Un assistant pour la conception et le développement des systèmes de RÀPC

Amélie Cordier, Béatrice Fuchs

LIRIS, UMR 5205 CNRS
Université Claude Bernard
43, Boulevard du 11 Novembre 1918
69622 VILLEURBANNE CEDEX
http://liris.cnrs.fr/
{acordier, bfuchs}@liris.univ-lyon1.fr

Résumé : Le raisonnement à partir de cas (RÀPC) est un paradigme de raisonnement complexe ce qui rend délicat le processus développement ou des systèmes qui mettent en œuvre ce type de raisonnement. Dans cet article, nous présentons une ébauche d'architecture pour un environnement d'assistance à la conception et au développement d'applications de RÀPC. Notre approche s'appuie sur les travaux en ingénierie des connaissances avec la prise en compte les deux niveaux: «connaissance» et «symbole». L'outil proposé s'appuie sur les représentations des connaissances par objets dont la richesse des fonctionnalités répond bien aux besoins du RÀPC mis en évidence.

Mots-clés : Raisonnement à Partir de Cas, Représentation des connaissances par objets, Adaptation, Apprentissage.

1 Introduction

Le Raisonnement à Partir de cas (RÀPC) est un paradigme de raisonnement qui s'appuie sur la remémoration de problèmes passés résolus, appelés les cas sources, pour résoudre de nouveaux problèmes, appelés les problèmes cibles (Aamodt & Plaza, 1994). Ce paradigme a été utilisé dans de nombreux systèmes industriels pour résoudre des problèmes dans des domaines d'application variés. Le RÀPC a souvent été présenté comme une solution au goulet d'étranglement de l'étape d'acquisition des connaissances grâce à l'utilisation de connaissances expérimentales, les cas, qui sont plus faciles à recueillir. Bien que certains arguments présentent le RÀPC comme une solution nettement plus facile à mettre en oeuvre que d'autres systèmes à bases de connaissances, force est de constater qu'en pratique, il est impossible de faire l'économie de l'effort d'acquisition de connaissances. Le développement des systèmes de RÀPC se heurte toujours à un problème d'ingénierie des connaissances. En particulier, les connaissances

d'adaptation sont difficiles à modéliser, les fonctionnalités d'apprentissage sont limitées à la mémorisation des cas et l'étape de révision, quant à elle, est trop souvent rudimentaire. Ces systèmes implantent partiellement les fonctionnalités du RÀPC et ont par conséquent un intérêt limité au regard des possibilités théoriques de ce type de raisonnement. Pourtant, des travaux ont été menés afin de modéliser les étapes du raisonnement et aider à leur développement. Mais dans les faits, il est complexe et coûteux de suivre les recommandations issues de ces études, d'autant qu'elles ne sont complètement réifiées dans aucun outil de développement. Même si certains environnements proposent quelques méthodes et outils pour assister la conception d'applications de RÀPC, ceux qui permettent de prendre en compte tous les besoins potentiels des concepteurs d'applications font cruellement défaut. Rares sont les systèmes dotés de possibilités d'extension suffisantes pour personnaliser une application particulière.

Dans cet article, nous nous intéressons à la problématique du développement d'applications de RÀPC ainsi qu'aux pistes de recherche associées. Inspirés par des travaux antérieurs (Fuchs, 1997), nous souhaitons exploiter les avantages des représentations des connaissances par objets pour de telles applications. Nous présentons une idée d'architecture pour la conception d'un environnement d'assistance au développement d'applications de RÀPC. Dans la deuxième partie, nous évoquons les différents travaux menés autour du RÀPC puis nous passons en revue les outils de développement associés que nous considérons en particulier sous l'angle de la représentation des connaissances. Dans la troisième partie, nous proposons l'exploitation de ces résultats au sein d'un outil unique, ouvert et extensible pour l'assistance à la conception d'applications de RÀPC.

