Raisonnement à partir de cas pour la configuration de jeux thérapeutiques destinés à des enfants autistes.

Karim Sehaba, Pascal Estraillier

L3i-Université de La Rochelle
Pôle Sciences et Technologie
17042 La Rochelle Cedex 1 – FRANCE
{ksehaba, pestraillier}@univ-lr.fr

Résumé : Les logiciels ludo-éducatifs classiques ont généralement une construction statique, permettant d'adapter les jeux qu'ils proposent seulement par rapport au profil de l'utilisateur, ce qui limite la qualité des résultats qu'ils sont susceptibles de fournir. Notre architecture utilise le raisonnement à partir de cas pour faire évoluer dynamiquement l'exécution des jeux en tenant compte naturellement du profil du joueur, mais aussi des consignes du concepteur du jeu et surtout du comportement du joueur. En particulier, nous observons les actions du joueur au cours de la session afin de "comprendre" son comportement et d'y répondre, de manière personnalisée et en temps réel. Nous avons appliqué nos résultats à un environnement logiciel conçu pour des enfants autistes par une équipe de pédopsychiatres.

Mots-clés : Architecture des logiciels ludo-éducatifs, Raisonnement à partir de cas, Adaptation, Comportement.

1 Introduction

Un des enjeux majeurs des médias interactifs, et en particulier *des jeux éducatifs*, consiste à maîtriser en temps-réel la chaîne de traitements complexes mettant en jeu une série d'extraction d'informations, d'optimisation de traitements, de comparaison avec des expériences passées en vue de déterminer la meilleure des réactions à provoquer.

Dans ce contexte, notre recherche consiste à définir un système capable de *"comprendre"* le comportement d'un *joueur* et d'y répondre, de manière personnalisée et en temps réel, par des activités adaptées en tenant compte des *consignes* du concepteur du jeu. L'objectif à terme est de faire interagir efficacement différents niveaux d'analyse de scènes et/ou d'événements pour les intégrer dans un système d'observation et d'analyse de comportement en vue d'aide à la décision.

Le contexte applicatif s'articule autour du projet *Autisme*, en partenariat avec le service de pédopsychiatrie de l'hôpital de La Rochelle. Il s'agit de proposer aux enfants souffrant d'autisme, un processus d'apprentissage efficace de consignes par manipulations interactives d'outils informatiques.

Dans un premier temps, nous avons établi une cartographie des comportements d'enfants présentant des troubles autistiques. L'observation de ces comportements, lorsqu'ils sont inscrits dans un registre de significations, réside dans la mise en évidence d'indicateurs précoces de risque de passages à l'acte ou même d'actions parasites comme la rupture, la colère, l'évitement,... Cette analyse permet, dans une seconde étape, d'améliorer l'adéquation entre les actions du système et les comportements des enfants dans le but de les maintenir attentifs et donc réceptifs vis-à-vis de l'outil informatique.

Il s'agit donc, à partir de différentes sessions de jeu, de faire évoluer une base de cas (bien évidemment qualifiés par l'équipe médicale). Lorsqu'une activité est en cours, il sera nécessaire d'identifier, à partir du comportement de l'enfant, le cas le plus approprié pour choisir, en fonction des directives du concepteur du jeu, l'activité correspondant à la meilleure évolution possible de la session de jeu.

Un paradigme adapté …

Le raisonnement à partir de cas est un paradigme de résolution de problèmes s'appuyant sur la réutilisation d'expériences passées, stockées dans une *base de cas*, pour résoudre de nouveaux problèmes appelés *cas cibles*. Le principe fondamental de ce paradigme consiste à chercher dans la base de cas des cas similaires au cas cible et de les adapter 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) est utilisé dans divers domaines d'applications telles que la supervision industrielle (Fuchs, 1997), l'aide à la navigation dans l'hypermédia (Trousse et *al.*, 1999), l'accidentologie et le droit médical (Despres, 2002),... On s'intéresse ici à l'utilisation du RàPC dans le domaine des systèmes d'apprentissage et des jeux éducatifs. Dans ce cadre, le RàPC a surmonté les limitations des approches basées sur une base de règles comme le souligne (Watson & Marir, 1994).

