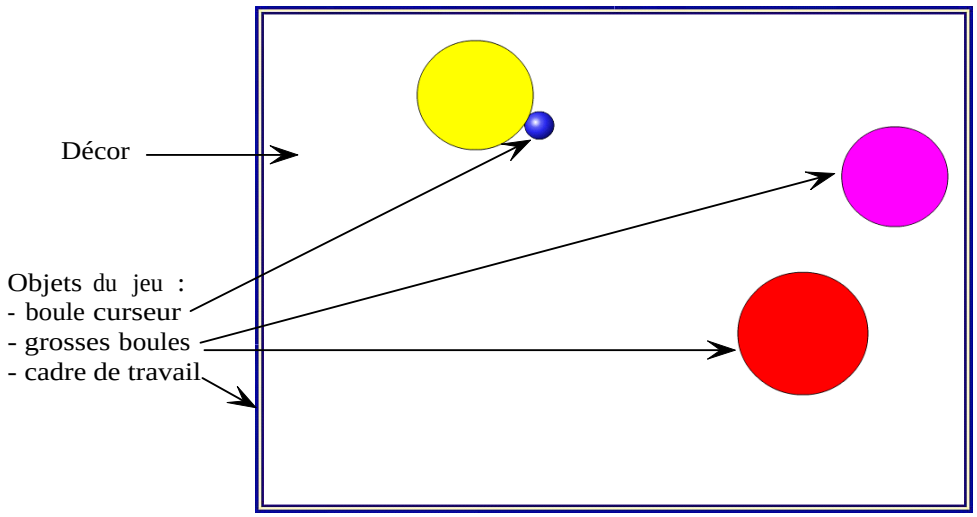

FIG. 2.3 – L'interface du jeu *Coucou Caché*

Présentons à présent le jeu selon la formalisation que nous avons adoptée en section 2.2. Il s'agit de présenter le *décor*, les *objets du jeu* et leurs comportements ainsi que quelques *objectifs* éducatifs du jeu.

2.3.1 Le décor

Pour ne pas perturber l'enfant autiste, souvent sensible aux changements même minimes, la majorité des jeux développés dans le cadre du projet présentent un décor statique. Dans ce jeu, le décor est défini par l'environnement délimité par le cadre de travail.

2.3.2 Les objets du jeu et leurs comportements

Ce jeu est muni, comme nous venons de voir, d'un certain nombre d'objets : il s'agit de la boule curseur, les grosses boules, le cadre de travail et les signales sonores.

L'exécution adaptative de ce jeu consiste à le reconfigurer de manière dynamique de telle sorte qu'il soit le plus cohérent possible aux compétences et besoins de son utilisateur. Ceci est rendu possible grâce à la définition dynamique du comportement des objets. Ces comportements définissent les formes d'interactions associées aux objets du jeu. Nous distinguons sept comportements :

1. **Rencontre boule curseur/grosses boules** : deux comportements sont possibles, changeant ainsi l'objectif éducatif du jeu. Soit la boule curseur disparaît à l'approche des grosses boules (comportement de « Disparition ») ; soit elle s'accole aux grosses boules (comportement d'« Accolement ») ;

2. **Taille de la boule curseur** : la boule curseur est parfois difficile à manipuler pour certains enfants. Ce comportement permet de la faire grossir, ce qui rend sa manipulation plus aisée ;
3. **Son de sortie du cadre** : ce comportement permet l'émission d'un signal sonore (traduisant l'éloignement du but à atteindre) lorsque l'enfant tente de sortir la boule curseur du cadre de travail ;
4. **Musique de réussite** : une fois activé, ce comportement permet l'émission d'un *jingle* lorsque la boule curseur rentre en contact avec l'une des grosses boules, symbolisant la réussite ;
5. **Couleur et nombre de boules** : ce comportement permet de choisir le nombre de boules à présenter à l'écran et leurs couleurs. Il influence beaucoup sur l'objectif de l'activité comme nous allons voir dans la section suivante ;
6. **Déplacement** : avec ce comportement, on peut rendre déplaçable les grosses boules. En effet, certains enfants ne comprennent pas toujours pourquoi c'est toujours la boule curseur qui devait se déplacer vers les grosses et pourquoi l'inverse est impossible ;
7. **Contour** : ce comportement permet de tracer un contour aux périphéries des grosses boules. Le contour permet de séparer l'objet de son environnement. Ce qui fait comprendre que les grosses boules ne font pas parti de l'environnement.

2.3.3 Quelques objectifs éducatifs et la configuration adéquate des objets

Les objectifs de ce jeu varient selon les différentes situations qui se présentent dans son déroulement. Il s'agit de l'utilisation de la boule curseur, le nombre et le comportement des grosses boules présentes à l'écran. Nous présentons dans cette section quelques objectifs éducatifs et la configuration comportementale adéquate que les objets doivent prendre pour assurer tels objectifs.

- Dans le cas de la présence uniquement de la boule curseur, son utilisation élémentaire permet à l'enfant d'établir une relation en première instance entre une action motrice directe (toucher l'écran) et l'effet produit (mouvement de l'objet) ;
- Peuvent être alors étudiés l'attention que l'enfant porte :
  - à la perception du curseur et à son déplacement ;
  - au rôle facilitateur ou inhibant des stimuli sonores et leur accès éventuel à la fonction de ce signal (c'est-à-dire qui communique une information partagée par l'enfant et son tuteur) et à la capacité de l'enfant à décontextualiser ce signal sonore (par exemple en l'utilisant dans d'autres lieux, mais avec la même signification) ;
- Dans le cas où une ou plusieurs grosses boules sont présentes à l'écran et où le curseur s'accole à elles, l'objectif est d'apprécier l'intention de l'enfant pour tenter un rapprochement entre la boule curseur qu'il dirige et celle qui reste fixe. Ce jeu

sera ensuite décontextualisé et assorti de commentaires qui reflèteront la proximité et l'éloignement relationnel ;
– Dans le cas où une ou plusieurs grosses boules sont présentes à l'écran et où le curseur disparaît à leur approche, l'objectif est d'analyser la capacité de l'enfant à se représenter les objets cachés et d'agir de manière pratique pour le faire réapparaître.

Chapitre 3

Observation et analyse de comportements

Ce chapitre concerne la problématique de l'analyse du comportement de l'utilisateur en interaction avec des applications interactives. Après une présentation générale de la problématique de l'exécution adaptative par observation et analyse de comportements, ce chapitre a pour objectif de répondre à la problématique de l'analyse de comportements. Dans la première section, nous définissons cette problématique. La deuxième section présente la notion du terme *comportement* qui peut être source de confusion. La troisième section relate les différentes approches de l'analyse de comportements, leurs principales caractéristiques et les problèmes de ces approches dans le cadre de notre étude. Nous abordons, par la suite, l'analyse de comportements par observation qui constitue notre première contribution dans cette thèse. Un accent particulier a été mis sur les fondements théoriques de notre approche ainsi que la formalisation adoptée.

3.1 Problématique de l'analyse de comportements

Notre première problématique consiste à analyser le comportement de l'utilisateur à partir des données hétérogènes provenant de différentes sources. Il s'agit des évènements générés par les actions de l'utilisateur. Ces évènements sont le résultat de l'analyse du flux vidéo (pour la direction du regard, mouvement de la tête, etc) ou à partir des actions effectuées sur les éléments de contrôles (clic de souris, sélection d'une entrée, bouton, etc). L'intérêt de cette étude, dans la problématique de l'exécution adaptative, réside dans l'identification de certains comportements (définis par l'expert du domaine) dans le but de contrôler l'exécution de l'application.

L'analyse de comportements est un domaine riche en expérimentations diverses et variées comme le montrent différentes exemples présentés dans le chapitre 1. Bien que le terme *comportement* ne soit pas de rigueur dans la majorité des cas (*scénario* dans [MLBT03], *plan* [Tes97], *action* [Int99], etc), le problème est bien le même. Ce problème consiste à instancier un ensemble de modèles représentant des comportements de l'utilisateur avec un ensemble d'éléments du flux.

En ce qui concerne le type de flux, deux catégories peuvent être trouvées. La première, constitue l'ensemble des approches ayant pour entrées un flux vidéo (pour le suivi de certains objets d'intérêts). La seconde, et celle que nous considérons ici, constitue l'ensemble des approches ayant pour entrée un flux d'évènements. Bien que les approches basées sur un flux vidéo sont de plus bas niveau, donc plus expressives, la majorité des approches basées sur un flux vidéo effectuent une conversion en flux d'évènements. Et c'est l'hypothèse que nous posons dans cette étude. En conséquence, notre problématique consiste à instancier des modèles, qu'il convient de déterminer, représentant des comportements à partir d'un flux d'évènements.

Dans la littérature, on trouve deux axes qui différencient l'ensemble des travaux menés en analyse de comportements :

- Le premier axe concerne le *type de représentation* utilisé pour décrire les comportements, c'est-à-dire le type de modèle. Ce dernier doit être capable de combiner et exprimer des contraintes sur un ensemble d'éléments de base. Ces derniers désignent des *formes* avec lesquelles la description du comportement est faite explicitement par un utilisateur expert, ou obtenue par apprentissage. Pour cela, le modèle doit représenter le comportement à identifier dans le cadre d'un formalisme donné. Le choix du formalisme contraint alors les possibilités de représentation et d'expressivité des formes ;
- La seconde différence entre l'ensemble des méthodes proposées dans la littérature réside dans les *techniques* établies pour *identifier des comportements*. Il s'agit de reconnaître les modèles instanciés représentant les comportements. De ce point de vue, la nature du type de représentation des modèles à reconnaître influence le choix des techniques d'identification.

Ces deux axes sont utilisés dans notre analyse des différentes approches existantes qui traitent la problématique de l'analyse de comportements présentés dans la section 3.3.

Avant cela, nous présentons, dans la prochaine section, la notion du terme comportement afin de lever toute ambiguïté.

3.2 Notion de *Comportement*

La description du terme *comportement* dans le langage quotidien est souvent teintée de subjectivité ce qui manifeste une difficulté de compréhension du terme en question. Pour bien cerner cette difficulté, la première étape de notre démarche consiste à définir c'est qu'un comportement. Ainsi, nous sommes certain de faire référence à la même notion. Plus la description est précise, plus l'accord entre différentes vues a des chances d'être entier.

Définition 1.
Comportement n.m **1.** Manière de se comporter, de se conduire ; ensemble de réactions d'un individu. **Comporter** v.pr. **1.** se conduire d'une certaine manière. *Se comporter en honnête homme.* **2.** Fonctionner, réagir d'une certaines façon, dans des conditions données. *Cette voiture se comporte bien dans les virages...* Petit Larousse, 1992.

Définition 2.
En psychologique, un comportement est défini comme étant un ensemble de *réactions*, *observable* objectivement, d'un individu qui agit en réponse à une *stimulation* venue de son milieu intérieur ou du milieu extérieur. Petit Larousse, 1992.

Définition 3.
En éthologique, un comportement est considéré comme une séquence motrice ordonnée susceptible de variations individuelles, effectuée en fonction du milieu intérieur et du contexte environnemental instantané du sujet. Grand Larousse Universel 1990-1992, Tome 4.

Du point de vue psychologique, la notion du *comportement* s'est d'abord confondue avec le « *behaviorisme* » (en englais *behaviorism*) de **J.B. Watson** et **H. Piéron** [DP91] qui ne considère que les comportements donnés en réponse à des évènements de l'environnement ou stimuli (dites S-R ou « stimulus-réponse »). Dans cette conception, on s'interdit à peu prés complètement de supposer l'existence d'évènements ou de variables qui soient intermédiaires entre le stimulus et la réponse, et qui ne puissent être réduits aux relations simples qu'on peut observer entre ceux-ci. La notion du comportement se limite ici aux activités directement *observables* de l'individu, ce qui exclut, du moins au premier abord, les états de conscience, les pensées, les sentiments, les représentations et les autres activités intérieures.

Le behaviorisme a beaucoup perdu de son crédit avec l'apparition du « *cognitivisme* »[10]. La conception stimulus-réponse a été remplacée par des conceptions dans lesquelles on s'efforce de se représenter sous forme de modèles, les variables internes (ce qui se passe à l'intérieur) des individus comme la faim, la fatigue, le stress, la souffrance physique qui peuvent avoir des répercussions importantes sur le comportement [Tha01]. Par exemple, une fatigue importante rendra une personne plus lente dans ses mouvements ; l'effet d'un gaz toxique l'endormira, etc. Les modèles qui représentent les variables internes sont soumises à une validation par le moyen de l'étude des comportements observés dans des conditions bien déterminées.

La fin du behaviorisme, l'avènement et le succès de la psychologie et des sciences cognitives, n'ont pas conduit à abandonner la notion de « comportement » mais à l'épurer comme le souligne [DP91]. En effet, la définition du comportement n'a pas changé, et les deux définitions se rejoignent, au moins, sur deux points :

- D'une part, il existe un lien fort entre le *comportement* et les *stimuli* (que soit des stimuli de l'environnement de l'individu ou suite d'un processus interne de l'individu) ;
- D'autre part, dans les deux courants, il s'agit d'un *observable* en réponse à une stimulation.

Le problème soulevé qui n'est pas évoqué dans les définitions précédentes par souci de généralité, concerne la caractérisation du comportement. Sur ce dernier point, plusieurs questions se présentent : quelles sont les unités pertinentes à observer pour déterminer un comportement ? où commence et où finit un comportement ? quel est le niveau d'observation et d'analyse où se placer ? On aura à faire un choix, pouvant aller de l'unité segmentaire de l'analyse mécanique du mouvement (chaque rotation repérable de la tête est, par exemple, enregistrée) à l'acte organisé par rapport à une finalité. Sur ce dernier point, nous pensons que les notions de *niveau d'analyse* et de *point de vue* sont importantes dans l'analyse de comportements.

Dans un contexte d'observation et d'analyse du comportement de l'utilisateur en interaction avec une application informatique, nous considérons que :

- Les stimuli évoqués dans les définitions précédentes correspondent aux objets de l'application informatique, présentant un intérêt particulier (par exemple les objets du jeu évoqués dans la formalisation des jeux que nous avons adopté - voir la page 18) ;
- L'identification du comportement de l'utilisateur est le résultat de l'analyse de ses actions, à différents niveaux (actions, évènements, comportement), inscrit dans un *processus d'interprétation* ;
- Le processus d'interprétation correspond à des instanciations de modèles par observation de différents éléments de base *pertinents* pour le comportement en question. Nous appelons ces éléments, "*formes*". Elles portent sur les expressions faciales de l'utilisateur et ses actions sur l'application.

[10] Ce courant s'institutionnalise lors de la fondation de *Center for Cognitive Studies* à l'Université de Princeton au début des années 1960.

À partir de cela, nous définissons un comportement comme étant un ensemble de réactions observables, à différents niveaux, en réponse à des stimulations des objets d'intérêt (appelés Stimuli) de l'application. Les réactions constitueront des *formes* calculables à partir du flux d'évènements générés par les actions de l'utilisateur.

3.3 Présentation de différentes approches d'analyse de comportements

Le comportement humain a été depuis longtemps un sujet d'analyse dans de nombreux domaines : philosophie, psychologie, sociologie, sciences cognitives, etc. Les outils informatiques sont de plus en plus intéressés par l'étude du comportement humain dans différents domaines tels que la simulation du comportement humain à travers des humanoïdes virtuels [Tha01], Web Usage Mining (WUM) qui consiste à analyser le comportement des utilisateurs d'un site Web, l'analyse du mouvement dansé [Che04], etc.

Plusieurs approches ont été utilisées pour l'analyse du comportement humain pendant les dix dernières années. Parmi celle-ci, deux approches principales ont émergées. Il s'agit des approches basées sur les graphes probabilistes et les approches mutli-agents. Dans cette section, nous présentons les principes de ces deux approches. Chaque approche est suivie d'un exemple d'architecture qui est analysé par rapport aux deux axes qui différencient la problématique de l'analyse de comportements, à savoir le type de représentation et la technique d'identification. Cette étude nous permet, dans une seconde étape, de déduire les inconvénients qui peuvent être rencontrés dans ces approches dans le cadre de l'analyse de comportements en interaction avec les applications interactives.

3.3.1 Approches des graphes probabilistes

3.3.1.1 Modèles de Markov Cachées

Les modèles de Markov cachés (HMMs : Hidden Markov Models) sont des outils statistiques pour le traitement des données séquentielles. Ils sont principalement utilisés dans l'identification des situations précises dans des successions de transitions. Ils ont été utilisés avec succès dans la reconnaissance de la parole [Rab89] et récemment dans l'interprétation du comportement humain [BOP97] [WB98].

Un Modèle de Markov Caché (MMC) est une sorte d'automate d'états finis probabiliste où les transitions entre les états représentent les causalités caractérisées par une distribution de probabilité, généralement apprises à partir d'un ensemble de cas déjà expérimentés dans la phase d'apprentissage. Un MMC est un processus markovien, par conséquent l'hypothèse de Markov, qui stipule que chaque état dépend seulement de l'état précédent est maintenue.

Plusieurs travaux ont exploité les modèles de Markov cachés pour la représentation et l'identification de comportement. Dans [BI98], les auteurs utilisent un ensemble de MMC pour l'identification des comportements simples tels que le mouvement d'une main dessinant un cercle. Ils ont développé aussi des MMC pour la reconnaissance des comportements complexes par une analyse syntaxique. Dans [BOP97], les auteurs utilisent une forme particulière de MMC (MMC couplés) pour identifier des comportements du gymnaste asiatique (*Tai'Chi)*. Un MMC couplé peut modéliser les interactions entre les processus tels que les mouvements des mains d'une personne. L'auteur de [Hoe01] utilise les MMC hiérarchiques afin d'identifier les expressions faciales caractérisant les émotions dans un processus d'interaction avec l'utilisateur.

3.3.1.2 Réseaux bayésiens

Les réseaux bayésiens sont des modèles graphiques probabilistes permettant de représenter les influences entre les évènements. Ils ont la capacité d'acquisition, de représentation et de manipulation des connaissances. Chaque nœud du graphe représente une variable aléatoire et les liens entre les nœuds représentent une causalité entre les différentes variables aléatoires ; les liens sont associés aux probabilités conditionnelles qui sont définies par apprentissage à partir d'un ensemble d'exemples.

Dans [BG95], les auteurs utilisent les réseaux bayésiens pour l'interprétation des flux vidéo dans un système de surveillance du trafic afin d'identifier des comportements particuliers tels que l'embouteillage. Les Réseaux bayésiens (RB) sont utilisés à deux niveaux différents : pour le calcul des caractéristiques simples mais incertaines d'un comportement donné et pour l'identification des comportements plus complexes.

Dans [HBN00], les auteurs utilisent un classifieur bayésien naïf pour identifier des comportements complexes dans un contexte de surveillance d'un parking de stationnement. Il s'agit d'identifier des comportements complexes à partir de plusieurs comportements simples caractérisés par des évènements. Par exemple, le comportement « ralentir à l'approche d'un objet » est reconnu à partir des évènements : « se diriger vers l'objet de référence », « ralentir à l'approche de l'objet de référence » et « la distance par rapport à l'objet de référence diminue » en utilisant des probabilités antérieures calculées durant la phase d'apprentissage du système.

3.3.1.3 Exemple d'architecture

Nous présentons dans cette section, un exemple d'architecture utilisant les réseaux bayésiens dans le processus d'identification de comportement. À travers cet exemple, nous présentons le modèle de représenation ainsi que le processus d'identification dans cette architecture. Il s'agit d'une architecture développée à l'INRIA Sophia-Antipolis pour l'identification des comportements violents faisant participer un groupe d'individus dans un contexte de surveillance d'une station de métro [CRTT97][MLBT03].

La figure 3.1 présente l'architecture générale du système proposée par les auteurs

[MLBT03]. Elle est composée, principalement, de deux modules :

- *module de vision* qui consiste à traiter le flux vidéo (segmentation, classification et suivi des objets en mouvement) et générer des évènements identifiant les caractéristiques visuelles des personnes ;
- *module d'interprétation* qui consiste à identifier les comportements prédéfinis dans les connaissances expertes stockées dans la base des connaissances.

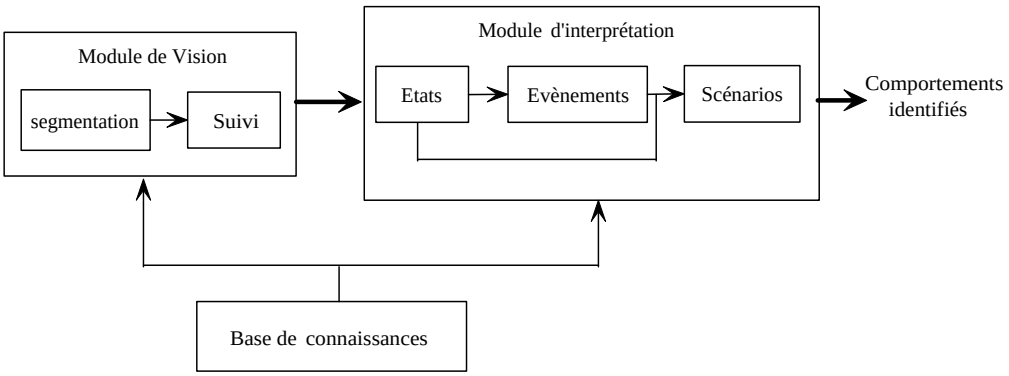

FIG. 3.1 – Architecture de système d'analyse de comportements utilisant les Réseaux Bayésiens[MLBT03]

Dans ce qui suit, nous présentons le modèle de représentation et d'identification de comportement utilisé dans cette architecture.

1. Représentation

Chaque comportement est défini d'une manière hiérarchique par trois entités :

- *Un état* : est une propriété caractérisant une personne dans un intervalle de temps ;
- *Un événement* : caractérise un changement d'état ;
- *Un scénario* : est une combinaison d'états, d'événements et/ou des sous-scénarios.

2. Identification

À partir de la représentation de la scène fournie par le module de vision et en se basant sur les connaissances expertes, le module d'interprétation utilise une approche bayésienne afin d'identifier d'une manière hiérarchique toutes les occurrences d'états, d'évènements et de scénarios, c'est-à-dire toutes les occurrences des comportements de l'utilisateur prédéfinis dans le module des connaissances.

L'approche utilise un formalisme basé sur un réseau bayésien récurrent (RBN : Recurrent Bayesian Network). Ce dernier est une forme particulière des Réseaux Bayésiens Dynamiques (Dynamic Bayesian Network) [DK89] destiné à l'identification des comportements. Chaque comportement est défini par les caractéristiques visuelles des personnes.

La structure d'un réseau bayésien récurrent (RBR) possède comme entrées les valeurs des caractéristiques visuelles exprimées dans un intervalle de temps limité.

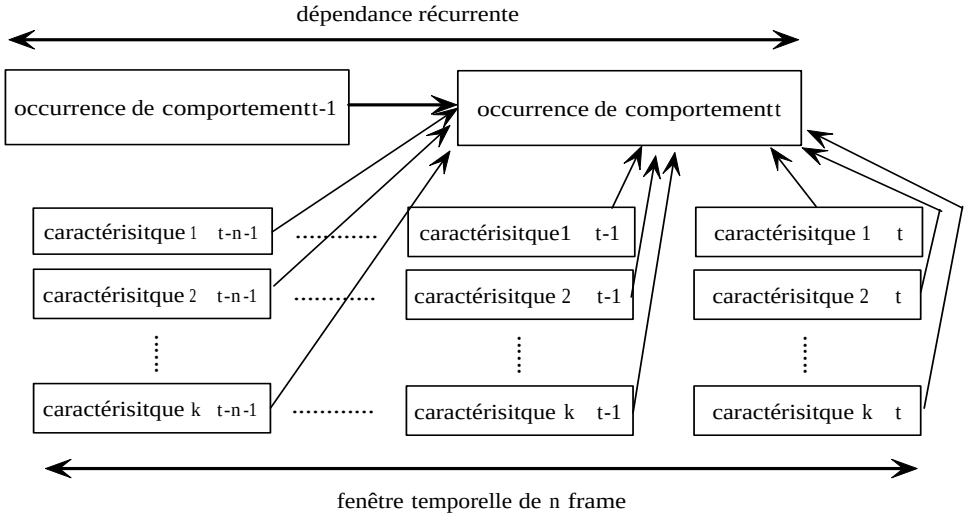

FIG. 3.2 – Structure du réseau bayésien récurrent pour l'analyse de comportements

La figure 3.2 montre la structure générale d'un RBR. La dépendance récurrente est chargée de diffuser l'information caractérisant les périodes précédentes (occurrences précédentes des comportements).

Formellement, soit un comportement B à reconnaître et O un ensemble de dépendances avec $O = \{R, C1_t, C1_{t-1}, ..., C1_{t-n}, C2_t, C2_{t-1}, ..., C2_{t-n,}, ..., Ck_t, Ck_{t-1}, ..., Ck_{t-n}\}$ où :

- R est la dépendance récurrente ;
- Ci_t est la dépendance de la caractéristique visuelle i à l'instant t.

Pour simplifier, on note les dépendances par $O = \{O_1, O_2..., O_{nk+1}\}$ avec n est la taille de la fenêtre temporelles et k le nombre de caractéristiques visuelles utilisées pour reconnaître l'occurrence du comportement.

L'idée consiste à comparer les probabilités conditionnelles $P(B|O)$ et $P(\overline{B}|O)$ afin de déterminer si le comportement B est reconnu ou non.

Selon le *théorème de Bayes*, si les dépendances O_i sont conditionnellement indépendantes à B (l'hypothèse de Bayes) on a :

$$P(B|O) = \frac{P(O_1, O_2, ..., O_{nk+1}|B) * P(B)}{P(O_1, O_2, ..., O_{nk+1})} = \frac{\Pi_i P(O_i|B) * P(B)}{P(O_1, O_2, ..., O_{nk+1})} \tag{3.1}$$

On a juste à comparer $P(B|O)$ avec $P(\overline{B}|O)$. La probabilité $P(O_1, O_2, ..., O_{nk+1})$ est constante et elle n'a aucune influence sur la comparaison. Afin de reconnaître B et $\overline{B}$ avec la même probabilité, on suppose que $P(B) = P(\overline{B})$, on a donc :

$$\frac{P(B|O)}{P(\overline{B}|O)} = \frac{\Pi_i P(O_i|B)}{\Pi_i P(O_i|\overline{B})} \tag{3.2}$$

Les probabilités $P(O_i|B)$ sont calculées durant le processus d'apprentissage du système à partir de l'ensemble $\{(b,o)_1, (b,o)_2, ..., (b,o)_m\}$ où chaque couple $(b,o)_i$ représente une probabilité définie manuellement par l'expert. Pour cela, l'expert doit caractériser par des valeurs : d'une part, le comportement B et d'autre part, toutes les dépendances O_i.

3.3.2 Approches multi-agents

L'approche multi-agents fait partie des méthodes utilisées pour l'analyse du comportement dans différents contextes applicatifs, en particulier dans les moteurs intelligents des robots [BS99] [KAU04]. Il s'agit, dans ce contexte, des agents qui communiquent entre eux afin d'identifier des situations particulières de l'environnement dans lequel ils évoluent, souvent complexe et dynamique, tels que chaque agent assure une tâche particulière. Plusieurs architectures utilisent le paradigme agent dans l'identification de certains comportements qui portent des intérêts particuliers. Par exemple, les auteurs dans [BS99] [Bre99] ont développé un robot, baptisé KISMET. Ce prototype voit et entend ce que fait la personne en face de lui et interprète ses paroles et ses gestes. Simultanément, il en déduit un comportement et répond par sa voix synthétique avec une tonalité spécifique, appuyée par des expressions faciales.

3.3.2.1 Exemple d'architecture

Dans cette section, nous présentons, une architecture distribuée développée dans un cadre d'interaction multimodale entre l'homme et le robot. L'architecture proposée dans [KAU04] [AKU03] consiste à percevoir une personne et à lui proposer des services en fonction du comportement qu'elle manifeste.

L'architecture du système, présentée dans la figure 3.3, est composée d'un ensemble de composants spécialisés dans différents domaines, appelés agents primitifs, et d'un serveur de connaissances, appelé gestionnaire de connaissances.

- *l'agent primitif* permet de gérer des entités logicielles qui accomplissent des tâches spécifiques, telles que la reconnaissance faciale, la reconnaissance de la parole, etc ;
- *le gestionnaire de connaissance* joue un rôle central dans le système. Il s'agit d'un composant, qui peut être considéré comme un agent primitif spécial, dont le rôle consiste à déclencher des actions spécifiques en fonction des informations reçues des agents primitifs.

La communication entre les différents composants s'effectue par envoi de messages (à la différence de la communication via une structure partagée - voir la section 5.5.3) basés sur la technologie XML-RPC [Use04]. Cet outil utilise le protocole RPC (Remote Procedure Calling protocol), qui permet l'appel des procédures à distance.

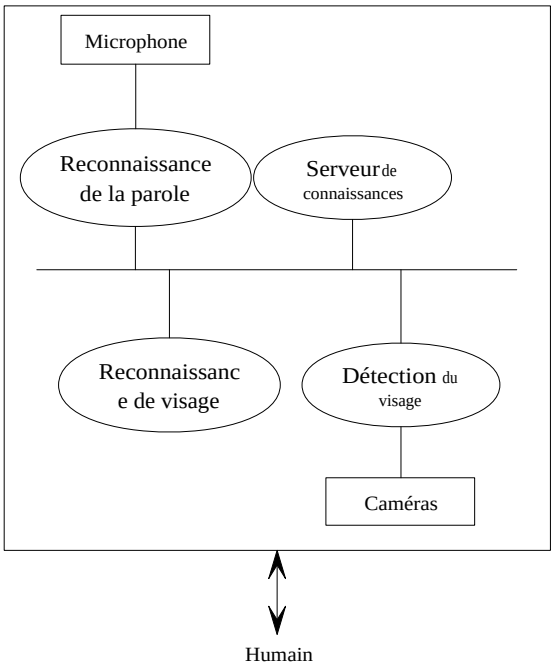

FIG. 3.3 – Architecture du système [KAU04]

1. Représentation

Le serveur de connaissance est basé sur la plate-forme SPAK (Software Platform for Agents and Knowledge)[AKU03]. Cette plate-forme utilise les *frames* [Min75] comme moyen de représentation des connaissances. Un frame est une structure de données permettant de représenter des situations stéréotypées de l'environnement. Ce dernier désigne des comportements particuliers, des personnes (par exemple, élève, professeur ou événement) en interaction avec le robot, dans des hiérarchies de classes et d'objets avec une possibilité d'appariement de la situation actuelle de l'environnement avec les frames.

2. Identification

Les agents primitifs déclenchent des évènements, envoyés au serveur de connaissance sous forme de messages, caractérisant l'état actuel de l'environnement. Les messages ont une structure de frames. Le comportement de ces derniers est défini par des contraintes sur les attributs caractérisant le comportement en question. Au cours de l'exécution du système, le serveur de connaissance génère certaines actions adéquates, à chaque fois qu'un frame est instancié avec les siens.

Par exemple, quand une situation particulière du visage d'une personne est détectée par l'agent primitif correspondant, un évènement est envoyé au serveur de connaissance sous forme d'un message dans une structure de frames. À la réception de cet évènement, l'instance du frame reconnaissant cet évènement est déclenchée.

3.4 Discussion

Le développement rapide de la reconnaissance automatique de la parole a beaucoup contribué au succès des modèles de Markov pour l'identification des comportements. Les perspectives scientifiques concernent donc la mise en forme des données, les variantes des algorithmes d'apprentissage qui assurent une meilleure discrimination entre des modèles concurrents, la construction de modèles combinant des données hétérogènes de natures différentes et l'amélioration de la robustesse. Néanmoins, ils présentent quelques faiblesses dans le cadre de la problématique de l'analyse de comportements telle que nous l'avons définie :

- D'une part, la pauvreté de la représentation à base d'automates dont la topologie est fixée a priori ;
- D'autre part, les modèles de Markov Cachées sont des processus markovien, par conséquent l'hypothèse de Markov qui stipule que chaque état dépend seulement de l'état précédent est maintenue. Ce constat peut poser des problèmes dans la mesure où la plupart des comportements humains ne sont pas des processus markoviens.

Les réseaux bayésiens constituent une extension majeure, car ils associent la théorie des probabilités à celles des graphes. L'avantage principal réside dans leur utilisation des connaissances antérieures, modélisant ainsi les causalités entre les caractéristiques des comportements. Ils fournissent également des outils intuitifs et naturels pour traiter des problèmes dans lesquelles l'incertitude et la complexité des données joue un rôle important. L'idée fondamentale des modèles graphiques est la modularité : un système complexe est construit en combinant des parties plus simples. Néanmoins, ils nécessitent une phase d'apprentissage qui est souvent longue. À titre d'exemple, dans l'architecture présentée dans la section 3.3.1.3, l'expert doit définir toutes les dépendances O_i.

L'avantage des approches basées sur le paradigme des systèmes multi-agents réside dans leurs capacités d'analyse de comportements dans un environnement dynamique complexe. Ceci est rendu possible grâce aux compétences des agents (autonomie, sociabilité, partage des connaissance, etc.) permettant de résoudre des problèmes complexes. Néanmoins, les approches actuelles dans le contexte de l'analyse de comportements manquent d'un formalisme générique permettant la définition des formes telles que nous l'avons présentée dans la définition du comportement (voir la section 3.2).

Lors de la conception du modèle de représentation et d'identification du comportement un certain nombre de choix et d'hypothèses de travail sont à définir.

Tout d'abord, le choix des comportements à identifier. Sur la base de notre définition du comportement, notre choix concerne les comportements de l'utilisateur en interaction avec une application informatique qui présente un certain nombre d'objets d'intérêt.

Ensuite, vient le choix du fondement théorique, c'est-à-dire sur quelle base, nous définissons les formes et leurs pertinences pour l'analyse du comportement. Il peut être d'origine psychologique, didactique, informatique, cognitive, etc. Et il conditionne les choix de représentation ultérieurs. En outre, selon qu'il est dépendant d'un domaine ou pas, son

application à la modélisation est plus ou moins directe.

Le choix du modèle dépend aussi du choix des *observables*, c'est-à-dire des éléments qui serviront à décrire le comportement pour le système. Ainsi, le choix du niveau d'analyse, ou encore d'échelle d'observation est crucial puisque tout modèle se construit à partir de cette base. Le résultat de l'analyse des comportements en dépend, puisqu'une caractéristique d'un comportement donné peut apparaître à un certain niveau mais pas à un autre.

À partir de cela, nous présentons notre approche de l'analyse de comportements du l'utilisateur basée sur l'observation.

3.5 Analyse de comportements par observation des actions

Dans cette section, nous allons d'abord, présenter le principe de notre approche pour l'analyse de comportements. Dans la deuxième partie nous présentons le fondement théorique basé sur l'observation, puis le formalisme que nous avons adopté, ainsi que le processus d'interprétation du comportement.

3.5.1 Principe de notre approche

La problématique consiste à identifier un ensemble de comportements prédéfinis par analyse des réactions de l'utilisateur face à des stimuli de l'application. Ces réactions peuvent être de nature visuelle (direction du regard par exemple) ou des actions sur l'application (clics sur un objet par exemple). Ainsi, il s'agit donc de qualifier des comportements à partir de données hétérogènes provenant de différentes sources (images ou autres). Dans cette optique, la nécessité d'aborder simultanément les évènements à différents niveaux s'avère nécessaire. Tout d'abord, l'acquisition des actions de l'utilisateur qui peuvent être *explicités* (provenant d'un flux vidéo) ou *implicites* (un clic de souris, sélection d'une entrée, bouton, etc.) sur les objets d'intérêt de l'application. Ces actions seront à l'origine des différents évènements. Ensuite, et à un niveau supérieur, être capable d'analyser ces évènements afin de reconnaître les *formes* qui constituent les éléments de base de l'analyse de comportements. Dans un troisième temps, il nous est nécessaire de donner un sens à ces formes dans le cadre d'un processus d'interprétation qui va permettre d'interpréter le comportement de l'utilisateur.

L'idée principale, que nous avons retenue, a été de considérer que, lors du processus d'interprétation, ce ne sont pas toujours les mêmes points de vue sur les formes qui sont utilisés [SCE06][SCE05a]. La sélection de tel ou tel point de vue, exprimé par un **observateur** donné pour un ensemble de formes, dépend du **contexte** dans lequel les formes se trouvent. Le contexte exprime le profil de l'utilisateur, la position de certaines formes par rapport aux autres, etc. Ainsi, chaque observateur est une instance d'une opération de transition du points de vue à partir d'un ensemble d'autres points de vue

préexistants. L'opération de transition permet d'expliquer le caractère compositionnel des différents niveaux d'analyse puisqu'elle permet de donner un point de vue nouveau à partir de plusieurs autres points de vue.