2 Raisonnement à partir de cas : modèles et outils

Le raisonnement à partir de cas est un paradigme de résolution de problèmes qui cherche à résoudre un problème cible en s'appuyant sur une base de cas passés résolus. Un cas est constitué d'un problème source et de sa solution source associée. Le RÀPC peut être modélisé par un cycle constitué de cinq étapes - élaboration, remémoration, adaptation, révision et apprentissage - gravitant autour d'une base de connaissances du domaine d'application. Chacune des étapes du cycle mobilise ces connaissances pour supporter la recherche de la solution du problème cible.

Le RÀPC trouve ses origines dans les travaux sur la mémoire dynamique de Schank (Schank, 1982) qui s'intéressaient particulièrement au rôle joué par la remémoration d'expériences passées dans les processus d'apprentissage et de résolution de problèmes. Le RÀPC s'inspire également du raisonnement par analogie dont il est considéré comme un cas particulier. Les travaux successifs sur le RÀPC ont conduit à la réalisation d'outils informatiques de résolution de problèmes s'appuyant entièrement sur ce mode de raisonnement. Si les premières recherches sont issues des travaux en psychologie cognitive, les recherches actuelles sont pluridisciplinaires. Elles ont des ramifications dans les domaines variés de l'intelligence artificielle : représentation des connaissances, classification, mesures de similarité, apprentissage, etc., ce qui en fait un mode de raisonnement complexe. Afin d'appréhender cette complexité, nous allons cerner certains de ces problèmes étape par étape puis plus globalement.

2.1 Les étapes du RÀPC

La première étape du RÀPC est de construire la spécification du problème à résoudre. À partir d'une requête initiale soumise au système, des informations et connaissances sont inférées afin de mieux orienter la recherche pour un objectif et une tâche précis. Élaborer un cas pertinent, c'est-à-dire qui à la fois, reflète le problème courant et soit exprimé en termes cohérents par rapport à la base de connaissances utilisée est une vraie question d'ingénierie des connaissances. Le problème cible élaboré est ensuite utilisé dans l'étape de remémoration pour retrouver un cas «jugé similaire» dans la base de cas. Déterminer la similarité entre deux cas est loin d'être une opération triviale. Elle implique l'utilisation de mesures et de connaissances de similarités fortement liées au domaine d'application. Les travaux qui s'intéressent à la similarité sont nombreux. Des mesures de similarité génériques ont été proposées, et une étude de celles-ci est proposée dans (Rifqi, 1996). Durant l'étape d'adaptation, la solution du cas source est adaptée pour obtenir une solution au problème cible. L'adaptation est un des processus les plus délicats du RÀPC. C'est d'ailleurs pour cette raison qu'il est très souvent négligé ou délégué à l'utilisateur de l'application RÀPC. Dans (Fuchs *et al.*, 2000), les auteurs ont proposé une approche de l'adaptation s'appuyant sur la notion d'influence du problème sur la solution qui, combinée avec les appariements effectués au moment de la remémoration, permet d'adapter la solution du cas cible. Longtemps, les connaissances utilisées pour la remémoration et pour l'adaptation ont été considérées comme distinctes. (Smyth & Keane, 1993) a proposé d'utiliser les connaissances d'adaptation au moment de la remémoration pour privilégier la remémoration des cas minimisant l'effort d'adaptation. Si la remémoration est faite en anticipant correctement sur l'adaptation, cette dernière en sera d'autant plus facilitée et l'on sera sûr d'avoir un cas adaptable. Les connaissances de similarité et d'adaptation apparaissent alors comme étant duales voire même confondues. C'est lors de la phase de révision que la solution proposée peut être corrigée, acceptée ou refusée par l'utilisateur. Cette étape permet d'évaluer l'adaptation. Elle permet également de préparer l'apprentissage puisqu'elle fait émerger de nouvelles connaissances : une nouvelle solution (acceptée ou refusée) mais aussi des connaissances d'adaptation si certaines corrections ont été apportées par l'utilisateur (Aamodt, 1991), ou des connaissances de remémoration si celle si n'est pas jugée satisfaisante (Fox & Leake, 1994). Le résultat de l'évaluation de la solution met en évidence une insuffisance de l'adaptation à produire une solution satisfaisante ou de la remémoration à sélectionner le cas adéquat. L'étape de révision permet également d'évaluer l'utilité du cas nouvellement résolu et d'élaborer une stratégie de rétention ou d'oubli des cas selon leur contribution à la compétence du système (Smyth & Keane, 1995). La phase d'apprentissage soulève également un certain nombre de problématiques de recherche. Les questions qui se posent sont avant tout de savoir quelles sont les connaissances qui doivent être apprises et comment les apprendre. La plupart des recherches portent sur l'apprentissage des cas passés résolus, des méthodes d'indexation et de l'organisation de la base de cas, mais plus rares sont les recherches qui comme (Aamodt, 1991) s'intéressent à l'apprentissage de connaissances implicites telles que les connaissances de similarité ou d'adaptation.