De fait, de nombreux systèmes d'apprentissage (Elorriaga & Fern, 2000) (Funk & Conlan, 2002) utilisent le RàPC dans leur processus de raisonnement pour générer des activités éducatives. Ces travaux mettent en avant l'aspect d'adaptabilité des activités proposées par le système par rapport au profil de l'utilisateur, mais sans tenir compte des aspects liés à la *dynamicité* du système, et à *l'observation du comportement* de l'utilisateur. Pourtant, dans de nombreuses applications liées aux logiciels éducatifs et thérapeutiques pour enfants, ces deux aspects sont importants pour plusieurs raisons :

- Les utilisateurs évoluent et les activités proposées par le système peuvent devenir incohérentes au cours de la session d'apprentissage ce qui motive le choix d'un système dynamique.
- L'analyse comportementale est un facteur clef pour les thérapeutes. Son intérêt réside dans la mise en évidence d'indicateurs précoces qui peuvent alerter sur le risque de décompensations, de passages à l'acte ou même d'actions parasites (comme 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 vis-à-vis de l'outil informatique.

Nous proposons une architecture d'un système permettant de :

- Construire une séquence d'activités répondant aux buts éducatifs que l'utilisateur veut/doit atteindre en tenant compte de son profil.
- Observer en permanence les actions de l'utilisateur au cours de la session et leurs associer une interprétation de 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 son observation, le système détecte les cas où les activités proposées ne répondent plus à l'évolution actuelle du comportement de l'utilisateur et met à jour la séquence d'activités en pareil cas.

La structure de cet article est la suivante : la deuxième partie définit les termes utilisés dans notre approche. La partie suivante donne une description générale de l'architecture du système. Nous détaillons ensuite le processus de décision. Il s'agit de présenter le modèle de représentation des cas, le processus de raisonnement ainsi que le mécanisme de gestion d'exceptions. Enfin, la dernière partie présente quelques résultats de l'application.

2 Définitions – Hypothèses

Avant de présenter notre approche, il nous semble nécessaire de présenter certains termes. Ainsi,

- Un ***Jeu*** possède des paramètres de configuration et des objectifs à atteindre. Il est caractérisé par :
  - un ***Décor*** statique,
  - une collection d'***Objets*** (pictogramme, boule, image, musique,...) statiques ou munis de comportements qui définissent les formes d'interactions associées.
  - des ***Actions*** possibles du joueur, en particulier la manipulation des objets au moyen de l'IHM.
  - des ***règles*** de fonctionnement (les règles du jeu) relatives aux objets et aux actions possibles.
- Une ***Activité*** est une instance d'un jeu (avec une configuration particulière et des objectifs qualifiés et quantifiés).
- Un ***Protocole*** est une séquence d'activités ordonnées, déterminée en vue de permettre aux joueurs de réaliser des objectifs complexes.
- Un ***Cas*** est un protocole associé à un certain profil du joueur, lui permettant d'atteindre des objectifs parfaitement caractérisés (qualitativement, quantitativement).
- Une ***Consigne*** (ou ***directive***) caractérise un état du système (et en particulier son évolution liée au comportement du joueur) 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***. Le Joueur est défini par un profil le caractérisant. L'Expert est caractérisé par son savoir-faire dans un domaine donné. C'est lui qui définit le Jeu, détermine les Activités associées et définit les différents Cas en adaptant un Protocole au profil du Joueur. Son rôle consiste également à définir les Consignes associées au Joueur.

3 Présentation générale de l'architecture

L'architecture générale que nous proposons, présentée dans la figure 1, est construite à partir de trois sous systèmes : un système d'observation et d'analyse de comportement, un système de décision, et un système d'action.