En résumé, la méthode que nous proposons pour la reconnaissance de comportement consiste à se doter d'un ensemble d'observateurs représentant des points de vue, à différents niveaux d'analyse, du comportement de l'utilisateur et d'un ensemble de formes représentant différents séquences d'évènements. Ainsi, on extrait du flux d'évènements les indices des différentes formes afin de calculer une description de celles-ci. À partir de cette description, on analyse la façon dont les réactions de l'utilisateur ont évolué afin d'obtenir une interprétation de son comportement. Dans la section suivante nous présentons les fondements théoriques de notre approche.

3.5.2 Fondements théoriques

3.5.2.1 La théorie des affordances

Le terme *Affordance*[11] constitue l'ensemble des aspects pertinents et significatifs de l'environnement d'un être vivant. La théorie des affordances est issue de l'approche écologique de la perception. Selon le fondateur de cette théorie, **James Jerome Gibson** [Gib79], les affordances sont des propriétés réelles des objets qui peuvent avoir une valeur utile pour leur observateur. Elles portent sur ce que l'on perçoit en fonction de ce sur quoi on peut agir. Par exemple, un bouton suggère que l'on peut appuyer dessus. Ainsi, la perception ne consiste pas à capter les dimensions et les propriétés absolues des objets de l'environnement, mais à capter les caractéristiques que ces objets *fournissent*, *exposent* aux *observateurs*. **Donald Norman**[Nor88] [Nor90] a fait l'interprétation suivante : « *la référence aux objets et concepts se fait tels qu'ils sont vus (perçus) par l'observateur* ».

Dans le contexte de l'analyse de comportement humain, l'approche écologique consiste à percevoir les formes personnels telles que les caractéristiques faciales, la voix, la façon de bouger, etc. Ces formes ont une fonction adaptative pour l'individu [BM86] [MZ98] ce qui constitue des affordances comportementales. Ainsi, percevoir adéquatement les formes faciales d'une personne permettrait à un individu d'avoir un comportement adapté et de savoir ce qu'il peut lui offrir et ce qu'il peut en attendre (*behavioral affordances*). Par exemple, le fait de détecter grâce aux formes faciales qu'un individu est un jeune enfant (grand front et petit menton, sourcils plus fins, yeux plus larges, cheveux plus clairs, lèvres plus larges et rouges, nez plus petit et éléments faciaux moins longs par rapport à l'adulte) entraînera chez un adulte des comportements de soin et de protection et inhibera les comportements violents [BM86] [MZ98].

[11] « *The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. The verb to afford is found in the dictionary, but the noun affordance is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment, etc.* » [Gib79], p.127

La théorie des affordances a été utilisée en informatique dans divers travaux, en particulier dans le contexte des *Agents Conversationnels* [SPS03] [CBVY00]. Par exemple, dans [SPS03], les auteurs ont proposés un langage de description d'agents, dédié à l'interaction en langue naturelle entre des utilisateurs humains et des composants logiciels de l'Internet. La théorie des affordances a été la base de la représentation du composant logiciel du point de vue supposé de l'utilisateur par les *Agents Dialogiques*[12]. Cette représentation permet à l'agent de répondre aux questions de l'utilisateur.

Dans la même perspective et dans un contexte de l'analyse de comportement humain, nous supposons que les « formes » perceptibles des expressions émotionnelles, direction du regard, gestes,... constituent des affordances spécifiquement comportementales. Ainsi, une forme est une partie d'une instance d'un modèle de comportement exprimée par un observateur, et potentiellement, une partie de plusieurs comportements, ce qui résume assez simplement la théorie des affordances. En effet, l'application de tel ou tel point de vue sur une forme donnée permet d'agir d'une manière appropriée et acceptable à la situation, satisfaisant ainsi le critère de l'exécution adaptative.

3.5.2.2 Les Frames

À partir des travaux de la psychologie cognitive sur l'organisation de la mémoire chez l'être humain, la notion de *schéma* a été proposée comme un modèle de représentation d'expériences passées mises à profit pour résoudre des problèmes nouveaux. Ce concept a été repris en informatique, et en particulier en Intelligence Artificielle, sous la notion de *frames*, proposés par **Marvin Minsky** [Min75] en tant qu'unité structurée de description. Dans son ouvrage "*Society of Mind*" [Min98], l'auteur part de l'idée que les humains disposent d'un certain nombre de structures pré-existantes représentant des modèles de situations qu'ils cherchent à adapter aux nouvelles situations. Il nomme ces structures *frames*. Ces derniers regroupent de façon structurée l'ensemble des connaissances relatives à un objet, un concept ou une situation typique, de sorte que chaque frame est composé d'un ensemble d'attributs (appelés *slots*) qui servent à le caractériser à travers les diverses notions relatives à la situation représentée.

La structure des frames a ensuite évolué avec les notions de *frames*, *attributs* et *facettes* (voir la figure 3.4-a) :

- *frame* correspond au nom de la frame et aux liens *sorte-de* qui défini un lien d'héritage et *est-un* qui défini une instanciation d'un frame ;
- *attributs* correspondent aux propriétés qui définissent la structure de la situation décrite par la frame ;
- *facettes* correspondent aux modalités descriptives ou comportementales d'un attribut. Par exemple, ils décrivent le type et le domaine de validité de l'attribut et la valeur par défaut que l'attribut peut prendre, etc.

Les frames étant par ces notions des entités génériques constituées d'attributs. Ces

[12]les Agents Dialogiques sont des composants logiciels dotés des capacités d'interaction avec les utilisateurs par la voie de la langue naturelle [SPS03]

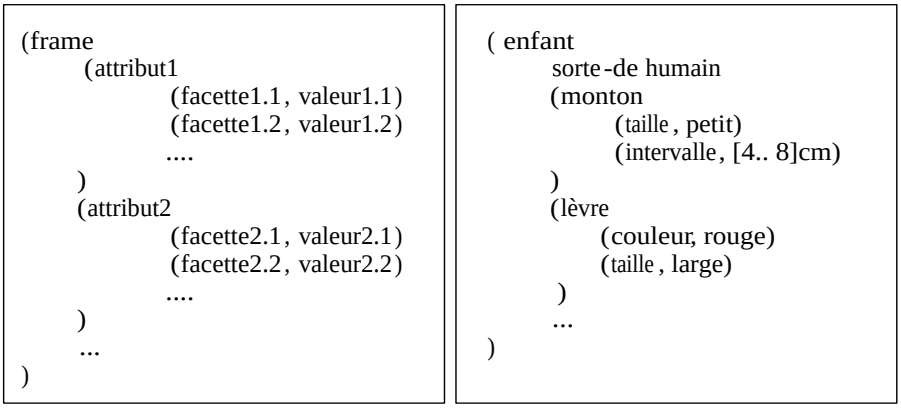

FIG. 3.4 – La structure des frames

derniers sont décrits par un ensemble de facettes possédant des valeurs. L'exemple de la figure 3.4-b montre un frame décrivant certaines expressions faciales caractérisant les enfants. Le frame en question contient deux attributs (menton et lèvre) ayant comme facettes (taille, intervalle) et (couleur, taille) avec les valeurs (petit, [4..8] cm), (rouge, large) respectivement.

De par leur définition, les frames sont proches des objets dans le contexte des langages de programmation orientée-objet. Toutefois, si des points communs existent entre les deux paradigmes comme, notamment, l'organisation des objets dans une hiérarchie de spécialisation où les objets plus génériques dominent les objets plus spécifiques, les différences fondamentales résident dans leurs finalités respectives :

- Les objets sont comme un langage de programmation, donc beaucoup plus rigide, alors que les frames implémentent un langage de représentation de connaissances ;
- Les langages de programmation orientée-objet sont destinés à écrire des programmes, les frames sont destinés à supporter des mécanismes d'inférences ;
- Les attributs du frame ont des facettes qui permettent de représenter différents *points de vue* sur un même attribut, ce qui n'est pas le cas des objets.

À partir de ces notions on peut en déduire les propriétés de base qui caractérisent les frames, elles concernent :

- le type du lien qu'ils considèrent, soit un seul lien *est-un*, soit deux liens (*sorte-de*, *est-un*) pour organiser et lier les frames ;
- l'héritage multiple ou simple qu'ils prennent en charge. L'héritage consiste à partager les propriétés entre frames reliés par les liens de type *sorte-de* ;
- le droit d'associer une valeur par défaut aux attributs. Il s'agit des valeurs qu'un attribut donné peut prendre selon qu'elles sont admises lors d'un manque d'information.

Une autre caractéristique importante est celle du type de frame, s'il s'agit d'un frame

générique (classe) ou de type classe/instance. Cette caractéristique est à l'origine de deux approches, à savoir *l'approche prototypique* et *l'approche Classe/Instance*.

1. Approche prototypique

Dans cette approche, il s'agit de représenter et mettre en facteur un seul type de frame qui contient les informations communes à toutes les situations qu'il représente et qui sont regroupées dans une catégorie. On utilise pour cela une structure particulière, que nous appelons*frame-prototype*, qui est attaché à chaque catégorie de situation. Une situation est décrite par la donnée du *frame-prototype* que l'on peut considérer comme la représentation moyenne de l'ensemble des situations qu'on veut représenter. Tout frame de cet ensemble est engendré à partir du frame-prototype ou d'un des sous-frames crées jusque là.

Les frames ont donc, dans cette approche, la capacité d'être recopiés afin de produire des copies modifiées appelées *sous-frames*. Par rapport à son sur-frame prototype, un frame hérite les informations qu'il ne redéfinit pas. Les caractéristiques nouvelles qu'il contient viennent enrichir la connaissance sur ce frame. Les informations qui viennent contredire ou masquer les informations portées par son sur-frame sont acceptées. Au moment de connaître toutes les informations disponibles sur ce frame, on hérite des connaissances contenues dans le sur-frame qui ne sont pas modifiées dans le frame.

2. Approche Classe/Instance

Dans cette approche, il s'agit de distinguer deux types de frames : les *classes* et les *instances*. Les classes sont les frames qui décrivent des catégories de situations. Les instances sont les frames qui représentent les individus appartenant aux situations décrites par les classes. Par opposition à l'approche prototypique où chaque frame peut servir de modèle pour une copie, les classes ont ici un rôle de frames génériques et générateurs alors que les instances sont des frames spécifiques non générateurs.

La figure 3.5 montre une représentation de l'approche classe/instance. Chaque classe peut hériter les propriétés des autres classes à travers le lien *sorte-de*. Par exemple, la classe $C1$ hérite toutes les propriétés des classes $C2$ et $C3$. $I1$ et $I2$ sont des instances de la classe $C1$ représentant deux individus appartenant aux situations décrites par la classe $C1$.

3. Discussion

L'approche prototypique présente des inconvénients vis-à-vis de la cohérence de la structuration de la connaissance, comme l'évoque **Ronald Brachman** dans [Bra85]. L'auteur souligne que les propriétés exprimées par les attributs d'un frame ne peuvent pas être considérées comme des conditions nécessaires d'appartenance à la situation qu'ils représentent, puisque ces propriétés peuvent être remises en cause par un sous-frame. Dès lors, les hiérarchies qui peuvent être établies ne sont pas fiables : les liens frame/sous-frame

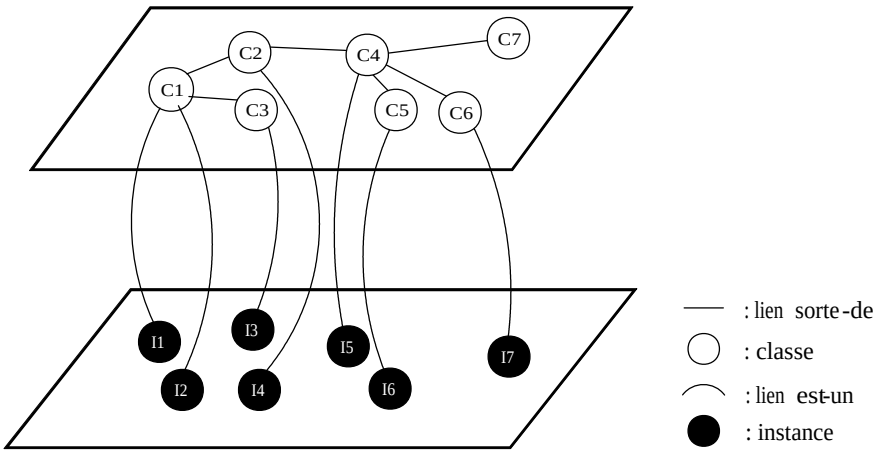

FIG. 3.5 – Approche Classe/Instance

ne peuvent refléter la réelle appartenance des sous-frames à la même famille d'individus représentée par le sur-frame, ni même l'inclusion structurelle d'un frame-concept dans ses sous-frames concepts.

Si l'approche prototypique se révèle utile dans la représentation des exceptions et la construction des hiérarchies non complètement définies, elle ne favorise pas la problématique de l'analyse de comportement. En particulier un aspect important de cette problématique est celui de la reconnaissance des instances de comportements. En plus, il nous semble difficile de représenter les comportements dans une structure hiérarchique qui peut être source d'incohérences sémantiques. En revanche, l'approche classe/instance, permet de part sa définition l'instanciation d'un certain nombre de comportement à partir de leurs classes.

À partir de cette analyse, nous avons retenu l'approche classe/instance, que nous considérons adéquate à la représentation et l'identification de comportement dont les classes représentent des catégories de comportements, et les instances représentent des comportements particuliers de ces catégories. Chaque classe peut hériter les propriétés d'une ou de plusieurs classes (héritage multiple). Par exemple, les propriétés qui décrivent le *regard* de l'utilisateur face à certains objets du jeu peuvent être utilisées pour décrire un comportement d'*évitement* (le fait que l'utilisateur ne se concentre pas sur l'activité[13] en cours d'exécution) et la *perception visuelle* (le fait que l'utilisateur suit ou non les objets en mouvement sur l'écran).

Dans cette optique, notre stratégie consiste à s'inspirer de la structure des frames pour décrire le comportement, où les attributs de la frame constituent les formes et la partie facettes-valeurs constituent les contraintes sur les formes.

[13]L'activité désigne ici, une instance d'un jeu (voir les définitions dans la page 18)

3.5.3 Genèse de l'analyse de comportements par observation

Rappelons que la problématique de l'analyse de comportements consiste à reconnaître des comportements représentés dans des modèles. Ces derniers dépendent de deux critères, à savoir le *type de représentation* utilisée pour décrire les comportements et la *technique d'identification* utilisée.

Notre approche considère l'analyse de comportements comme un processus d'interprétation des formes à différents niveaux d'analyse. Cette analyse est effectuée à partir des observations assurées par des éléments externes, appelés observateur, sur un environnement d'observation. Nous nous appuyons pour affirmer cela sur la théorie des d'affordances (voir la section 3.5.2.1) et la théorie de *la sémantique procédurale*. Cette dernière introduite par **Ludwig Wittgenstein** [Wit53], et reformulée par [Jl77] [Woo81] pour les applications informatiques, pose les bases d'une vision fonctionnelle de la signification. En effet, ses initiateurs proposent de considérer le sens d'un symbole et par extension, de toute observation, comme sa réalisation ou encore son application dans le monde réel par *l'action de l'observateur*. La sémantique d'une action n'a donc pas d'existence autre que celle de son expression dans *un environnement*.

Le *type de représentation* consiste à instancier les formes dans un formalisme permettant d'exprimer des contraintes dessus. La *technique d'identification* est un processus d'interprétation qui consiste à donner des points de vue sur les formes à différents niveaux d'analyse. Nous distingons trois niveaux, *évènements*, *formes* et *comportements* :

- *Niveau des évènements* : Les évènements sont générés à partir de l'acquisition des actions implicites et explicites de l'utilisateur sur les stimuli de jeu.
- *Niveau des formes* : Les *formes* constituent les éléments de base pertinents pour l'identification d'un comportement donné. Elles sont reconnues à partir de l'analyse des évènements du niveau 1.
- *Niveau des comportements* : Le troisième niveau permet de donner un sens aux *formes* du niveau 2 dans le cadre d'un processus d'interprétation qui va permettre d'identifier certains comportements de l'utilisateur.

Nous avons considéré pour cela un ensemble d'**observateurs** communicant entre eux via un **environnement d'observation** (voir la figure 3.6) :

- L'observateur constitue la connaissance de base permettant de donner une interprétation sur ce qu'il observe sur l'environnement d'observation. C'est à travers l'interaction entre les observateurs que les comportements peuvent être identifiés ;
- L'environnement d'observation constitue une structure partagée par les différents observateurs. Il a aussi une autre tâche, décrite dans la section 3.5.6, qui consiste à récupérer les actions de l'utilisateur à partir d'un flux d'évènements et de les représenter dans des formes.

Les observateurs ne sont pas considérés comme une partie d'un formalisme retraçant de manière déclarative l'association entre les formes et une sémantique comportementale, mais davantage comme des **règles de production** dont la partie condition serait les contraintes sur les formes, et la partie action serait des points de vue sur les formes.

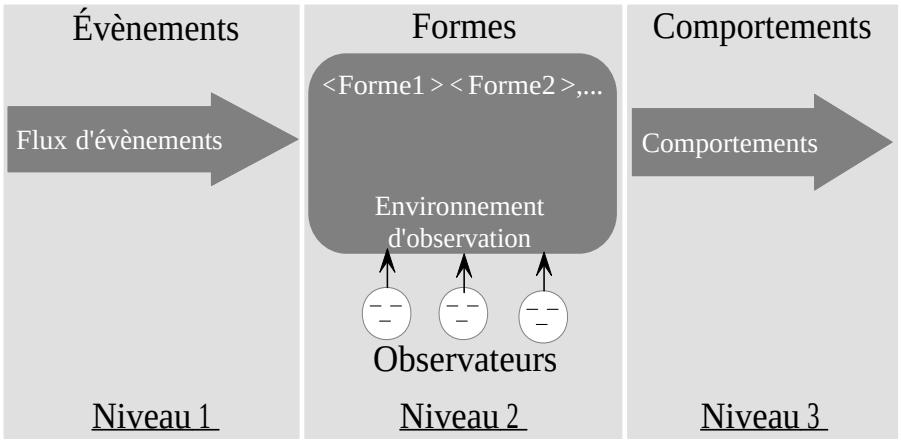

FIG. 3.6 – Principe d'analyse

Chaque observateur repose donc sur deux concepts, contexte et point de vue où :

1. le **contexte** permet de décrire les structures qui vont contenir les formes dans une logique proche de celle des frames. Il exprime les conditions de la règle de production ;

2. le **point de vue** permet de décrire l'action à prendre dans le processus d'interprétation du comportement. Il exprime l'action de la règle de production.

Le contexte est inspiré de la représentation des frames où les attributs de la frame constituent les formes et les facettes-valeurs constituent les contraintes sur les formes. Un observateur spécifie donc les relations entre les instances du contexte et les interprétations produites qui vont donner un point de vue sur les formes observables de manière à participer à l'analyse de comportements. Ces productions vont permettre d'effectuer des actions adaptables au niveau de l'application répondant ainsi au critère de l'exécution adaptative.

3.5.4 Formalisation

Les observateurs dont la structure est décrite dans la figure 3.7, définissent les relations liant les formes (représentées dans des instances de frames) observées, issues d'un flux d'évènements, et des points de vue sur le comportement de l'utilisateur. Chaque observateur contient deux composants : un composant **contexte** et un composant **point de vue**. Les deux composants peuvent être vus comme une description des formes, comme bases élémentaires qui décrit un comportement face à une situation donnée.

Dans ce qui suit, nous présentons les deux composants du modèle de l'observateur.

Contexte
< Sorte-de > : < Contexte>
< Formes > : (forme1, forme2, forme3)
<Contraintes contenu> :
forme1 ← valeur1
forme2 ↔ forme3

< Contraintes relations > :

< Contraintes profil > :

Point de vue
< Mode d'activation > : < réactif > / < proactif>
< Procédure d'activation > :

FIG. 3.7 – Définition des observateurs

3.5.4.1 Composant *Contexte*

Un contexte est défini par une étiquette qui sera utilisée dans le reste de la déclaration pour y référer (dans un héritage par exemple) et par un ensemble de caractéristiques :

Sorte-de : Ce champ désigne le ou les contextes desquels le contexte courant va hériter les formes et les contraintes. Le contexte a ainsi accès à tous les éléments définis dans ses contextes parents (son contexte parent direct, le contexte parent de celui-ci, et ainsi de suite). L'héritage ici ne concerne que les propriétés descriptives du contexte. Il faut noter que contrairement aux langages de programmation, aucun mécanisme ne permet de limiter l'accessibilité des éléments telles que les déclarations privées, protégées ou publiques dans les langages orientés objet.

Formes : Ce champ décrit les formes et liaisons qui définissent la description du comportement. C'est donc lui qui permet de donner le cadre du contenu informationnel des instances, du contexte et leur composition.

Contraintes : Chaque forme est définie par un type, qui peut être un contexte (dans le cas de description d'un comportement complexe) ou un type atomique (une forme élémentaire). Les formes peuvent être vues comme des réceptacles, leurs valeurs n'étant pas spécifiées explicitement, mais définies par des contraintes exprimées par des relations sur lesquelles elles s'appliquent. Nous distinguons trois types de relations (assignation, identification, prédicat) et trois types de contraintes (contenu, relation, profil).

Les relations :

- **relation d'assignation (←)** : Elle force une forme à être remplie par une certaine valeur. La partie gauche de l'assignation est forcément une forme, sa partie droite est forcément une valeur constante ;
- **relation d'identification (↔)** : Elle contraint ses deux parties à avoir le même contenu. Les deux parties sont soit des formes simples, soit des expressions mettant

en jeu une seule forme ;
- **relation de prédicats** : Outre les deux relations définies ci-dessus, le formalisme de définition des contraintes permet d'utiliser des contraintes prédicatives dont la sémantique est décrite de manière extérieure au modèle. On peut notamment définir ainsi des contraintes arithmétiques. Par exemple : *plus-petit-que(forme, valeur).*

Les contraintes :

- **contraintes contenu** : Elles définissent les relations qui doivent être vérifiées entre les différentes formes (y compris les formes héritées). Par exemple, *posture.tête à 45°* ;
- **contraintes relation** : Elles définissent les relations qui doivent être vérifiées entre les formes et les stimuli de l'application. Par exemple, *mouvement.boule* $\leftrightarrow$ *orientation.regard* ;
- **contraintes profil** : Elles définissent les caractéristiques du profil de l'utilisateur. Ce dernier est détaillé à la section 4.4.5.

3.5.4.2 Composant *Point de vue*

Le composant *point de vue* permet d'exprimer l'action de l'observateur. Chaque action est caractérisée par :

Un mode d'activation : Il permet de déterminer le comportement de l'observateur. L'utilisateur expert fait le choix entre les deux modes :

- *réactif* s'il réagit à la présence de certaines formes dans l'environnement d'observation ;
- *proactif* s'il réagit au fait que d'autres observateurs attendent des formes que l'observateur en question sait produire.

Une procédure d'action : La procédure d'action peut avoir trois actions différentes :

- Si l'observateur produit seulement une information pour d'autres observateurs, il génère d'autres formes par exemple : la forme *clic_sur_écran(50, 300)* qui désigne un clic sur la position de coordonnées *(50, 300)* deviendra *clic_sur_boule (couleur = '0000FF', position ='(60,310)', rayon = '20')* ;
- Si l'observateur doit agir sur l'agent de décision, il effectue une opération sur ce dernier, en lui envoyant son résultat d'interprétation du comportement de l'utilisateur. Par exemple, il envoie le message <*consigne, non-suivie*> qui veut dire que l'utilisateur n'a pas suivi la consigne souhaitée de l'activité. À la réception de ce message, l'agent de décision déclenche un nouvel épisode. Il s'agit d'un mécanisme permettant l'adaptation du scénario (voir le chapitre 4) ;
- Si l'observateur doit agir sur l'utilisateur, il produit un message pour celui-ci.

3.5.5 Processus d'interprétation

Le processus d'interprétation est vu comme une succession de transitions de points de vue sur les formes observées sur l'environnement, suivies de production qui consiste à reconnaître des comportements particuliers de l'utilisateur. Il s'agit des comportements qui ne sont pas cohérents par rapport à l'activité en cours.

Le processus d'interprétation est une boucle qui s'exécute sans fin, et à chaque tour, on suit le cheminement suivant :

1. Dans une première étape, les formes nouvellement entrées dans l'environnement d'observation sont étudiées par les observateurs. Les formes sont générées soit par l'action des observateurs, soit par les actions de l'utilisateur. Dans ce denier cas, Elles sont le résultat de l'analyse du flux d'évènements (voir la section 3.5.6) ;
2. En priorité, on tente de satisfaire les demandes des observateurs en attente de certaines formes dans l'environnement d'observation. Pour cela, on sélectionne les observateurs dont le composant contexte satisfait les formes présentes dans l'environnement et qui peuvent produire d'autres formes attendues par d'autres observateurs. Cette étape permet de classer les observateurs à activer plus tard, et surtout à autoriser l'activation des observateurs uniquement proactifs ;
3. Ensuite, à partir des formes nouvellement arrivées, on va chercher les observateurs réactifs et des observateurs proactifs rendus activables dans l'étape 2 ;
4. En exécutant l'action de l'observateur, on vérifie si les observateurs, ainsi recensés, deviennent ou non activables ;
5. Si leur contexte n'est que partiellement reconnu, alors l'algorithme met en mémoire les formes qui manquent pour vérifier le contexte, et elles sont recherchées en priorité lors de la prochaine itération ;
6. Si le contexte est totalement vérifié, la procédure d'action est déclenchée, provoquant éventuellement l'arrivée de nouvelles formes dans l'environnement ou produisant des messages envoyés à l'agent de décision. Ces messages contiennent des interprétations du comportement.

L'algorithme converge vers des interprétations locales, mais ne termine jamais, des interprétations faites par les observateurs pouvant toujours être utilisées par la suite.

3.5.6 Les formes

Rappelons que les formes sont générées à partir du flux d'évènements résultat des actions de l'utilisateur. Il s'agit de capter les interactions de l'utilisateur avec les objets d'intérêt de l'application. Les évènements peuvent être résultat d'une interaction directe (l'approche action) avec les objets d'intérêts tel qu'un clic sur l'objet « boule curseur » du jeu *Coucou caché* (voir la section 2.3) ou des interactions indirectes (l'approche vision) s'il s'agit, par exemple, de suivre le regard de l'utilisateur sur un objet particulier du jeu

telle une boule en mouvement sur l'écran afin d'analyser le niveau de perception du joueur (voir par exemple le jeu *Roule la boule* en annexe D).

Nous avons adopté une formalisation avec des balises de la représentation XML pour la représentation des formes. Chaque balise possède une identification unique et un contenu. Pour cela, nous avons choisi la représentation de balisage suivante :

$$<T\ id\ =\ '\%ID\%',\ attribut1\ =\ 'valeur1',\ \ldots\ ,attributN\ =\ 'valeurN'>\ldots</T\ id\ =\ '\%ID\%'>\ldots<T\ id\ =\ '\%ID\%'>\ldots</T\ id\ =\ '\%ID\%'>$$

Où

- T : est le type utilisé pour la reconnaissance ;
- *%ID%* : est une clef unique dans l'environnement d'observation permettant d'identifier la forme ;
- *Attribut, valeur* : sont des informations utilisables par les observateurs dans le processus d'interprétation présenté dans à la section 3.5.5. Il s'agit des attributs et des valeurs à comparer avec les contraintes du composant *contexte* de chaque observateur.

Les portions de l'environnement d'observation délimitées par une balise de type $<T$ *id='xxx' v1='val1', v2='val2'>...</T id='xxx'>* sont ainsi marquées comme pouvant être interprétées comme des entités T avec les attributs *v1* et *v2* ayant des valeurs *val1* et *val2* respectivement. On aura par exemple :

$$<clic\ id='1',\ xpos\ =\ '50'\ ,\ ypos\ =\ '300'>\ <clic\ id='1'>$$

qui représente un clic du joueur sur l'écran à l'endroit de coordonnées *(50,300)*.

3.6 Conclusion

Dans ce chapitre, nous avons défini la problématique de l'analyse du comportement, ainsi que les différentes notions proposées concernant le terme *comportement*. Notre définition de ce dernier a la spécificité de prendre en compte les objets d'intérêt de l'application avec laquelle l'utilisateur est en interaction. Après avoir présenté les approches proposées dans la littérature concernant l'analyse de comportements, nous avons conçu un modèle basée sur l'observation des actions de l'utilisateur afin de déterminer des situations particulières définies dans des observateurs.

Nous avons notamment introduit la notion des formes et la notion d'observation dans le processus d'interprétation. Ce dernier permet d'identifier des comportements particuliers, définis par l'expert, par analyse à différents niveaux et en appliquant différents points de vue selon le niveau de granularité considéré. Notre proposition repose sur un fondement théorique issu de la psychologie et qui peut être implémenté informatiquement et

interfacé via le formalisme des frames, avec les deux composants : contexte et point de vue. L'approche que nous avons développée dans ce chapitre sera utilisée par les agents observateurs afin de déterminer les comportements des enfants autistes qui portent un intérêt particulier.

Du point de vue de notre problématique générale, les comportements identifiés dans cette partie seront injectés dans l'agent de décision afin d'adapter le scénario. Le chapitre suivant présente, l'approche que nous avons adopté pour cela. L'idée étant de définir une stratégie sous forme de scénario d'activités permettant d'atteindre l'objectif souhaité. Durant l'exécution du scénario, à chaque fois qu'un comportement est identifié, l'agent de décision déclenche un nouvel épisode afin de mettre à jour le scénario.

Chapitre 4

Contrôle d'exécution adaptative

Ce chapitre concerne le contrôle d'exécution des activités en considérant le comportement de l'utilisateur. Il s'agit d'établir un modèle de scénarisation permettant de produire des activités adaptées à chaque utilisateur, et de suivre leur déroulement afin de répondre à la problématique de l'exécution adaptative telle qu'elle a été définie dans le chapitre 1. Après la définition de la problématique dans la première section, nous étudions différentes approches de contrôle. Il s'agit des systèmes à base de connaissance, systèmes à base de procédures, systèmes de classeurs et le storytelling interactif. Par la suite, nous présentons quelques problèmes pouvant être rencontrés dans ces approches dans le cadre de l'exécution adaptaitve afin d'aborder notre approche « contrôle d'exécution à partir de cas » basée sur le raisonnement à partir de cas.

4.1 Contrôle d'exécution des systèmes interactifs

Le contrôle d'exécution des systèmes interactifs, évoluant dans un environnement dynamique et complexe, est caractérisé par l'aptitude de ces systèmes à accomplir différentes tâches de façon robuste et efficace, dans des situations plus ou moins favorables. Dans le cas des systèmes en interaction avec l'utilisateur humain, ces situations dépendent des actions de l'utilisateur effectuées sur les activités proposées par le système. Autrement dit de son comportement tel que nous l'avons défini dans le chapitre 3 (voir la section 3.2). Pour cela, différentes capacités complémentaires doivent être mises en œuvre. Il s'agit de la boucle *perception-scénarisation-exécution*. Cette boucle peut se présenter de différentes manières en fonction du domaine d'application visé. Ainsi deux catégories se présentent :

- Dans les applications qui relèvent généralement du domaine industriel ou spatial, il est préférable de conserver un expert qui garde le contrôle d'exécution (la boucle *B* de la figure 4.1). Le rôle du système, dans ce cas, consiste à assister l'expert. Pour cela, le système dispose de capacités lui permettant d'atteindre un but, et propose à l'expert un certain nombre d'actions à exécuter. L'expert peut les valider, les amender ou même exiger une autre marche à suivre. La validation n'intervient alors que pour certaines étapes du contrôle ; les autres, plus routinières, sont exécutées automatiquement pour décharger l'expert ;
- D'autres applications, dites « *intelligentes* », visent à une autonomie totale du système et implémentent une boucle fermée (boucle *A* de la figure 4.1). Il s'agit des systèmes capables d'agir de façon rationnelle en fonction de leurs perceptions et leurs capacités d'interaction sans intervention explicite de l'expert.

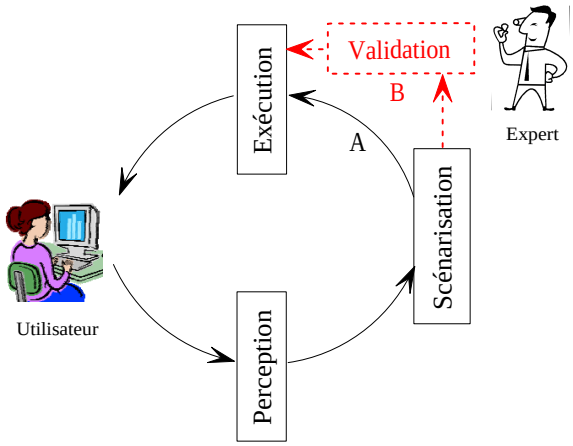

FIG. 4.1 – Boucle *perception-scénarisation-exécution*

L'exécution adaptative, telle que nous l'avons définie dans le chapitre 1, se place dans la deuxième catégorie dans la mesure où le système doit être capable de reconfigurer son comportement, au cours de l'exécution, sans même l'assistance explicite de l'expert mais en tenant compte de ses directives.

Notre problématique, dans ce chapitre, consiste à concevoir des mécanismes permettant au système de contrôler le déroulement du scénario de manière à qu'il soit adapté aux besoins (exprimés par des objectifs liés au domaine d'application), profil (qui présente une description qui caractérise l'utilisateur) et comportement de chaque utilisateur afin d'assurer une exécution adaptative par observation et analyse de comportements.

Les caractéristiques attendues dans le cadre cette problématique sont :

- *La réactivité* : à cause de l'aspect évolutif et imprévisible du comportement de l'utilisateur, le système doit pouvoir réagir aux changements brusques de celui-ci dans des délais adaptés à la dynamique de ces évolutions ;
- *Un comportement intelligent* : le comportement global du système doit être guidé par ses objectifs[14] d'où le besoin d'une capacité de scénarisation lui permettant de les réaliser ;
- *L'adaptabilité* : le système étant conçu pour assurer l'exécution adaptative des activités de façon autonome, il doit être capable d'affiner leurs descriptions de manière à prendre en compte les conditions du moment concernant le comportement de l'utilisateur, et orienter leur contrôle selon les circonstances définies par le profil de l'utilisateur.

L'obtention de ces caractéristiques dépend des propriétés des différents agents qui implémentent la boucle *perception-scénarisation-exécution* (format des données échangées, type de représentation des connaissances, mécanisme de décision, etc.) mais aussi de leurs interactions.

L'agent responsable de la scénarisation occupe un rôle particulier dans la mesure où il est le seul composant à pouvoir agir sur l'application. Il s'agit de l'agent de décision, de l'architecture du système présentée dans la section 2.2, qui consiste à recevoir en permanence les résultats de l'analyse de comportements et générer des réponses adéquates, en exécutant les actions prédéfinies dans le but de contrôler l'exécution de l'application. Pour cela, il dispose d'un ensemble de connaissances expertes liées au domaine d'application. La nature de ces connaissances varie en fonction du type du modèle utilisé pour les représenter et établir des méthodes de raisonnement. Parmi ces modèles, on trouve : *les systèmes à base de règles* [TBH86], *les automates* [BFKSD97], *les règles de contrôle de situation* (SCR : Situation Control Rules) [LH95], *les systèmes de classeurs* [HR78] [Gér02], *les systèmes à base de procédures* [IG92] [EF96], etc. Les principales approches et leurs caractéristiques sont présentées dans ce qui suit.

4.2 Différents types de modèles de contrôle

Cette section, présente une typologie des principaux modèles rencontrés dans la littérature concernant le contrôle d'exécution. Il s'agit des systèmes à base de connaissances, systèmes à base de procédures, systèmes de classeurs et le storytelling interactif.

[14]Par exemple, les objectifs éducatifs des jeux destinés à des enfants autistes, voir la section 2.3 ou le tableau 5.1

4.2.1 Systèmes à base de connaissances

Les systèmes à base de connaissances [TBH86] permettent de résoudre, sur un domaine en général limité, des problèmes sur la base d'inscriptions de connaissances, conformément codées, pour être l'origine de calculs bien définis aux résultats les plus contrôlés possibles. Le principe de ces systèmes consiste à produire des scénarios d'activités à partir d'un ensemble de règles de contrôle définis par l'expert. Une règle de contrôle établit la correspondance entre un état de l'environnement (le déclencheur de la règle) et une action à exécuter pour cet état.