2.2 Les modèles du RÀPC

Plus globalement, certaines études s'intéressent au RÀPC dans son ensemble, par exemple en s'appuyant sur des explications du raisonnement (Massie *et al.*, 2004). Ces explications sont souvent très utiles pour comprendre pourquoi certaines solutions qui paraissent incohérentes ont été proposées et donc pour identifier les failles dans le raisonnement. Elles peuvent également servir de point de départ et être complétées par l'utilisateur pour fournir une solution plus élaborée. On trouve également des travaux visant à proposer un modèle unifié du RÀPC (Fuchs *et al.*, 1999). D'autres travaux se sont intéressés à la représentation des connaissances pour le raisonnement à partir de cas, et en particulier à la représentation des cas ainsi qu'à leur structuration dans une base de cas : (Gomez-Albarran *et al.*, 1999) propose une modélisation du RÀPC en logiques de descriptions, (Napoli, 1992) et (Lieber, 1997) se sont intéressés au lien entre le RÀPC et le raisonnement par classification dans le cadre des RCO, (Aamodt, 1991) a proposé un environnement de représentation des connaissances de type réseau sémantique et des mécanismes de raisonnement associés pour le RÀPC et (Fuchs, 1997), propose de traiter ces aspects comme des problèmes de représentation des connaissances. Un système de RÀPC est un système à base de connaissances et s'intéresser à la conception d'applications de RÀPC nécessite donc de s'intéresser également aux outils de représentation des connaissances.

2.3 Le développement des systèmes de RÀPC

Dans le domaine du développement des systèmes de RÀPC, deux tendances ont été explorées. La première est celle des outils de développement d'applications, comme par exemple Orenge (tec:inno, 2000) ou CBR*TOOLS (Jaczynski & Trousse, 1998), dont l'étude se situe en grande partie au niveau «symbole» et relève donc principalement du génie logiciel. Ces outils implantent une partie des fonctionnalités du RÀPC (Lenz *et al.*, 1998) en proposant en particulier des stratégies de remémoration efficaces, mais ils n'intègrent pas toujours explicitement l'adaptation et l'apprentissage dans le processus de développement, ni un modèle explicite du raisonnement. Le modèle implicite sous-jacent ne permet pas facilement de personnaliser une application nécessitant des fonctionnalités spécifiques. Les mécanismes de ces outils sont la plupart du temps prédéfinis et adaptés à certaines classes d'applications (diagnostic, aide à la décision par exemple).

La seconde catégorie d'environnements de développement est celle des systèmes de représentation de connaissances (SRC) tels que Creekl (Aamodt, 1991) ou Noos (Plaza & Arcos, 1993). Ces systèmes ne sont pas dédiés au RÀPC, mais sont plutôt des systèmes génériques de développement de systèmes à base de connaissances. Ils ne comportent donc pas de modèle pour guider la conception des systèmes de RÀPC et sont trop généraux pour aider efficacement le développement. Dans cette tendance, (Fuchs, 1997) a proposé une modélisation du cycle de raisonnement qui intègre notamment la tâche d'élaboration et explicite les autres étapes et a implanté un outil adéquat pour mettre en œuvre ce modèle.