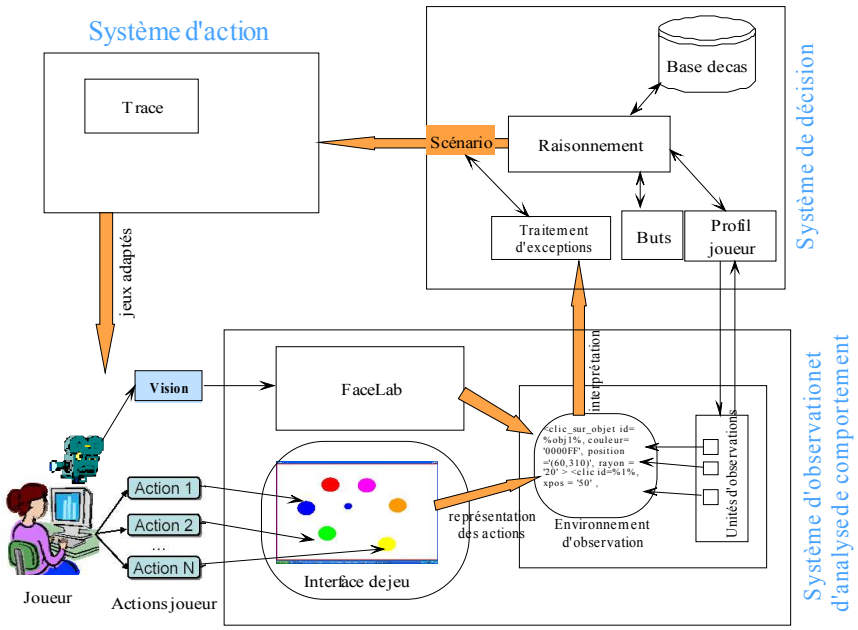

Fig. 1 – L'architecture générale.

Système d'observation et d'analyse de comportement : En s'inspirant de *la théorie des affordances* (Gibson, 1977) (Gibson, 1979) *et la théorie de la sémantique procédurale* (Wittgenstein, 1953) (Johnson-laird, 1977) (Woods, 1981), ce système est chargé de récupérer les actions du joueur et de leurs associer des mots d'état caractérisant son comportement. L'observation est faite selon deux approches, *Action logicielle* et *Vision*. Dans la première approche, il s'agit de récupérer les actions du joueur sur les périphériques : souris, écran tactile, clavier… Dans l'approche vision, assurée par le système logiciel-matériel **FaceLab**, il s'agit de mesurer les

caractéristiques concernant la représentation 3D du visage et de l'orientation du regard.

Système de décision : ce système utilise le raisonnement à partir de cas (Kolodner, 1993), pour générer des protocoles adaptés au profil du joueur. Les protocoles engendrés par ce système peuvent et doivent être modifiés lorsqu'ils se révèlent obsolètes au cours de la session d'apprentissage. Ce qui garantit une exécution adaptative.

Système d'action : ce système est chargé d'exécuter les activités du protocole fourni par le système de décision. Une autre tâche importante de ce système est la sauvegarde de la trace d'exécution. La trace d'exécution doit permettre de retrouver exactement ce que le système avait prévu de faire faire au joueur (les activités qui lui proposées et ce qu'il avait réalisé). Ces informations ont plusieurs intérêts. Dans le contexte de l'autisme, la trace peut permettre de noter l'évolution de l'enfant; elle sera aussi 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"...

4 Système de décision

Le système de décision doit fournir au système d'action, un protocole adapté au profil du joueur et aux buts éducatifs à atteindre. Pour assumer cette tâche, ce système contient cinq modules : **Buts**, **Profil du joueur**, **Base de cas**, **Raisonnement** et **Traitement d'exceptions**.

Dans chaque session, certains **buts** éducatifs sont en activité dans le système de décision et certaines informations caractérisant le joueur, sont tenues dans le module **Profil du joueur**. Étant donnés les buts existants et le contenu du profil du joueur, le module de **Raisonnement** permet de générer un protocole *adapté* à ces informations. Le protocole généré sera ainsi placé dans le système d'action pour son exécution. Le module de raisonnement utilise le raisonnement à partir de cas (Kolodner, 1993) qui consiste à réutiliser les connaissances spécifiques des expériences précédemment expérimentées appelées **cas sources** et stockées dans une **base de cas**, pour résoudre de nouveaux problèmes. Un nouveau problème, appelé **cas cible**, est résolu en *sélectionnant* un cas source similaire à celui-ci. Le cas sélectionné sera *adapté* à la nouvelle situation du cas cible. Toute nouvelle expérience dont le degré de similarité aux cas sources excède un certain seuil sera *mémorisée* dans la base de cas, la rendant immédiatement disponible pour des problèmes futurs.