Parmi les outils basés sur ce type de représentation on trouve les systèmes experts temps réel [RRAM95]. Leur principal avantage est la facilité de mise en œuvre qui ne nécessite que peu de connaissances. En revanche, aucune information ne permet de prévoir l'état de l'environnement après l'exécution d'une séquence d'actions, ce qui rend le contrôle difficile. De plus, l'ensemble des règles de contrôle étant statique ce qui rend l'adaptabilité difficile.

Dans [LH92] [LH95], les auteurs présentent une architecture composée de deux éléments en interaction : un *reactor* et un *planificateur*. Le premier, en prise directe avec l'environnement, réagit en fonction de son état courant à l'aide de règles de contrôle appelées réactions (composé d'une condition de déclenchement, et d'une action à réaliser par un effecteur). Le deuxième, raisonne à partir d'un modèle de l'environnement et des effecteurs, d'une description du reactor, et d'un but à atteindre.

Le rôle du planificateur est de modifier en continu le *reactor* (par ajout ou suppression de réactions) afin que son comportement permette d'atteindre le but. En retour, il reçoit de ce dernier des données perceptuelles sur l'état de l'environnement.

Dans [MDS95], les auteurs adoptent le même découplage des activités délibératives (sans temps de réponse borné) et purement réactives (soumises à des contraintes temps-réel). Ainsi, l'AIS (AI Subsystem) construit dynamiquement des scénarios de contrôle, constitués d'un ensemble de TAP (Test Action Pairs) qui associent à un état :

- une action ;
- des ressources nécessaires à cette action ;
- les durées de test et d'exécution dans le pire cas, et
- la fréquence maximale d'appel de cette action.

L'AIS exploite pour cela un modèle de l'environnement sous la forme d'un graphe d'états, dont les transitions sont des évènementielles, liées à une action ou temporelles. Ces scénarios de contrôle sont ensuite exécutés par le module RTS (Real Time Subsystem).

L'exécution adaptative par utilisation des systèmes à base de connaissances montre plusieurs limites vu la nature de la représentation des connaissances utilisées dans ces systèmes et le caractère dynamique des situations de l'exécution adaptative. Dans [WM94], les auteurs présentent quelques limites de cette approche :

- la mise en œuvre de la connaissance dans un ensemble de règles est une tâche longue et coûteuse ;

– la maintenance les connaissances est une tâche difficile dans la mesure où le modèle à base de connaissances gère un grand volume d'informations ;
– l'adaptation de la stratégie de scénarisation à des nouveaux domaines et aux situations imprévues à l'avance paraît difficile.

Beaucoup d'efforts sont utilisés pour introduire des possibilités formelles de prise en compte du caractère dynamique des situations, par l'adjonction des possibilités d'apprentissage à ces systèmes. Bien qu'elle semble être un secteur prometteur, seulement quelques améliorations individuelles ont été faites (voir par exemple [Dil89][Gut92]).

4.2.2 Systèmes à base de procédures

Les systèmes à base de procédures offrent un certain nombre d'avantages par rapport aux règles de contrôle. Alors que ces dernières sont essentiellement guidées par l'état du système, les systèmes à base de procédure utilisent des procédures permettent de répondre à des buts venant de l'utilisateur ou du scénario en cours d'exécution de manière à les rendre plus proches des connaissances expertes.

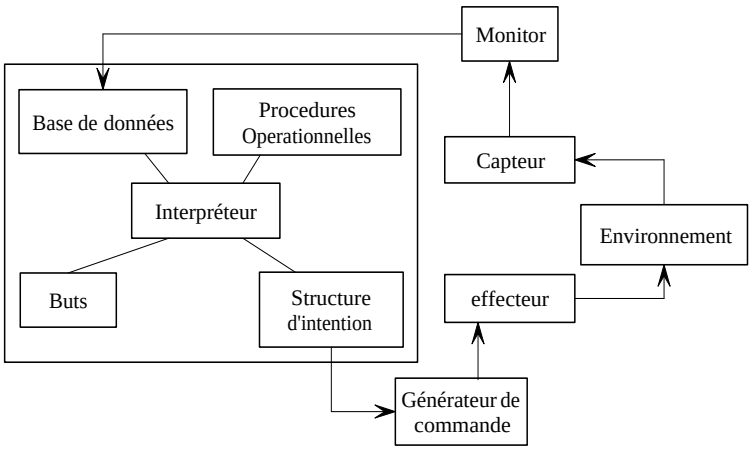

FIG. 4.2 – Structure du système PRS [GI89]

Le système de raisonnement procédural (Procedural Reasoning System - PRS) [GI89] [IG90] [IG92], présenté dans la figue 4.2, utilise des procédures, appelées OP (*Operational Procedures*, également désignées par *Knowledge Areas* dans les premières versions de PRS), déclenchées par un but, ou en réaction aux évènements provenant de l'environnement. Chaque procédure dispose d'un contexte d'application qui permet, lorsque plusieurs procédures permettent d'atteindre le même but, de sélectionner les plus adéquates. Enfin, un champ des OP précise la façon de l'exécuter : certains se décomposent en sous-buts, et d'autres qui constituent les éléments terminaux de cette décomposition, sont directement associés à une action primitive.

Des procédures particulières, appelées méta-procédures ou meta-level OP, peuvent également être utilisées pour aider le contrôle dans le choix d'un OP à exécuter, lorsque

plusieurs procédures sont disponibles pour un même but ; elles jouent alors le rôle d'une heuristique du domaine.

Le système RAP (Reactive Actions Package) [Fir94] propose une approche très semblable à celle du PRS ; les *Reactive Actions Package* sont un ensemble de méthodes procédurales permettant d'atteindre le même but, selon des contextes variables. La plupart des champs décrivant les OP se retrouvent dans les RAP. Ces derniers comportent toutefois quelques informations supplémentaires, telles que la durée d'exécution estimée ou les ressources consommées [Fir95]. Néanmoins, on ne retrouve pas d'équivalent dans RAP pour les meta-level OP.

4.2.3 Systèmes de classeurs

Les systèmes de classeurs sont des systèmes à base de règles qui exploitent des régularités de la dynamique de l'environnement. L'idée originale de ces systèmes revient à **John H. Holland** [Hol76] [HR78]. Le principe consiste à caractériser les états par plusieurs attributs représentant autant de propriétés perceptibles de l'environnement. Un état est un vecteur de plusieurs valeurs discrètes, une pour chacune des caractéristiques perçue de l'environnement.

Un système de classeurs est caractérisé par un ensemble de règles de décision appelées *classeurs*. Chacun de ces classeurs est caractérisé par une partie *Condition*, une partie *Action* et une *Force*.

- La partie *Condition* spécifie dans quelles situations le classeur est applicable. Il s'agit d'un vecteur d'attributs. Chaque attribut peut prendre les valeurs discrètes 0, 1 ou une valeur générale #. Un vecteur d'attributs est apparié avec un autre lorsque leurs attributs de même rang sont appariés deux à deux. La table de vérité de l'opérateur d'appariement pour les attributs est donnée dans le tableau 4.1 ;
- La partie *Action* spécifie la réponse du système à ces situations. Ainsi, le classeur « [11100111] [01] » propose l'action « [01] » dans le contexte défini par la condition « [11100111] » ;
- Si, dans une certaine situation, plusieurs classeurs proposent chacun une *action*, leurs *Forces* respectives sont utilisées pour sélectionner le classeur dont l'action est exécutée dans l'environnement.

Un attribut de valeur # est apparié avec n'importe quelle valeur. Dans la partie *Condition*, il opère donc comme un *joker* permettant d'ignorer la valeur d'un attribut. Par contre, un attribut avec une valeur particulière (0 ou 1) spécifie une restriction pour l'ensemble des vecteurs qui sont appariés avec la condition.

L'architecture générale du système de classeurs est présentée dans la figure 4.3. Elle se composent d'une interface d'entrée, une interface de sortie, une liste de classeurs et une liste de messages.

L'interface d'entrée exprime les perceptions du système en messages d'entrées qui peuvent être appariés avec les conditions des classeurs. L'interface de sortie traduit les

x	y	appariement
0	0	V
1	0	F
#	0	V
0	1	F
1	1	V
#	1	V
0	#	V
1	#	V
#	#	V

TAB. 4.1 – Table de vérité de l'appariement dans les systèmes de classeurs

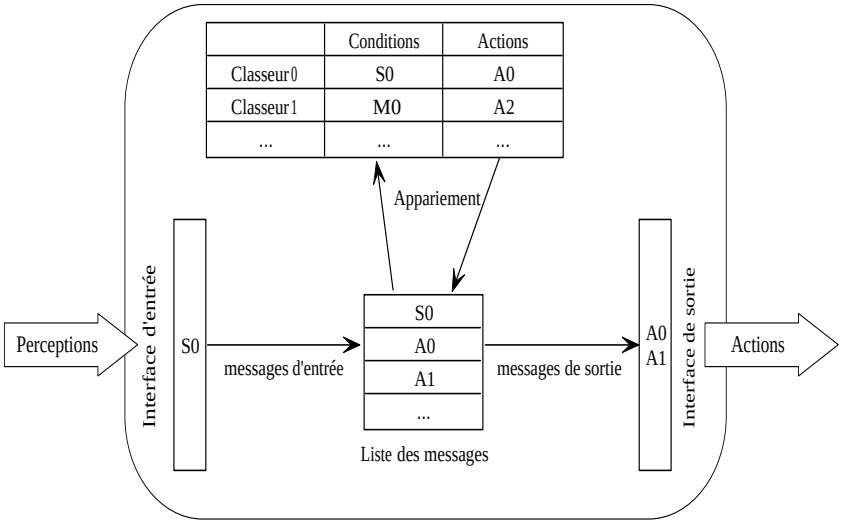

FIG. 4.3 – Architecture d'un système de classeurs [Gér02]

messages de sortie en actions effectives de système sur l'environnement. Ainsi, au cours de l'exécution, le système reçoit un message correspondant à une situation perçue, et décide quel message renvoyer et donc quelle action entreprendre. Pour cela, les messages d'entrée sont envoyés sur la liste des messages du système de classeurs et appariés par la suite avec les conditions des classes caractérisant le système. Si la condition d'un classeur est appariée avec un message de la liste, le classeur est activé. Les messages ayant provoqué une telle activation disparaissent alors de la liste. Chaque classeur activé envoie ensuite le message correspondant à sa partie *Action* sur la liste des messages. À niveau, certains messages ont été supprimés de la liste, d'autres y ont été ajoutés. Les messages qui peuvent être interprétés par l'interface de sortie sont traduits en actions effectives sur l'environnement et supprimés de la liste.

4.2.4 Storytelling interactif

Le storytelling interactif constituent actuellement un courant de recherche actif, en particulier dans le secteur des applications interactives qui inclut l'utilisateur humain. La recherche courante sur ce paradigme suit une grande diversité d'approches qui recouvrent : le storytelling immersif [SHG+01] [NT99], le storytelling émergent [Ayl99] [Dau98], le raisonnement à partir de l'expérience tracée [SHM05], etc. Ces approches correspondent à différentes solutions techniques liées au problème de production d'activités interactives (appelées récits interactifs) en particulier dans le cadre des jeux vidéo. Il s'agit de développer des méthodes de collecte, de gestion et de restitution des histoires comme vecteurs de connaissance puissants pour ceux qui les content, les écoutent, les réutilisent. Une des propriétés remarquables est que le récit retrace une séquence d'événements considérés comme pertinents pour rendre compte d'un processus sous-jacent.

L'exploitation des histoires nécessite une description de leur contenu. Cette description de contenus peut passer par une décomposition en « storiettes » par exemple, devant elles-mêmes être décrites d'une manière ou d'une autre pour en faciliter la recherche, l'exploitation, le partage, etc. Dans cette perspective et dans un contexte de jeux vidéo, les auteurs [CCM02] représentent les comportements des personnages en termes de rôle. Il s'agit d'une représentation narrative des buts et des actions des personnages qui permettent de les réaliser. Cette représentation est dans un Réseau Hiérarchique de Tâche (RHT) (Hierarchical Task Networks - HTN). Un RHT correspond à plusieurs décompositions possibles de la tâche principale en sous tâches. Ce qui peut être vu comme une représentation implicite pour l'ensemble des solutions possibles [ENS95]. Pour la production des activités interactives, les auteurs ont proposé une technique de planification qui est la base du comportement des personnages.

L'exécution adaptative du comportement des personnages est nécessaire dans la mesure où l'environnement dans lequel ils évoluent est de nature dynamique. Ceci est dû aux évolutions qui peuvent changer constamment l'environnement suite aux actions des autres personnages ou en raison des interventions de l'utilisateur. Le remède traditionnel consiste à assembler planification et exécution, de sorte que les décisions prises soient constamment adaptées à la situation actuelle de l'environnement. En outre, les mesures prise par un personnage peuvent échouer en raison des facteurs externes, ce qui nécessite que le comportement des personnages doit incorporer des capacités de replanification.

4.3 Discussion

Les différents types de modèles existants, présentées dans la section précédente, nous permettent de disposer d'une base de réflexion pour l'élaboration d'un modèle de contrôle d'exécution des systèmes interactifs proposant des activités informatiques. L'intérêt de ces approches réside dans leurs capacités formelles de vérification des propriétés concernant le comportement du système (comportements réactifs). Néanmoins, il est difficile de parler d'une exécution adaptative par observation et analyse du comportement de l'utilisateur,

pour ce type d'approches. En effet, d'une part, la description des systèmes de contrôle à l'aide de règles, procédures ou classeurs utilisent un ensemble de tâches élémentaires, auxquelles sont associées des événements, de précondition, de postcondition ce qui rend difficile de prévoir l'état du système après exécution d'une séquence d'actions. D'autre part, ces approches ne nous semblent pas appropriées pour l'intégration de fonctions de scénarisation de la boucle *perception-scénarisation-exécution* (voir la figure 4.1). La description utilisée dans ces approches met essentiellement en avant la réactivité du système, et paraît donc mal adaptée pour envisager une exécution adaptative avec un ***Contrôle délibératif***.

Le courant de recherche du Storytelling interactif est particulièrement intéressant pour répondre à la problématique du contrôle d'exécution par l'exploitation des histoires comme vecteurs de connaissances. Il s'agit de générer des scénarios interactifs en fonction de l'état de l'environnement. Néanmoins, ce nouveau courant de recherche ne met pas l'accent sur le comportement de l'utilisateur en interaction avec les objets d'intérêt de l'application.

L'étude que nous avons faite sur l'état des différentes approches de contrôle d'exécution, nous amène à retenir un certain nombre de principes pour l'élaboration d'un *modèle de contrôle à partir de cas*. Par opposition aux approches précédentes, nous désignons par ces termes les architectures de contrôle qui utilisent des techniques permettant de raisonner, à partir d'une modélisation de l'utilisateur et des moyens d'actions, sur la mise en œuvre d'un modèle permettant l'exécution adaptative. Cela nous amène à retenir un certain nombre de fonctionnalités pour l'élaboration de ce modèle :

- *Représentation de l'utilisateur* : Il s'agit d'établir un modèle permettant de définir :
  - le profil de chaque utilisateur et les éléments qu'il contient ;
  - les besoins de l'utilisateur exprimés sous forme d'objectifs ;
  - ainsi qu'une capacité du modèle à tenir compte des comportements de l'utilisateur en interaction avec les activités en exécution.
- *Représentation des connaissances* : il s'agit d'établir un modèle permettant d'organiser les connaissances de l'expert du domaine de manière à les réutiliser dans un processus de raisonnement. Le modèle doit être capable d'adapter ces connaissances à des situations similaires qui peuvent apparaître dans l'interaction ;
- *Processus de raisonnement* : C'est la fonctionnalité la plus importante du système. Il s'agit de définir des mécanismes intelligents permettant de générer des scénarios d'activités adaptées à la situation définie par la représentation de l'utilisateur (son profil, ses besoins et/ou son comportement). Pour cela, cette fonctionnalité doit établir un lien entre la représentation de l'utilisateur et la représentation des connaissances dans une architecture délibérative.

Dans la section suivante, nous présentons notre approche de contrôle d'exécution des activités.

4.4 Contrôle d'exécution à partir de cas

Dans cette section, nous allons d'abord, présenter le fondement théorique de notre approche basé sur le raisonnement à partir de cas. Dans la deuxième partie nous présentons le principe de notre approche pour le contrôle d'exécution [SE05b]. Par la suite, les modèles de représentation des connaissances et du profil de l'utilisateur sont détaillés. La dernière partie est consacrée au processus de décision.

4.4.1 Fondement théorique : *Raisonnement à partir de cas*

Le raisonnement à partir de cas est un paradigme de résolution de problèmes s'appuyant sur la réutilisation d'expériences passées, appelées *cas source* et stockées dans une *base de cas*, pour résoudre de nouveaux problèmes appelés *cas cibles*. L'ambition de ce paradigme, s'inspirant des principes issus de travaux en psychologie [Sch82], consiste à diminuer l'effort lié à l'acquisition des connaissances, pour les réduire toutefois à une méthode efficace de résolution des problèmes. Le calcul dans ce paradigme est moins grand que dans les approches présentées plus haut puisqu'il s'agit essentiellement d'un raisonnement analogique : un épisode de résolution de problème est constitué d'une partie ***problème*** et d'une partie ***solution***.

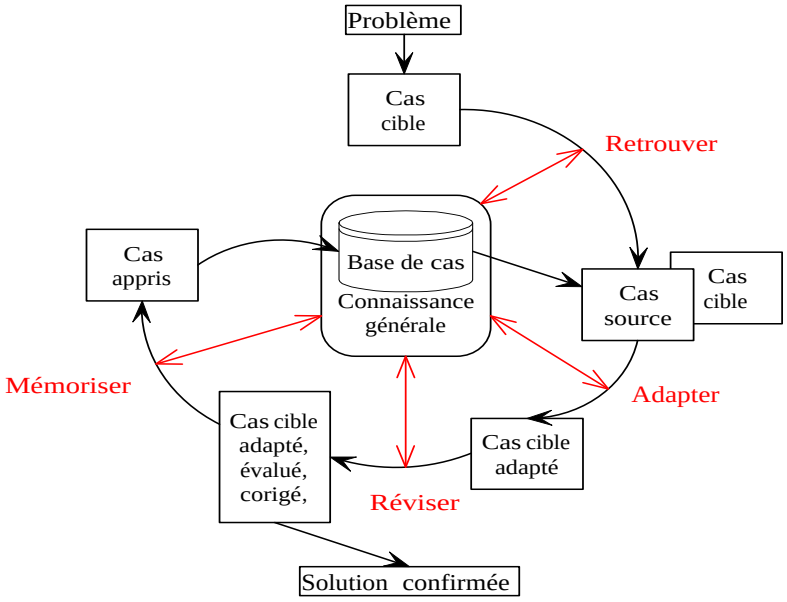

FIG. 4.4 – Processus du raisonnement à partir de cas

Le principe fondamental de cette méthode, présentée dans le schéma de la figure 4.4, consiste à chercher dans la base de cas des cas similaires au cas cible et d'adapter la solution des cas sources à la nouvelle situation du cas cible. Toute nouvelle expérience peut être mémorisée dans la base de cas la rendant immédiatement disponible pour les problèmes futurs.

Le raisonnement à partir de cas (RàPC) a surmonté les limitations des approches basées sur une base de règles comme le souligne [WM94]. De fait, de nombreux systèmes utilisent le RàPC dans divers domaines d'applications telles que l'accidentologie et le droit médical [Des02] [DC]. Dans [DC], les auteurs présente une architecture basée sur le raisonnement à partir de cas qui à partir d'un groupe d'accidents de la route survenu dans un secteur géographique particulier permettra d'y associer un profil de scénario d'accidents et qui fournira un ensemble d'objectifs et de principes d'actions à entreprendre pour aménager le secteur étudié. D'autres travaux, utilisent le raisonnement à partir de cas dans divers domaines : la supervision industrielle [Fuc97], l'aide à la navigation dans l'hypermédia [TJK99], etc.

4.4.2 Principe de notre approche

La majorité des travaux qui utilise le RàPC mettent en avant l'aspect de personnalisation des activités proposées par le système par rapport au profil de l'utilisateur, mais sans tenir compte des aspects liés à l'exécution adaptative par observation et analyse du comportement de l'utilisateur (voir par exemple les travaux [EFnC00] [FC02] dans le contexte de logiciels éducatifs). Pourtant, dans de nombreuses applications, en particulier les logiciels éducatifs et thérapeutiques pour enfants, ces aspects sont importants pour plusieurs raisons [SE05b] :

- Les utilisateurs évoluent et les activités du scénario proposées par le système peuvent devenir incohérentes au cours de la session ce qui motive le choix d'un contrôle dynamique ;
- L'analyse comportementale est un facteur clé pour les thérapeutes. Son intérêt réside dans la mise en évidence d'indicateurs précoces qui peuvent alerter sur le risque de décompensations, de passage à l'acte ou même d'actions parasites (telles que des stéréotypes) ;
- Cette analyse permet d'améliorer l'adéquation entre les actions du système et les comportements des utilisateurs dans le but de les maintenir attentifs et donc réceptifs à l'outil informatique.

Pour cela, notre stratégie de contrôle consiste à :

- construire une séquence d'activités répondant aux buts éducatifs que l'utilisateur veut/doit atteindre en tenant compte de son profil ;
- observer en permanence, durant l'interaction, les actions de l'utilisateur ;
- détecter les cas où les activités proposées par le système ne répondent plus à l'évolution actuelle du comportement de l'utilisateur, et déclencher un nouvel épisode de raisonnement dans ces cas, afin de mettre à jour le scénario.

4.4.3 Représentation des cas

En se basant sur le RàPC dans le processus de décision, les connaissances sur la façon d'accomplir les buts sont représentées dans le système par un ensemble de cas sources.

Chaque cas est défini par deux composants :

```
 Problème
[un ensemble de descripteurs]
 Solution
[une séquence d'activité]
```

1. **Problème :** Ce composant correspond au *Contexte d'Application* d'un cas. Il contient des descripteurs liés aux buts, au profil de l'utilisateur et à certains comportements de l'utilisateur qui peuvent apparaître dans l'interaction avec les activités du scénario. Les descripteurs sont représentés par un couple de la forme « attribut, valeur ».
   - Les descripteurs liés aux objectifs décrivent la nature des buts. Par exemple : « Perception-auditive, haut », « Perceptionvisuelle, moyen » ;
   - Les descripteurs liés au profil contiennent des informations concernant les compétences et les préférences de l'utilisateur. Par exemple : « Type-utilisateur, débutant », « couleur, vert », « Durée-session, [15.30] » ;
   - Les descripteurs liés aux comportements décrivent des situations particulières d'interaction. Par exemple : « consigne, suivie ».

   Les descripteurs de cas sont utilisés pour calculer le degré de similarité entre le cas cible et les cas sources dans le processus de raisonnement (voir la section 4.4.6).
2. **Solution** : ce composant correspond à la partie *Scénario*. Il contient une séquence d'activités permettant la réalisation des objectifs répondant à la description du contexte d'application.

Dès que la structure d'un cas a été présentée, un autre problème de base est celui de l'organisation de la base de cas [Sch82] [RS89]. Dans le cas d'une grande base, une organisation linéaire, telle qu'une liste, est très inefficace pour la recherche. Dans la prochaine section nous abordons la structure de la base de cas.

4.4.4 Organisation de la base de cas

L'organisation de la base de cas permet de structurer les cas de manière à établir des mécanismes optimaux d'investigation et de sauvegarde. Pour cela, nous nous somme inspirés du *modèle à mémoire dynamique* de **Roger C. Schank** [Sch82]. L'idée fondamentale consiste à organiser les différents cas ayant des similarités propres sous la forme d'une structure plus générale appelée *épisode généralisé* ou *EG*. Un *EG* contient trois types d'objets :

- *Norme (Norm)* : elle représente les données communes à tous les cas indexés sous l'*EG*. Elle est représentée par une liste de couples de la forme [attribut, valeur] ;
- *Index (Indices)* : ils représentent les différences entre les cas d'un même *EG*. Chaque index est lié à un attribut concret d'un descripteur[15] et contient une liste de paires

[15] L'ensemble d'index attaché à un *EG* peut utiliser différents attributs pour se pointer vers plusieurs nœuds.

[valeur, nœud[16]]. Chaque paire d'index lie son *EG* avec un autre nœud (cas ou *EG*) correspondant ;

– *Cas (Cases)* : ils représentent des expériences individuelles respectant la représentation donnée dans la section 4.4.3.

Ainsi se forme un graphe hiérarchique dont les nœuds sont soit des *EG* soit des cas. Les arcs représentent les liens entre les index et les nœuds.

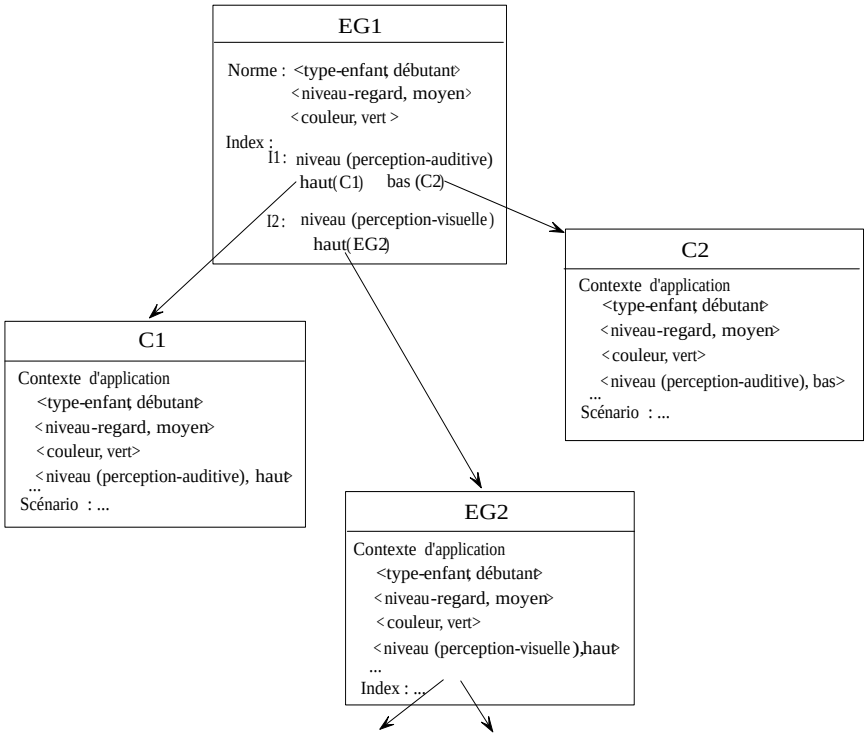

FIG. 4.5 – Organisation de la base de cas

Le schéma de la figure 4.5 illustre un exemple d'une base de cas qui représente les profils des enfants autistes. Il montre l'épisode généralisé *EG1* avec une norme contenant des attributs partagés par ses descendants (*C1, C2* et *EG2*) ; concrètement il indique que les scénarios[17] sous l'*EG1* sont pour des enfants débutants <*type-enfant, débutant*> dont le niveau du *regard* est moyen <*niveau-regard, moyen*> et la *couleur* préférée est verte <*couleur, vert*>. *EG1* est lié à deux cas (*C1, C2*) et à un épisode généralisé différent (*EG2*) au moyen de deux index (*I1, I2*). L'index *I1* utilise les valeurs de l'attribut du niveau de perception auditive <*niveau (perception-auditive)*> pour distinguer les deux cas *C1* et *C2*. Dans *C1* la valeur est <*haut*> et dans *C2* elle est <*bas*>. L'index *I2* lie *EG1* avec *EG2* à travers l'attribut <*niveau (perception-visuelle)*>. Dans la norme de *EG2*, on peut remarquer que les attributs communs sont ceux de *EG1* plus l'attribut <*niveau*

[16]nœud de cas ou de *EG*.

[17]Le scénario est considéré, dans ce cas applicatif, comme une séquence de jeux éducatifs destinés aux enfants autistes.

(perception-visuelle)>.

4.4.5 Profil de l'utilisateur

Le profil de l'utilisateur a des fonctionnalités multiples utilisées à différents moments par l'agent de décision, plus particulièrement, dans le processus de raisonnement présenté dans la prochaine section. Il intervient aussi dans le processus d'interprétation du comportement de l'utilisateur (présenté dans la section 3.5.4) par les agents observateurs. Plusieurs types d'informations concernant l'utilisateur sont présents :

- informations générales ;
- compétences ;
- préférences, et
- histoire.

Les informations générales concernent l'identité de l'utilisateur, par exemple : nom, prénom, date de naissance, etc. Les compétences et les préférences donnent une description du profil de l'utilisateur identique à celle mentionnée dans le composant *Contexte d'application* du cas (présentée à la section 4.4.3). Ces informations sont donc représentées par une liste de descripteurs de la forme [attribut, valeur].

L'histoire[18] est vue comme un agenda de tout ce qui est arrivé à l'utilisateur lors de chacune des sessions. Elle doit permettre de retrouver exactement ce que le système avait prévu de faire faire à l'utilisateur, les activités qui lui ont été proposées et ce qu'il a réalisé. L'intérêt de ces informations réside dans plusieurs aspects. Par exemple, l'intérêt de l'histoire dans le cadre du PROJET AUTISME consiste à :

- garder une trace de ce que le système et l'enfant ont fait lors d'une session, ceci permet au système de revenir sur certains détails si, dans son processus de décision, l'enfant a besoin davantage de précisions pour accomplir certains buts ;
- évaluer la pertinence du jeu. Pour cela, l'expert peut visualiser la trace d'exécution sous deux formes : *animation graphique* ou *données statistiques* (voir l'annexe C pour plus de détail sur la trace d'exécution). À partir de son évaluation, l'expert peut modifier les scénarios, les règles de fonctionnement du jeu, etc ;
- suivre l'évolution de l'enfant.

L'historique peut être à l'origine de certaines règles : « si l'on a déjà présenté un tel jeu la semaine dernière, alors proposer un autre cette semaine » ou encore, « si l'enfant n'a pas eu d'activité de tel type depuis un mois, alors on introduit cette activité dans la session », etc.

[18] Le terme histoire ici est différent que celui de *Storytelling*, voir la section 4.2.4

4.4.6 Processus de raisonnement

Le processus de raisonnement permet de générer des scénarios d'activités adaptés à la situation. Pour cela, la stratégie adoptée, basée sur le RàPC [Kol93], consiste à sélectionner des cas sources, de la base de cas, similaires au cas cible et de les adapter à la situation actuelle du cas cible. Le nouveau cas ainsi généré peut être mémorisé dans la base de cas s'il répond à certains critères détaillés par la suite. Nous avons recensé trois étapes dans le processus de raisonnement : *remémoration*, *adaptation* et *mémorisation*.

4.4.6.1 La remémoration

La remémoration, des cas sources similaire au cas cible, est une étape qui intègre deux processus, *appariement* et *recherche*. Le processus d'appariement utilise une fonction qui calcule le degré de similarité entre les cas. Le processus de recherche consiste à élaborer des méthodes d'investigation optimales dans la base de cas qui tiennent compte de sa structure et de ses propriétés.

1. Appariement

L'appariement est défini dans [Kol93] comme étant « *un processus permettant de comparer deux cas entre eux et de déterminer leur degré de similarité* »[19]. La méthode d'appariement que nous avons utilisée est *le plus proche voisin* (k-nearestneighbors). Cette méthode, utilisée dans [EFnC00] [ED94] [Lea96], possède l'avantage d'être « très simple à implémenter » [Bis00] et le fait qu'elle utilise directement la notion de similarité pour mesurer la correspondance entre chaque cas source et cas cible. La similarité est mesurée en terme d'attributs appropriés des descripteurs de cas (cible ou source). Rappelons que les descripteurs d'un cas concernent les objectifs, profil de l'utilisateur et les comportements.

Nous distinguons deux filtres dans le processus d'appariement, le premier filtre sélectionne les cas sources dont la similarité avec le cas cible est supérieure à un seuil défini par l'expert. Le deuxième filtre sélectionne les cas sources qui minimisent l'effort d'adaptation.

Le premier filtre utilise la fonction numerique ϕ qui calcule le degre de similarité entre deux cas. La fonction ϕ est définie comme étant la moyenne pondérée des valeurs de similarité sur chacun des attributs du contexte d'application de chaque cas multipliée par l'importance de l'attribut.

$$\phi(C_1, C_2) = \frac{\sum_{i=1}^{n} w_i * \varphi(v_i^1, v_i^2)}{\sum_{i=1}^{n} w_i} \tag{4.1}$$

Le relation (4.1) exprime cette fonction de similarité où :

[19]Traduction du « *Matching is the process of comparing two cases to each other and determining their degree of match* ».

- C_1, C_2 sont des cas définis par un ensemble de descripteurs $d_i (i \in [1..n])$ du contexte d'application ;
- w_i est le coefficient d'importance du descripteur d_i. Les descripteurs les plus importants doivent être pris en considération, dans le calcul de la similarité, avec des valeurs plus élevées que d'autres descripteurs moins importants ;
- $\varphi(v_i^1, v_i^2)$ est la similarité entre deux valeurs v_i^1 et v_i^2 du descripteur d_i associées aux cas C_1 et C_2 respectivement.

Chaque attribut d'un descripteur a un mode de comparaison. La comparaison entre les deux valeurs v_i^1 et v_i^2 suivant un type donné est définie par la fonction φ. Pour un attribut de type numérique, la similarité est calculée à partir de la différence $|v_i^1 - v_i^2|$. Or, cette manière de procéder risque de fausser le résultat lorsque les attributs décrivant les cas ont des valeurs définies dans des vecteurs de tailles différentes. Dans ce cas, les attributs dont les valeurs sont définies dans des vecteurs de petites dimensions sont implicitement favorisés. Ce qui conduit à augmenter exagérément la valeur de similarité finale. Il est donc nécessaire de normaliser la valeur de similarité dans ces cas. La solution consiste plutôt à exprimer de manière explicite le domaine de définition des attributs et à intégrer cette information dans le calcul de la similarité.

Les types d'attributs que nous avons retenus et leurs fonctions φ correspondantes sont définis dans le tableau 4.2 avec :

- $d = |v_i^1 - v_i^2|$;
- $rang(v)$: est le rang de la valeur v dans le vecteur V. Ce dernier représente les valeurs possible que peut prendre un descripteur dont la dimension est strictement supérieur à 1. Par exemple, le descripteur *niveau* peut prendre trois valeurs définies dans le vecteur <*bas, moyen, haut*>.

Type d'attribut	La fonction φ
numérique strict	$\begin{cases} 1 & \text{si d} = 0 \\ 0 & \text{si d} \neq 0 \end{cases}$
numérique non strict	$\frac{1}{1+d}$
symbolique strict	$\begin{cases} 1 & \text{si } v_i^1 = v_i^2 \\ 0 & \text{si } v_i^1 \neq v_i^2 \end{cases}$
numérique ou symbolique gradué, défini dans le vecteur V	$1 - \frac{\lvert rang(v_i^1) - rang(v_i^2) \rvert}{Dim(V)-1}$

TAB. 4.2 – Types d'attributs et leur fonction de similarité

Le deuxième filtre minimise l'effort d'adaptation des cas sources à la situation actuelle du cas cible. L'effort d'adaptation, défini par la fonction 4.2, est calculé en fonction des descripteurs du cas cible non assurés par le cas source. Plus ces données sont importantes plus l'effort d'adaptation est important, moins la sélection du cas source est retenue.

$$\gamma(C_1, C_2) = \frac{\sum_{i \in E} w_i}{w_{max} * card(E)} \tag{4.2}$$

où

– E représente l'ensemble des descripteurs du cas cible C_1 non assurés par le cas source C_2 ;
– w_{max} est le poids maximum autorisé.

2. Recherche

La recherche dans la base de cas dépend fortement de sa structure. Dans une structure hiérarchique, la recherche est guidée par les normes partagées de chaque EG[20] et les index. Ce problème de recherche présente quelques propriétés spéciales.

D'abord, la structure hiérarchique de la base de cas permet une recherche heuristique guidée par la fonction de similarité. Cette dernière est limitée aux attributs des descripteurs du cas cible inclus dans la norme de l'*EG*.

Une autre caractéristique de la recherche réside dans l'absence d'un critère d'arrêt : un cas partageant tous les attributs du cas cible est rarement trouvé. Par conséquent, le critère de sélection est basé sur une comparaison entre la valeur de la fonction de similarité et un seuil entre 0 et 1 défini par l'expert. Pour chaque cas source dont la valeur de la fonction de similarité avec le cas cible dépasse le seuil sera retenu comme cas candidat pour la deuxième étape (l'étape d'adaptation).