Le problème de la représentation des connaissances est un problème qui est né avec l'intelligence artificielle et qui se pose toujours. Les premières représentations symbo-

liques étaient les logiques du premier ordre. Bien que très expressives, ces logiques étaient difficiles à manipuler. Rapidement, de nouveaux formalismes plus visuels tels que les réseaux sémantiques et les graphes conceptuels (Chein & Mugnier, 1992) ont émergé. En parallèle, les représentations à base de frames, inspirées des frames de Minsky, ont fait leur apparition. Ces représentations sont particulièrement adaptées à la description des modèles mentaux. Des travaux antérieurs se sont en particulier intéressés à deux de ces formalismes : les *logiques de descriptions* (Napoli, 1997) et les *représentations des connaissances par objets* (RCO) (Euzenat, 1998).

Les logiques de descriptions s'inspirent fortement des logiques du premier ordre mais aussi du concept de *frame*. Elles ont une sémantique forte et présentent l'avantage de produire des représentations dans lesquelles ont peut rapidement déduire de nouvelles connaissances par un processus d'inférence sur la base. En contre-partie, les logiques de descriptions présentent l'inconvénient de ne pas permettre de représenter l'aspect dynamique des éléments modélisés, puisque, comme leur nom l'indique, elles sont essentiellement descriptives.

Les RCO s'inspirent des langages à base de frames et sont à l'origine des langages de programmation par objets. Elles permettent une modélisation structurée des connaissances. Ces représentations s'appuient sur un élément de base, l'objet, qui représente un «concept» du monde réel. Un objet a à la fois des caractéristiques descriptives (propriétés structurelles), souvent appelées attributs et des propriétés comportementales, les méthodes, qui renseignent sur les comportements que peut avoir l'objet. Les objets sont regroupés en classes sur la base de propriétés structurelles et comportementales communes. Les classes sont organisées en hiérarchie les unes par rapport aux autres grâce à la relation d'héritage. Ainsi, contrairement aux logiques de descriptions, les représentations à objets permettent de prendre en compte l'aspect dynamique des éléments modélisés au sein même du système. Elles offrent également des mécanismes d'inférence variés tels que l'héritage, la classification, le filtrage et les facettes procédurales, ce qui permet d'implanter une variété de méthodes de raisonnement ainsi que des fonctionnalités très utiles en pratique (Ducournau *et al.*, 1998). De par leur structure hiérarchique, les représentations à objets permettent de considérer plusieurs niveaux de granularité dans la représentation. L'ensemble de ces propriétés font des RCO des outils permettant une transposition intuitive et aisée des connaissances du monde réel vers des représentations symboliques.

Les RCO et les logiques de descriptions partagent de nombreux principes de représentation et d'inférence. Les classes en RCO peuvent être assimilées aux concepts des logiques de descriptions. L'analogie est la même entre instance et individu. La différence majeure entre les deux formalismes réside dans la notion de rôle. Les rôles dans les logiques de descriptions sont souvent matérialisés par des attributs-liens et ne sont pas hiérarchisés, contrairement aux méthodes des RCO. Certains systèmes de représentations des connaissances par objets, comme AROM (Page *et al.*, 2000) notamment, vont même plus loin en réifiant les associations, ce qui permet de représenter facilement les relations n-aires et les différents rôles qu'elles sous-tendent par exemple.

Certains travaux ont exploité la richesse des représentations des connaissances pour le RÀPC à plusieurs niveaux. Ainsi, (Salotti & Ventos, 1998) propose une modélisation du RÀPC en logiques de descriptions, tandis que (Fuchs, 1997) a développé ROCADE, un

outil de RCO et en a proposé un exemple d'utilisation pour le RÀPC. A un autre niveau, (D'Aquin *et al.*, 2004) s'est intéressé à la notion de points de vues pour la représentation des connaissances d'adaptation, en s'inspirant essentiellement de la façon dont les points de vues sont représentés dans les RCO telles que TROEPS (Marino, 1993).