Atteindre un but revient à construire un protocole. Mais il ne s'agit pas d'une exigence rigide, les protocoles engendrés par le système de décision peuvent et doivent être modifiés lorsqu'ils se révèlent désuets dans le cas de l'apparition d'événements exceptionnels au niveau des actions du joueur. Les exceptions sont utilisées pour notifier une situation indésirable qui empêche l'exécution standard du protocole. **Le module de traitement d'exceptions** permet d'identifier, de signaler et de traiter ces exceptions.

4.1 Représentation des cas

Les connaissances sur la façon d'accomplir les buts sont représentées dans le système par un ensemble de cas sources. Dans notre architecture, un cas source contient deux composants : *une description* et un *protocole d'activités.*

- **La description de cas :** ce composant contient des *descripteurs* liés aux buts et au profil du joueur. Les descripteurs sont représentés par un couple de la forme « attribut, valeur ». Les descripteurs liés aux buts décrivent la nature des buts. Par exemple : *« Perception-auditive, haut »*, *« Perception-visuelle, moyen »*. Les descripteurs liés au profil contiennent des informations concernant les connaissances et les préférences du joueur - par exemple : *« Type-joueur, débutant »*, *« couleur, vert »*, *« Durée-session, [15.30] »*. Les descripteurs des cas sont utilisés pour calculer le degré de similarité entre le cas cible et les cas sources dans le processus de raisonnement (voir 4.3).
- **Le protocole** : ce composant contient une séquence d'activités.

Une fois que la structure d'un cas a été présentée, un autre problème de base est celui de l'organisation de la base de cas (Schank, 1982) (Schank, 1989).

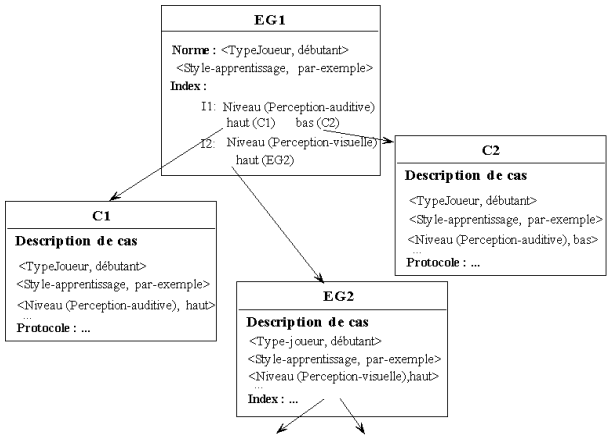

Fig. 2 –Structure de la base de cas.

L'organisation de la base de cas que nous adoptons est basée sur le modèle à mémoire dynamique de Schank (Schank, 1982). L'idée fondamentale est d'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 ».
- ***Cas (Cases)*** : ils représentent des expériences individuelles.
- ***Index (Indices)*** : ils représentent les différences entre les cas d'un même *EG*. Chaque index est lié à un attribut concret d'un descripteur et contient

une liste de paires « valeur, nœud ». Chaque paire d'index lie son *EG* avec un autre nœud (cas ou *EG*) correspondant.

Ainsi se forme un graphe hiérarchique (voir l'exemple de la figure 2) dont les nœuds sont soit des *EGs* soit des cas. Les arcs représentent les liens entre les index et les nœuds.

4.2 Profil du joueur

Le profil du joueur a des fonctionnalités multiples, utilisé à différents moments par le système de décision, plus particulièrement, dans le processus de raisonnement présenté à la section 4.3. Il intervient aussi dans l'interprétation des actions du joueur par le système d'observation et d'analyse de comportement. Plusieurs types d'informations concernant le joueur sont présents :

- Des informations générales
- Des connaissances du domaine
- Des préférences
- L'histoire

Les informations générales concernent l'identité du joueur, p. ex. : nom, prénom, date de naissance,... Les connaissances du domaine et les préférences donnent une description du profil du joueur identique à celle mentionnée dans la description de cas (voir la section 4.1). Ces informations sont donc représentées par une liste de couples de la forme « attribut, valeur».