Ces propriétés nous permettent d'établir des techniques de sélection basées sur des méthodes de recherche classiques de l'Intelligence Artificielle. Plusieurs algorithmes de recherche peuvent être combinés (par exemple : *A* recherche, Hill-climbing search*). Une stratégie de sélection établit des appels à des algorithmes de recherche de base avec leurs paramètres. Les paramètres d'un appel sont l'*identifiant de l'algorithme*, la *fonction heuristique* (quand c'est un algorithme heuristique), la *fonction de similarité* et le *seuil de sélection*.

4.4.6.2 Adaptation

L'étape d'adaptation permet de modifier les cas candidats sélectionnés lors de l'étape de remémoration, pour qu'ils répondent aux mieux à la description du cas cible. Dans la plupart des systèmes, l'étape d'adaptation nécessite une intervention humaine pour compléter une solution partielle ou tout simplement pour générer une solution entièrement à partir des cas. Ceci est dû à la difficulté de l'implémentation de cette étape comme le souligne [DBLN04] et à la nécessité de nombreuses connaissances [EFnC00] et un coût en terme de temps et d'efforts.

Dans la problématique de l'exécution adaptative, notre choix se porte sur une adaptation automatique sans intervention humaine. Ce choix est motivé par deux raisons :

– D'une part, par définition, l'exécution adaptative exige une reconfiguration des activités du scénario d'une manière automatique sans intervention explicite de l'utilisa-

[20] épisode généralisé

teur ;
- D'autre part, étant donné que notre application est destinée aux enfants autistes accompagnés, une intervention du tuteur au cours de la session peut perturber l'enfant avec un risque de rupture.

Notations

Selon la représentation de cas que nous avons adoptée, nous définissons un cas C par un ensemble de descripteurs et un scénario d'activités. L'ensemble des descripteurs du cas C liés aux objectifs est noté d_G^C et celui liés au profil et aux comportements est noté d_P^C.

Soit un cas cible C_{cible} et l'ensemble des cas candidats $C_{candidats}$ sélectionnés lors de la première étape de raisonnement (étape de remémoration), avec $C_{candidats} = \{C_1, C_2, ..., C_n\}$. L'étape d'adaptation consiste à adapter les cas candidats $C_i \in C_{candidats}$ à la description du cas cible C_{cible}. Pour cela, nous avons adopté une stratégie qui distingue deux types d'adaptations, globale et locale comme le montre l'algorithme 1.

1. Adaptation globale

Il s'agit de remplacer des sous-scénarios des cas candidats par d'autres répondant au mieux aux objectifs du cas cible. Pour cela, il faut recenser tous les objectifs du cas cible non assurés par les cas candidats, et chercher les activités qui peuvent compléter ces buts. Ces activités seront ainsi insérées dans les scénarios des cas candidats correspondants. Le meilleur cas candidat similaire au cas cible est considéré comme le cas répondant aux besoins du cas cible.

L'insertion des nouvelles activités est faite selon un ordre établi par l'expert. Concernant la cohérence des activités du scénario à fait l'objet de plusieurs études, notamment [CPE05] [CS05] basé sur la logique linéaire.

2. Adaptation locale

Il s'agit de régler les paramètres de configuration des activités du scénario du cas candidat avec des valeurs mieux adaptées à la description du cas cible, en d'autres termes adaptées aux préférences et aux compétences de l'utilisateur.

Algorithme 1 : Algorithme d'adaptation

Entrée :
- L'ensemble des cas candidats $C_{candidats}$
- Le cas cible C_{cible}

Sortie :
- *cas* : cas source adapté à la description du cas cible

Initialisation :
- $L :=$ Liste des cas candidats $C_{candidats}$/ $C_{candidats} = \{C_1, C_2, ..., C_i, ..., C_n\}$;
- $max := 0$;

Adaptation Globale

while *(L est non vide)* **do**
  Choisir et supprimer de L un cas C_i ;
  for *(chaque descripteur* $d \in d_G^{C_i}$*)* **do**
    if *(*$d \notin d_G^{C_{cible}}$*)* **then**
      Chercher une activité qui satisfait le descripteur d ;
      Ajouter cette activité au scénario du cas C_i ;
    end
  end
  $sim := \phi(C_{cible}, C_i)$;
  if *(*$sim > max$*)* **then**
    $max := sim$;
    $cas := C_i$;
  end
end

Adaptation Locale

for *(chaque descripteur* $d \in d_P^{cas}$*)* **do**
  if *(la valeur de d n'est conforme à la description* $d_P^{C_{cible}}$*)* **then**
    Changer la valeur de d avec une valeur conforme de $d_P^{C_{cible}}$;
  end
end

4.4.6.3 La mémorisation

La mémorisation consiste à mettre à jour la base de cas après chaque session. Il s'agit d'évaluer, en termes d'apports éducatif et informatique, le nouveau cas et de le sauvegarder dans la base de cas en tenant compte de sa structure et ses propriétés. Cette étape se compose donc de deux phases : évaluation et sauvegarde.

1. Évaluation

Pendant la phase d'évaluation, chaque scénario est évalué dans deux dimensions : *éducative* et *informatique* :

- Dans la dimension éducative, le souci des données concerne uniquement le profil de l'utilisateur. Fondamentalement, les changements des préférences et compétences recueillies pendant la session et les erreurs faites par l'utilisateur. Nous pensons que l'automatisation de cette dimension est difficile dans la mesure où elle dépend du contexte applicatif. Par exemple, dans le cadre du PROJET AUTISME, l'enfant est très sensible aux perturbations de son environnement. Ces perturbations ne peuvent pas êtres contrôlées par le système (bruits, ouverture de la porte de la salle, etc.). Le tuteur évalue donc ce qu'a fait l'enfant durant la session en tenant compte des perturbations de l'environnement et prend la décision de l'utilité de sauvegarder le cas ou non. Le fait de ne pas automatiser cette étape, ne met pas en cause l'exécution adaptative dans la mesure où, la phase de mémorisation s'effectue à la fin de la session. Alors que l'exécution adaptative concerne la reconfiguration des activités au cours de leurs exécutions ;
- Dans la dimension informatique, l'objectif de l'évaluation consiste à estimer la qualité de chaque nouveau cas pour créer un nouveau cas source. Notre principe est que seulement les cas sensiblement différents des cas sources seront mémorisés. Il y a deux possibilités : quand le nouveau cas est assez semblable à n'importe quel cas source (par exemple quand le nouveau cas a été obtenu par une légère transformation d'un cas source), il ne sera pas mémorisé. En revanche, quand le nouveau cas a été obtenu par une transformation substantielle d'un cas source (le degré d'adaptation, globale et locale, excède un seuil défini par l'expert) ou le cas était appliqué à une situation différente (le degré de similarité excède un autre seuil défini par l'expert), un nouveau cas sera construit et ajouté à la base de cas. Une réorganisation des liens de la structure de la base est produite comme effet secondaire.

2. Sauvegarde

La phase de sauvegarde nécessite des mécanismes robustes pour maintenir la structure et les propriétés de la base de cas à chaque ajout d'un nouveau cas. Ceci implique trois étapes :

- *le choix d'un EG*[21] *qui contiendra le nouveau cas* : Le choix d'un *EG* candidat pour accueillir le nouveau cas est un processus de recherche semblable à celui du processus de recherche de l'étape de remémoration. Cependant, deux différences surgissent :
  - Le nœud cible est un *EG* ;
  - L'algorithme peut élaguer les nœuds du graphe puisque le nouveau cas doit complètement satisfaire la norme de *EG* candidat.
- *l'enchaînement du nouveau cas à l'EG* : L'enchaînement du nouveau cas à l'*EG* est

[21] épisode généralisé

atteint en choisissant un attribut d'index qui remplit quelques conditions. La paire « attribut, valeur » doit distinguer le nouveau cas parmi les autres descendants de l'*EG*, et la valeur doit être liée ;

– *la généralisation des cas/EG* : Afin de maintenir une structure optimale de la base de cas, cette phase devrait vérifier périodiquement la base de cas après chaque insertion pour détecter des situations dans lesquelles il est recommandé de généraliser. Quand un ensemble de nœuds appartenant à un *EG* et partageant un certain nombre de descriptions est détecté, une généralisation est déclenchée ; elle crée une nouvelle abstraction (*EG*) et réorganise les liens parmi les nœuds impliqués.

4.4.7 Approche méthodologique

Dans cette section, nous présentons la méthodologie que nous avons adoptée lors de l'utilisation de notre approche. Plus de détail est donnée dans le chapitre 5 (voir la section 5.8). La figure 4.6 illustre notre approche méthodologique.

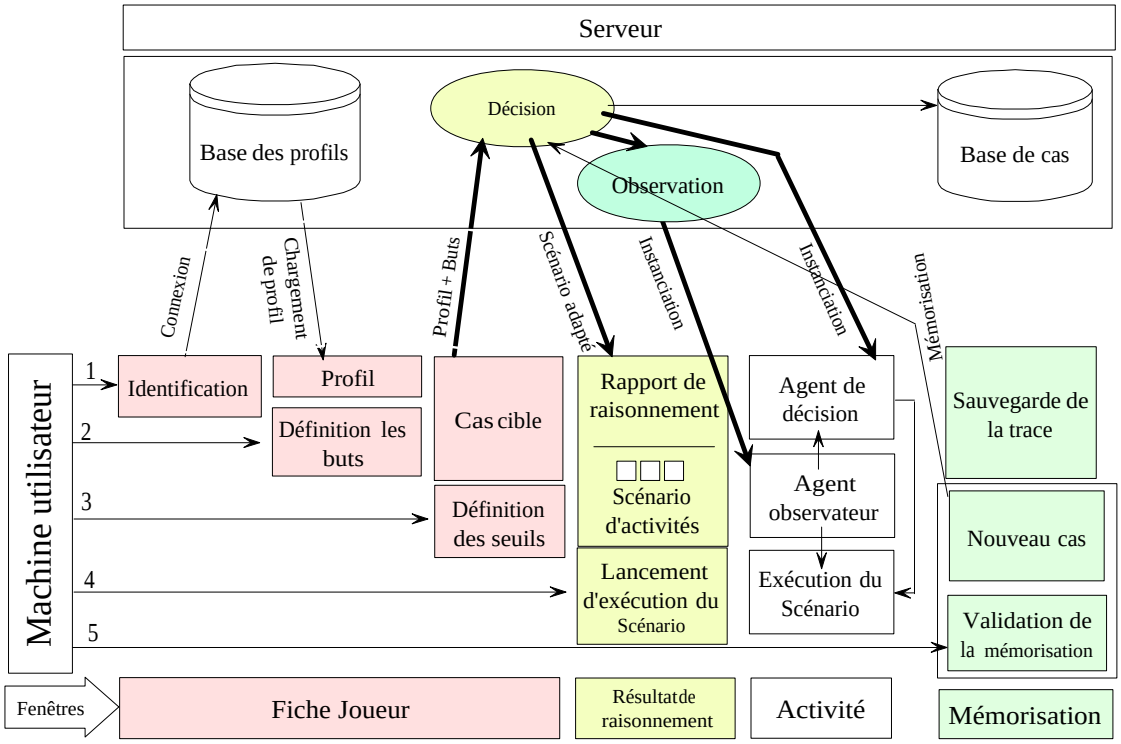

FIG. 4.6 – Approche méthodologique

Chronologiquement, l'utilisateur se connecte au système, son profil est alors chargé. Ensuite, l'utilisateur spécifie les buts à atteindre. Une fois ces informations entrées, un cas cible est créé. L'utilisateur définit les seuils de raisonnement. Le cas cible ainsi que les seuils sont transmis par message au système. Le message est intercepté par l'agent de raisonnement. Le rôle de cet agent est de générer un scénario adapté à la situation

actuelle du cas cible en utilisant le RàPC. Un rapport de raisonnement sera envoyé à l'utilisateur. Ce rapport contient le cas source choisi, les valeurs de la similarité et de l'effort d'adaptation entre le cas cible et le cas source ainsi que les modifications faites au niveau des activités du scénario du cas source sélectionné.

L'utilisateur lance les activités. Pendant la session, une instance de l'agent observateur suit en permanence les actions de l'utilisateur. Chaque exception détectée au niveau du comportement de l'utilisateur sera traitée par l'agent de décision en déclenchant un nouvel épisode de raisonnement. À la fin de la session, l'utilisateur valide (dans la dimension éducative) ou non le cas. Dans le cas favorable, le système procède à l'évaluation dans la dimension informatique pour sa mémorisation.

4.5 Conclusion

Dans ce chapitre nous avons défini la problématique de contrôle d'exécution. Dans la deuxième partie, nous avons dressé un état de l'art des méthodes de contrôle d'exécution. Les différentes approches se distinguent par la représentation des connaissances utilisées et les mécanismes de raisonnement adoptés. Les nombreuses études comparatives effectuées dans ce domaine affirment qu'il n'existe pas de méthode qui soit meilleure que les autres. La plupart de ces méthodes permettent de parcourir d'une manière limitée une base de connaissances définie par l'expert.

Dans un second temps, nous avons présenté notre modèle basé sur le paradigme du raisonnement à partir de cas. L'idée principale de diminuer l'effort lié à l'acquisition des connaissances, s'inspirant des principes issus de travaux en psychologie [Sch82] pour les réduire toutefois à une méthode efficace de résolution de problèmes. Notre principe consiste à déclencher un nouvel épisode à chaque fois qu'une incohérence entre l'activité en cours d'exécution et le comportement de l'utilisateur est détectée. L'approche que nous avons utilisée dans ce chapitre sera utilisée par l'agent de décision de l'architecture générale du système.

Nous présentons, dans le chapitre suivant, l'application de nos contributions dans le cadre du PROJET AUTISME.

Chapitre 5

Application : Le Projet Autisme

La technologie informatique est devenue un objet d'étude dans le cadre de l'autisme comme le montre plusieurs travaux de recherche. Le travail présenté ici entre dans le cadre du Projet Autisme, en partenariat avec le service psychiatrique de l'hôpital de La Rochelle, qui consiste à mettre en œuvre un système logiciel et matériel d'aide à la structuration des enfants autistes. L'objectif du projet consiste à proposer aux enfants souffrant d'autisme, un processus thérapeutique par manipulations interactives des outils informatiques. Il s'agit des séquences de jeux suffisamment précis selon les capacités et les caractéristiques de chaque enfant. Ce cadre d'étude présente les caractéristiques d'une exécution adaptative et interactive adaptée au comportement de l'utilisateur, comportement appréhendé par différents moyens (réponses au jeu et expressions faciales).

La première section, de ce chapitre, relate les différents travaux qui conjuguent les technologies informatiques et structuration des personnes autistes. La deuxième section présente le cahier des charges que nous avons établies avec nos partenaires du secteur médical. La section suivante présente les spécificités de cette application. Il s'agit de quelques comportements qui portent un intérêt dans ce domaine, les différents types d'utilisateurs impliqués dans le système et les spécificités des jeux. Par la suite, nous justifions notre choix du paradigme des systèmes multi-agents pour l'implémentation du système, ainsi que l'architecture adoptée. Les sections suivantes présentent l'implémentation et les modalités d'utilisation du système. Il s'agit de présenter les différentes bibliothèques de classes, les différentes interfaces pour chaque type d'utilisateur et le processus d'utilisation du système. La dernière section présente quelques études de cas. Il s'agit des bilans évolutifs concernant quelques enfants autistes qui ont été suivis par les médecins.

5.1 Autisme et technologie informatique

L'autisme infantile est un trouble global et précoce du comportement apparaissant avant l'âge de trois ans. Il est considéré comme un trouble du développement neurophysiologique par la communauté scientifique internationale. Il se manifeste par un certain nombre de signes caractérisés par un fonctionnement déviant et/ou retardé dans le domaine de l'interaction sociale et de comportement, mais il s'agit essentiellement d'une difficulté, voire d'une impossibilité de communication avec l'entourage. Parmi les signes les plus caractéristiques, on trouve que les personnes autistes sont fascinés par les objets qui tournent, ne craignent pas le danger, établissent difficilement des contacts avec autrui, préfèrent s'isoler, etc. Malgré la diversité de ces signes, diverses méthodes d'évaluation et de traitement sont proposées dans la littérature [SRL88] :

- *le traitement psychologique et comportemental* qui vise à ouvrir l'enfant à la notion de l'autre en lui donnant les moyens de construire et reproduire des actions concrètes qui procure un minimum de maîtrise sur son environnement ;
- *la méthode TEACCH* qui permet dans une première étape d'évaluer les handicaps tels que la capacité d'initiation et la capacité de compréhension. À partir de cette évaluation, la méthode donne des outils de communication ;
- *la communication facilitée* qui permet aux enfants de communiquer avec autrui par l'intermédiaire d'un tiers, l'ordinateur par exemple. Dans ce cas les enfants sont aidés par une personne *tuteur*.

En se basant sur la dernière méthode (la communication facilitée), plusieurs travaux scientifiques dans le domaine de l'autisme soutiennent le champ de recherche théorico-clinique sur l'usage de l'informatique dans l'autisme. À titre d'exemple, l'article [Gep01] concerne la malvoyance du mouvement dans l'autisme infantile, d'autres travaux [FM05] s'intéressent rôle des émotions dans l'encodage des perceptions, etc.

Comme conséquence, la technologie informatique pour la structuration des personnes autistes est devenue un objet de recherche qui occupe une place particulière, comme le montrent plusieurs travaux de recherche[22]. L'intérêt de la technologie ici est justifié par les différences interindividuelles très importantes des personnes autistes. De fait, les nouvelles technologies peuvent donc être considérées comme particulièrement pertinentes puisqu'elles permettent de faire intervenir de manière contrôlée les comportements communicatifs et émotionnels de chaque personne autiste.

En conséquence, de nombreux systèmes interactifs d'aide à la structuration des enfants avec autisme sont développés [Dau00] [FM05] [Mon03] [RDtBB04]. Il s'agit de systèmes robotiques ou de logiciels qui peuvent interagir avec l'utilisateur humain. Parmi les systèmes robotiques, citons le robot KISMET [BS99] et la poupée ROBOTA [Bil00] [BDH98]. KISMET est un robot humanoïde qui peut exprimer des interactions sociales. De telles in-

[22]Plusieurs conférences, organisées ces dernières années, conjuguent réhabilitation des personnes autistes et technologie, par exemple AAATE (The Association for the Advancement of Assistive Technology in Europe), CWUAAT (Universal Access and Assistive Technology), RVISA (Robotic and Virtual Interactive Systems in Autism Therapy), etc.

teractions peuvent être considérées comme un moyen pour le développement des rapports sociaux. La poupée ROBOTA est un robot humanoïde développé comme jouet interactif utilisant l'imitation, la parole et les gestes. D'autres systèmes [FM05] [Par01] utilisent un environnement virtuel pour la compréhension des expressions émotionnelles des enfants avec autisme. Dans [GMN04], les auteurs s'intéressent à la conception d'IHM destinées aux personnes autistes.

5.2 Cahier des charges et approche adoptée

Les approches présentées précédemment permettent d'assurer une interactivité avec les personnes autistes. Néanmoins, nous constatons, d'une part, que le concept de l'exécution adaptative n'est pas abordé, d'autre part, ces approches ne tiennent pas en compte des directives de l'expert[23] dans un processus éducatif, c'est-à-dire elles ne sont pas *ouverts* (paramétrable) de manière que l'expert peut modifier les connaissances du système.

Notre approche avec le service de pédopsychiatrie de l'hôpital de La Rochelle consiste à utiliser les jeux éducatifs comme moyen d'interaction avec les enfants autistes dans un processus éducatif spécifique. Le principe de l'interactivité avec des outils informatiques existe depuis 20 ans dans le service de pédopsychiatrie de l'hôpital de La Rochelle dirigé par Monsieur **D. Lambert** [Lsq]. Il s'agit de jeux adaptés qui présentent sur écran des stimuli représentant des objets ayant retenus préalablement l'attention de l'enfant et comportant une valeur émotionnelle de satisfaction. Ces objets vont être amenés à subir des transformations physiques (vitesse du mouvement par exemple) qui permettront l'établissement des catégories élémentaires (fort - doux, vite - lent, grand - petit) que l'enfant pourra même reproduire également dans d'autres situations éducatives.

Le cahier des charges que nous avons établi avec nos partenaires du secteur médical, dans le cadre de ce projet, consiste à :

- compléter les démarches d'évaluation psychologique et éducative plus traditionnelles, en proposant des *jeux* permettant d'apprécier les capacités d'attention et de comprendre le comportement de l'enfant face aux stimuli présentés (par exemple, voir si l'action produite est liée à une compréhension entre la cause et l'effet) ;
- améliorer la compréhension du comportement de l'enfant afin d'anticiper des comportements de colère, départ, répétitif, etc, et de les maintenir attentifs et réceptifs vis-à-vis de l'outil informatique ;
- modifier les croyances de l'enfant en proposant des images virtuelles qui interagissent avec l'enfant, mais en tenant compte de ses spécificités autistiques, par exemple en ralentissant le mouvement des stimuli pour que l'enfant puisse en extraire une information générale, utile pour rééduquer ses troubles émotionnels, langagiers, perceptifs et cognitifs ;
- favoriser le besoin de « pareil » sécurisant des enfants autistes, tout en introduisant des procédures d'introduction du « pas pareil » afin d'éviter l'enfermement dans l'im-

[23] Le terme *expert* désigne le médecin clinicien dans ce chapitre

muabilité de la répétition ;
– développer le codage temporel par la subordination du logiciel à une fonction narrative qui spécifie l'enfant dans une chronologie en interaction avec son environnement.

Pour répondre à cela, nous présentons un système permettant d'observer les actions de l'enfant afin de comprendre son comportement dans le but de lui proposer des activités adaptées en tenant compte des consignes de l'expert. Chaque enfant, de par son niveau et ses compétences particulières, nécessite un traitement et un accompagnement adapté. L'adaptabilité permet d'éviter la généralisation sans précaution d'une méthode particulière et favorise la prise en charge des déficits spécifiques observés chez chaque enfant. Il est donc important de repérer et interpréter avec soin ces comportements, intrinsèques à chaque enfant, afin de l'aider dans son développement.

Dans cette optique, le système doit permettre [SEL05] [LES+05] :

– de construire un scénario d'activités, sous forme de jeux éducatifs, répondant aux objectifs éducatifs que l'enfant veut/doit atteindre en tenant compte de son profil ;
– d'observer en permanence les actions de l'enfant au cours de la session afin de comprendre son comportement. L'observation est faite sur les actions effectuées sur les périphériques : souris, écran tactile, clavier,... et sur les gestes, mouvements corporels, orientations des yeux...
– à partir de l'observation du comportement de l'enfant, le système détecte les cas où les activités proposées ne répondent plus à une démarche standard et met à jour le scénario d'activités en pareil cas.

Rappelons qu'un scénario regroupe un certain nombre d'activités dans un ordre défini pour atteindre des objectifs éducatifs et thérapeutiques complexes. L'ordre dans lequel les activités d'un scénario sont présentées est important dans la mesure où certaines activités nécessitent une maîtrise de certaines actions de l'enfant qui doivent être maîtrisés dans d'autres activités. Par exemple, une tâche impliquant une motricité fine avancée, à savoir tracer une figure complexe, ne sera pas tentée avant qu'une motricité fine plus élémentaire, par exemple, pointer sur une cible à l'écran ne soit maîtrisée.

5.3 Spécificités de l'application

En plus d'être socialement important, le contexte applicatif du projet constitue paradoxalement un environnement à hypothèses simplificatrices où l'exécution adaptative par observation et analyse du comportement de l'utilisateur est primordiale pour mener à bien l'intération entre l'enfant et le jeu.

Dans cette section, nous présentons les spécificités de cette application. Il s'agit de décrire quelques comportements de l'enfant qui portent un intérêt particulier lors du déroulement du jeu, les différents types d'utilisateurs qui interviennent dans le système et la nature des jeux.

5.3.1 Comportements à observer

Les comportements que nous avons considéré lors du déroulement du jeu ont pour but d'attirer l'attention de l'enfant. Ces comportements peuvent être des comportements généraux qui ne dépendent pas du jeu, ou des comportements spécifiques aux jeux.

5.3.1.1 Comportements généraux

La figure 5.1 présente le schéma général du contrôle d'exécution des jeux destinés à des enfants autistes. Il s'agit d'un contrôle par observation et analyse du comportement de l'enfant autiste. Après le lancement du jeu, ce dernier présente des stimuli à l'écran. Le système capte les réactions de l'enfant vis-à-vis des stimuli. Les comportements qui portent un intérêt particulier dans un premier temps concernent la *rupture* et l'*évitement* :

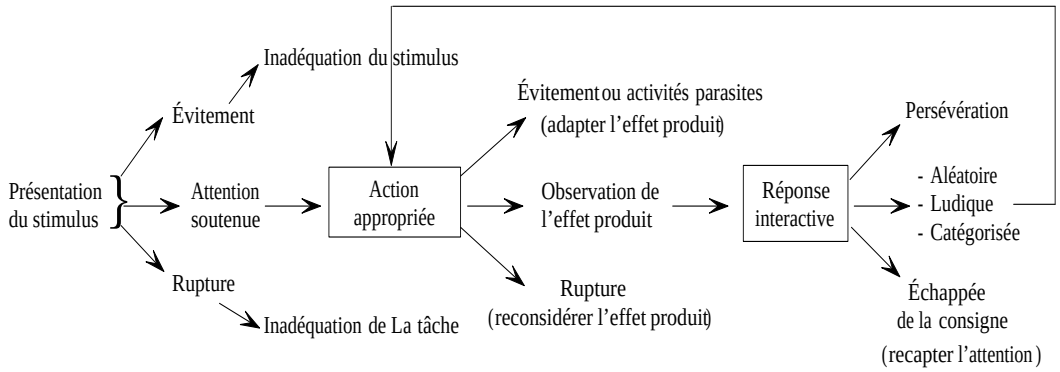

Fig. 5.1 – Contrôle d'exécution des jeux par analyse de comportements

- La *rupture* de l'enfant est définie par son éloignement physique de l'outil informatique lors de la session. La rupture de l'enfant est un comportement qui montre l'inadéquation du jeu à l'enfant. Dans ce cas, l'exécution adaptative consiste à relancer une autre activité ;
- L'*évitement* de l'enfant est défini par son raccrochement à l'activité mais sans réaction de sa part sur celle-ci. On trouve deux types d'évitement : *temporel* et *visuel* :
  - Le premier concerne toute orientation du regard dans une direction autre que celle de l'écran où le jeu se déroule ;
  - Le deuxième concerne une passivité de l'enfant pendant un certain temps. Il se manifeste par aucune action sur le jeu pendant une durée dépassant un certain temps défini par l'expert.

  L'évitement de l'enfant est dû le plus souvent à une inadéquation du stimulus à l'enfant. Dans ce cas, le système change soit l'objet du jeu soit son comportement.

5.3.1.2 Comportements spécifiques au jeu *Coucou Caché*

Nous présentons dans cette section quelques comportements liés au jeu *Coucou caché*. Il s'agit des comportements qui portent un intérêt particulier dans le déroulement de ce jeu. Rappelons que ce jeu a été présenté en détail dans la section 2.3.

Une des consignes de ce jeu consiste à faire réapparaître la boule curseur quand elle sera cachée derrière une grosse boule. Le non respect de cette consigne est considéré comme un comportement d'échappe à la consigne <*consigne, non-suivie*>. Dans ce cas, l'exécution adaptative du jeu consiste à recapter l'attention de l'enfant par la réduction du nombre de boules.

FIG. 5.2 – Comportements spécifiques au jeu *Coucou caché*

Un autre comportement présente un intérêt particulier dans ce jeu. Il s'agit des séquences des grosses boules empruntées par la boule curseur. La figure 5.2 montre deux exemples de séquences (rouge, bleu et jaune) et (noir, bleu et ciel). Plus le jeu est maîtrisé plus le nombre de séquences est important. Dans ce cas l'exécution adaptative consiste à augmenter le niveau de complexité en ajoutant d'autres boules à chaque fois que le système détecte une maîtrise du jeu.

5.3.2 Différents types d'utilisateurs

Du point de vue de l'utilisateur, nous considérons trois catégories distinctes : les experts, les tuteurs et les joueurs.

5.3.2.1 L'expert

Nous pensons qu'il est nécessaire de prendre en compte les experts du domaine dès la conception des jeux afin d'obtenir leurs adhésions et de répondre à leurs besoins éducatifs et thérapeutiques.

Pour cela, deux approches sont possibles. La première consiste à donner un rôle de concepteur et de réalisateur aux experts et de les laisser développer eux-mêmes les outils informatiques qui répondrent à leurs besoins. Cela demande du temps et des compétences techniques que les experts doivent acquérir. L'émergence des systèmes d'aide à la conception de logiciels éducatifs, dans une certaine mesure, permet de réduire ces difficultés. Cette première approche ne fait pas l'objet de notre stratégie.

La deuxième approche consiste à favoriser l'intégration des experts du domaine dans une équipe de production pluridisciplinaire et de créer des jeux ouverts, paramétrables par l'expert qui les utilisera selon ses propres besoins. Reste à définir comment permettre ce paramétrage : sur quoi doit-il porter ? est-ce techniquement possible ? comment le mettre en œuvre ? ...

Si les nouvelles technologies, avec les avancées dans les méthodes et les concepts de développement du génie logiciel, sont maîtrisées par le concepteur informaticien, leur application dans le cadre des jeux éducatifs et thérapeutiques relève de l'imagination et des compétences de transposition des experts. Le concepteur informaticien ne peut plus anticiper les besoins à partir d'un cahier des charges, sinon il court le risque de développer un produit trop technologique, sans réalité dans le domaine étudié, offrant un excès de fonctionnalités par rapport aux objectifs des experts.

C'est pourquoi, nous pensons que cette intégration ne doit pas se limiter à une prise en compte de l'expert en tant que simple utilisateur, c'est-à-dire prescripteur (adaptation, configuration, etc.) mais aussi et surtout en tant qu'**auteur** dès la conception des jeux (choix et définition des éléments du jeu, des règles de jeu, des directives associées au jeu, etc). C'est cette prise en compte de l'expert auteur que nous adoptons dans l'écriture des jeux. Pour cela, nous mettons à disposition de l'expert des outils lui permettant d'écrire le jeu (voir l'annexe B) et de le configurer (voir la section 5.7.1)

À partir de cette analyse, le rôle de l'expert consiste à :

- spécifier les activités, c'est-à-dire définir des instances de jeux avec des configurations particulières ;
- définir quelques objectifs éducatifs et les associer aux scénarios appropriés ;
- caractériser le profil de chaque joueur ;
- accéder et suivre l'histoire des enfants par analyse de la trace d'exécution.

5.3.2.2 Le tuteur

C'est la personne qui accompagne l'enfant dans la session de jeu. Il faut que le tuteur soit, de préférence, une personne familière que l'enfant apprécie afin de faciliter la

communication.

Le rôle du tuteur consiste à :

- déclencher les logiciels ;
- donner des consignes spécifiques ;
- assister l'enfant dans les tâches de l'activité ;
- suivre/contrôler l'interaction entre l'enfant et le scénario.

Par exemple, dans le cadre du jeu *Coucou Caché*, le tuteur lance l'activité et dit : « Fait bouger la petite boule dans ce cadre », si l'enfant ne comprend pas le langage, il peut être désorienté et se mettre à taper sur l'écran, mettre les mains à la bouche, ou simplement ne pas répondre, ou même, quitter la table. Le tuteur peut simplifier la tâche en montrant du doigt la petite boule, puis la fait bouger. Il peut même prendre la main de l'enfant et la guider.

5.3.2.3 Le joueur

Le Joueur est défini par un profil le caractérisant et des objectifs éducatifs à atteindre. Il s'agit des enfants autistes dont les profils respectent la formalisation que nous avons adoptée à la section 4.4.5.

5.3.3 Jeux

L'utilisation des outils informatiques dans le cadre des prises en charge thérapeutiques et d'évaluation peut favoriser l'émergence de performances révélant les capacités cognitives et des procédures d'apprentissages sous-évaluées des enfants avec autisme. Dans cette optique, plusieurs travaux utilisent les jeux comme moyen d'interaction [DRD+05]. Les jeux, que nous considérons, sont caractérisés par une simplicité afin de ne pas perturber l'enfant autiste, souvent sensible aux environnements complexes. C'est la raison pour laquelle les jeux possèdent un décor statique, comportent très peu d'objets et demandent une manipulation simple des objets. Néanmoins, chaque objet possède plusieurs comportements afin de permettre d'envisager une exécution adaptative.

La finalité de ces jeux permet aux enfants souffrant d'autisme, de troubles envahissants du développement et de troubles dysharmoniques, d'accéder par des manipulations interactives à des performances révélatrices de compétences dans le domaine perceptif, mnésique, spatial et temporel, et d'intégrer une communication codée s'apparentant à un langage (pictogrammes, images) [LES+05]. Ces objectifs sont soit saisis par le tuteur ou générés directement par le système si besoin dans le cas où le système aperçoit, par l'analyse du comportement (voir chapitre 3), que pour atteindre un objectif donné, le joueur à besoin d'un certain prérequis.

Les buts des activités peuvent être de diverses natures, le tableau 5.1 montre quelques activités et leurs buts dans le cas de l'autisme. L'annexe D présente quelques jeux développés dans le cadre du PROJET AUTISME.

Objectifs	**Activités**
Perception	Observer un objet mobile, Regrouper des objets similaires
Perception auditive	Associer le cri de trois animaux avec ceux-ci
Activités motrice	Pointer sur une cible à l'écran

Tab. 5.1 – Exemples de quelques objectifs éducatifs dans le domaine de l'autisme

5.4 Choix du paradigme multi-agents

Les analyses et les propositions, faites et décrites dans les chapitres précédents, dessinent une stratégie d'exécution adaptative intégrant l'observation et analyse de comportements dans un contexte de jeux éducatifs destinés aux enfants autistes. L'architecture que nous avons mise en œuvre permet aux experts du domaine, d'une part, de spécifier des jeux souples et modulables en fonction de chaque enfant autiste, d'autre part de suivre l'interaction entre le joueur et le jeu. Vue la centralité de la coopération et l'interaction dans les systèmes mutli-agents, nous avons utilisé ce paradigme dans la mise en œuvre de notre architecture. Le choix des systèmes multi-agents (SMA) n'est qu'un moyen de mise en œuvre *fonctionnelle* de l'ensemble des modules de notre système.

L'annexe A présente un bref état de l'art sur les SMAs. Cette présentation se veut à la fois didactique mais non exhaustive tant le sujet est vaste. Ces préliminaires permettront de se positionner par rapport aux différentes architectures existantes.

Plusieurs principes de base dans l'analyse des besoins justifient ce choix :

- La complexité de l'organisation des connaissances : les différents types de connaissances sont représentés dans divers formalismes (représentation du profil des joueurs, connaissances de la base de cas, connaissances d'observateur, etc.) ;
- La complexité de la communication entre les différents types d'utilisateurs (joueur, expert et tuteur) avec le système et entre eux ;
- Avoir des parties interdépendantes et autonomes de l'architecture ;
- La répartition des informations et des connaissances sur, d'une part, un serveur et, d'autre part, des clients où le jeu se déroule, afin d'avoir un ensemble modules structurés et qui communiquent entre eux.

La meilleure manière de satisfaire ces impositions, c'est l'adoption du paradigme multi-agent. Car les agents sont indépendants pour prendre des décisions basées sur des plans d'action ou des règles, tournent de façon indépendante, sont auto stimulés pour exécuter des tâches et peuvent envoyer des messages vers d'autres agents. De cette façon, nous pouvons avoir un système où ses composants sont caractérisés par leurs autonomies.

D'autres points vont dans le sens de l'exécution adaptative. Il s'agit de la capacité des SMAs à [Fer95] :

- construire des architectures ouvertes, distribués, hétérogènes et souples, capables d'offrir une grande qualité de service dans un travail collectif, sans imposer une structure a priori ;

– définir simultanément la structure des agents et celle de leur environnement ;
– analyser à partir d'une spécification globale la société que l'on veut obtenir et les propriétés recherchées.

Cela signifie que nous pouvons créer des agents dès que l'analyse de besoins montre la nécessité d'un nouvel agent pour atteindre un nouvel objectif identifié, mais aussi quand, dans les tests unitaires, nous découvrons la nécessité de faire des ajustements dans le système. Cet agent peut être intégré sans aucun problème à la structure du logiciel déjà fonctionnel, sans nécessiter la moindre re-programmation ou changement en profondeur. Nous pouvons alors tester les agents individuellement dès qu'ils sont prêts.