De nombreuses fonctionnalités des RCO ont motivé notre choix pour ce mode de représentation des connaissances pour les systèmes de RÀPC. En effet, il existe dans les RCO une grande variété de mécanismes de raisonnement utiles pour les besoins du RÀPC. D'autre part, dans un objectif de développement de système à base de connaissances, l'approche centrée objet présente de nombreux avantages tels que la modularité et la réutilisabilité. Finalement la possibilité de disposer de facettes procédurales associées éventuellement à un langage support pour personnaliser une application est indispensable dans cette optique.

3 Un outil support pour la conception d'applications de RÀPC

La conception et le développement de systèmes de RÀPC est, comme pour tout système à base de connaissances, un processus particulièrement complexe. Des outils tels que ReMind (Barletta, 1993) parmi les plus anciens ou plus récemment CBR*Tools (Jaczynski & Trousse, 1998) par exemple ont été développés pour proposer un support aux concepteurs.

L'approche du système CBR*Tools est intéressante et mérite que l'on s'y attarde. L'objectif de CBR*Tools est de faciliter le développement d'applications de RÀPC grâce à un ensemble de composants réutilisables et de méthodes extensibles. Il se présente comme une plate-forme à objets développée en Java et exploitant l'atelier de modélisation Rational Rose. L'approche «repose sur la définition d'une architecture abstraite modélisant les concepts du RÀPC» et «l'architecture intègre des points d'ouverture qui peuvent être configurés par spécialisation ou par intanciation». (Jaczinski, 1998) (p112). Dans cet outil, l'accent est mis sur le développement. Un système de RÀPC est obtenu par réutilisation de classes Java représentant les concepts du RÀPC en utilisant les techniques classiques de la programmation par objets. Il y a donc une bijection entre les concepts et mécanismes du RÀPC et les classes et méthodes de la plate-forme. Les conséquences sont que le niveau connaissance est réduit au minimum et que la connaissance est exprimée essentiellement au niveau «symbole» sous forme de classes Java. CBR*Tools est un outil intéressant pour développer les systèmes RÀPC, ouvert et extensible mais il n'est pas adapté pour la gestion des connaissances du RÀPC. Le logiciel Orenge (tec:inno, 2000) a été développé par la suite dans la même veine que CBR*Tools.

L'approche développée dans le système Rocade, un *framework* développé en 1995 (Fuchs & Mille, 1995), est similaire à CBR*Tools, mais à un niveau d'abstraction différent. Il permet la modélisation au niveau «connaissance» d'un système et des tâches de raisonnement; il constitue donc un environnement de gestion des connaissances à part entière permettant l'explicitation des connaissances. Dans Rocade, le niveau «symbole» est constitué des classes réutilisables du framework comprenant des méthodes d'infé-

rences (héritage, classification, indexation, etc.). Ces méthodes mettent en œuvre le raisonnement à partir de cas et peuvent être spécialisées à deux niveaux : au niveau du framework lui-même et au niveau des facettes des objets et des classes d'objets définis dans le RCO.

Dans la veine du système Rocade, notre approche pour aborder la conception d'un système de RÀPC se rapproche de celle d'un système à base de connaissances. Elle s'apparente à un processus d'ingénierie des connaissances qui vise à expliciter les connaissances et à les modéliser. Cette approche se situe à deux niveaux : le niveau «connaissance» et le niveau «symbole». L'aspect symbolique du système est le «support» sur lequel s'appuie le niveau connaissance. Notre objectif est de proposer un environnement pour le développement d'applications de RÀPC. Cet environnement présente deux facettes distinctes mais néanmoins fortement connectées : un cadre théorique formel d'une part et un ensemble de composants logiciels d'autre part. Le cadre théorique permet de formaliser et de structurer l'ensemble des connaissances dont on dispose sur le RÀPC. Les composants logiciels permettent quant à eux de réifier les concepts énoncés dans le cadre théorique. Ils se présentent sous forme de «briques de base» spécialisables de manière à être réutilisées lors du développement des applications cibles. Il existe un lien fort entre les concepts proposés par le cadre théorique et les composants au niveau symbolique. Grâce à ce lien, un concepteur d'application peut conduire son raisonnement au travers du cadre formel, en restant au niveau connaissance, même si, *in fine*, il manipule des composants logiciels. L'outil idéal fournirait aux concepteurs d'applications de RÀPC un ensemble de composants réutilisables, spécialisables et extensibles pour faciliter et accélérer le développement d'applications spécifiques. L'architecture générale d'un tel système est illustrée par la figure 1.

