Université Claude Bernard Lyon I

Laboratoire LIRIS

Gestion des Connaissances

pour des

Systèmes à Base de Connaissances hybrides

Amélie Cordier

Mémoire de DEA DISIC
Documents multimédia, Images et Systèmes d'Information Communicants

Encadré par Béatrice Fuchs
Équipe CEXAS
Axe 1 du LIRIS, Données, Documents et Connaissances

Lyon, Juin 2004

Table des matières

Introduction

La problématique de la gestion des connaissances est, à l'heure actuelle, une question omniprésente dans le monde de la recherche. De nombreuses sociétés et industries sont également confrontées à cette question. C'est notamment le cas du groupe avec lequel nous travaillons : EADS CCR.

EADS CCR[1] mène des travaux de recherche, développe des prototypes et accompagne les transferts technologiques pour les entités du groupe EADS. Le département "Ingénierie et Technologies de l'Information" s'intéresse en général à la gestion des connaissances et en particulier aux systèmes à base de connaissances.

Ce département développe des prototypes de systèmes à base de connaissances hybrides. Lors de la réalisation de ces applications, de nouveaux besoins ont émergé en ce qui concerne la modélisation des connaissances. En effet, des problèmes apparaissent à la fois au niveau de l'interopérabilité des systèmes et de la réutilisation des connaissances. De nombreuses questions se posent. Comment échanger des connaissances d'une application à l'autre ? Comment réutiliser des éléments connus ? En d'autres termes, comment traiter les problèmes d'interopérabilité, de capitalisation et de partage c'est-à-dire de *gestion des connaissances*.

Afin d'apporter des réponses à ces questions, il est nécessaire de disposer d'outils efficaces pour l'acquisition et la modélisation des connaissances. C'est à ce problème que nous nous sommes intéressés. Nous proposons plus particulièrement un cadre de gestion des connaissances adapté à la conception de systèmes à base de connaissances hybrides tels que ceux conçus par EADS CCR. Nous avons élaboré ce cadre en nous focalisant particulièrement sur les aspects *modélisation*, *interopérabilité* et *réutilisation* des connaissances.

Afin de mieux cerner les problèmes qui nous étaient posés, nous avons effectué une analyse des besoins liés aux domaines d'application du groupe EADS. Nous avons réalisé un inventaire des différentes catégories de connaissances mises en œuvre dans les prototypes. A l'issue de cette analyse, nous avons décidé de restreindre notre étude à un système hybride gérant des cas et des règles. Nous justifierons ces choix dans le premier chapitre.

Dans un deuxième temps, nous avons dressé un état de l'art des méthodes et outils d'ingénierie des connaissances qui nous semblaient pertinents d'après l'analyse des besoins. Le résultat de cette étude sera présenté dans le deuxième chapitre de ce document.

L'étude des applications et des besoins nous a conduit à proposer plusieurs modèles de connaissances. En particulier, nous nous sommes focalisés sur la réalisation d'un modèle de règles et d'un modèle de cas. Nous nous somme également intéressés spécifiquement à la notion de point de vue comme support d'aide à la structuration des connaissances pour la réutilisation et la visualisation. Ces notions seront abordées dans le troisième chapitre.

Nous discuterons des résultats obtenus dans le quatrième chapitre. Enfin, nous présenterons le travail qui sera effectué dans un avenir proche et nous exposerons également les pistes de recherche ouvertes par ce travail.

[1] EADS CCR : Centre Commun de Recherche du groupe EADS

Chapitre 1

Contexte de travail et problématique

Dans ce chapitre, nous allons présenter quelques systèmes à base de connaissances développés par EADS CCR. Nous verrons les limites et les problèmes qu'ils soulèvent et nous nous appuierons sur ces observations pour définir plus précisément notre problématique de recherche.

1.1 Environnement de travail

EADS CCR développe de nombreuses applications à base de connaissances à destination de l'aéronautique et en particulier d'Airbus. La plupart sont des applications hybrides manipulant à la fois des systèmes à base de règles, du raisonnement à partir de cas, du raisonnement à partir de modèles, des réseaux de neurones, etc. Dans notre étude, nous nous sommes restreints aux applications à base de règles et au raisonnement à partir de cas.

Dans ce paragraphe, nous allons présenter deux applications spécifiques afin de mieux comprendre les problématiques auxquelles nous sommes confrontés.

1.1.1 Aide à la spécification d'outillage de maintenance

Cette première application à pour but d'assister un utilisateur à la spécification d'un outil de levage ainsi qu'à la rédaction du cahier des charges correspondant. Elle fonctionne grâce à une combinaison d'un système à base de règles et d'un système de raisonnement à partir de cas. Structurellement, elle est composée de deux modules : l'un pour la mise à jour du modèle de connaissances et l'autre à destination de l'utilisateur final.