5.5 Architecture du système

L'architecture d'un système informatique est un ensemble de structures comprenant chacune : des agents, des propriétés extérieurement visibles de ces agents et les relations que ces agents entretiennent. À partir de cette définition inspirée de [BCK98] et pour bien décrire l'architecture développée pour le système, il faut d'abord décrire les agents qui en font partie et leurs fonctions, présenter le schéma de l'architecture du système et finalement présenter la communication. L'architecture de notre système [SE05a] [SEL05], sera développée dans cette section à partir de ces trois parties.

5.5.1 Les agents du système

L'architecture du système doit permettre d'assurer une exécution adaptative des jeux éducatifs destinés à des enfants autistes. Il s'agit donc d'adapter les jeux en tenant compte des comportements de l'enfant. Certains de ces comportements ont été définis dans la section 5.3.1. Notre démarche, basé sur les approches que nous avons développées dans le chapitre 3 et 4, consiste à observer l'enfant afin de comprendre son comportement et répondre en temps réel par des actions adoptées en tenant compte des consignes de l'expert. Pour cela trois agents principaux composent le système. Le premier assure le processus d'observation qui consiste à observer, analyser et interpréter le comportement de l'enfant. Le deuxième consiste à adapter et contrôler l'exécution des jeux et le troisième consiste à exécuter les jeux.

5.5.1.1 Agents observateurs

Les agents observateurs restent actifs pendant toute la session d'interaction de l'enfant avec le système. Le mécanisme d'observation que nous avons adopté (voir la figure 5.3) pour ces agents a été développé dans le chapitre 3. Il s'agit donc des agents réactifs qui permettent de :

– capter les flux d'évènements provenant des actions du joueur ;
– représenter ces actions dans l'environnement d'observation ;

– détecter les comportements décrits et présents dans le *composant contexte* et donner un *point de vue* sur ce contexte.

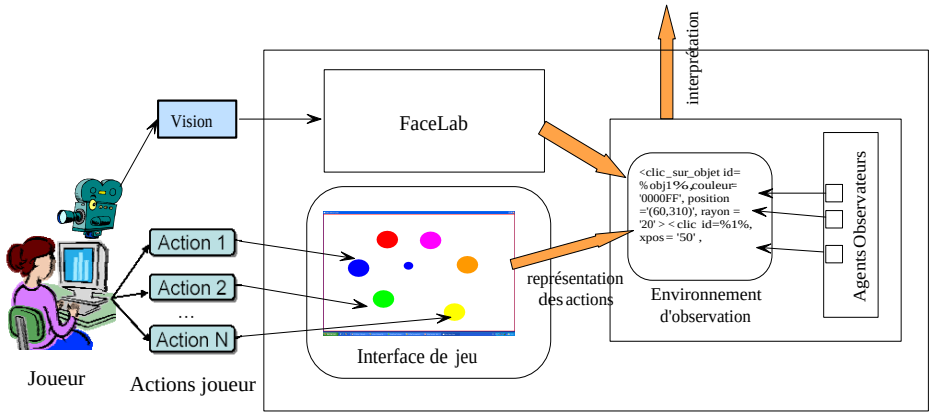

FIG. 5.3 – Mécanisme d'observation adopté par l'agent de décision

Le processus d'interprétation du comportement de l'utilisateur n'est autre que le résultat de la communication entre les agents observateurs. La communication que nous avons adoptée est une communication indirecte (via l'environnement d'observation) dans la mesure où aucun agent n'est censé connaître les compétences des autres agents.

5.5.1.2 Agent de décision

L'agent de décision doit fournir à l'agent d'exécution, des scénarios adaptés au profil du joueur et aux objectifs éducatifs à atteindre. Pour assumer cette tâche, cet agent, de nature cognitive contient quatre modules, comme le montre la figure 5.4 : *Buts*, *Profil joueur*, *Base de cas* et *Raisonnement*.

Dans chaque session, certains objectifs éducatifs sont en activité dans le module *Buts* et certaines informations caractérisant le joueur, sont tenues dans le module *Profil joueur* dont la formalisation adoptée est présentée dans la section 4.4.5. Étant donnés les objectifs existants et le contenu du profil du joueur, le module de *Raisonnement* permet de générer des scénarios adaptés à ces informations en utilisant les connaissances présentes dans la base de cas. Pour cela, l'agent utilise le processus de raisonnement développé dans la section 4.4.6. Les scénarios générés seront ainsi envoyés à l'agent d'exécution.

Atteindre un objectif revient à construire un scénario. Mais il ne s'agit pas d'une exigence rigide, les scénarios engendrés par l'agent de décision peuvent et doivent être modifiés lorsqu'ils se révèlent désuets dans le cas de l'apparition de certains comportements. Il s'agit des actions du joueur qui se révèlent pas adéquats par rapport à l'activité en cours. Dans ce cas, les agents ont la charge de notifier ces situations indésirables qui empêchent l'exécution standard du scénario. Il s'agit d'un processus d'interprétation (voir la section 3.5.5).

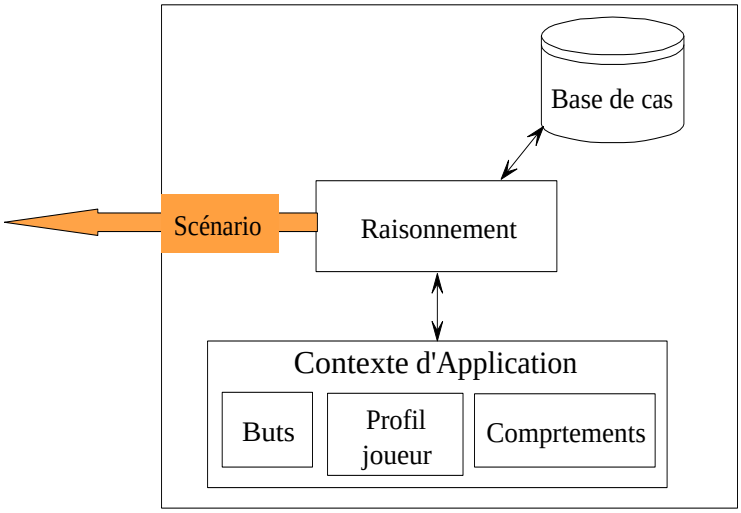

FIG. 5.4 – Architecture de l'agent de décision

À chaque fois qu'un comportement incohérent du joueur se présente, un nouvel épisode est déclenché. Il s'agit, soit d'un nouveau processus de décision, dans le cas où le joueur ne respecte pas la consigne du jeu par exemple, soit d'un comportement réactif de l'agent, par exemple, en cas de rupture une musique familière à l'enfant sera déclenchée.

5.5.1.3 Agent d'exécution

L'agent d'exécution, comme son nom l'indique, est chargé d'exécuter les scénarios d'activités générés et envoyés par l'agent de décision. En parallèle, il est capable de recevoir toutes les instructions provenant de l'agent de décision lors de l'exécution des activités ce qui permet d'assurer une exécution adaptative.

Une autre tâche assurée par cet agent concerne la sauvegarde de la trace d'exécution. La trace est sauvegardée dans un fichier XML. Elle permet de retrouver tous les évènements importants qui se sont produits dans la session du jeu. Ce qui permet de la visualiser par l'expert pour l'analyse. Un autre mode de visualisation de la trace est assuré par cet agent. Il s'agit de visualiser le déroulement du jeu en temps réel, tel qu'il est manipulé par le joueur, sur un poste distant.

5.5.2 Architecture générale

La figure 5.5 montre l'architecture de notre système multi-agents, conçue à partir des contributions des chapitre 3 et 4. Rappelons que notre objectif était de concevoir une architecture qui assure une exécution adaptative par observation et analyse du comportement de l'utilisateur. Il s'agit, à partir des interactions entre le joueur et le système, de détecter les cas où le comportement du joueur n'est pas cohérent par rapport à l'activité

en cours et de réagir par modification du comportement de l'activité.

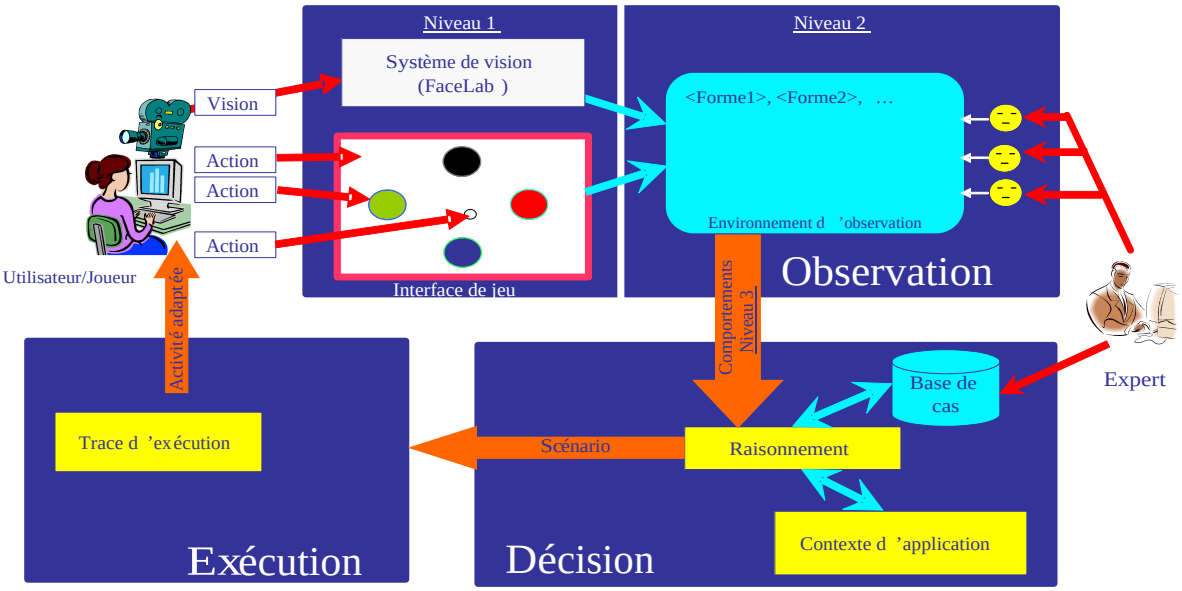

FIG. 5.5 – Architecture du système

L'interaction entre le joueur et le système n'est autre que l'interaction du joueur avec le jeu, en particulier les objets du jeu (stimuli du jeu). Il s'agit des actions du joueur appliquées directement sur les stimuli ou des expressions faciales. Ces réactions seront représentées dans l'environnement d'observation constituant des *formes* particulières et adoptant une formalisation décrite dans la section 3.5.6.

Par raffinements successives, que nous avons appelé *processus d'interprétation*, les agents observateurs vont donner des points de vue sur ces formes. Les points de vue constituent des actions des observateurs sur l'environnement d'observation suivie des productions qui consiste à détecter des instances de certaines formes sous certaines contraintes.

À chaque détection d'un comportement, un épisode est déclenché par l'agent de décision. Un épisode constitue une nouvelle phase de raisonnement. Il s'agit d'un processus qui génère des scénarios d'activités adaptés à la situation. La situation est décrite ici par le *contexte d'application* représentant les buts éducatifs à atteindre, le profil du joueur et son comportement vis-à-vis des stimuli du jeu.

Le processus de raisonnement de l'agent de décision est basé sur la méthode développée au chapitre 3. Il s'agit donc, d'un cycle de raisonnement à partir de cas, avec ses différentes phases : remémoration, adaptation et mémorisation. Les connaissances de la base de cas, les différents profils ainsi que les comportements à observer, utilisés par cette agent sont introduits par l'expert à travers les différentes interfaces qui seront présentées dans la section 5.7.

Durant la session, tout épisode déclenché par l'agent de décision provoque un ensemble d'opérations à effectuer sur le scénario en cours d'exécution. Ces opérations seront envoyées

à l'agent d'exécution. À leur réception, ce dernier interrompt le scénario en cours et procède à l'exécution de ces opérations. Il s'agit des modifications des comportements des objets du jeu. Par exemple, changement dans l'ordre dans lequel les activités se présentent, l'ajout d'autres activités, ou même l'arrêt complet du scénario et le lancement d'autres activités qui sortent du cadre éducatif de l'enfant. C'est le cas de rupture ou d'évitement du joueur évoqués ci-dessus.

5.5.3 Communication

La communication est la base de l'interaction et de l'organisation des SMA, ce qui lui donne une place primordiale dans le paradigme des SMA. C'est en communiquant que les agents peuvent échanger des informations. Dans cette section, nous présentons la notion de ce terme, les différents types de communication ainsi que la structure des messages que nous avons adoptée.

Définition 1.
Communication : **1.** Le fait de communiquer, d'établir une relation avec *qqn, qqch* (correspondance, rapport). Toute relation dynamique qui intervient dans un fonctionnement (information). **2.** Action de communiquer *qqch a qqn* ; résultat de cette action (information, message). **3.** Moyen technique par lequel des personnes communiquent (transmission). Le micro Robert, 1995

Définition 2.
Communication : est le lien établi de manière temporaire entre deux partenaires par l'intermédiaire d'un moyen de transmission et qui permet l'échange d'informations entre ces correspondants. Dictionnaire de l'informatique Larousse, 1996

À partir de ces définitions, nous constatons que la communication est un lien entre correspondants. Le médium ou canal (qui établit le lien) ne caractérise pas seulement un type mais également est la communication elle-même. Selon **Ferber** [Fer95], la communication s'exprime comme une forme d'interaction dans laquelle la relation dynamique entre les agents s'exprime par l'intermédiaire de médiateurs, appelés *signaux*, qui une fois interprétés, vont produire des effets sur les agents. Nous retenons le terme "*message*" pour référer à la chose à *partager* et non le mot *signal* ou information qui peuvent porter de nombreuses implications, notamment par rapport à la sémiologie et à la Théorie de l'information.

5.5.3.1 Types de communication

Selon [Foi98] la communication est définie comme étant une action locale d'un agent vers les autres agents, et qui peut prendre deux formes : *directe* ou *indirecte*. Dans la première les agents échangent les messages directement, dans le deuxième ils accèdent à une structure partagée dans laquelle les messages sont envoyés.

1. Communication par envoi de messages

La communication par envoi de messages suppose la définition d'un protocole de communication entre les agents. Il s'agit de l'établissement d'une communication, la transmission des données et la clôture de la communication. Il existe différentes variantes de protocoles : soit l'émetteur du message connaît le destinataire, c'est ce que l'on appelle la communication point à point, soit l'émetteur envoie le message à tous les agents sans connaître le destinataire intéressé par l'information, c'est la communication par diffusion, soit une diffusion restreinte. Dans le premier cas, cela suppose que les liaisons de communications sont fixes ; cette difficulté est contournée dans une communication par diffusion, ce qui permet aux agents de changer, d'apparaître et de disparaître en fonction des besoins.

De nombreuses variantes et cas d'applications traitées dans la littérature ont utilisé ce type de communication, par exemple, l'architecture Quiz développée dans [Fut90] utilise une communication point à point.

Dans la mesure où les connaissances du domaine de chaque agent de notre architecture est totalement distribuée, nous avons adopté ce type de communication pour faire échanger les informations entres les agents *Observateurs*, de *Décision* et d'*Exécution*. La structure des messages échangés est présentée dans la section 5.5.3.2.

2. Communication par partage d'information

Dans ce type de communication, on suppose qu'il existe un support de communication commun aux agents (un environnement ou une structure partagée). Ce type de communication est utile lorsqu'il y a recouvrement des domaines d'expertise de chaque agent. Ceci étant effectué en faisant la supposition que les agents ne possèdent qu'une connaissance limitée du domaine d'activité des autres agents [Iff92].

Les systèmes impliquant ce type de communication sont principalement basés sur l'architecture par *tableau noir* (*blackboards*) [HR85]. Cette architecture fut développée à l'origine en intelligence artificielle classique pour réaliser des systèmes de reconnaissances de la parole avec le système HEARSEY II [LE77]. Le tableau noir est généralement divisé en plusieurs régions appelées niveau. Les agents, appelés *sources de connaissances* dans cette architecture, peuvent écrire des messages, insérer des résultats partiels de leurs calculs et obtenir des informations du tableau noir.

Ce type de communication (par structure partagée) a été adopté par les agents observateurs dans leurs processus d'interprétation (voir section 3.5.5). La structure partagée par les agents ici est représentée par l'environnement d'observation. Ce dernier ressemble sur plusieurs aspects aux sources de connaissances des tableaux noirs, mais il en diffère sur deux formes :

- les niveaux dans un tableau noir, sont organisés de manière linéaire, comme un empilement, alors que les formes sont librement structurées ;
- les sources de connaissances sont attachées à un niveau du tableau noir, alors que

les observateurs peuvent requérir l'instanciation d'un nouveau contexte, ainsi que déterminer les contextes dans lesquels ils choisissent leurs constituants,

5.5.3.2 Structure des messages

Pour que les différents types d'agents communiquent entre eux, nous devons mettre au point un protocole d'échange de messages. Ce protocole doit être capable de fournir aux agents des mécanismes leurs permettant d'identifier le message et donc déclencher le traitement approprié. Ainsi, nous proposons une structuration en deux parties inspirée de la représentation des formes. À savoir une structure regroupant un *en-tête* et un *corps* :

- l'*entête* est un identifiant unique dont l'ensemble des valeurs varie en fonction des comportements à observer (communication entre les agents observateurs et l'agent de décision), des opérations à appliquer sur le jeu (communication entre l'agent de décision et l'agent d'exécution), etc ;
- le *corps* de message et un objet générique (de la classe Object en java) dont le type dépendra de l'en-tête ; c'est aux agents concernés par ce message de connaître le type du corps du message (par exemple, une liste de comportements, une demande d'interruption de l'activité en cours, etc.) et le traitement approprié.

Message	**Sémantique**
<FrameTouched id='1'> </FrameTouched>	signal un touché du cadre
<AdvoiceTempral id='2', time = '5'> </AdvoiceTempral>	signal un évitement temporal
<AdvoiceVisual id='3', eyesAngle = '60'> </AdvoiceVisual>	signal un évitement visuel
<Break id='4'> </Break>	signal une rupture

Tab. 5.2 – Représentation des messages échangés entre les agents du système

Nous avons adopté la représentation XML (voir le tableau 5.2) pour l'implémentation de cette structure de messages. Le choix de la représentation XML est justifié par :

- la capacité de XML de transmettre tout type de données, et cela indépendamment de la plate-forme utilisée et des langages de programmation. Ce qui permet une communication facile entre les jeux développés en C++ et le SMA développé en Java ;
- la facilité d'utilisation de la représentation XML. En effet plusieurs librairies permettent de produire ce type de représentation dans toute sorte de langage (Xerces : librairie créée par le groupe apache et disponible en C++ et Java), libxml : librairie créée par GNU, tinyxml, etc.

5.6 Implémentation

Nous présentons dans cette section certains aspects de l'implémentation du système. Il s'agit d'une plate-forme qui offre une panoplie de services permettant d'effectuer cer-

taines opérations sur les données. Ainsi, la plate-forme regroupe des bibliothèques de classes définissant les représentations et les méthodes de raisonnement permettant d'assurer l'exécution adaptative par observation et analyse de comportements. Leur mise en œuvre est basée sur les approches dans les chapitres précédents. C'est aussi un ensemble de classes utilisées par les agents du système.

En tant que bibliothèque, l'implémentation peut être vue comme un ensemble de classes permettant d'observer les actions de l'utilisateur sur l'activité en cours et d'appliquer certaines opérations sur cette activité. Toutes les opérations appliquées sont délibérées par les agents du système dans la mesure où les agents sont caractérisés une autonomie de prise de décision, ce qui caractérise leur différence par rapport au paradigme objet.

Dans cette section, nous présentons dans la première partie le choix de la plate-forme que nous avons utilisée pour développer les agents. Dans la seconde partie, nous présentons l'organisation que nous avons adoptée pour l'implémentation du système. Il s'agit d'un modèle client-serveur. Dans la deuxième partie, nous présentons trois bibliothèques de classes : *Descriptor* permettant de représenter les descripteurs des cas et le calcul de la similarité entre cas, *Adapter* permettant d'assurer l'adaptation des cas à la description du cas cible et la bibliothèque *Observer* permettant d'observer les actions de l'utilisateur et de détecter certains comportements particuliers.

5.6.1 Choix de la plate-forme

Les plates-formes sont des outils permettant de faciliter la construction et l'exploitation des systèmes multi-agents. Elles peuvent prendre différentes formes, allant d'outils d'ordre méthodologiques, à des outils de développement, ou de support d'exécution.

Certaines plates-formes sont issues du domaine du génie logiciel *commonKADS* [SWdH94] *DESIRE* (framework for DEsign and Specification of Interacting REasoning components) [BDKT97]. Ces plates-formes se concentrent particulièrement sur les aspects d'analyse et conception, et n'abordent pas l'aspect de développement. D'autres sont issues de l'Intelligence Artificielle telle que *JAM* (Java Agent Model) [Mar99]. Ces plates-formes prennent rarement en compte les interactions entre agents.

Il existe d'autres projets ayant pour but de normaliser le domaine des systèmes multi-agents, citons par exemple la *FIPA* (Foundation for Intelligence Physical Agents) et *AUML* (AgentUML). La *FIPA* n'abordent que les aspects normatifs. L'inconvénient de ces projets c'est qu'ils sous-entendent un modèle d'agent *complet* pour des applications générales ce qui augmente la complexité de la modélisation. Il faut, en effet, être capable de mettre en œuvre la communication par actes de langage définie par *ACL* (Agent Communication language). Ce qui n'est pas le cas dans notre architecture.

Du point de vue technique, différentes solutions ont été choisies : *AgentBuilder* [Ret99] et *Zeus* [CNT99] sont plutôt des "éditeurs *BDI*", alors que *Jack* [BRHL99] est un langage de programmation agent et *Madkit* [GF97] est un environnement d'exécution multi-agents. Chacune de ces solutions a des avantages et des inconvénients. Les éditeurs BDI sont très

efficaces car ils permettent de programmer à un très haut niveau d'abstraction, mais ils sont limités à une seule architecture d'agents. D'un autre coté, les langages de programmation d'agents exigent un effort de programmation qui nécessite plus de code à écrire, car le niveau d'abstraction est plus bas.

Une meilleure solution serait une plate-forme modulaire qui donne une librairie riche de modèles d'agents, où chaque modèle possède ces propres propriétés et caractéristiques. Pour cela, nous avons choisi la plate-forme DIMA.

DIMA (Développement et Implémentation de Systèmes Multi-Agents) [GMB01] est un environnement de développement de systèmes multi-agents dont le développement a débuté en 1993, dans le cadre de la thèse de **Guessoum** [Gue96]. DIMA a été utilisé pour développer plusieurs applications réelles (Ventilation artificielle, simulation de modèles économiques, etc.) par les auteurs mais également par d'autres personnes n'ayant pas participé à son développement (voir par exemple [Bou98]). Ces applications peuvent être des simulations, des résolutions de problèmes ou des systèmes de contrôle ayant éventuellement des contraintes temps réel. La première version de DIMA a été implémentée en Smalltalk-80 et a été ensuite portée en JAVA.

Plusieurs raisons justifient le choix de DIMA. D'abord, DIMA se caractérise par une approche résolument orientée Objet, l'objectif étant d'étendre les objets pour en faire des agents dotant des propriétés manquantes, en particulier l'autonomie, la proactivité, l'adaptation et la sociabilité. Également, l'architecture de la plate-forme proposée est modulaire. Un agent est composé de différents composants, chacun implémentant un comportement particulier. Ces comportements peuvent être réactifs ou cognitifs. L'orientation objet permet la composition et l'héritage des comportements. Les agents sont adaptatifs, grâce à un mécanisme de méta-comportement qui leur permet de choisir le comportement le plus adapté.

5.6.2 Aspect « organisation »

L'architecture du système est fondée sur le modèle client-serveur. L'application a été déployée dans un contexte Web, où nous utilisons un serveur Apache [Apa04] qui héberge et gère l'accès aux différentes connaissances et informations du système qui peuvent se situer localement ou à distance.

Les parties principales du système sont le serveur et les clients. Le serveur gère l'accès à différentes sources : base de cas, profils des joueurs, les configurations des activités, etc. C'est aussi un gestionnaire de données, car il peut répondre aux requêtes des utilisateurs (experts ou tuteurs) en les exécutants et en les redirigeant vers les bases contenant ces données. Pour cela, il redirige les connexions du client en se connectant aux ressources de données et en envoyant les réponses sous forme de pages HTML avec génération dynamique.

Le client représente la machine où le jeu se déroule. L'utilisateur se connecte au serveur à travers n'importe quel navigateur Web. Il peut charger, modifier et sauvegarder toutes

sortes de ressources du système.

5.6.3 Aspect « bibliothèque de classes »

Nous décrivons une partie du système en UML (Unified Modeling Language), en prenant en compte le fait qu'une partie des objets doit permettre de définir les cas et les comportements à observer, et qu'une autre partie doit permettre l'utilisation de ces objets dans le modèle agent correspondant.

5.6.3.1 Framework de description : *Descriptor*

Les structures utilisées pour la description des cas correspondent aux représentations données à la section 4.4.3. La figure 5.7 présente brièvement les classes les plus importantes du framework *Descriptor*. Il s'agit d'une description conçue pour décrire les cas en vue de les comparer à l'aide des fonctions *similarity* et *adaptationEffort* responsable du calcul de la similarité entre deux cas (cas cible et cas source) et l'effort d'adaptation en fonction des descripteurs du cas cible non assurés par le cas source.

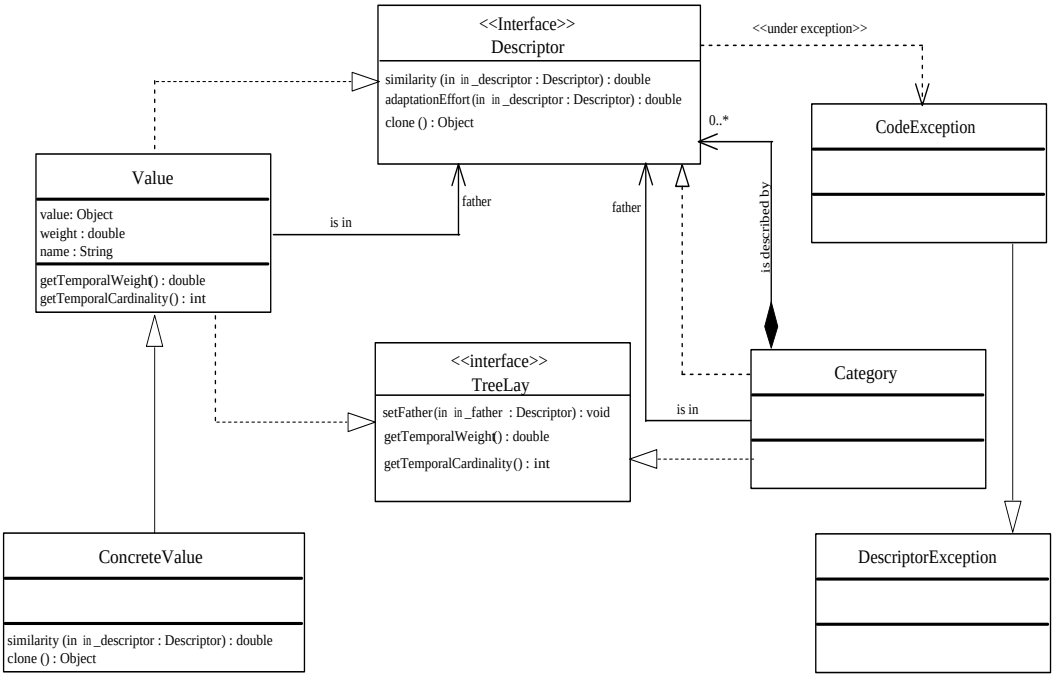

Fig. 5.6 – Diagramme de classe de la bibliothèque *Descriptor*

Nous avons adopté une structure hiérarchique dans la description des cas. La classe *Category* permet de distinguer les feuilles (représentant les cas) et les nœuds (représentant les *épisode généralisés*). Ainsi, elle offre des fonctions pour le parcours de l'organisation des cas. Les principales classes de ce framework sont *Descriptor*, *Value* et *ConcretValue*.

Classe Descriptor : Cette classe permet de décrire les cas et de les comparer avec les cas stockés dans la base de cas.

Classe Value : Cette classe est une classe abstraite qui contient la valeur du descripteur sous la forme d'un objet (classe *Object* de Java), ce qui nous permet de placer n'importe quel type d'object (Integrer, Double, String, Vector, etc.). Pour pouvoir utiliser la classe *Value*, il faut qu'elle soit spécialisée en classe *ConcretValue*.

Classe ConcretValue : Cette classe permet de définir la fonction de similarité entre deux cas grâce à la fonction *similarity*.

5.6.3.2 Framework raisonnement : *Adapter*

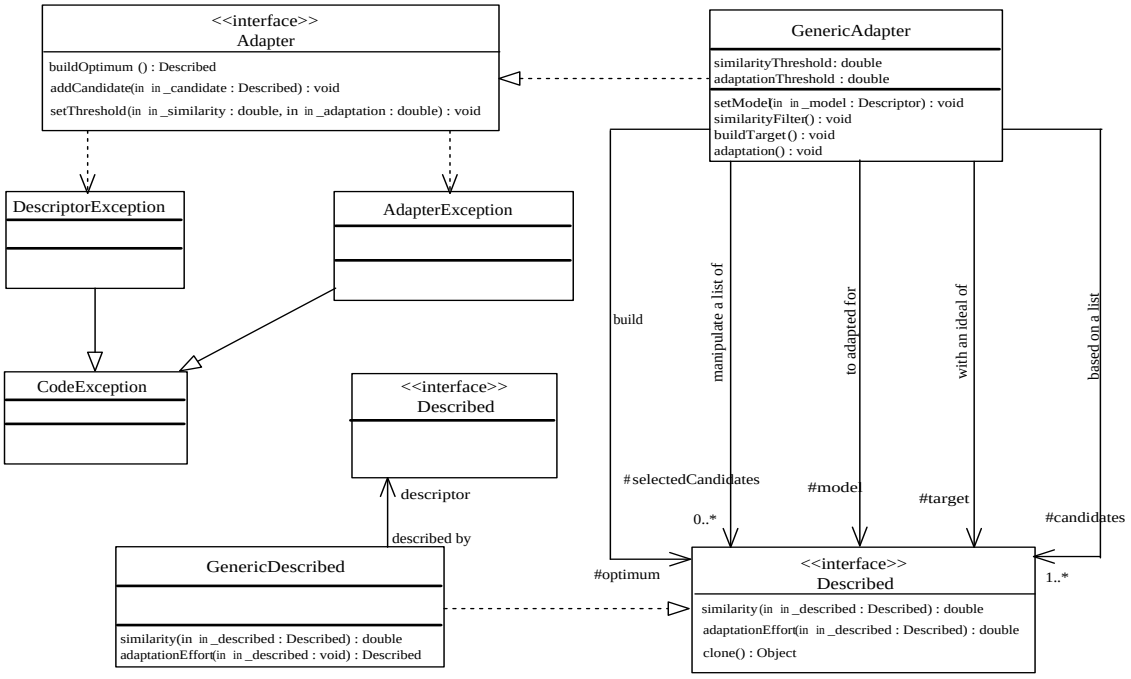

FIG. 5.7 – Diagramme de classe de la bibliothèque *Adapter*

Ce framework permet de sélectionner quelques cas de la base de cas (CandidatesCases) et de les adapter à la description du cas cible. Pour cela, il utilise des seuils de sélection (à maximiser) et d'adaptation (à minimiser).

La classe *GenericAdapter* constitue la classe principale de ce framework. Elle permet de construire un cas répondant aux exigences de la description du cas cible. Pour cela, elle sélectionne une liste de cas sources répondant aux critères de similarités, puis de voir les descripteurs de cas cible qui n'ont pas été satisfaits (descripteurs non existants ou valeurs non conformes) par le cas candidat et faire des modifications au niveau des activités du cas cible.

Initialement, le cas cible n'est autre qu'une description. Dans le cadre du PROJET AUTISME, il représente le profil de l'enfant et les objectifs éducatifs que l'utilisateur souhaite atteindre, c'est alors un objet de la classe *Descriptor*. À partir de cette description, cette classe permet de créer un cas cible avec la fonction *buildTarget()*. Le cas cible sera utilisé par la suite dans la phase d'adaptation pour la génération des scénarios. Pour cela, la classe *GenericAdapter* utilise les seuils définis dans les variables *similarityThreshold* et *adaptationThreshold*.

5.6.3.3 Framework d'observation : *Observer*

Le modèle de description des comportements basé sur les frames (voir chapitre 3) a été utilisé dans l'implémentation du framework *Observer*. La figure 5.8 montre les classes les plus importantes de ce framework dans le cadre du jeu *Coucou caché*. Il s'agit des comportements à observer durant le déroulement de l'activité. Deux catégories de comportement se présentent :

- La catégorie des comportements détectés par observation des actions du joueur sur le jeu. Il s'agit des actions effectuées sur les objets du jeu *Coucou caché*. Toutes les classes de cette catégorie hérite de la classe *Action* qui surveille toutes les actions du joueur sur le jeu ;
- La catégorie des comportements détectés par analyse des flux d'évènement par un système de vision[24]. Toutes les classes de cette catégorie héritent de la classe *Event*, responsable de surveiller les évènements provenant du système de vision.

La classe *ActivityManager* permet de déclencher les actions délibérées par l'agent de décision à chaque fois qu'un des comportements décrit par ces deux catégories se présente. Dans ce qui suit nous présentons les classes de chaque catégorie.

Classes d'observation par action logicielle : L'observation par action logicielle consiste à capter les actions du joueur sur le jeu afin de détecter les actions qui ont un intérêt particulier sur le déroulement du jeu. Dans le cas du jeu *Coucou caché*, nous avons recensé quatre comportements : *cadre touché*, *grosse boule touchée*, *boule curseur cachée derrière une grosse boule*, et *les séquences des grosses boules empruntées par la boule curseur*. Ces quatre comportements sont assurés par les classes *FrameTouched*, *BallHidden BallTouched ActionSequence*.

Classes d'observation par vision : Trois classes sont responsables de l'observation du comportement du joueur dans cette catégorie. Il s'agit des classes *TemporalAvoid VisualAvoid* et *Break*. Ces classes permettent d'observer respectivement trois comportements du joueur lors de la session : comportement de rupture, d'évitement temporel et visuel. Pour cela, en se basant sur la formalisation que nous avons adopté au chapitre 3, ces classes permettent d'analyser le flux d'évènements générés par le système de vision **FaceLab** et de réagir en temps réel lorsqu'un comportement est détecté.

[24]Il s'agit du système **FaceLab** - voir l'annexe E

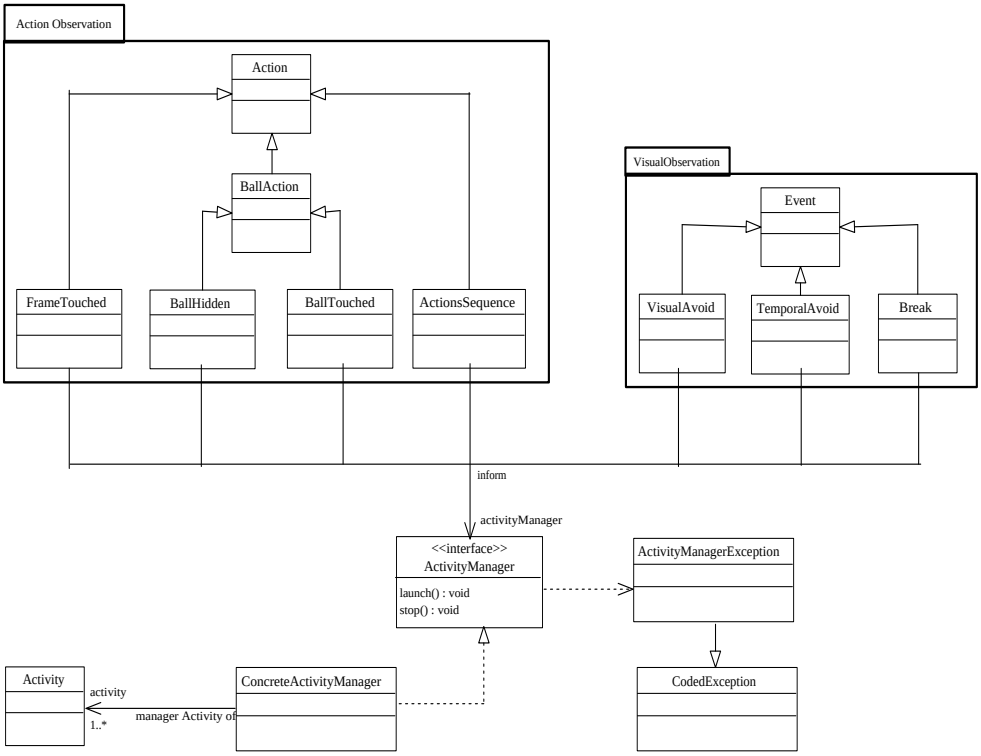

FIG. 5.8 – Diagramme de classe de la bibliothèque *Observer*

Les classes des deux catégories ont uniquement la charge de détecter ces comportements, quand à la réaction du système face à ces comportements, elle est assurée par l'agent de décision. Il s'agit dans ce cas, de lancer une musique familière à l'enfant dans le cas de rupture, d'une image famillière dans le cas d'un évitement visuel.