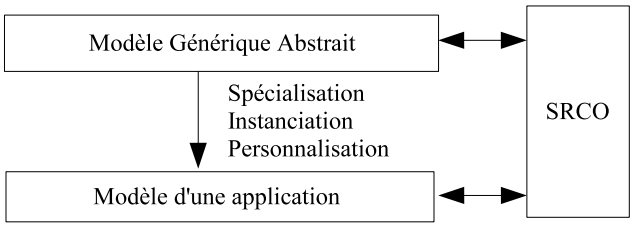

FIG. 1 – *L'architecture générale d'un assistant pour la conception et le développement de systèmes de* RÀPC. *Un modèle générique abstrait est dérivé en un modèle spécifique d'une application par spécialisation. Les deux modèles s'appuient sur une* RCO.

Pour réaliser un tel cadre, nous avons besoin d'un outil de représentation des connaissances. Cet outil servira à la fois à représenter les connaissances sur le raisonnement et les connaissances spécifiques au domaine d'application lors du passage à l'étape de conception. Comme nous l'avons vu, il semble que les représentations des connaissances par objets répondent bien aux besoins exprimés. Dans ce cadre, nous souhaitons présenter une modélisation objet du RÀPC ainsi qu'un modèle de décomposition du raisonnement en sous-tâches de raisonnement, comme cela a été proposé dans (Fuchs, 1997). Selon le principe «diviser pour régner», une telle démarche permet de décompo-

ser chaque tâche en sous-tâches suffisamment simples pour être traitées. La décomposition doit rester générique afin de garder une indépendance franche avec tout domaine d'application. Les sous-tâches élémentaires obtenues pourront être réifiées dans la partie logicielle de l'environnement. A chaque tâche correspond un composant. Les composants ainsi définis pourront alors être réutilisés par les concepteurs d'applications qui auront tout le loisir de les spécialiser pour qu'ils répondent à leurs besoins. Le modèle générique est présenté à la figure 2.

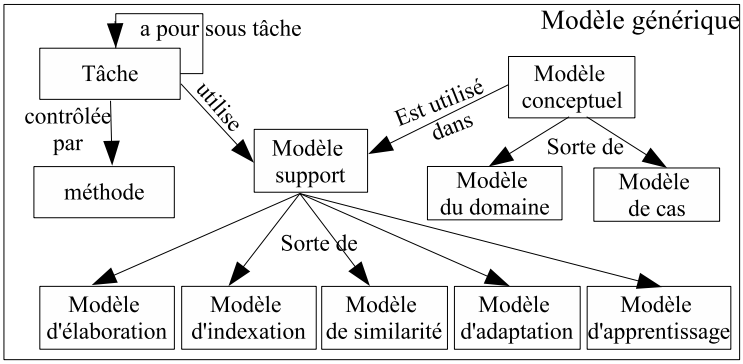

FIG. 2 – *Le modèle générique du* RÀPC.