L'historique est vu comme un agenda de tout ce qui est arrivé au joueur lors de chacune des sessions. Il doit permettre de retrouver exactement ce que le système avait prévu de faire faire au joueur, les activités qui lui ont été proposées et ce qu'il avait réalisé.

4.3 Processus de raisonnement

Le module de raisonnement du système de décision utilise le raisonnement à partir de cas (Kolodner, 1993), pour générer des protocoles en sélectionnant des cas sources, de la base de cas, similaires au cas cible et en les adaptant à la situation actuelle du cas cible. Le nouveau cas 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 : *La remémoration*, *L'adaptation* et *La mémorisation*.

4.3.1 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.

L'appariement est défini dans (Kolodner, 1993) par *"Matching is the process of comparing two cases to each other and determining their degree of match"*. La méthode d'appariement que nous avons utilisée est *le plus proche voisin* (k-nearest-neighbors). Cette méthode, utilisée dans (Elorriaga & Fern, 2000) (Voß, 1994) (Leake, 1996), possède l'avantage d'être *"très simple à implémenter"* (Bisson, 2000) 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 termes d'attributs appropriés des descripteurs de cas (cible ou source). Rappelons que les descripteurs d'un cas concernent les buts et le profil du joueur.

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 numérique Φ qui calcule le degré de similarité entre un cas cible *C* et un cas source *S*. La fonction Φ est définie comme étant la moyenne pondérée des valeurs de similarité sur chacun des descripteurs de *C* et *S*. Le schéma (1) montre la fonction de similarité Φ où :

- *D* est l'ensemble des descripteurs du cas cible *C*.
- W_d est le coefficient d'importance du descripteur *d*. 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.
- $\varphi_d(C,S)$ est la similarité entre deux valeurs du descripteur *d* associés à *C* et *S*.

$$\phi\ (C,S) = \frac{\sum_{d\in D} W_d * \varphi_d(C,S)}{\sum_{d\in D} W_d} \qquad (1)$$

Le deuxième filtre minimise l'effort d'adaptation des cas sources à la situation actuelle du cas cible. L'effort d'adaptation est calculé en fonction des buts et 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.

La recherche dans la base de cas dépend fortement de sa structure. Dans une structure hiérarchique, la recherche est guidée par les attributs partagés de chaque *EG* (normes) 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 le seuil. 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 les méthodes classiques de recherche d'intelligence artificielle. Plusieurs algorithmes de recherche peuvent être combinés (p. ex.: *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*.

4.3.2 L'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 (d'Aquin *et al*. 2004) et à la nécessité d'une base de connaissances intensives (Elorriaga & Fern, 2000) et un coût en termes de temps et d'efforts.

Étant donné que notre application est destinée aux enfants autistes accompagnés, une intervention de l'accompagnateur au cours de la session peut perturber l'enfant avec un risque de rupture, ce qui motive une stratégie d'adaptation automatique. La stratégie que nous avons adoptée distingue deux types d'adaptation, *globale* et *locale*.

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

L'adaptation locale permet de régler les paramètres de configuration des activités du protocole du cas candidat avec des valeurs mieux adaptées à la description du cas cible, en d'autres termes adaptées aux préférences et aux connaissances du joueur.

4.3.3 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 : l'évaluation et la sauvegarde.

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

Dans la dimension éducative, le souci des données concerne juste le profil du joueur : fondamentalement, les changements des préférences et connaissances recueillies pendant la session et les erreurs faites par le joueur. Nous pensons que l'automatisation de cette dimension est difficile dans le contexte de l'autisme, dans la mesure où l'enfant est très susceptible aux perturbations de son environnement. Ces perturbations ne peuvent pas êtres contrôlées par le système (bruits, ouverture de la

porte de la salle,…). L'accompagnateur 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.

L'objectif de l'évaluation dans la dimension informatique 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.