5.7 Le système vu par son interface

Le plus souvent l'interface utilisateur désigne l'interface joueur, c'est-à-dire celui qui utilise le système pour s'évoluer. Nous considérons que le joueur est un cas particulier de l'utilisateur. Étant donné que le système a pour utilisateurs non seulement des joueurs mais aussi des experts et des tuteurs. Le terme général utilisateur est plus approprié pour désigner l'ensemble experts, tuteurs et joueurs.

L'interface utilisateur est le vecteur de communication entre l'utilisateur et le système, c'est-à-dire, dans notre cas, entre les utilisateurs vus comme des agents humains et les agents au SMA. Nous présentons, dans cette section, les interfaces de l'utilisateur expert et du tuteur. Quant aux joueurs, leur interface n'est autre que l'interface du jeu (voir l'annexe D).

5.7.1 Interfaces de l'expert

Afin d'avoir une interface qui offre des fonctionnalités toujours disponibles nous avons adopté pour l'interface expert l'utilisation de deux cadres (figure 5.9), le premier présente toujours la barre de navigation d'une interface à l'autre. Les interfaces sélectionnées de la barre d'outils seront affichées dans le deuxième cadre.

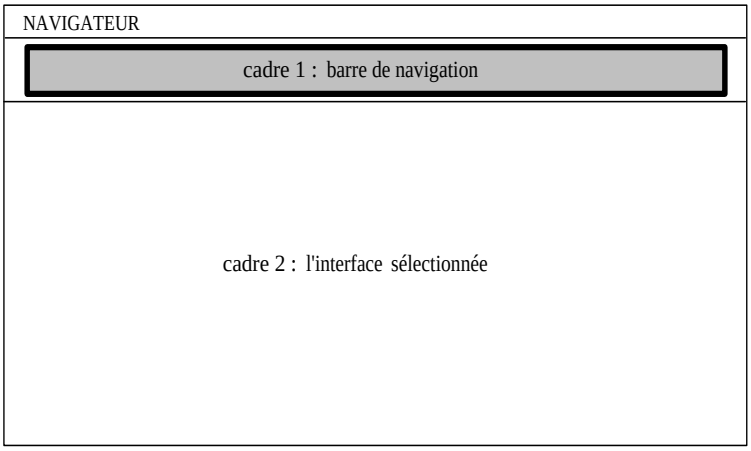

Fig. 5.9 – Interface experts

La barre de navigation offre quatre interfaces : *Cas*, *Activité*, *Descripteurs* et *Joueurs*.

L'interface Cas : Cette interface permet à l'expert de définir les cas (figure 5.10). Il s'agit de définir les activités à présenter au joueur afin d'atteindre les objectifs, définis dans la partie but de l'interface, dont le profil est proche de la description du cas.

L'interface Activités : Cette interface permet à l'expert de définir les activités (figure 5.11). Rappelons qu'une activité est une instance du jeu. Il s'agit donc de définir les comportements des objets du jeu. Ceci est assuré par les valeurs que peut prendre chaque paramètre. Les objectifs de l'activité ont été ajoutés afin d'intervenir dans le processus d'adaptation (en particulier l'adaptation globale) qui consiste à chercher des activités qui répondent au mieux à la description du cas cible.

L'interface Descripteurs : Les descripteurs sont utilisés dans la phase de remémoration du processus de décision. La phase de remémoration consiste à calculer la similarité entre le cas cible et les cas sources en utilisant la fonction numérique ϕ. Les types des attributs des descripteurs que nous avons retenus sont présentés dans le tableau 4.2.

L'interface du descripteur permet à l'expert de définir les distances entre les différentes valeurs des descripteurs dans le cas où le type d'attribut est un *vecteur*. La figure 5.12 défini les distances entres les valeurs du descripteur *Niveau* dont les attributs sont définis dans le vecteur <*bas, moyen, haut*>.

Interface Joueurs : Cette interface permet à l'expert de définir les profils des joueurs. On trouve des informations générales, des préférences, des compétences et l'histoire contenant

la trace d'exécution.

Interface comportement : Cette interface permet à l'expert de définir les comportements qui ont un intérêt particulier dans le déroulement du jeu. La figure 5.2 présente une interface représentant certains comportements du jeu *Coucou caché*.

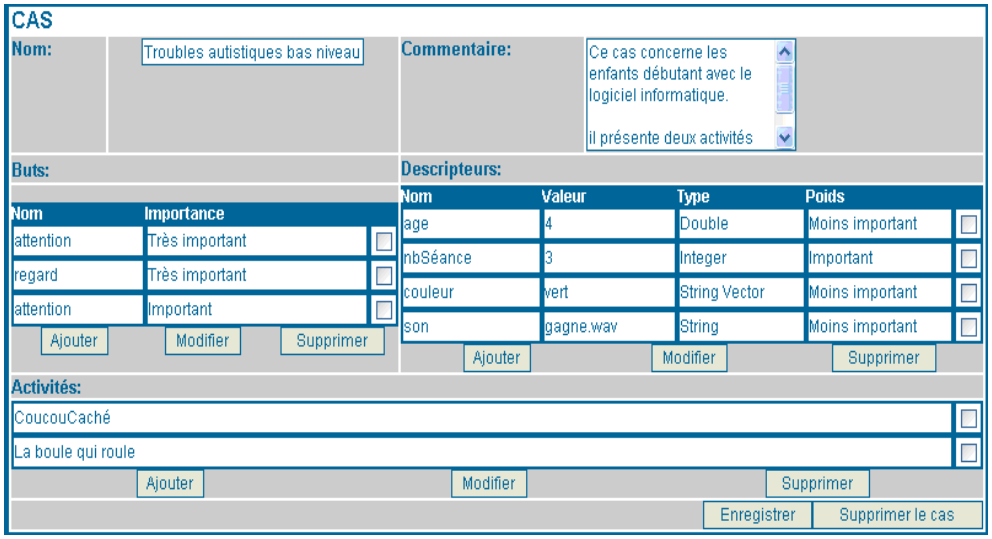

FIG. 5.10 – Fenêtre de définition des cas

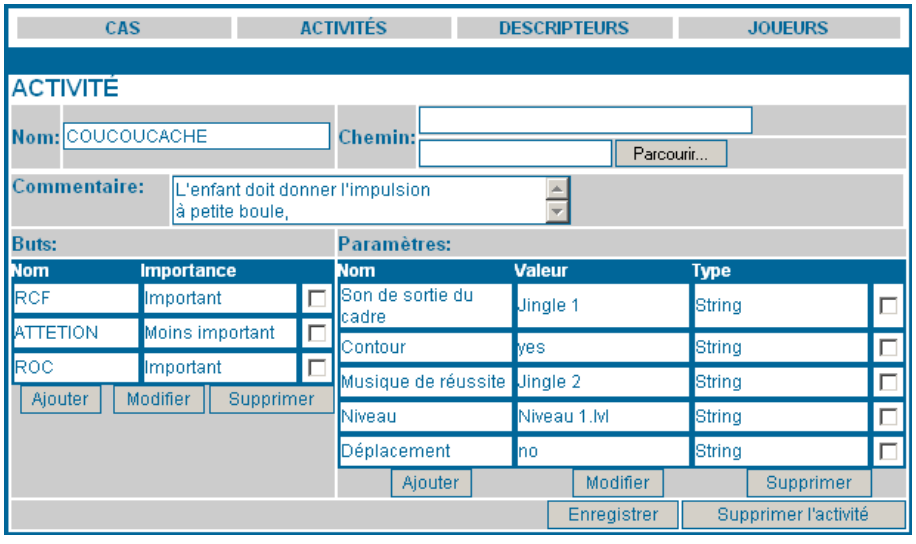

FIG. 5.11 – Fenêtre de spécification des activités

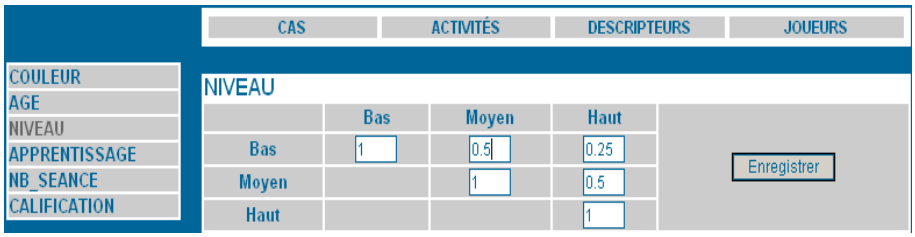

Fig. 5.12 – Définition des distances entre les valeurs de l'attribut *Niveau*

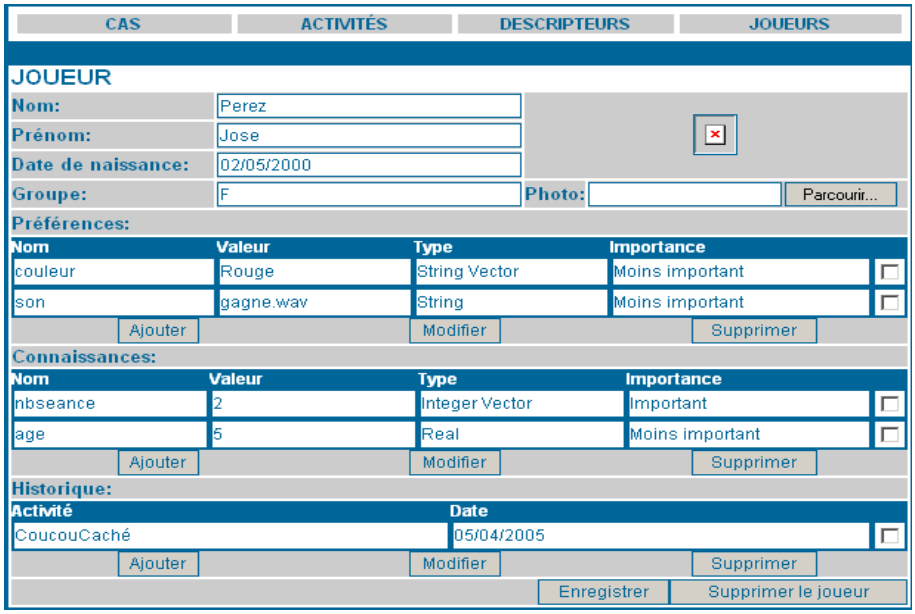

Fig. 5.13 – Fenêtre de définition du profil de l'enfant

5.7.2 Interfaces du tuteur

Le tuteur a la charge de charger le profil du joueur et de définir les objectifs éducatifs à atteindre. Suite à ces informations le système génère un scénario d'activités adaptées. L'interface présentée dans la figure 5.14 représente *la fiche joueur* qui permet à l'expert de spécifier ces informations.

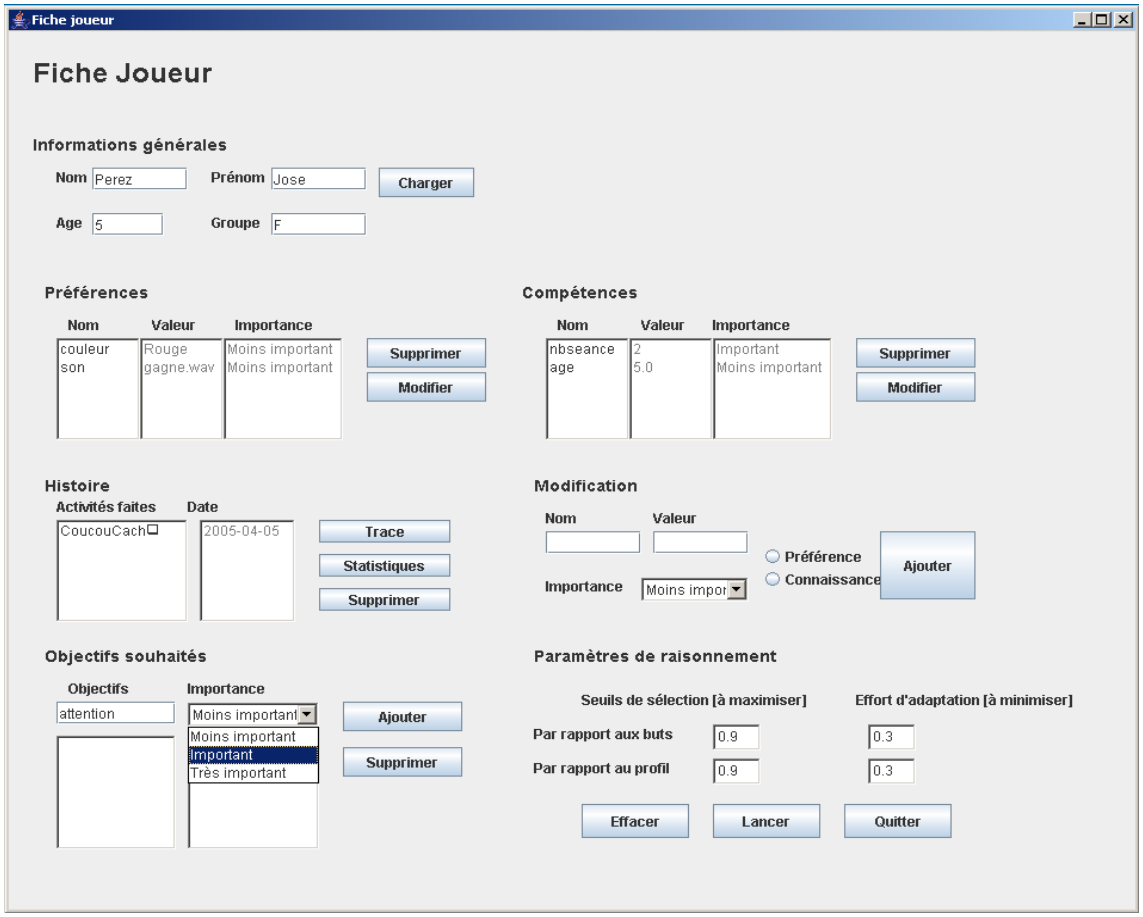

Fig. 5.14 – Fiche joueur

5.8 Méthodologie d'utilisation du système

Dans cette section, nous présentons les différentes phases à entreprendre lors de l'utilisation du système. Nous entendons par utilisateur ici, la personne qui a la charge d'assister le joueur (l'enfant autiste) pendant la session du jeu. Il s'agit de l'expert, le tuteur ou toute autre personne proche de l'enfant.

La figure 4.6 montre la fiche joueur qui permet de réaliser les différentes phases du

scénario d'utilisation[25] :

- L'utilisateur se connecte au serveur et charge le profil du joueur. L'utilisateur ne nécessite pas d'informations particulières (login ou mot de passe) pour se connecter au serveur. Il suffit juste d'introduire le nom et le prénom du joueur et cliquer sur charger. Par cette action, toutes les informations caractérisant le profil seront chargées : informations générales, préférences, compétences ainsi que l'histoire du joueur. Il peut ajouter ou modifier les compétences et les préférences du joueur. Comme il peut aussi visualiser la trace d'exécution sous différentes formes (statistique ou animation graphique) ;
- Ensuite, l'utilisateur spécifie les objectifs éducatifs qu'il veut faire atteindre au joueur. Chaque objectif possède une valeur d'importance parmi les trois valeurs possible (moins important, important, très important). Une fois ces informations entrées (le profil et les buts éducatifs), un cas cible est créé. Il s'agit d'une description conforme au contexte d'application que nous avons adopté. Donc, la description est sous forme <attribut, valeur> dans laquelle on trouve des informations liées au profil du joueur et des objectifs éducatifs à atteindre ;
- L'utilisateur défini les seuils de raisonnement. Il s'agit des valeurs entre 0 et 1 qui servent dans le processus de décision. Elles regroupent les seuils de remémoration pour le filtre de sélection (à maximiser) et le filtre de l'effort d'adaptation (à minimiser). Nous avons distingué les seuils liés au profil du joueur et ceux liés au objectifs de la session. Ce qui permet de donner plus de lisibilité à l'expert en terme de précision des valeurs à entrer ;
- Le cas cible ainsi que les seuils de raisonnement sont transmis par message au serveur. Le message est intercepté par l'agent de décision. Ce dernier génère un scénario adapté à la situation actuelle du cas cible. Après la phase de raisonnement de l'agent de décision, ce denier envoie à l'utilisateur le scénario d'activités adapté à la description du cas cible ainsi qu'un rapport de raisonnement ;
- Le rapport de raisonnement, présenté dans la figure 5.15, contient le cas source sélectionné avec les valeurs de la similarité et de l'effort d'adaptation calculé entre les cas cible et source, ainsi que l'adaptation faite au niveau des activités du scénario du cas source sélectionné. Il s'agit des activités ajoutées ou enlevées du scénario dans le cas d'une adaptation globale. Dans le cas d'une adaptation locale, il s'agit des changements des comportements des objets des activités du scénario ;
- L'utilisateur lance les activités, s'il est satisfait du résultat généré par le système ;
- Durant le déroulement du jeu, des instances d'agents observateurs suivent en permanence les actions du joueur. Chaque comportement incohérent du joueur vis-à-vis des stimuli des activités sera détecté par les agents observateurs et signalé à l'agent de décision qui procède à mettre à jour le scénario de manière à ce qu'il soit adapté à la situation par le déclenchement d'un nouvel épisode de raisonnement ;
- À la fin de la session, l'utilisateur valide, dans la dimension éducative, la mémorisation du cas s'il constate que le scénario du nouveau cas ainsi généré présente des résultats éducatifs satisfaisants. Dans ce cas, le système procède à l'évaluation dans la dimension informatique la mémorisation. Pour cela, il doit comparer le nouveau

[25]Nous supposons que les phases de calibration du système de vision **FaceLab** sont faite

cas avec les cas sources de la base de cas en utilisant les seuils d'adaptation définis par l'utilisateur.

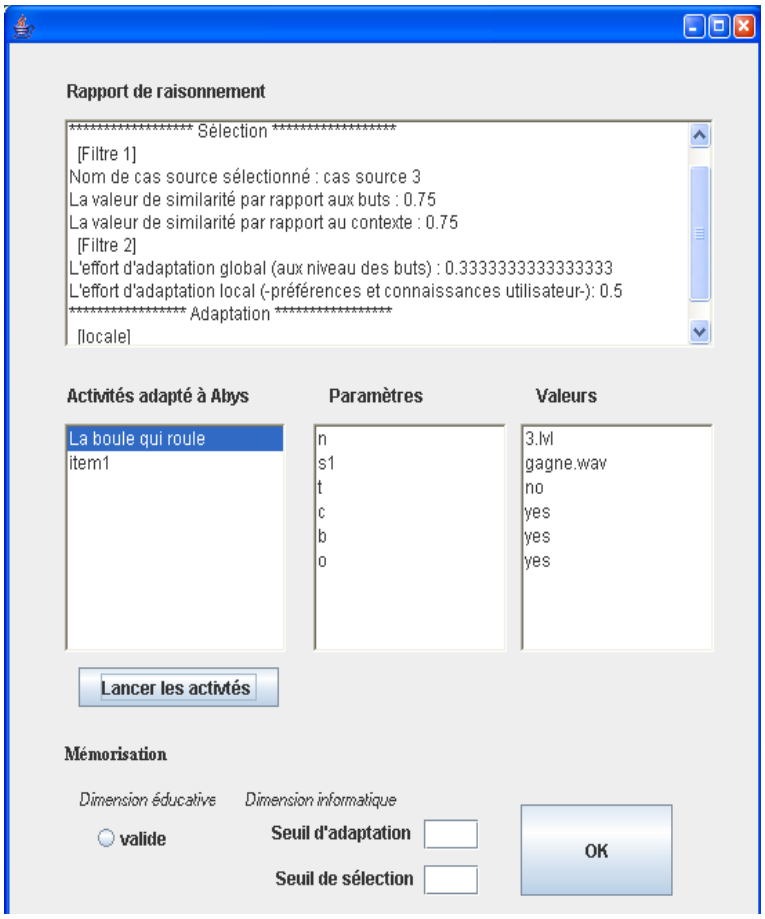

FIG. 5.15 – Rapport du processus de raisonnement

5.9 Expérimentations

Une étude a été menée avec nos partenaires du secteur médical pour valider l'approche que nous avons développée. Il s'agit d'évaluer si l'approche proposée permet d'améliorer les résultats en termes de stratégie éducative destinée à des enfants autistes.

Le programme éducatif, administré par une équipe de pédopsychiatre, cible spécifiquement les fonctions problématiques de l'enfant. En particulier, l'attention, la perception, la motricité et le contact. Des outils d'évaluation, aussi bien assurés par le système que l'observation directe par les psychologues, ont été utilisés pour évaluer l'évolution les enfants dont les déficits primaires concernent l'isolement, l'évitement du regard et certains stéréotypies [26].

[26] Gestes étranges, répétés et vides de sens, chargés de soulager l'enfant de son angoisse, comme par

Pour participer à l'étude, les enfants ont dû venir au centre pour jouer au moins une fois par semaine. La session du jeu dure entre 5 à 20 minutes selon la capacité de l'enfant défini dans son profil. Le nombre d'enfants suivi à l'hôpital est de 50. Les observations et les suivis effectués dans le cadre de cette expérimentation se portaient sur trois enfants : *Dylan*, *Xavier* et *Eliza*. Dans cette section, nous décrivons l'environnement dans lequel les sessions des jeux se déroulaient ainsi que le bilan évolutif de ces trois cas suivis.

5.9.1 Description de l'environnement

L'étude a été entreprise au centre de service de pédopsychiatrie de l'hôpital de La Rochelle dans une de leurs salles de thérapie. La salle est de surface 3×2 m^2 avec un ordinateur situé dans le coin afin qu'il y ait moins d'objets qui perturbe l'enfant durant la session. L'ordinateur est sans clavier avec écran tactile réglable de manière à ce que l'enfant soit confortable dans la session de jeu.

Les enfants sont soutenus par une personne familière (le tuteur). Le rôle de ce dernier est défini dans la section 5.3.2.2. Un observateur est impliqué également dans l'expérimentation, son rôle consiste à observer l'interaction enfant-jeu afin d'évaluer l'évolution de l'enfant et les réactions de l'outil.

5.9.2 Bilan évolutif des enfants

Les trois cas que nous présentons dans cette section ont été suivi par Mr. **D. Lambert**, psychiatre des hôpitaux et directeur du service du pédopsychiatrie du Centre Hospitalier Marius Lacroix à La Rochelle. Il s'agit de présenter les différentes phases d'évaluations effectuées par le psychiatre. Dans chaque phase, l'enfant manipulait les jeux développé dans le cadre du projet. Certains de ces jeux sont présentés en annexe D.

Étude du cas 1

Nom de l'enfant : Xavier GUIZIOU, Date de naissance : 28/11/99

Évaluation antérieures : Xavier a attiré l'attention du pédiatre pour son retard de la communication langagière et sociale, par ses jeux répétitifs et par ses troubles fonctionnels en particulier des présentations d'éveil alors même qu'il était seul dans une chambre dans l'obscurité. Le bilan qui a été pratiqué a confirmé l'existence d'un autisme sévère. Les symptômes dominants de Xavier sont :

- évitement regard
- évitement changement
- évitement du contact

exemple remuer les doigts devant le visage, bouger les épaules, etc.

Évaluation du janvier 2004 : Xavier maintient un contact avec la sensorialité selon un mode d'adhésion au réel de nature autistique. Les quelques mots associés à des gestes d'accompagnement paraissaient avoir pour objet d'éviter la reconnaissance de la perte et/ou du changement. Aucune signification n'était perçue qui permettait de relier ces séquences entre elles.

Évaluation du 8 octobre 2004 : L'enfant à révéler une meilleure communication par le regard et une meilleure tenue du corps. Les objets et les situations sont mieux reconnus, mais ils restent toujours mal reliés les uns aux autres.

Après cette évaluation, les objectifs établis par les médecins consiste à associer les situations éparses dans une trame signifiante l'inscrivant dans une histoire.

Évaluation du 30 Septembre 2005 : Dans certains cas, Xavier peut manifester une attention soutenue avec des réussites intéressantes dans le domaine du graphisme et du langage. Mais, il peut perdre le contact et se réfugier dans une rêverie apragmatique.

Étude du cas 2

Nom de l'enfant : Eliza BOGDANICK, Date de naissance : 9/11/95

Évaluation antérieures : Eliza s'inscrit dans une pathologie des troubles envahissants du développement ce qui, dans la classification française des troubles mentaux, correspond à une dysharmonie évolutive de personnalité. Elle donne l'impression d'une enfant "écorchée vive" qui veut agir sur l'environnement à sa guise, maîtriser la relation avec les adultes et avec les enfants. Quand elle se heurte aux limites tangibles et frustrantes posées par l'entourage, elle manifeste par des colères, des cris, des pleurs qui ne peuvent trouver de sédation que par la contention physique.

Les apprentissages sont très limités et en dehors des activités perceptives qui permettent des réussites assez satisfaisantes (assemblage de formes diverses du jeu Pictécran - voir la section D.2), elle n'accède pas aux activités de type scolaire et donc pas à la lecture/écriture.

La présentation des jeux a permis son accès à l'utilisation des signes (pictogrammes) reconnus par leur signification et agencés sur l'écran comme s'il s'agissait d'écrire une phrase pictographique. Les progrès générés par ce mode de prise en charge se poursuivent et certains estiment raisonnable le projet d'accès à un déchiffrage élémentaire.

Étude du cas 3

Nom de l'enfant : Dylan MONTERO, Date de naissance : 21/03/2000

Évaluation antérieures : Dylan se présente comme un enfant dont l'autisme se caractérise essentiellement par une résistance aux changements, une agitation qui amène le pédiatre consultant à parler d'hyperactivité et enfin, par des stéréotypies marquées en

particulier lorsqu'il est confronté à une situation nouvelle. Les symptômes dominants de Dylan sont :

- Isolement
- Stéréotypies
- Hyperactivité

Au sein de l'unité, elle manifeste une activité rituelle et quelques fois une opposition aux incitations de l'entourage. Son regard se fixe difficilement sur les personnes mais elle lui est possible ponctuellement de croiser le regard d'autrui dans des situations où elle est apaisé et très attentif à ce qui se dit alentour.

5.10 Conclusion

Nous avons présenté, dans ce chapitre, différents aspects du système développé dans le cadre du PROJET AUTISME en partenariat avec le service pédopsychiatrie de l'hôpital de La Rochelle. Ce travail a été pour nous une occasion pour mettre en œuvre et valider les différents principes et approches proposés dans les chapitres précédents.

Nous avons ainsi pu implémenter des agents observateurs pour l'analyse de comportements à différents niveau d'analyse, de l'action au comportement dans un processus d'interprétation. Nous avons également utilisé le raisonnement à partir de cas pour contrôler l'exécution des jeux. Cette technique nous offre un modèle permettant, d'une part, à l'expert d'intervenir d'une manière rapide dans la définition de ces connaissances. D'autre part, un apprentissage du système qui est un véritable moyen pour assurer l'adaptabilité du système face à des nouvelles situations.

Les expérimentations faites sur les trois cas (de *Dylan*, *Xavier* et *Eliza*) que nous avons présentés ont conduit à une certaine évolution chez les enfants. Cette évolution a été aussi constatée par les parents des enfants comme le témoigne Mr. **D. Lambert**, psychiatre des hôpitaux et directeur du service du pédopsychiatrie du Centre Hospitalier Marius Lacroix à La Rochelle.

Chapitre 6

Conclusion et perspectives

L'exécution adaptative par observation et analyse de comportement est un domaine promoteur pour les applications interactives qui incluent l'utilisateur humain. Les avantages de l'analyse de comportements tels que la personnalisation de l'application à son utilisateur, l'évaluation de l'application par l'expert, la compréhension du comportement de l'utilisateur vis-à-vis des objets d'intérêts de l'application et l'exécution adaptative sont en général intéressants pour améliorer le développement des applications interactives.

Dans cette thèse, nous avons essayé d'apporter notre contribution à la problématique de l'exécution adaptative par observation et analyse de comportements. La première partie de notre travail a concerné l'étude des approches d'analyse de comportements. Nous considérons deux axes qui différencient l'ensemble d'approches menées : le type de représentation et les techniques établies pour identifier des comportements. Nous avons utilisé ces deux critères pour analyser les différentes méthodes qui existent dans la littérature. L'étude a montré l'existence de différents problèmes lors de la mise en place de ces approches. En particulier, la notion du *comportement* qui est différente ou incomplète par rapport à notre définition dans le cadre de la problématique de l'exécution adaptative. Notre définition, basée sur un fondement théorique issu de la psychologie, a la spécificité de prendre en compte les objets d'intérêt de l'application avec lesquelles l'utilisateur est en interaction. Ainsi, à partir des observations effectuées sur les actions de l'utilisateur, les éléments de contrôle et également le suivi du regard et le mouvement de la tête, nous construisons des formes qui constituent les éléments de base pertinents pour le comportement considéré. Les formes, considérées comme des affordances comportementales, sont analysées par la suite par des agents observateurs dans un processus d'interprétation. Ce dernier consiste à donner des points de vue à différents niveaux d'analyse dans une méthodologie incrémentale.

La seconde partie de ce travail a concerné l'étude des systèmes de scénarisation et de contrôle d'exécution. Il s'agit d'établir des modèles permettant d'une part, de générer des scénarios adaptés à la situation. D'autre part, contrôler le déroulement du scénario en tenant compte le comportement de l'utilisateur. L'étude des approches proposées dans la littérature, notamment, les systèmes à base de connaissances, les systèmes à base

de procédures et les systèmes de classeurs, par leurs caractères réactifs, présentent une difficulté de contrôle qui se manifeste dans la prévoyance de l'état du système après exécution des séquences d'actions générées par ses systèmes. D'autre part, ces approches ne nous semblent pas appropriées pour l'intégration de fonctions de scénarisation de la boucle perception-scénarisation-exécution. Le storytelling interactif a montré un intérêt pour répondre à la problématique du contrôle d'exécution par l'exploitation des histoires comme vecteurs de connaissance. Il s'agit de générer des scénarios interactifs en fonction de l'état de l'environnement. Néanmoins, ce courant de recherche ne met pas l'accent sur le comportement de l'utilisateur en interaction avec les objets d'intérêt de l'application.

À partir de cette étude, nous avons défini un modèle fondé sur le raisonnement à partir de cas permettant la génération répondant à la description du profil de l'utilisateur ainsi que ses besoins. Le raisonnement à partir de cas permet la résolution de problèmes s'appuyant sur la réutilisation de cas source, stockées dans une base de cas, pour résoudre des cas cibles. Toute nouvelle expérience peut être mémorisée dans la base de cas, la rendant immédiatement disponible pour les problèmes futurs ce qui permet un apprentissage du système. Notre stratégie de contrôle consiste à déclencher un épisode de raisonnement à chaque fois que les agents observateurs détectent un comportement qui porte un intérêt particulier. Il s'agit des cas où le comportement de l'utilisateur n'est pas cohérent par rapport à l'activité en cours. L'avantage de notre stratégie est de diminuer l'effort lié à l'acquisition des connaissances, de calcul dans le processus de raisonnement et de l'apprentissage du système :

- Les connaissances sont considérées comme des expériences (cas) définies par l'expert qui sont réutilisées pour résoudre des cas concrets ;
- Le calcul est beaucoup moins important que dans les approches réactives puisqu'il s'agit essentiellement d'un raisonnement analogique ;
- L'apprentissage permet de doter le système de contrôle de capacité d'adaptation aux nouvelles situations ce qui répond aux exigences de l'exécution adaptative.

Nous avons enfin présenté, dans la dernière partie, notre système qui consiste à assurer une exécution adaptative par observation et analyse du comportement des enfants autistes dans un cadre éducatif par manipulations interactives des outils informatiques. Il s'agit d'un système proposant des séquences de jeux éducatifs et thérapeutiques suffisamment précis selon les capacités et les caractéristiques de chaque enfant. Ce système est intégré dans le Projet Autisme et a constitué un cadre d'application et de validation de notre contribution. Notre méthode d'analyse de comportements a prouvé son efficacité dans l'identification de certains comportements. De même, notre stratégie de scénarisation et de contrôle a permis d'augmenter de manière significative la cohérence du comportement de l'utilisateur et le comportement des objets du jeu.

Il est bien entendu indispensable de poursuivre ces recherches, à la fois pour valider le fait que le modèle puisse tenir compte d'autres comportements, ainsi pour valider son efficacité sur le terrain. C'est un des aspects que nous comptons améliorer dans nos prochains travaux.

Nous pensons que la thèse de l'exécution adaptative par observation et analyse de

comportements défendue dans ce mémoire peut être utilisée dans d'autres domaines applicatifs. En particulier, La future direction de l'interaction sociale entre les robots et les êtres humains. Il s'agit de mettre en place des systèmes robotiques qui ne planifient pas uniquement les actions à exécuter pour atteindre leurs objectifs, mais également tenir compte de la dynamique de leur environnement. En outre, le domaine des jeux vidéo constitue une application potentielle dont l'exécution adaptative peut occuper une importante. En particulier, pour le comportement des personnages du jeu. Il s'agit ici de développer des jeux dont le scénario n'est pas prédéterminé à priori comme le scénario d'un film dont la fin est fixe quelque soit le comportement de ceux qui le voient, mais plutôt dynamique dont on connaît la trame scénaristique mais pas le chemin exact qui mène vers la fin du jeu, ce qui renouvelle toujours l'intérêt du jeu et offre plus de diversité dans le comportement. Parmi d'autres domaines applicatifs, on peut aussi citer : les systèmes d'apprentissage, les systèmes de contrôle industriel, etc.

Plusieurs pistes scientifiques inhérentes à la problématique de l'exécution adaptative peuvent être considérées. Parmi celles-ci, nous envisageons axer nos futures recherches sur différents points intervenant dans les trois parties traitées dans cette thèse. Nous pouvons citer :

- *Modification dynamique du profil* : Il s'agit de mettre à jour le profil à chaque évolution de l'utilisateur. Rappelons que le profil contient, des informations générales, préférences, compétences et l'histoire. Dans l'approche que nous avons proposée, il y a que l'histoire que s'évolue d'une manière dynamique par l'ajout de la trace d'exécution après chaque utilisation du système. Néanmoins, les composants de préférences et de compétences sont statiques tant que l'expert ou le tuteur ne les change pas. Une mise à jour de ces composants peut s'avérer primordiale dans l'exécution adaptative dans la mesure où le processus de décision génère les scénarios en fonction des informations contenues dans ces composants. Cette mise à jour peut être faite à partir de l'analyse du comportement de l'utilisateur afin de détecter des éventuels changements de préférences ou compétences de l'utilisateur ;
- *Construction d'objectifs* : le système doit être apte à construire automatiquement les objectifs en analysant et classifiant le profil et les comportements de l'utilisateur. Il s'agit d'identifier une description générale du travail en cours et définir les objectifs que l'utilisateur doit atteindre. Ceci doit forcément dépendre du domaine d'application ;
- *Vérification des propriétés* : La génération des scénarios à partir de cas peut être l'origine de certaines incohérences. Il s'agit des propriétés que l'expert souhaite qu'elles soient respectées dans le processus de décision du système afin de garantir un déroulement cohérent du jeu. Il s'agit par exemple de propriétés de sûreté (exprime que sous certaines conditions, un évènement ne peut jamais se produire) ou de vivacité (exprime que sous certaines conditions un évènement se termine), etc. Une vérification des ces propriétés s'avère nécessaire dans ces cas. Dans [CS05] [CPE05], une approche basée sur la logique linéaire pour la modélisation et la validation de l'interaction dans le cadre des jeux est proposée ;
- On peut aussi envisager de formaliser la description des cas avec des modèles formelles plus rigide telles que les travaux sur les logiques de descriptions [dLN05] qui semblent

bien adaptés pour le type de représentation des cas que nous avons proposée. Ces travaux ont l'avantage d'avoir une syntaxe et une sémantique bien définie. D'ailleurs, cette formalisation permet de faciliter la vérification des propriétés évoquées au-dessus.

Annexe A

Paradigme des systèmes multi-agents

A.1 C'est quoi un agent ?