Le modèle des tâches décrit les différentes tâches de raisonnement et les méthodes associées. Les modèles conceptuels décrivent les connaissances du domaine ainsi que les cas. Le lien entre ces deux types de modèles est effectué par les modèles supports qui mettent en relation les cas et connaissances du domaine avec les tâches de raisonnement précisant leur rôle et leur utilisation dans la résolution de problèmes. Cette approche présente deux intérêts majeurs. Comme nous l'avons déjà dit, elle se situe au niveau connaissance. Toutes les connaissances utiles sur le cycle du RÀPC sont explicitées et sont donc compréhensibles par un spécialiste du domaine. Cet aspect facilite grandement la compréhension du RÀPC par l'ensemble des acteurs du processus de conception des applications. Autre avantage, elle propose un cadre de travail modulaire, évolutif et non limitatif puisqu'elle repose sur le principe de la spécialisation des élements en fonction des besoins. Par exemple, dans la tâche d'élaboration, il serait possible de proposer un outil permettant de guider l'utilisateur sur l'ensemble des connaissances à acquérir pour construire un bon cas. Un tel outil pourrait s'appuyer sur des traces d'utilisation qui joueraient alors un rôle de «facilitateur d'acquisition des connaissances», mettant ainsi l'utilisateur au centre du système. On pourrait également mettre en œuvre un outil centré sur la modélisation de l'adaptation proposé dans (Fuchs *et al.*, 2000). Cet outil proposerait des fonctions d'influence et d'adaptation génériques qui pourraient ensuite être spécialisées en fonction du domaine d'application considéré. Nous pensons que l'étape d'apprentissage est l'étape durant laquelle le plus grand nombre de connaissances est susceptible d'émerger. Or cette étape se résume en règle générale à un simple apprentissage des cas résolus. Si l'on exploitait les connaissances extractibles des interactions entre l'utilisateur et le système lors de la phase de révision, il serait pos-

sible d'apprendre des connaissances d'adaptation qui ne préexistent pas dans la base de connaissance. C'est donc sur ce point que nous allons focaliser notre travail dans un premier temps.

4 Conclusion

Cet article décrit une ébauche d'architecture pour un assistant de conception d'applications de RÀPC. Il présente une justification de l'approche «niveau connaissance» que nous avons suivie. Il montre également l'importance de la représentation des connaissances dans de telles applications et insiste en particulier sur le formalisme sur lequel nous avons choisi de nous appuyer : les représentations des connaissances par objets.

Les perspectives de ce travail sont multiples. Nous projetons d'une part de présenter une formalisation aboutie de cette architecture. D'autre part, nous nous pencherons plus particulièrement sur la question de l'apprentissage des connaissance et en particulier des connaissances d'adaptation, lors des phases de révision et d'apprentissage.

Références

AAMODT A. (1991). *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning*. PhD thesis, University of Trondheim, Norwegian Institute of Technology, Department of Computer Science.

AAMODT A. & PLAZA E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AICOM*, **7**, 39–59.

BARLETTA R. (1993). *Remind Developper's Manual*. Rapport interne, Cognitive Systems.

K. BRANTING, K.-D. ALTHOFF & R. BERGMANN, Eds. (1999). *Third International Conference on Case-Based Reasoning - ICCBR'99*, Seeon, Germany. LNAI, Springer, Berlin.

CHEIN M. & MUGNIER M.-L. (1992). Conceptuals graphs: Fundamental notions. *Revue d'Intelligence Artificielle*, **6**(4), 365–406.

D'AQUIN M., LIEBER J. & NAPOLI A. (2004). Représentation de points de vue pour le raisonnement à partir de cas. In *LMO*, p. 245–258.

DUCOURNAU R., EUZENAT J., MASINI G. & NAPOLI A. (1998). *Langages et modèles à objets, état des recherches et perspectives*. INRIA.

EUZENAT J. (1998). *Représentation de connaissance par objets*, In *(Ducournau* et al.*, 1998)*, chapter 10, p. 293–319. INRIA.

FOX S. & LEAKE D. B. (1994). Using introspective reasoning to guide index refinement in case-based reasoning. In *Proceedings of the sixteenth annual Conference of the Cognitive Science Society*.

FUCHS B. (1997). *Représentation des connaissances pour le raisonnement à partir de cas : le système ROCADE*. Thèse d'université, Université Jean Monnet, Saint-Etienne, France.

FUCHS B., LIEBER J., MILLE A. & NAPOLI A. (1999). Towards a Unified Theory of Adaptation in Case-Based Reasoning. In *(Branting* et al.*, 1999)*, p. 104–117.