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* qui contiendra le nouveau cas, l'enchaînement du nouveau cas à *l'EG*, et la généralisation des cas/*EG*.

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* est 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.

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 Gestion d'exceptions

Le système de décision établit un lien entre les informations concernant le profil du joueur et les buts pour générer un protocole adapté. L'aptitude du protocole à répondre de façon satisfaisante à l'ensemble des actions du joueur au cours de la session, en d'autres termes au cours de l'exécution du protocole, dépend directement du caractère exhaustif des événements prévus par le protocole. Cette approche ne peut convenir que pour les domaines d'applications quasi-statiques. Cette hypothèse est contraignante dans le contexte de l'autisme dans la mesure où, d'une part, il est difficile que le protocole puisse recenser toutes les actions possibles que l'enfant peut

effectuer. D'autre part, des comportements atypiques (p.ex. rupture, évitement,…) de l'enfant peuvent apparaître. Dans ce contexte et dans un souci de fiabilité du système, aussi bien lors de la phase de génération du protocole qu'en cours de son exécution, il faut que le système de décision ait la capacité de réagir à tout moment aux actions (typiques et atypiques) de l'enfant afin de pouvoir adapter le protocole aux nouveaux besoins qui apparaîtront et de nouvelles possibilités qui s'offriront.

Les actions du joueur sont considérées par le système comme des événements qui peuvent êtres des événements classiques (associés aux actions typiques) ou des événements exceptionnels (associés aux actions atypiques). Les premiers peuvent être traités par le protocole standard généré par le module de raisonnement.

Dans le contexte de la programmation, les événements exceptionnels (souvent appelés exceptions (Goodenough, 1975)) sont gérés par des mécanismes de gestion d'exceptions (Weinreb, 1983) (Meyer, 1988) (Koenig & Stroustrup, 1989). Le principe de ces mécanismes est utilisé aussi dans le contexte d'agents (Souchon *et al.*, 2004). Dans notre contexte, les exceptions sont utilisées pour notifier une situation indésirable qui empêche l'exécution standard du protocole de se poursuivre, telles qu'un évitement ou une rupture de la part de l'enfant. Le module de traitement d'exceptions fournit des structures de contrôle et des mécanismes permettant de *détecter*, de *signaler* et de *traiter* les exceptions.

La détection d'une exception interrompt l'exécution standard du protocole. Sa continuation est alors prise en charge par le module de gestion d'exceptions qui procède au signalement de l'exception, c'est-à-dire à la recherche d'un traitement pouvant être appliqué pour gérer cet événement. De tels traitements sont définis dans des gestionnaires d'exceptions. L'exécution des actions correctives, définies par les gestionnaires d'exceptions, permet soit de reprendre l'exécution là où le protocole a été interrompu (repris), soit en un autre point, en abandonnant tout ou partie du protocole en cours d'exécution au moment de la détection de l'exception (terminaison).

5 Résultats

Dans cette section, nous présentons l'implémentation de notre architecture dans le cadre du projet *Autisme*. Dans un premier temps, nous avons développé une plate-forme qui assure le processus de raisonnement et le traitement d'exceptions. Pour le traitement d'exception, nous avons développé un agent qui observe les actions du joueur pendant la session et réagit en cas d'exceptions en modifiant le protocole.

Dans une seconde étape, nous avons développé une interface permettant à l'expert de définir les activités (voir la figure 3), les cas et les distances entre les différentes valeurs des attributs de chaque descripteur. Ces informations ainsi que le profil du joueur sont stockés dans un serveur de données.