La définition d'un agent est aussi emblématique que l'a été (et l'est toujours) la question de ce qu'est l'*intelligence* pour l'intelligence artificielle. Le problème générale posé peut être formulé ainsi « *comment définir et modéliser un agent ?* ». La question est vaste et a, en conséquence, généré de nombreuses réponses dont nous présentons les lignes générales.

En dépit de l'absence d'une définition consensuelle de ce qu'est un agent, nous retiendrons la définition minimale proposée par **Jacques Ferber** et **Malik Ghallab** [JG98] : « *On appelle agent une entité physique ou abstraite qui est capable d'agir sur elle-même et sont environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de sa connaissance et de ses interactions avec les autres agents.* »

Par ailleurs, les auteurs dans [WJ94] proposent deux définitions : l'une dite large et l'autre plus restrictive. La première, similaire à celle de Ferber et Ghallab, identifie les propriétés nécessaires d'autonomie, de capacités sociales (communication), de réactivité et de proactivité[27].

De ces définitions générales, deux courants antagonistes ont émergé conduisant à distinguer les agents *réactifs* des agents *cognitifs*. Ces deux architectures seront présentées dans la prochaine section.

[27] c'est-à-dire que l'agent peut décider d'agir de lui-même et pas seulement en réponse à un stimulus

A.2 Architectures d'agents

A.2.1 Les agents cognitifs

L'architecture des systèmes basés sur les agents cognitifs est aussi appelée architecture *horizontale*. Un SMA cognitif comprend un petit nombre d'agents de forte granularité : chaque agent possède des capacités de raisonnement sur une base de connaissances, et sur les représentations symboliques de l'environnement dans lequel il évolue. Les agents cognitifs sont de très haut niveau et chacun d'entre eux peut résoudre des problèmes très complexes.

Parmi les architectures horizontales proposées, on trouve le célèbre modèle BDI (Belief - Desire - Intention) de **Michael Bratman** qui satisfait les désirs ou buts, se traduisant par des intentions (plans d'action intermédiaires) à partir des croyances (connaissances supposées vrai d'un agent sur lui-même et son environnement) [BIP88]. Le comportement d'un agent ici résulte donc de l'influence de plusieurs « attitudes » vis-à-vis du monde qui l'entoure. Les architectures IRMA (Intelligent Resource-bounded Machine Architecture) [HS96], PRS [GI89], COSY [BS92], GRATE* [Jen93], [CSE03] etc., sont issues de cette approche.

A.2.2 Les agents réactifs

L'architecture des systèmes basés sur les agents réactifs, à contrario de l'architecture cognitive horizontale, correspond à une architecture dite *verticale*. Dans un SMA réactif, à l'instar des sociétés d'insectes, le comportement complexe émerge de la coexistence de la coopération des comportements simples. Les systèmes à base d'agents réactifs sont composés d'un grand nombre d'agents de faible granularité dont le comportement est simple. Il s'agit de décomposer l'activité générale en un ensemble de tâches simples affectées à chaque agent. La résolution du problème résulte de l'interaction de ces agents. Le fonctionnement de ces derniers, est similaire à celui d'un objet dans le sens plus « primitif » du terme c'est-à-dire qu'il obéit à la loi de stimuli/réponse ou règles de production.

La principale différence entre les SMA à base d'agents réactifs et les systèmes centralisés, tels que les systèmes experts, réside dans le fait que l'intelligence demeure dans l'univers et pas dans le système désincarné. La résolution des problèmes alors émerge de l'interaction des agents.

Cette approche est apparue avec l'architecture de *subsomption* de **Rodney Brooks** [Bro86], dans le domaine de la robotique. Par la suite, d'autres architectures alternatives, comme celle de [Ste94] ou encore de [AC87] basées sur des règles et automates sont apparues.

Annexe B

LevelEditor

LevelEditor[28] dont l'interface est présentée dans la figure B.1 permet de créer des activités du jeu *Coucou Caché*. Il s'agit d'un logiciel qui permet à l'expert de faciliter l'écriture du jeu (définir les objets du jeu et leurs comportements) de manière à fournir un environnement de développement souple et confortable afin d'assurer une exécution adaptative. Ce logiciel contient deux parties, *Un entête* et *un environnement de travail*.

L'entête permet à l'expert de définir quelques comportements des objets de l'activité (visualiser les informations sur la boule : taille, couleur, position, etc.), et aussi de sauvegarder et charger les activités déjà définies. L'environnement de travail permet de placer les objets du jeu sur le décor (rappelons que le décor est statique dans le cas des jeux destinés à des enfants autistes).

Dans les sections suivantes nous décrivons en détail ces deux parties du logiciel.

B.1 Entête

L'entête représente une palette de configuration qui permet de caractériser les objets du jeu. Il permet également de charger, modifier et de sauvegarder les activités. Décrivons dans ce qui suit l'entête de droite à gauche telle qu'il se présente dans le logiciel *LevelEditor* :

– *Suppression des boules (Delete Balls)* : permet d'activer (On) ou désactiver (Off) la suppression des boules ;

[28] ce logiciel a été developpé par **Dimitrios V. Mouchritsas** dans le cadre d'un stage ERASMUS

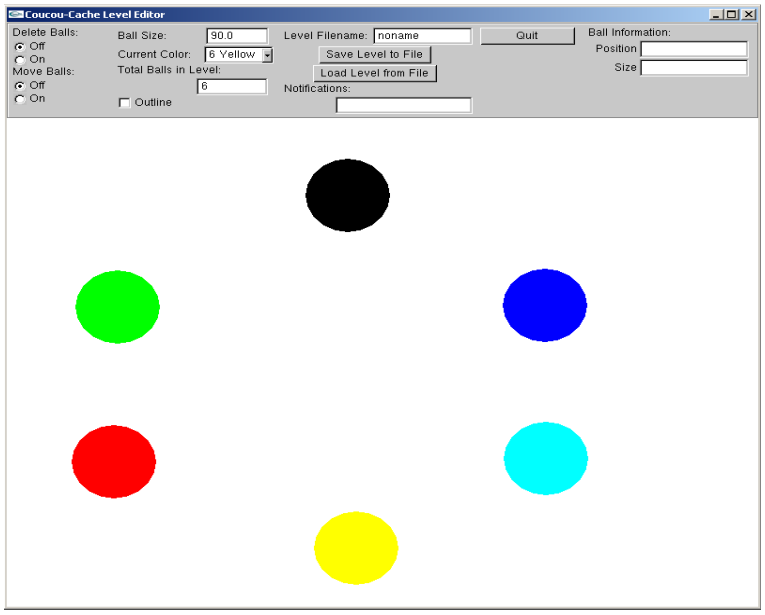

FIG. B.1 – Fenêtre *Level Editor*

- *Déplacement des boules (Move Balls)* : permet d'activer (On) ou désactiver (Off) le déplacement de la boule. Ces deux premières options seront décrites dans la prochaine section ;
- *Taille de boules (Balls Size)* : permet de définir la taille des boules. Les boules déjà placées ne peuvent pas changer de taille ;
- *Couleur Courante (Current Color)* : permet de choisir une des huit couleurs prédéfinies : Noir, Rouge, Vert, Bleu, Ciel, Jaune, Orange, Rose ;
- *Nombre de boules (Total Balls in Level)* : indique le nombre de boules de l'activité en question. Ceci afin de vérifier si le nombre de boules défini dans ce champ est adéquat avec le nombre de boules présent dans l'environnement de travail. Si le nombre n'est pas adéquat, il se peut que des boules se recouvrent entre elles ;
- *Contour (Outline)* : ajoute un contour noir aux périphéries des grosses boules ;
- *Nom de fichier (Level FileName)* : définit le nom du fichier de l'activité. Les activités sont stockées dans un dossier spécial. Le format du fichier est *nom_fichier.lvl* ;
- *Sauvegarde de l'activité (Save Level to File)* : permet de sauvegarder l'activité courante dans un fichier dont le nom est défini dans la zone de texte *Level FileName*. Si le fichier existe déjà, il sera écrasé par la nouvelle activité ;
- *Charger l'activité (Load Level from File)* : permet de charger l'activité dont le nom est indiqué dans *Level FileName* ;
- *Notifications* : permet de présenter plusieurs informations concertant la manipulation du logiciel. Par exemple, indique qu'un échec est survenu lors d'une écriture dans un fichier, etc ;
- *Quitter (Quit)* : permet d'arrêter le logiciel ;
- *Information sur les boules (Ball Information)* : permet de fournir des informations sur ce qui caractérise la boule pointée par la sourie. À savoir, sa position et sa taille ;

B.2 Environnement de travail

L'environnement de travail est représenté par un cadre dont l'utilisateur expert place les objets du jeu, à savoir les boules dans le cas du jeu *Coucou Caché*. L'expert peut placer la boule avec un clic gauche n'importe où sur l'environnement. Avant qu'il place la boule il devrait considérer la taille et la couleur de la boule car si le clic est près du cadre qui délimite l'environnement de travail une partie de la boule sera cachée.

En autorisant l'expert à ajouter les objets boules peut entraîner des erreurs : boules mal positionnées, tailles ou couleurs non conformes, etc. Pour corriger ces erreurs, inévitables, deux modes de corrections sont offertes : suppression et déplacement.

- *Mode Suppression (Delete Balls)* : une fois ce mode est activé, tout clic gauche sur une boule provoque sa suppression. Si l'utilisateur clic sur un espace (blanc), rien ne sera produit. Tout clic sur des boules qui se recouvrent provoque la suppression de toutes les boules en questions ;
- *Mode déplacement (Move balls)* : Une fois ce mode est activé, l'expert peut sélectionner et déplacer la boule à une nouvelle position. Lors du déplacement, une fois le curseur entre dans une autre boule, elle sera sous la première. Dans ce cas, l'utilisateur est conseillé de supprimer toutes les deux boules et de redéfinir les objets de l'activité.

Dans le cas où les deux modes sont activés, le mode suppression est plus prioritaire que le mode déplacement.

Annexe C

Trace d'exécution

Pour mieux analyser les résultats de la réalisation de chaque enfant autiste, deux logiciels complémentaires de « trace » ont été mis en œuvre[29]. Ces logiciels conservent une trace de l'activité de l'enfant. Ils enregistrent la durée de l'activité, le nombre d'essais qu'il fait jusqu'au moment où il réussit ainsi que le nombre de fois où le tuteur est intervenu pour aider l'enfant.

C'est d'abord un outil pour l'expert afin d'évaluer la performance de l'enfant pendant qu'il utilise le système. Plusieurs autres intérêts ont été cités dans la section 4.4.5. L'enregistrement de la trace est constitué par le nom de l'activité et la date à laquelle l'enfant a réalisé cette activité et d'autres informations liées à l'activité même.

La visualisation de la trace d'exécution des interactions enfant-jeu peut se faire selon deux approches : *animation graphique* et d*onnées statistiques*.

C.1 Animation graphique

Dans l'animation graphique toutes les informations issues des d'interactions enfant-jeu telle que « l'enfant a cliqué 3 fois sur l'objet ; ensuite il a sélectionné l'image, puis il a touché le cadre de la fenêtre, etc. » sont enregistrées et visualisées par l'expert.

Dans le cas du jeu *Coucou Caché*, l'animation graphique concerne tous les mouvements et les actions de l'enfant. Ces informations ainsi que les tailles et les positions de départ des objets sont stockés dans un fichier. Cela permet de refaire « le film » de la session.

[29] ces logiciels ont été developpés par **Dimitrios V. Mouchritsas** dans le cadre d'un stage ERASMUS

Ce « film » permet de mettre en évidence le déplacement de la boule curseur au cours du temps et la vitesse de manipulation.

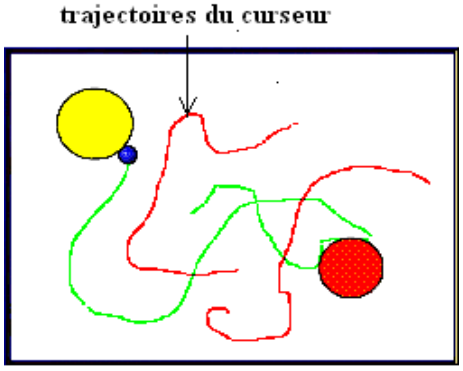

FIG. C.1 – Animation graphique

La figure C.1 montre un exemple d'une animation graphique du jeu *Coucou Caché*. Il s'agit de la trajectoire empruntée par la boule curseur. Deux couleurs sont associées à ces trajectoires : le vert indique que l'enfant a bien déplacé la boule curseur alors qu'une trace rouge indique que l'enfant a touché l'écran sans agir sur aucun objet, ce qui peut être perçu comme une forme d'échec.

Le tuteur peut enclencher puis arrêter l'enregistrement d'une trace en appuyant sur « Entrée ». Avec la touche clavier « t » il peut ensuite l'afficher ou pas. Enfin avec la touche « i » il peut l'imprimer.

C.2 Données statistiques

Les informations visualisées par l'animation graphique ne sont que des « informations » brut qui donnent un aperçu des actions de l'enfant mais elles sont difficiles à analyser. Pour donner un intérêt et une facilité d'analyse en termes d'état de structuration de l'enfant, des statistiques sur la trace d'exécution de l'activité deviennent pertinentes pour l'expert.

La figure C.2 montre une représentation des statistiques d'exécution du jeu *Coucou Caché*. Il s'agit de :

- **La durée de la session** : c'est l'heure de début de la session moins l'heure à la fin de la session ;
- **Temps passif** : c'est la durée dans laquelle l'enfant ne manipule pas le jeu durant la session ;
- **Touché du cadre** : c'est le nombre de fois que l'enfant touche le cadre avec la boule curseur ;
- **L'intervention du tuteur** : c'est le nombre de fois que le tuteur intervient pour aider l'enfant à réaliser la tâche souhaitée ;

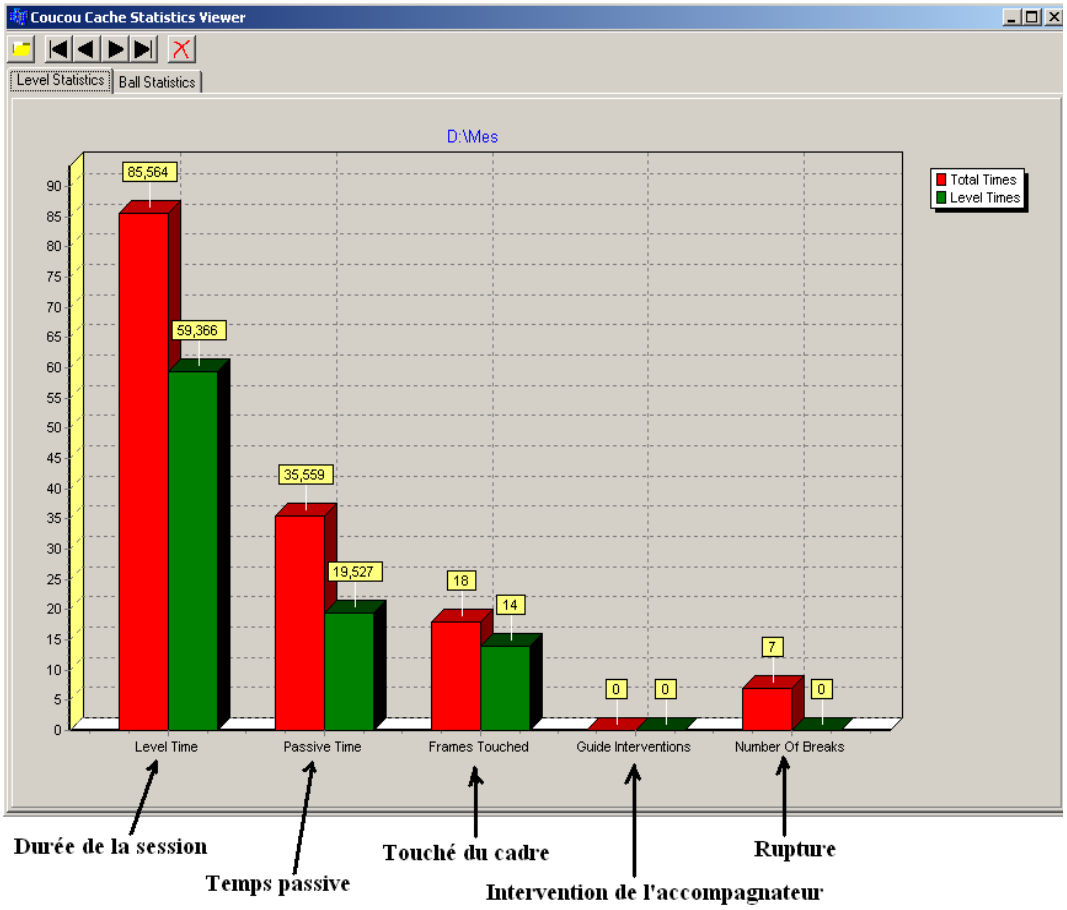

FIG. C.2 – Données statistiques - Fenêtre de *statics Viewer*

– **Rupture** : c'est le nombre de ruptures de l'enfant.

La figure C.3 montre le nombre de fois que les boules avec leurs couleurs correspondantes étaient touchées.

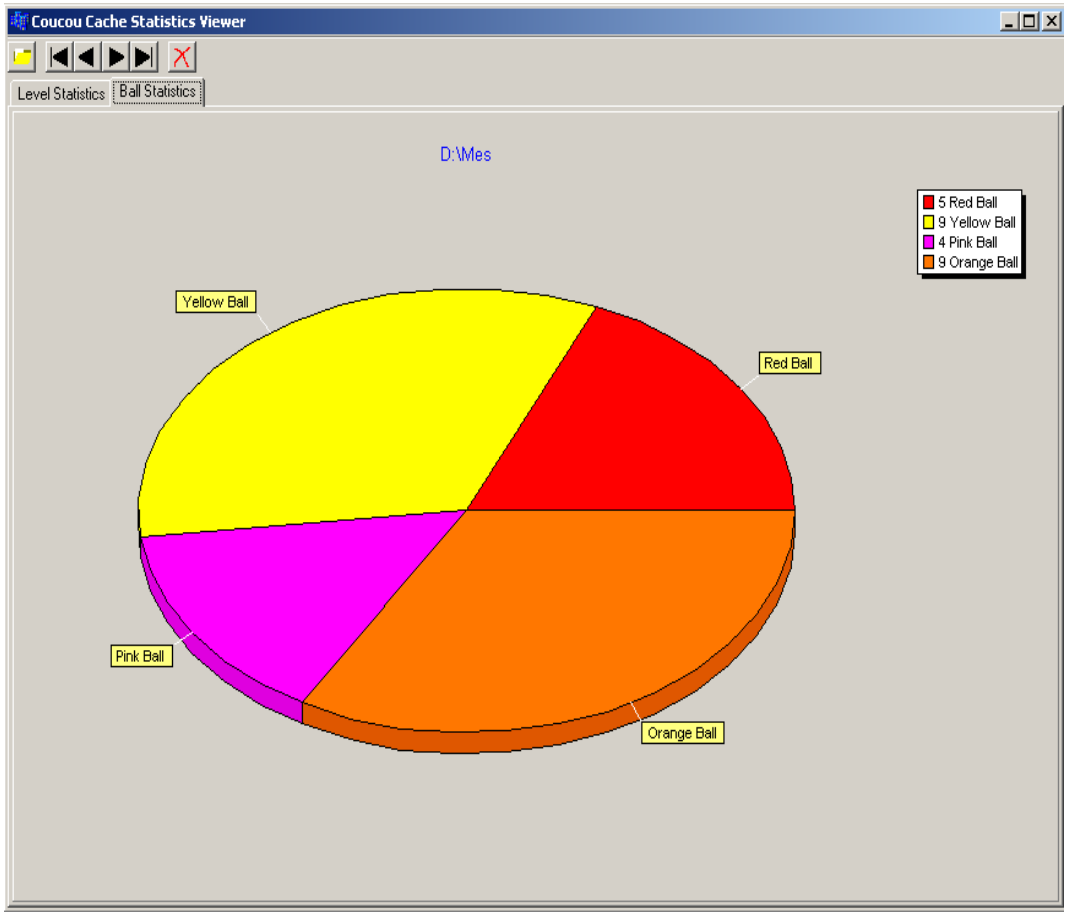

FIG. C.3 – Données statistiques - Le nombre de touché des boule du jeu *Coucou Caché*

Annexe D

Les jeux

Dans cette annexe, nous présetons quelques jeux destinés aux enfants autistes. Ces jeux ont été développés dans le cadre du PROJET AUTISME. Il s'agit des jeux, *Roule la boue*, *Pictécran* et *Fenêtre centrale*. Pour chaque jeu, nous présentons les objectifs éducatifs ainsi que les objets et leurs différents comportements. Dans la majorité des cas, ces jeux sont manipulés à travers un écran tactile, ce qui présente un lien direct entre les objets du jeu et l'action de l'enfant. Ceci offre une facilité d'utilisation par rapport à une manipulation via les éléments de contrôle, clavier, souris ou autres. Ces derniers présentent une complexité supplémentaire dans la mesure où il faut faire le lien entre les forces appliquées sur ces éléments et se qui se passe à l'écran.

D.1 Roule La boule

Ce jeu présente à l'enfant une boule en relief ou un disque de couleur, comme il est montré dans la figure D.1. Cette boule peut évoluer dans un cadre de travail, en suivant une progression horizontale, verticale ou libre et en émettant une musique. Pour cela, l'enfant doit donner l'impulsion à ce mouvement en utilisant la méthode du « Glisser-Jeter ». La boule poursuivra alors son déplacement en rebondissant contre les bords de l'écran et en tournant sur elle-même.

D.1.1 Objectifs éducatifs du jeu

Ce jeu, par le principe visuel d'une boule en mouvement libre ou selon un axe vertical ou horizontal, vise à apprécier et développer les capacités de perception et d'anticipation

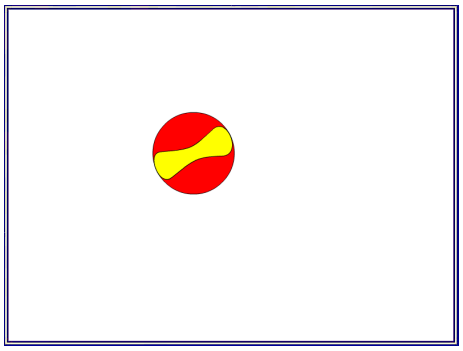

FIG. D.1 – Fenêtre principale du jeu *Roule la boule*

d'une action sur l'environnement de l'enfant. Il nécessite des possibilités d'attention, de mémorisation et d'évaluation de la situation. Il permettra à l'enfant de comprendre que son action a engendré un déplacement de la boule et pourra ainsi mieux assimiler la relation cause à effet.

D.1.2 Objets du jeu et leurs comportements

L'exécution adaptative de ce jeu consiste à reconfigurer le jeu de manière à qu'il soit le plus cohérent possible aux compétences et besoins de son utilisateur. Ceci est possible grâce à la définition des différents comportements qui peuvent avoir les objets du jeu. On peut en citer :

- *Son de sortie du cadre* : l'intérêt est de recadrer automatiquement l'enfant vers l'objectif premier de l'activité. Il s'agit d'émettre un signal sonore (traduisant l'éloignement du but à atteindre) lorsque le curseur de souris sort du cadre de travail ou lorsqu'un appui sur l'écran est effectué en dehors de ce cadre. De même, lorsque le curseur revient dans ce cadre ou qu'un appui est à nouveau effectué dans le cadre de travail, un autre signal sonore est émis (traduisant le rapprochement du but à atteindre) ;
- *Sens des déplacements* : Il s'agit de définir si les déplacements de la boule doivent être horizontaux, verticaux ou libres. Si le déplacement est horizontal ou vertical, la boule se déplacera au milieu de l'écran. Elle n'aura donc qu'un seul degré de liberté. Bien entendu, en mouvement libre, la boule peut être déplacée partout sur l'écran et peut suivre tous types de mouvements ;
- *Vitesse de déplacement* : Il s'agit de choisir la vitesse avec laquelle la boule va se déplacer à l'écran. Ceci dépendra de la capacité de la perception de l'enfant, plus elle est élevée plus le déplacement de la boule est plus rapide ;
- *Musique de réussite* : Une musique de réussite peut être considérée comme une récompense ou un symbole de réussite pour certains enfants. Il s'agit d'émettre une petite musique lorsque la boule est suffisamment déplacée suite à l'action de l'enfant ;
- *Couleur* : Il s'agit de choisir le style de présentation de la boule (en *Relief* ou *Unie*

sous la forme d'un *Disque*) et la couleur de la boule. En effet, les enfants sont souvent plus disposés à utiliser certaines couleurs. Le 'Relief' ou 'Unie' a pour unique but de montrer une rotation de la boule. La rotation de la boule peut effectivement être plus ou moins bien accueillie en fonction de chaque enfant.

D.2 Pictécran

Ce jeu se base sur une zone de travail où l'enfant déposera ses pictogrammes, il pourra choisir ces derniers à l'aide d'un catalogue accessible dans une barre en bas de l'écran. La zone de travail constitue l'emplacement principal de cette activité. C'est sur cette zone que l'enfant va pouvoir poser les pictogrammes qu'il choisit dans le catalogue. Il pourra les déplacer comme bon lui semble avec toutefois quelques restrictions, il lui est impossible de faire sortir un pictogramme du cadre de travail, parfaitement délimité par un contour noir. De plus, les pictogrammes ne peuvent en aucun cas se superposer.

La figure D.2 montre la fenêtre principale de ce jeu. Elle est composée de deux parties : une première partie, occupant la majeure partie de l'écran, est destinée à accueillir les pictogrammes déposés par l'enfant, et une seconde présente les pictogrammes disponibles. Les pictogrammes peuvent êtres ajoutés, déplacés, ou supprimés. L'enfant peut ajouter des pictogrammes autant de fois que l'écran le permet et il peut aussi ajouter le même pictogramme autant de fois qu'il le désire. L'ajout d'un pictogramme se fait donc en le glissant de la barre vers la zone de travail et de manière intuitive. La suppression se fait en suivant le chemin inverse, c'est-à-dire en glissant le pictogramme de la zone de travail vers la barre.

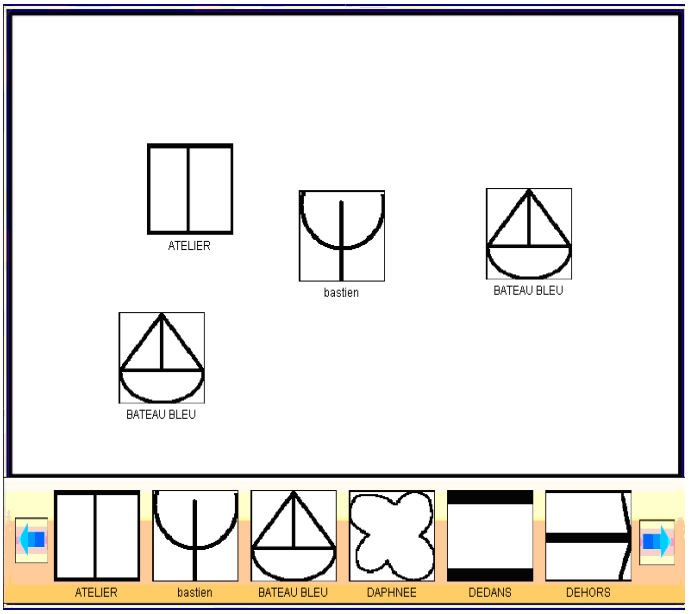

Fig. D.2 – Fenêtre principale du jeu *Pictécran*

D.2.1 Objectifs éducatifs du jeu

Ce jeu offre la possibilité aux enfants d'intégrer un code personnel ou collectif à partir de pictogrammes ou de représentations graphiques à fortes valences émotionnelles. Ce code amorce les procédures d'écriture et de lecture pouvant s'organiser autour de photos, dessins personnels, ou d'autres supports.

D.2.2 Objets du jeu et leurs comportements

Deux comportements sont possibles lors des déplacements des pictogrammes. Le choix de l'un ou de l'autre dépend de la satisfaction de l'ensemble des utilisateurs.

- Le premier comportement consiste à faire en sorte que le pictogramme en cours de déplacement puisse être placé n'importe où sur la zone de travail sans tenir compte des autres. Ceux-ci seront alors déplacés pour faire en sorte de laisser la place au pictogramme '*actif*'. Ce comportement est adapté aux utilisateurs qui ont une évolution de la gestion des déplacements mais il reste peu convaincant dans le cadre d'une utilisation par des enfants autistes. En effet, ce comportement entraîne parfois des déplacements de plusieurs pictogrammes à la fois pour laisser de la place ce qui peut perturber l'enfant ;
- Le second comportement s'est avéré plus adapté. Cette fois, c'est le pictogramme actif qui se place en fonction de ceux qui sont déjà présents. Si lors du déplacement d'un pictogramme la souris passe au-dessus d'un autre pictogramme, celui que nous déplaçons se placera alors à l'endroit le plus proche de la souris sans recouvrir un autre pictogramme.

D'autres comportements sont considérés :

- Le premier permet d'afficher ou non le nom de chaque pictogramme juste au-dessous dans la zone de travail. Pour l'affichage du nom, la zone de texte qui le contient est un objet à part entière qui suit toujours l'objet pictogramme auquel il est lié. Le comportement permet simplement d'afficher ou non cette zone de texte ;
- Le second définit si l'enfant verra tous ses pictogrammes ou si une présélection est faite auparavant. Ce comportement constitue une part importante, car il permet de choisir si l'enfant travaillera avec tous ses pictogrammes ou s'il n'en utilisera que quelques-uns qui seront sélectionnés par le tuteur ;
- Le troisième permet de choisir une image familière de fond parmi les photos ou les travaux de l'enfant. Ce comportement permet d'incruster une image de fond sur la zone de travail pour placer l'enfant dans un contexte ou dans un environnement qu'il affectionne. Après chaque clic sur l'une des cases à cocher, les photos ou les travaux de l'enfant sont chargés et affichés dans une liste présentée dans l'ordre prédéfini.

D.3 Fenêtre centrale

Fenêtre centrale est un logiciel qui permet d'importer des photos, ce qui permet au programme de s'adapter à chaque enfant. La photo est tout d'abord recouverte par un écran noir, et au fur et à mesure que l'enfant touche l'écran, l'image se découvre en commençant par le centre (Figure D.3). Lorsque toute la photo est découverte, une musique de réussite est déclenchée et un personnage recouvre à nouveau l'écran de noir (Figure D.4). L'exercice continue alors avec la photo suivante.

FIG. D.3 – Apparition de l'image dans le jeu *Fenêtre centrale*

FIG. D.4 – Animation de recouvrement dans le jeu *Fenêtre centrale*

D.3.1 Objectifs éducatifs du jeu

Ce jeu permet de montrer des objets connus de l'enfant par le biais des photos dans un contexte différent de l'endroit ou il pourrait les voir normalement. Les photos doivent donc

avoir une relation avec le vécu de l'enfant. Cela permet ainsi un renforcement son environnement de l'enfant. La série de photos peut aussi permettre de retracer un cheminement lorsqu'elles s'enchaînent dans un ordre précis.

Cependant l'objectif était parfois détourné par certains enfants. En effet, l'animation du bonhomme qui marche pour faire disparaître l'image (Figure D.4) était tellement attrayante pour certains enfants qu'ils ne prêtaient plus aucune attention au contenu des images. Ils s'empressaient de toucher l'écran pour voir cette animation.

Afin de remédier à ce problème, deux comportements de découvrement peuvent être choisis. Ces comportements seront détaillés dans la prochaine section.

D.3.2 Objets du jeu et leurs comportements

- *Le comportement Gomme* : Lorsque l'enfant passe son doigt sur l'écran, des zones rectangulaires noires qui se situent sous son doigt disparaissent pour faire apparaître l'image. Ce comportement force l'enfant à parcourir toutes les zones de l'écran pour découvrir la photo, ce qui rend l'activité plus difficile mais aussi plus interactive. Les zones noires ont cependant été choisies grandes afin de simplifier l'utilisation par rapport à une gomme de type « *palette* » ;
- *Le comportement Clic* : Il est très semblable au comportement « *gomme* » à l'exception près que la disparition d'un rectangle noir s'effectue lorsque le doigt de l'enfant quitte l'écran. L'enfant doit donc appuyer sur chaque zone et ne peut pas faire glisser son doigt sur l'écran pour faire disparaître plusieurs zones.

Annexe E

FaceLab

FaceLab, dont l'interface principale est présentée dans la figure E.1, est un système logiciel et matériel permettant la capture et l'analyse en temps réel des mouvements du visage et des yeux de l'utilisateur. Basé sur un système d'analyse d'image, FaceLab contient une bibliothèque de programme qui utilise un ensemble de caméras comme un dispositif de mesure passif. Le flux d'images provenant des caméras sont analysées pour établir des caractéristiques du visage de l'utilisateur, y compris la position courante et l'orientation de la tête dans l'espace 3D, la direction du regard et plusieurs autres mesures.

À la différence des équipements qui nécessitent de porter un casque et une caméra, Facelab libère l'utilisateur de toute technologie embarquée : la totalité des mesures sont effectuées depuis deux caméras positionnées face à l'utilisateur sans aucun capteur (voir la figure E.2). En revanche, Il nécessite une procédure de calibrage. Lors de cette procédure, des points caractéristiques du visage de l'utilisateur sont définis, comme le montre la figure E.1. Cela permet au système de les suivre afin de comprendre le comportement de l'utilisateur dans des diverses situations.

E.1 Ce que mesure FaceLab

Dans ce qui suit, nous présentons les caractéristiques que FaceLab peut mesurer :

- Position et orientation de la tête ;
- Regard : FaceLab fournit deux vecteurs de regard, chacun pour un œil de l'utilisateur ;
- Saccades : elles sont définies comme des mouvements rapides de l'œil pour changer la position du regard entre deux points de fixation. Il s'agit d'une donnée binaire qui peut prendre deux valeurs *vrai* ou *faux* ;

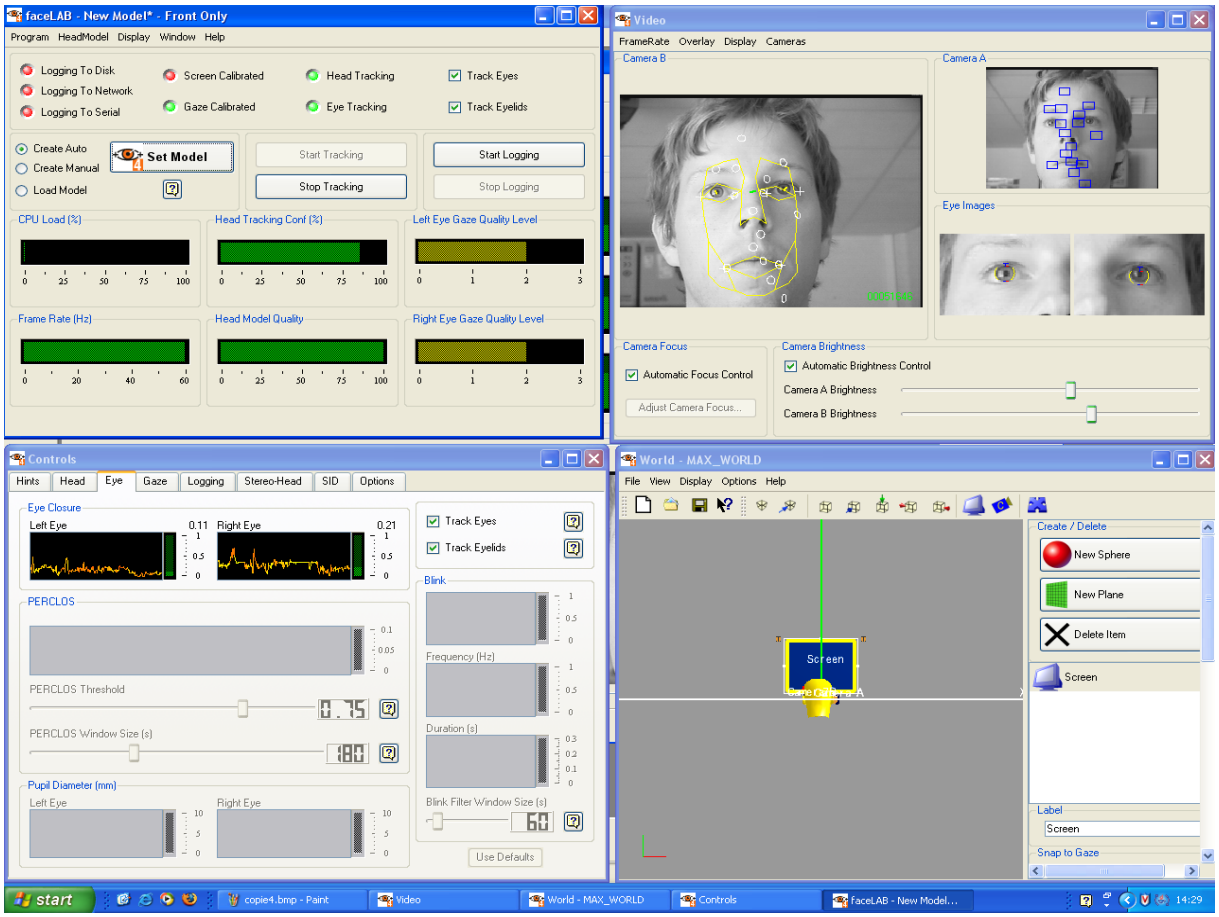

FIG. E.1 – L'éditeur de FaceLab

- Les objets que l'utilisateur regarde ;
- Fermeture des yeux : pour chaque œil est en pourcentage pour être significatif ;
- La mesure de la fatigue, l'hypovigilance et la perte d'attention : en fonction de la durée de fermeture et d'ouverture de l'œil ;
- Position des pupilles ;
- Position et taille des points caractéristiques du visage. Les quatre régions que FaceLab peut mesurer sont la tête, la bouche, chaque œil et chaque pupille.