FUCHS B., LIEBER J., MILLE A. & NAPOLI A. (2000). An Algorithm for Adaptation in Case-Based Reasoning. In *Proceedings of the 14th European Conference on Artificial Intelligence - ECAI 2000*, p. 45–49, Berlin: IOS Press.

FUCHS B. & MILLE A. (1995). Objective c extensions for knowledge representation. In *Object-Oriented Computing in the Natural Sciences, OOCNS-95: Second Multidisciplinary International Workshop of Scientific Computing Environments*, IMAG, Grenoble, France.

GOMEZ-ALBARRAN M., GONZALES-CALERO P., DIAZ-AGUDO B. & FERNANDEZ-CONDE C. (1999). Modelling the Life Cycle Using Description Logics. In *in (Branting* et al., *1999)*, p. 147–161.

JACZINSKI M. (1998). *Modèle et plate forme à objets pour l'indexation des cas par situations comportementales : application à l'assistance à la navigation sur le web.* Thèse d'université, Université de Nice - Sophia Antipolis, Nice, France.

JACZYNSKI M. & TROUSSE B. (1998). An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners. In *6Th German Workshop on Case-Based Reasoning*, Berlin, Germany.

LENZ M., BARTSCH-SPÖRL B., BURKHARD H.-D. & WESS S. (1998). *Case-Based Reasoning Technology, from foundations to applications*. LNAI 1400. Berlin: Springer.

LIEBER J. (1997). *Raisonnement à partir de cas et classification hiérarchique. Application à la planification de synthèse en chimie organique*. Thèse d'université, Université Henri Poincaré Nancy 1, Nancy, France.

MARINO O. (1993). *Raisonnement classificatoire dans une representation a objets multi-points de vue*. Thèse d'université, LIFIA / IMAG / INRIA Rhône-Alpes, Université Joseph Fourier - Grenoble I.

MASSIE S., CRAW S. & WIRATUNGA N. (2004). Visualisation of Case-Base Reasoning for Explanation. In *7th European Conference on Case-Based Reasoning*, p. 135–144, Madrid, Spain.

NAPOLI A. (1992). *Représentations à objets et raisonnement par classification en intelligence artificielle*. Thèse d'état, Université de Nancy I, Nancy.

NAPOLI A. (1997). *Une introduction aux logiques de description.* Rapport interne, INRIA.

PAGE M., GENSEL J., CAPPONI C., BRULEY C., GENOUD P. & ZIÉBELIN D. (2000). Représentation de connaissances au moyen de classes et d'associations : le système AROM. In *Langages et Modèles à Objets (LMO'00)*, p. 91–106, Mont Saint-Hilaire (QC, CA).

PLAZA E. & ARCOS J.-L. (1993). *Noos: an integrated framework for problem solving and learning*. Rapport interne, Institut d'investigació en Intelligència Artificial, Barcelona, Spain, Report IIIA-RR-97-02.

RIFQI M. (1996). *Mesures de comparaison, typicalité et classification d'Objets flous : théorie et pratique*. Thèse d'université, Université Pierre et Marie Curie, Paris VI, Paris.

SALOTTI S. & VENTOS V. (1998). Study and Formalization of a Case-Based Reasoning System Using a Description Logic. In *EWCBR98*, p. 286–297.

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

SMYTH B. & KEANE M. T. (1993). Retrieving adaptable cases. the role of adaptation knowledge in case retrieval. In M. M. RICHTER, S. WESS, K.-D. ALTHOFF & F. MAURER, Eds., *First European Workshop on Case-Based Reasoning - EWCBR-93*, p. 209–220, Kaiserslautern, Germany: LNAI, vol. 837, Springer, Berlin.

SMYTH B. & KEANE M. T. (1995). Remembering to forget: A competence-preserving deletion policy for CBR systems. In *14th Joint Conference on Artificial Intelligence, IJCAI-95*, p. 377–382, Montréal, Canada.

TEC:INNO (2000). *The orenge Framework. An Architecture for Intelligent Services in the Electronic business Based on XML and Java.* Rapport interne, tec:inno.