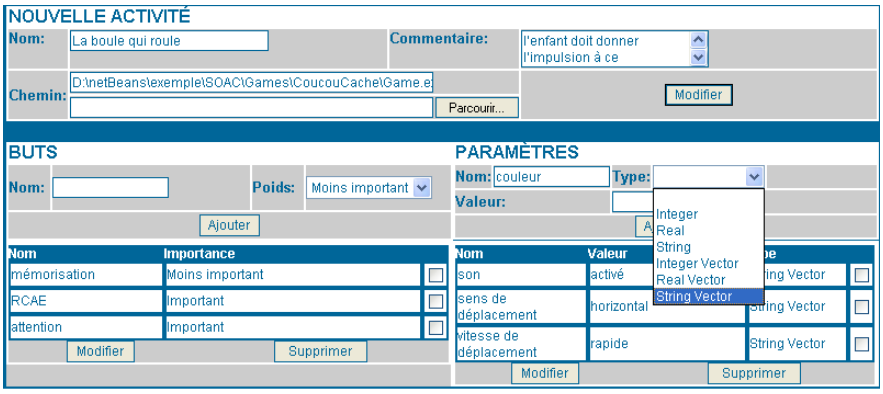

Fig. 3 – Fenêtre de création d'une activité.

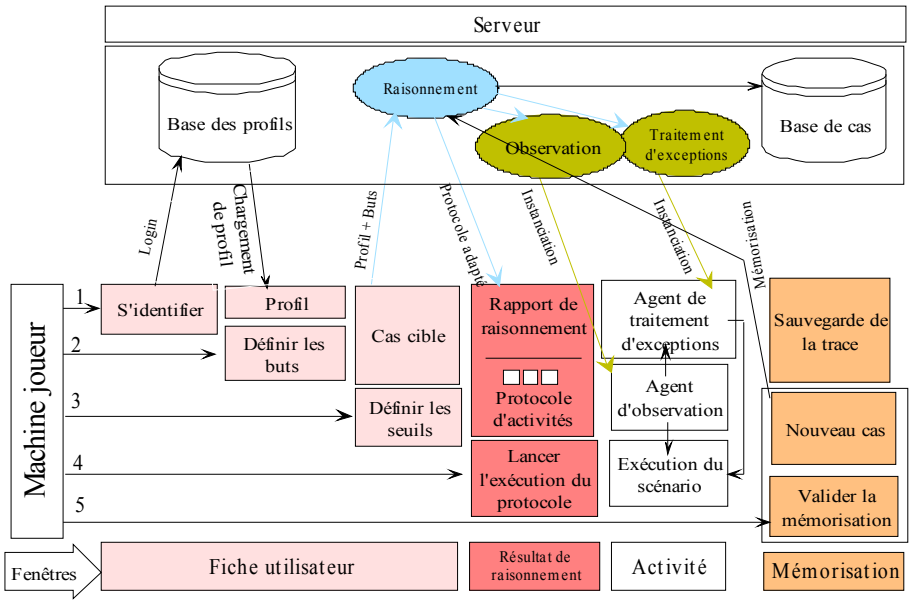

Fig. 4 –Implémentation.

La figure 4 illustre notre exemple. Chronologiquement, le joueur se connecte au serveur, son profil sera alors chargé. Ensuite, le joueur est prié de spécifier les buts qu'il veut atteindre. Une fois ces informations entrées, un cas cible est créé. Le joueur définit les seuils de raisonnement. Le cas cible ainsi que les seuils sont transmis par message au serveur. Le message est intercepté par l'agent de raisonnement. Le rôle de cet agent est de générer un protocole adapté à la situation actuelle du cas cible en utilisant le RàPC (voir la section 4.3).

Un rapport de raisonnement sera envoyé au joueur. 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 protocole du cas source sélectionné. Le joueur lance les activités. Pendant la session, une instance d'agent d'observation suit en permanence les actions du joueur. Chaque exception détectée par l'agent observateur sera traitée par l'agent de traitement d'exceptions.

A la fin de la session, le joueur valide (dans la dimension éducative) ou non le cas. Dans le cas favorable, le logiciel procède à l'évaluation dans la dimension informatique pour sa mémorisation.

6 Conclusion

Dans cet article, nous avons développé une architecture permettant au système visé d'adapter de façon dynamique son exécution par observation et analyse de comportement. Le principe de l'architecture repose sur l'optimisation de l'extraction de l'information en fonction du contexte. L'observation et l'analyse de comportement consistent à déterminer le comportement du joueur à partir de ces actions, en tenant compte des consignes de l'expert. L'approche que nous avons adoptée est inspirée des travaux issus de l'Interaction en Langue Naturelle.