Les résultats de l'analyse sont présentés en temps réel en local ou exploités par le réseau. Trois possibilités sont offertes pour l'exploitation des mesures :

- L'enregistrement dans un fichier ;
- L'envoi des données sur le réseau ;
- L'envoi sur un port série.

Les deux premières sont particulièrement intéressantes pour la trace d'exécution, la troisième est consacrée pour assurer l'exécution adaptative, en particulier pour l'analyse du comportement de l'utilisateur.

FIG. E.2 – FaceLab

E.2 Configuration des caméras

Il y a trois types de configuration que l'on peut choisir lors de la création d'un modèle de tête :

1. *Champ de vision large (Wide field of view)* : Dans ce type de configuration, la précision du regard est considérablement plus faible que pour les deux autres configurations. Les paramètres du suivi de l'œil ne sont pas ajustables et le regard ne peut pas être calibré. Ce mode n'est pas fait pour le suivi du regard, mais plus pour le suivi du visage ;
2. *Classique (classic)* : le zoom des caméras est moyen, la précision du suivi du regard aussi. Toutes les options sont disponibles pour ce mode, qui constitue un bon compromis entre précision et liberté de mouvement ;
3. *Regard de précision (Precision gaze)* : une des deux caméras zoome sur le visage de l'utilisateur. Cette configuration permet ainsi une plus grande précision du suivi du regard. En revanche, comme une des caméras ne filme que le visage (sourcils, yeux, bouche) ce mode limite les mouvements de tête de l'utilisateur. En effet, un léger mouvement latéral de la tête fait facilement sortir un des yeux de l'utilisateur du champ de la caméra. Ce mode est le seul où le suivi de regard peut atteindre la qualité maximale de suivi du regard (3 sur une échelle entière de 0 à 3).

E.3 Présentation de *Screen Intersection Display*

Le suivi du regard s'appuie sur un bon suivi de la tête dans la mesure où la connaissance fiable des positions des coins des yeux est très importante pour le suivi du regard.

Pour un suivi de précision, il faut s'assurer de commencer avec un bon modèle de suivi du visage et des yeux. Le *Screen Intersection Display* (SID)est un outil qui permet de calibrer l'intersection du regard avec l'écran. Le principe est simple : il s'agit de fixer successivement neuf points blancs qui apparaissent les uns après les autres pendant quelques secondes, en différents endroits sur l'écran noir.

FaceLab enregistre la position estimée de l'œil sur ces neuf points, et la position réelle des points, calcule l'erreur moyenne de distance commise, l'erreur angulaire moyenne et l'écart type (standard deviation) angulaire et en distance. La figure E.3 montre un exemple de résultats d'un SID en mode précision.

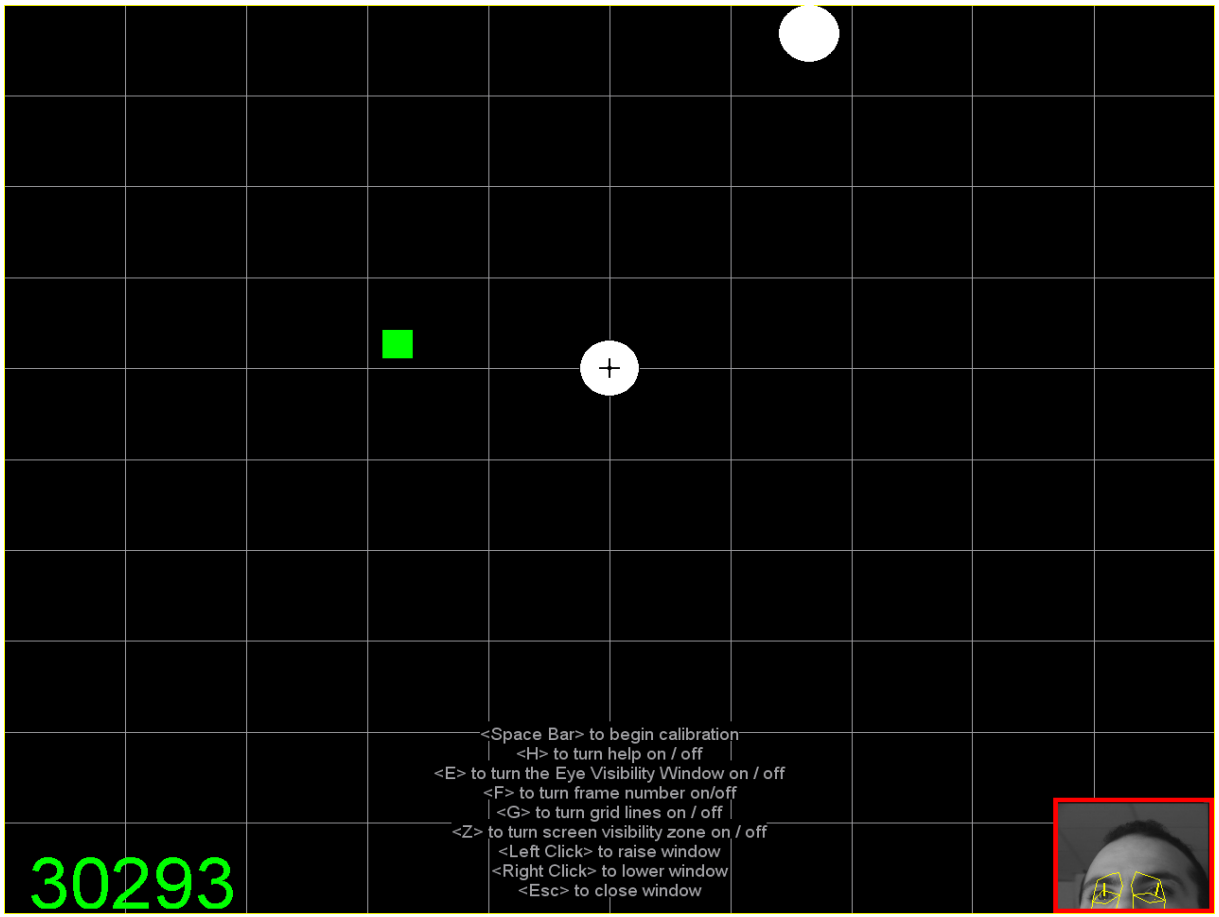

FIG. E.3 – Résultat de suivi du regard avec *Screen Intersection Display*

Bibliographie

[AC87] P. Agre and D. Chapman. Pengi : an implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

[AKU03] V. Ampornaramveth, P. Kiatisevi, and H. Ueno. Toward a software platform for knowledge management in human-robot environment. Rapport technique 83, IEICE, 2003.

[Apa04] The Apache Software Foundation, http ://httpd.apache.org/. *Apache HTTP Server Project*, 2004.

[Ayl99] R. Aylett. Narrative in virtual environments - towards emergent narrative. In *AAAI 1999 Fall Symposium on Narrative Intelligence*, 1999.

[Bal99] N. Balacheff. *Apprendre la preuve*, chapter Le concept de preuve à la lumière de l'intelligence artificielle, pages 197–236. Paris : PUF, 1999.

[BCK98] L. Bass, P. Clements, and P. Kazman. *Software Architecture in Practice*. Addison Wesley, 1998.

[BDH98] A. Billard, K. Dautenhahn, and G. Hayes. Experiments on human-robot communication with robota, an imitative learning and communicating doll robot. In *Proceeding Socially Situated Intelligence Workshop*, Zurich (CH), 1998.

[BDKT97] M.T. Brazier, R.N. Dunin-Keplicz, B.M. Jennings, and J. Treur. Desire : modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Information Systems*, 6(1), 1997.

[BFKSD97] M. Barbeau, M. Frappier, F. Kabanza, and R. St-Denis. A supervisory control synthesis case study : The antenna control system. In *35th Allerton Conference on Communication, Control and Computing*, Octobre 1997.

[BG95] H. Buxton and S. Gong. Advanced visual surveillance using bayesian networks. In *International Conference on Computer Vision*, 1995.

[BI98] A.F. Bobick and Y.A. Ivanov. Action recognition using probabilistic parsing. *Computer Vision and Pattern Recognition*, 1998.

[Bil00] A. Billard. Play, dreams and imitation in robota. In *Proceeding AAAI Fall Symposium Socially Intelligent Agents - The Human in the Loop*, 2000.

[BIP88] M. Bratman, D.I. Israel, and M.E. Pollack. Plans and resource-bounded practical reasoning. *Computer Intelligence*, 4 :349–355, 1988.

[Bis00] G. Bisson. La similarité : une notion symbolique/numérique. apprentissage symbolique-numérique (tome 2). In *CEPADUES*, pages 169–201, 2000.

[BM86] D.S. Berry and L.Z. McArthur. Perceiving character in faces : The impact of age-related craniofacial changes on social perception. *Psychological Bulletin*, 100(1) :3–18, 1986.

[BOP97] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. *Computer Vision and Pattern Recognition*, 1997.

[Bou98] N. Boukhatem. *L'approche multi-agent pour un contrôle adaptatif de réseaux ATM.* Thèse de doctorat, Université de Versailles Saint-Quentin-en-Yvelines, janvier 1998.

[Bra85] R. J. Brachman. I lied about the trees, or defaults and definitions in knowledge representation. *AI Magazine*, 3(6) :80–93, 1985.

[Bre99] C. Breazeal. Robot in society : Friend or appliance ? In *Proceedings of the 1999 Autonomous Agents Workshop on Emotion-Based Agent Architectures*, pages 18–26, Seattle, 1999.

[BRHL99] P. Busetta, R. Ronnquist, A. Hodgson, and A. Lucas. Jack intelligent agents components for intelligent agents in java. Technical report, http ://www.agent-software.com.au/, 1999.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :12–23, Mars 1986.

[Bru01] P. Brusilovsky. Adaptive hypermedia. user modeling and user adapted interaction. In *Ten Year Anniversary Issue (Alfred Kobsa, ed.)*, volume 11, pages 87–110, 2001.

[BS92] B. Burmeister and K. Sundermeyer. Cooperative problem-solving guided by intention and perception. In E. Werner and Y. Eds Demazeau, editors, *Artificial intelligence 3*, pages 77–92, Holland, 1992.

[BS99] C. Breazeal and B. Scassellati. How to build robots that make friends and influence people. In *Proceeding IROS99*, Kyonjiu : Korea, 1999.

[CBVY00] J. Cassell, T. Bickmore, H. Vilhjálmsson, and H. Yan. More than just a pretty face : Affordances of embodiment. In *Proceedings of 2000 International Conference on Intelligent User Interfaces*, New Orleans, Louisiana, 2000.

[CCM02] M. Cavazza, F. Charles, and S.J. Mead. Interacting with virtual characters in interactive storytelling. In *Conference on Autonomous Agents and Multiagent Systems (AAMAS2002)*, pages 318–325, Bologna, Italy, 2002.

[CGLSN05] J. Cole, M. Gray, J. Lloyd, and K. Siong Ng. Personalisation for user agents. In *Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, volume 2, pages 603–610, Utrecht, The Netherlands, Juillet 2005.

[Che04] F. Cheneviere. *Contribution à l'analyse du mouvement dansé.* Thèse de doctorat, Université de La Rochelle, La Rochelle, 2004.

[Clé02] C. Cléder. *Planification didactique et construction de l'objectif d'une session de travail individualisée : modélisation des connaissances et du raisonnement*

mis en jeu. Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, 2002.

[CNT99] J. Collis, D. Ndumu, and S. Thompson. Zeus methodology documentation, role modelling guide, three case studies, application realisation guide, and runtime guide. Technical report, http ://www.labs.bt.com/projects/agents/zeus/, 1999.

[CPE05] R. Champagnat, A. Prigent, and E. Estraillier. Scenario building based on formal methods and adaptative execution. In *International Simulation and Gaming Association ISAGA'2005*, eorgia Institute of technology, juin 2005.

[CPM02] P.A. Champin, Y. Prié, and A. Mille. Une approche fondée sur les usages pour l'assistance à l'utilisateur sur le web sémantique. In *Actes 13es Congrès Francophone de Reconnaissance de Formes et d'Intelligence Artificielle*, pages 633–642, Angers, janvier 2002.

[CRTT97] N. Chleq, C. Regazzoni, A. Teschioni, and M. Thonnat. A visual surveillance system for the prevention of vandalism in the metro stations. In *In European, Multimedia, Microprocessor Systems and Electronic Commerce EMMSEC'97*, Florence, Italy, November 1997.

[CS05] F. Collé and K Sehaba. Scenario analysis based on linear logic. In *IMAGINA'05 (Papier jeune chercheur)*, Monaco, Février 2005.

[CSE03] F. Collé, K. Sehaba, and P. Estraillier. A framework for business simulator : a first experience. In *International Conference on Enterprise Information System (ICEIS'03)*, volume 3, pages 540–543, Angers, April 2003.

[Dau98] K. Dautenhahn. Story-telling in virtual environments. In *ECAI Workshop on Intelligent Virtual Environments*, Brighton, UK, 1998.

[Dau00] K. Dautenhahn. Design issues on interactive environments for children with autism. In *Proceeding 3rd Intl Conf. Disability, Virtual Reality & Assoc. Tech*, pages 153–159, Alghero, Italy, 2000.

[DBLN04] M. D'aquin, S. Brachais, J. Lieber, and A. Napoli. Vers une acquisition automatique de connaissances d'adaptation par examen de la base de cas - une approche fondée sur des techniques d'extraction de connaissances dans des bases de données. In *12ème Atelier de Raisonnement à Partir de Cas - RàPC'04*, pages 41–52, Université Paris Nord, Villetaneuse., 2004.

[DC] S. Després and V. Ceausu. Raisonnement à partir de cas pour contribuer à améliorer l'aménagement du réseau urbain en prenant en compte la sécurité.

[Des00] O. Despouys. *Une architecture intégrée pour la planification et le contrôle d'exécution en environnement dynamique*. Thèse de doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes LAAS-CNRS, 2000.

[Des02] S. Despres. *Contribution à la conception de méthodes et d'outils pour la gestion de connaissances*. Habilitation à diriger des recherches, Université de Paris V, Paris, 2002.

[Dil89] P. Dillenbourg. Designing a self-improving tutor : Proto-teg. *Instructional Science*, pages 193–216, 1989.

[DK89] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3) :142–150, 1989.

[dLN05] M. d'Aquin, J. Lieber, and A. Napoli. Vers une utilisation du critère pessimiste de wald pour aider à la décision à partir de cas. In *13ème Atelier Raisonnement à Partir de Cas*, Nice, 2005.

[DP91] R. Doron and R. Parot. *Dictionnaire de psychologie*. 1991.

[DRD+05] M. Davis, B. Robins, K. Dautenhahn, C. Nehaniv, and S. Powell. Comparison of interactive and robotic systems in therapy and education for children with autism. In *Assistive Technology : From Virtuality to Reality AAATE*, pages 353–357, Lille, 2005. IOS Press.

[ED94] VOß ED. Similarity concepts and retreival methods. FABEL project Technical Report 13, Gesselschaft für Mathematik und Datenverarbeitung (GMB), Sankt Augustin, 1994.

[EF96] C. Earl and R.J. Firby. Combined execution and monitoring for control of autonomous agents. Rapport technique TR-96-19, Department of Computer Science, University of Chicago, Novembre 1996.

[EFnC00] J. Elorriaga and I. Fern ndez Castro. Using case-based reasoning in instructional planning : towards a hybrid self-improving instructional planner. *International Journal of Artificial Intelligence in Education*, pages 416–449, 2000.

[ENS95] K. Erol, D. Nau, and V. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. In *Artificial Intelligence 76*, pages 75–88, 1995.

[FC02] P. Funk and O. Conlan. Case-based reasoning to improve adaptability of intelligent tutoring systems. In *Workshop on Case-Based Reasoning for Education and Training at 6th European Conference on Case Based Reasoning, CBRET'2002*, pages 15–23, Aberdeen, Scotland, 9 2002. Robert Gordon University.

[Fer95] J. Ferber. *Les systèmes multi-agents*. InterEditions, Paris, 1995.

[Fir94] R.J. Firby. Task networks for controlling continuous processes. In *Second International Conference on AI Planning Systems*, Juin 1994.

[Fir95] R.J. Firby. The rap language manual - version 1.0. Rapport technique TR-95-6, Department of Computer Science, University of Chicago, Mai 1995.

[FM05] M. Fabri and D.J. Moore. The use of emotionally expressive avatars in collaborative virtual environments. In *Proceeding of Symposium on Empathic Interaction with Synthetic Characters, held at Artificial Intelligence and Social Behaviour Convention 2005 (AISB 2005)*, University of Hertfordshire, 2005.

[Foi98] R. Foisel. *Un modèle de réorganisation de système multi-agents : une approche descriptive et opérationnelle*. Thèse de doctorat, Université Henri Poincarré - Nancy 1, 1998.

[Fuc97] B. Fuchs. *Représentation des connaissances pour le raisonnement à partir de cas : le système Rocade*. Thèse de doctorat, Université Jean Monnet de Saint Etienne, 1997.

[Fut90] M. Futtersack. *QUIZ : Une architecture multi-agents pour un tuteur intelligent*. Thèse de doctorat, Université Paris 6, Paris, 1990.

[Gep01] B. Gepner. « malvoyance » du mouvement dans l'autisme infantile ? une nouvelle approche neuropsychopathologique développementale. *La Psychiatrie de l'enfant*, pages 77–126, 2001.

[GF97] O. Gutknecht and J. Ferber. Madkit : Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical Report 97188 97188, LIRMM, 1997.

[GI89] M.P. Georgeff and F.F. Ingrand. Dicision-making in an embedded reasoning system. In *International Joint Conference on Artificial Intelligence, 11th*, pages 972–978, Detroit, 1989.

[Gib77] J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford (dir.), editors, *Perceiving, acting and knowing. Toward an ecological psychology*, pages 67–82, Hillsdale, NJ : Erlbaum., 1977.

[Gib79] J.J. Gibson. The ecological approach to visual perception. chapter 14 : The theory of information pickup and its consequences. Boston : Houghton Miflin Co, 1979. Lawrence Erlbaum Associates.

[GMB01] Z. Guessoum, T. Meurisse, and J.P. Briot. Construction modulaire d'agents et de systèmes multi-agents adaptatifs en dima. *Techniques et Sciences Informatiques*, 1(1), 2001.

[GMN04] O. Grynszpan, J.C. Martin, and J. Nadel. Interfaces homme-machine pour personnes autistes : une étude en cours basée sur des jeux éducatifs dialogiques. In *Premières rencontres du Réseau national d'études cognitives et neurocognitives de l'autisme*, Paris, 2004.

[Gou99] J. Gout. *Intégration de la planification temporelle dans l'architecture décisionnelle d'un robot autonome*. Thèse de doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse, Mai 1999.

[Gér02] P. Gérard. *Systèmes de classeurs : étude de l'apprentissage latent*. Thèse de doctorat, l'Université Paris 6, Paris, Décembre 2002.

[Gue96] Z. Guessoum. *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents*. Thèse de doctorat, Paris 6, LAFORIA, Paris, 1996.

[Gut92] E. Gutstein. Using expert tutor knowledge to design a self-improving intelligent tutoring system. In *Intelligent Tutoring Systems, 2nd Int. Conference ITS'92*, pages 625–632. Frasson, C., Gauthier C., McCalla, G.I. (Eds.), LNCS Springer-Verlag, 1992.

[HBN00] S. Hongeng, F. Brémond, and R. Nevatia. Bayesian framework for video surveillance application. In *International Conference on Pattern Recognition*, 2000.

[Hoe01] J. Hoey. Hierarchical unsupervised learning of facial expression categories. *IEEE Workshop on Detection and Recognition of Events in Video*, 2001.

[Hol76] J.H. Holland. Adaptation. In *Progress in theorical biology*, 1976.

[HR78] J.H. Holland and J.S. Reitman. Cognitive systems based on adaptive algorithms. In *Pattern Directed Inference Systems*, pages 125–149, 1978.

[HR85] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3) :251–321, 1985.

[HS96] A. Haddadi and A. Sundermeyer. Belief - desire - intention agent architectures. In G.M.P. O'Hare et N.R. Jennings eds, editor, *Foundations of Distributed Artificial intelligence*, pages 169–185, 1996.

[Iff92] C. Iffnecker. *Un système multi-agents pour le support des activités de conception de produits.* Thèse de doctorat, Université de Pierre et Marie Curie, 1992.

[IG90] F.F. Ingrand and M.P. George. Managing deliberation and reasoning in real-time ai systems. In *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning*, Santa Diego, Novembre 1990.

[IG92] F.F. Ingrand and M.P. George. An architecture for real-time reasoning and system control. In *IEEE Expert*, volume 6, pages 33–44, Décembre 1992.

[Int99] S.S Intille. *Visual Recognition of multi-agent action.* Phd thesis, Massachusetts Institute of Technology, September 1999.

[Jen93] N.R. Jennings. Specification and implementation of belief-desir-joint-intention architecture for collaborative problem solving. *International Journal on Intelligence Cooperative information Systems*, 2(3) :289–318, 1993.

[JG98] Ferber J. and M. Ghallab. Problématiques des univers multi-agents intelligents. In *Journées nationale sur l'IA*, Toulouse, mars 1998.

[Jl77] P. Johnson-laird. Procedural semantics. In *Cognition*, volume 5, pages 189–214, 1977.

[KAU04] P. Kiatisevi, V. Ampornaramveth, and H. Ueno. A distributed architecture for knowledge-based interactive robots. In *2nd International Conference on Information Technology for Application (ICITA 2004)*, Harbin, China, January 2004.

[Kol93] J. Kolodner. *Case-based reasoning.* Morgan Kaufmann Pub, 1993.

[LE77] V.R. Lesser and L.D. Erman. A Retrospective View of the HEARSAY-II Architecture, 1977.

[Lea96] D. Leake. Cbr in context : The present and the future. In *Case-BasedReasoning : Experiences, Lessons, and Future Directions*, 1996.

[LES+05] D. Lambert, P. Estraillier, K. Sehaba, V. Courboulay, E. Bouchaud, and V. Gabet. Informatique appliquée aux troubles autistiques. présentation d'un projet de partenariat entre équipe de pédopsychiatrie et le laboratoire universitaire d'informatique. In *Journées Nationales (SFPEADA'05)*, Tours, juin 2005.

[LH92] D.M. Lyons and A.J. Hendriks. A practical approach to integrating reaction and deliberation. In *First International Conference on Artificial Intelligence Planning Systems*, pages 153–162, 1992.

[LH95] D.M. Lyons and A.J. Hendriks. Exploiting patterns of interaction to achieve reactive behaviour. *Artificial Intelligence*, 73 :117–148, 1995.

[Lsq] D. Lambert and son équipe. Communications personnelles. Technical report, Le service de pédopsychiatrie de l'hôpital de La Rochelle.

[Lue99] V. Luengo. A semi-empirical agent for learning mathematical proof. In SP Lajoie and M. Vivet (Eds), editors, *Artificial Intelligence in Education*, 1999.

[Mar99] J.M. Marcus. Jam : A bdi-theoretic mobile agent architecture. In *International Conference on Autonomous Agents (Agents'99)*, pages 236–243, Seattle, Mai 1999.

[MDS95] D.J. Musliner, E.H. Durfee, and K.G. Shin. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence*, 1995.

[MHCF04] J.C. Marty, J.H. Heraud, T. Carron, and L. France. A quality approach for collaborative learning scenarios. *Learning Technology newsletter of IEEE Computer Society*, 6(4) :46–48, 2004.

[Min75] M. Minsky. *A framework for Representing Knowledge*. The Psychology of Computer Vision, MacGraw-Hill, New York, wiston (ed.) edition, 1975.

[Min98] M. Minsky. *Society of Mind*, chapter A theory of memory, pages 81–92. Touchstone books edition, 1998.

[MLBT03] N. Moënne-Loccoz, F. Brémond, and M. Thonnat. Recurrent bayesian network for the recognition of human behaviors from video. In *icvs*, 2003.

[Mon03] J. Montemayor. *Physical programming : tools for kindergarten children to author physical interactive environments*. Thèse de doctorat, department of computer science, University of Maryland, USA, 2003.

[MRU01] P. Mathieu, J.C. Routier, and P. Urro. Un modèle de simulation agent basé sur les interactions. In *Journées Francophones Modèles Formels de l'interaction*, Toulouse, Mai 2001.

[MZ98] J.M. Montepare and L.A. Zebrowitz. Person perception comes of age : the salience and significance of age in social judgments. *Advances in Experimental Social Psychology*, 30 :93–161, 1998.

[New90] C. Newell. *Unified Theories of Cognition*. 1990.

[Nor88] D.A. Norman. *The psychology of everyday things*. New York : Basic Books, Paris : PUF, 1988.

[Nor90] D.A. Norman. *The design of everyday things*. New York : Doubleday, 1990.

[NT99] R. Nakatsu and N. Tosa. Interactive movies. In *Handbook of Internet and Multimedia, Systems and applications*. B. Furht (Ed), CRC Press, 1999.

[NV03] S. Natkin and L. Vega. A petri net model for the analysis of the ordering of actions in computer games. In *GAME ON 2003*, Londres, Octobre 2003.

[NY05] S. Natkin and C. Yan. Analysis of correspondences between real and virtual worlds in general public applications. In *CGIV05 Computer Graphics, Imaging and Visualization*, Pékin, juillet 2005.

[Par01] S. Parsons. Social conventions in virtual environments : Investigating understanding of personal space amongst people with autistic spectrum disorders. In *Robotic & Virtual Interactive Systems in Autism Therapy*, U.K, 2001.

[Pau99] P. Paulo. *GD.Visu@l : Environnement Distribué Interactif pour l'Apprentissage Humain de la Géométrie Descriptive*. Thèse de doctorat, Université du Maine - 5, novembre 1999.

[Rab89] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 1989.

[RDtBB04] B. Robins, K. Dautenhahn, R. te Boekhorst, and A. Billard. Effects of repeated exposure of a humanoid robot on children with autism. In *Universal Access and Assistive Technology (CWUAAT)*, Cambridge, UK, Mars 2004.

[ReM92] *Cognitive Systems*. ReMind Developer's Reference Manual, Boston, 1992.

[Ret99] Reticular Systems, www.agentbuilder.com. *AgentBuilder User's guide*, 1999.

[RMPB03] J.H. Réty, J.C. Martin, C. Pelachaud, and .N Bensimon. Coopération entre un hypermédia adaptatif éducatif et un agent pédagogique. In *Proceedings of H2PTM'2003*. Hermès, 2003.

[RRAM95] N. Rakoto-Ravalontsalama and J. Aguilar-Martin. *Supervision de processus à l'aide du système expert G2*. Hermes ed. paris edition, October 1995. ISBN 2-86601-499-5.

[RS89] C.K. Riesbeck and R.C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[SC05] K. Sehaba and V. Courboulay. L'aide à la décision par observation et analyse de comportement. Rapport interne, L3i, Université de La Rochelle, Janvier 2005.

[SCE05a] K. Sehaba, V. Courboulay, and P. Estraillier. Interactive system by observation and analysis of behavior for children with autism. In *Assistive Technology : From Virtuality to Reality AAATE*, pages 358–362, Lille, 2005. IOS Press.

[SCE05b] K. Sehaba, V. Courboulay, and P. Estraillier. L'aide à la décision par observation et analyse de comportement dans le contexte des jeux éducatifs. In *Journée Intelligence Artificielle et Jeux*, Paris, Juin 2005.

[SCE06] K. Sehaba, V. Courboulay, and P. Estraillier. Interactive system by observation and analysis of behavior for children with autism. *Accepté à Technology and Disability Journal*, 2006.

[Sch82] R.C. Schank. *Dynamic Memory : a Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.

[SE05a] K. Sehaba and P. Estraillier. A multi-agent system for rehabilitation of children with autism. In *AAMAS-05 Workshop on Agent-Based Systems for Human Learning (ABSHL'05)*, 2005.

[SE05b] K. Sehaba and P. Estraillier. Raisonnement à partir de cas pour la configuration de jeux thérapeutiques destinés à des enfants autistes. In *13ème Atelier Raisonnement à Partir de Cas*, Nice, 2005.

[SEKPG97] D. Simon, B Espiau, K. Kapellos, and R. Pissard-Gibollet. Orccad : Software engineering for real time robotics, a technical insight. *Robotica*, 15(1) :111–116, 1997.

[SEL05] K. Sehaba, P. Estraillier, and D. Lambert. Interactive educational games for autistic children with agent-based system. In *4th International Conference on Entertainment Computing (ICEC'05)*, pages 422–432, Sanda, Japan, September 2005. Springer.

[SH94] L. Spector and J. Hendler. The use of supervenience in dynamic world planning. In *Second International Conference on AI Planning Systems*, pages 158–163, Juin 1994.

[SHG+01] W. Swartout, R. Hill, J. Gratch, W.L. Johnson, C. Kyriakakis, C. LaBore, R. Lindheim, S. Marsella, D. Miraglia, B. Moore, J. Morie, J. Rickel, M. Thiebaux, L. Tuch, R. Whitney, and J. Douglas. Toward the holodeck : Integrating graphics, sound, character and story. In *Proceedings of the Autonomous Agents 2001 Conference*, 2001.

[SHM05] A. Stuber, S. Hassas, and A. Mille. L'expérience tracée comme support potentiel de négociation de sens entre agents informatiques et humains. In *13ème Atelier Raisonnement à Partir de Cas, Plateforme AFIA 2005*, Nice, 2005.

[Sig04] O. Sigaud. *Comportements adaptatifs pour des agents dans des environnements informatiques complexes*. Habilitation à diriger des recherches, Université PARIS 6, 2004.

[SPS03] J.P. Sansonnet, G. Pitel, and N. Sabouret. Un langage de description d'agents dédié à l'interaction dialogique. In *Modèles Formels d'Interactions*, pages 295–299, Lille, 2003.

[SRL88] E. Schopler, R.J. Reichler, and M. Lansing. *Stratégies éducatives de l'autisme Et des autres troubles du développement*. Collection Médecine et psychothérapie, 1988.

[Ste94] L. Steels. Emergent functionality in robotic agents through on-line evolution. In *Alife IV*, 1994.

[SWdH94] G. Schreiber, B. Wielinga, and R. de Hoog. Commonkads : Acomprehensive methodology for kbs development. *IEEE Expert*, pages 28–36, Décembre 1994.

[TBH86] J.M. Truong, A. Bonnet, and J.-P. Haton. *Systèmes experts, vers la maîtrise technique*. InterEditions, 1986.

[Tes97] C. Tessier. Reconnaissance de scènes dynamiques à partir de données issues de capteurs : le projet perception. In *Agard-CP-595 - Multi-sensor and data fusion for telecommunications, remote sensing and radars*, Lisbonne, Portugal, September 1997.

[Tha01] D. Thalmann. The foundations to build a virtual human society. In Springer-Verlag, editor, *Intelligent Virtual Actors (IVA), Lecture Notes in Artificial Intelligence (2190)*, pages 1–14, Madrid, Spain, 2001.

[TJK99] B. Trousse, M. Jaczynski, and R. Kanawati. Une approche fondée sur le raisonnement à partir de cas pour l'aide à la navigation dans un hypermédia. In *Hypertexte & Hypermedia : Products, Tools and Methods*, Paris, 1999.

[Use04] UserLand Software, http ://www.xmlrpc.org/. *XML-RPC Home Page*, 2004.

[WB98] A.D. Wilson and A.F. Bobick. Recognition and interpretation of parametric gesture. In *ICCV International Conference on Computer Vision*, pages 329–336, 1998.

[Wit53] L. Wittgenstein. *The philosophical investigations.* New York : Macmillan, 1953.

[WJ94] M. Wooldridge and N.R. Jennings. Agents theories, architectures, and languages : A survey. In Wooldridge et Jennings Eds., editor, *lecture notes in artificial Intelligence 89*, pages 1–39, 1994.

[WM94] I. Watson and F. Marir. Case-based reasoning : A review. *The Knowledge Engineering Review*, pages 327–354, 1994.

[Woo81] W. Woods. Procedural semantics as a theory of meaning. In A. Joshi, B. Webber, and I. Sag, editors, *Elements of Discourse Understanding.*, MIT Press, Cambridge, MA, 1981.

Résumé

La prise en compte du comportement de l'utilisateur dans les applications interactives, qui incluent l'utilisateur humain, apparaît évidente pour la mise en œuvre d'un mécanisme de prise de décision rapide et adaptée à la situation. Il s'agit de définir un système capable de "comprendre" le comportement de l'utilisateur et d'y répondre, de manière personnalisée et en temps réel, par des activités adaptées en tenant compte des consignes du concepteur de l'application. Dans ce cadre, nous avons apporté notre contribution à la problématique de l'exécution adaptative par observation et analyse du comportement.

La première partie de notre travail concerne l'étude des approches d'analyse de comportements. Nous avons défini un formalisme basé sur l'observation. Il s'agit d'observations effectuées sur les *actions explicites* (clavier, souris, écran tactile, etc.) et *implicites* (concernant les expressions faciales) de l'utilisateur.

La seconde partie concerne les modèles du contrôle d'exécution en tenant compte du comportement de l'utilisateur. Nous avons conçu un modèle fondé sur le raisonnement à partir de cas permettant la génération des activités répondant à la description du profil de l'utilisateur, ses besoins et son comportement. Le principe de contrôle, que nous avons élaboré, consiste à déclencher un épisode de raisonnement à chaque fois qu'un comportement portant un intérêt particulier est détecté.

La dernière partie de notre travail est consacrée à la présentation du système que nous avons développé dans la cadre du Projet Autisme. Ce système propose des jeux éducatifs destinés à des enfants autistes. Il constitue un cadre d'application et de validation de notre contribution. Ce cadre applicatif présente les caractéristiques d'une exécution adaptative et interactive adéquate au comportement de chaque enfant autiste, comportement appréhendé par différents moyens (réponses au jeu et expressions faciales). Ainsi, le système en question, basé sur le paradigme *multi-agents*, inclut les connaissances de l'expert, une description du profil de l'enfant autiste et la dynamique de leurs interactions.

Mots-clés: Interaction Homme-Machine, exécution adaptative, raisonnement à partir de cas, analyse de comportements, observation, jeux, autisme.

Abstract

Taking into account of the user's behavior in the interactive applications, which include the human user, appears obvious for the formation of a mechanism of fast decision-making and adapted to the situation. It consists in extracting conclusions relating to user's behavior and provides in a real time personal way adequate activities, keeping in mind the expert's advice. Within this framework, we have contributed to the problem of the adaptive execution by observation and analysis of behavior.

The first part of our work is about behavior analysis. We thus defined a formalism based on the observation of the user's behavior. It consists in the observation carried out on the explicit and implicit actions of user. The first one, recovering the user's actions carried out on : mouse, touch screen keyboard. The second one, concern facial expression.

The second part of our work is about execution control of the activities by taking into account of the user's behavior. We have defined a model based on the case-based reasoning that allows the generation of activities answering the description of the user's profile, his behavior and these needs. Our strategy of control consists in launching an episode of reasoning each time that a particular behavior is detected.

The last part of our work is devoted to the presentation of our system developed within the framework of the Autism Project. It is about a system proposing the educational games intended for children with autism. This system constituted a framework of application and validation of our contribution. This application framework reflects significant characteristics of user-adapted interactions according to the perceived behavior, behavior apprehended by various means (answers to the game and expressions facial). Thus, the system in question, based on the *multi-agents* paradigm, includes the knowledge of therapists, the children profiles and the dynamics of their interactions.

Keywords: Human-Machine Interaction, adaptive execution, Case-Based Reasoning, analysis of behavior, observation, games, autism.