Nous avons utilisé le RàPC pour la prise de décision. Le RàPC nous a permis de doter le système d'un mécanisme intelligent de création d'activités adaptées au profil du joueur. Afin de doter le système de décision d'un mécanisme dynamique lui permettant de réagir à tout moment, même au cours de la session, nous avons utilisé le mécanisme de gestion d'exception qui tient compte des consignes de l'expert. Il s'agit de détecter des comportements particuliers, définis par l'expert, et de répondre par des actions au niveau du protocole.

Le système que nous avons développé est actuellement testé au service de pédopsychiatrie de l'hôpital de La Rochelle.

Références

BISSON G. (2000). La similarité : une notion symbolique/numérique. Apprentissage symbolique-numérique (tome 2). In CEPADUES, Ed. p. 169–201.

D'AQUIN M., BRACHAIS S., LIEBER J. & NAPOLI A. (2004). 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. p.41-52. Université Paris Nord, Villetaneuse.

DESPRES S. (2002). Contribution à la conception de méthodes et d'outils pour la gestion de connaissances. Habilitation à diriger des recherches**,** Université de Paris V.

ELORRIAGA J. & FERN NDEZ-CASTRO I. (2000). Using case-based reasoning in instructional planning : towards a hybrid self-improving instructional planner. In International Journal of Artificial Intelligence in Education, p. 416–449.

FUCHS B. (1997). Représentation des connaissances pour le raisonnement à partir de cas : le système Rocade. Thèse de doctorat, Université Jean Monnet de Saint-Etienne.

FUNK P. & CONLAN O. (2002). 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, p. 15–23.

GIBSON J.J. (1977). The theory of affordances. In R. Shaw & J. Bransford (eds.), Perceiving, Acting and Knowing. Hillsdale, NJ: Erlbaum.

GIBSON J.J. (1979). The ecological approach to visual perception. Chapter 14: The theory of information pickup and its consequences. p. 238-263. Boston: Houghton Miflin Co.

GOODENOUGH J. B. (1975). Exception handling : issues and a proposed notation. In Communications of the ACM.

JOHNSON-LAIRD P. (1977). Procedural semantics. In Cognition, p. 189–214.

KOENIG A. R. & STROUSTRUP B. (1989). Exception handling for c++. In Proceedings of the C++ at Work conference, p. 322–330.

KOLODNER J. (1993). Case-based reasoning. In Morgan Kaufmann Pub.

LEAKE D. (1996). Cbr in context : The present and the future. In Case-BasedReasoning : Experiences, Lessons, and Future Directions.

MEYER B. (1988). Disciplined exceptions. In Technical report tr-ei-22/ex, Interactive Software Engineering.

SCHANK R. (1982). Dynamic memory ; a theory of reminding and learning in computers and people. In Cambridge University Press.

SCHANK R. & RIESBECK C.K. (1989). Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Inc. Hillsdale, New Jersey.

SOUCHON F., VAUTTIER S., URTADO C. & DONY C. (2004). Fiabilité des applications multiagents : le système de gestion d'exceptions sage. In Actes JFSMA'04, LAVOISIER, Ed., p.121–134.

TROUSSE B., JACZYNSKI M. & KANAWATI R. (1999). Une approche fondée sur le raisonnement à partir de cas pour l'aide à la navigation dans un hypermédia, In proceedings of Hypertexte & Hypermedia : Products, Tools and Methods, Paris.

VOß (ED) 1994. *Similarity Concepts and Retreival Methods*. FABEL project Technical Report number 13. Gesselschaft für Mathematik und Datenverarbeitung (GMB). Sankt Augustin.

WATSON I. & MARIR F. (1994). Case-Based Reasoning: A Review. The Knowledge Engineering Review, p. 327-354.

WEINREB D. L. (1983). Signalling and handling conditions. In Technical report, ambridge, MA, USA.

WITTGENSTEIN L. (1953). The philosophical investigations. New York : Macmillan.

WOODS W. (1981). Procedural semantics as a theory of meaning. In E. of discourse understanding, ed., a. joshi, b. webber and i. sag.