Afin de permettre la rédaction du cahier des charges, le système pose un ensemble de questions à l'utilisateur. Au fur et à mesure que celui-ci répond, la liste des questions posées est mise à jour. Le processus principal est guidé par un ensemble de règles qui définissent, en fonction d'un ensemble de réponses, quelles sont les questions à poser par la suite. Le raisonnement à partir de cas intervient à chaque fin de cycle pour apporter de l'aide à l'utilisateur en suggérant des réponses. Les éléments fournis par le cycle de raisonnement à partir de cas sont donnés uniquement à titre indicatif, l'utilisateur reste le seul à décider de ce qu'il souhaite retenir.

Dans cette application, nous sommes clairement confrontés au problème de l'interopérabilité des applications. Ce système utilise JRules (de la société ILOG[1]) pour les règles et Kaïdara (de la société Kaïdara Software[2]) pour les cas. Par conséquent, lorsqu'il y a une mise à jour des données, qui sont partagées par les deux applications, elle doit être effectuée dans les deux systèmes pour assurer la cohérence des bases. Cette mise à jour est faite grâce à un programme intermédiaire (en JAVA) qui a été développé spécifiquement pour cette application.

[1] www.ilog.com/products/jrules/
[2] www.kaidara.com

1.1.2 Aide à la conception d'outillages d'emboutissage

Ce second système offre une assistance à la conception et à la spécification d'outils. Il offre plusieurs fonctionnalités et permet notamment l'évaluation des coûts des outils conçus. Il permet également d'effectuer des transferts de connaissances vers les novices du domaine en réutilisant les connaissances et les expériences acquises.

Dans cette application, le moteur à base de règles et le moteur à base de cas fonctionnent en parallèle. Ils proposent tous les deux des solutions que l'utilisateur peut choisir ou non.

Au départ, l'application devait fonctionner uniquement avec un système à base de cas mais les concepteurs se sont vite rendus compte qu'il existait de nombreuses connaissances représentables par des règles qui pourraient être très utiles pour les novices dans le domaine, d'où cette intégration.

L'application est basée sur KAÏDARA pour ce qui est des cas. Le règles, quant à elles, sont saisies au fur et à mesure par les différents utilisateurs du système. Par conséquent, la base de règles devient rapidement très difficile à maintenir.

1.2 Problématique

EADS dispose de méthodologies d'acquisition des connaissances. Une fois acquises, ces connaissances sont modélisées et intégrées à des applications fonctionnelles. L'ensemble des outils permettant de manipuler les connaissances de bout en bout de cette chaîne existent : documentation des connaissances, explication des modèles, outils de traçage, gestion des flots de données, etc. Mais une lacune importante se fait ressentir au niveau de la modélisation des données. Comment représenter une règle ou un cas de façon suffisamment générique pour qu'ils soient réutilisables dans différentes applications ?

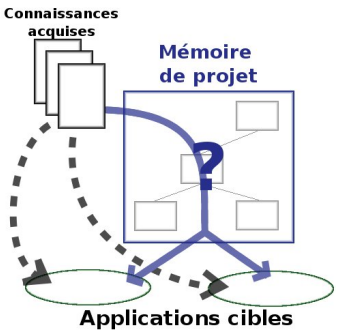

FIG. 1.1 – Représentation de la problématique

Comme nous pouvons le voir sur la figure 1.1, les connaissances acquises et explicitées sont stockées dans des fichiers dont on ne maîtrise pas les formats. La modélisation de connaissances dans les systèmes est faite par l'intermédiaire d'opérateurs humains. C'est un processus long et fastidieux. Plutôt que de passer directement des connaissances explicitées aux applications cibles, nous souhaitons les capitaliser dans une mémoire de projet afin de pouvoir l'utiliser plus facilement dans les applications cibles. Or, pour réaliser cela, il manque un élément central :

l'ensemble des modèles de connaissances qui doit être utilisé dans la mémoire de projet et qui permet cette *capitalisation*.

Un second aspect est manquant : la *réutilisation*. Pour pouvoir bénéficier d'une mémoire de projet dans laquelle les connaissances sont capitalisées, il faut pouvoir réutiliser des sous ensembles de ces connaissances à tout moment.

Enfin, nous avons besoin de pouvoir échanger facilement des connaissances d'une application à l'autre. Nous avons donc un problème d'*interopérabilité*.

Ces trois préoccupations relèvent de la gestion des connaissances. Nous allons les présenter plus en détails ci-dessous.

1.2.1 Capitalisation des connaissances

Les connaissances modélisées dans les différents projets sont présentes dans les outils cibles et dans la documentation associée au projet. Mais, dans l'état actuel, il n'est pas possible de tirer profit du processus d'ingénierie des connaissances pour extraire et représenter de manière unifiée cette connaissance explicitée.

Il est donc nécessaire de disposer d'un outil de gestion des connaissances permettant de résoudre ce problème. Compte tenu de notre contexte de travail, le modèle de représentation des connaissances de cet outil doit être à la fois suffisamment générique pour être réutilisé et étendu dans un maximum de situations mais également suffisamment clair et utilisable dans un cadre industriel. Il doit offrir un *guide* de modélisation général qui puisse ensuite être spécifié pour répondre aux besoins des applications spécifiques.

1.2.2 Réutilisation des connaissances

La notion de réutilisation des connaissances est très importante. Elle peut être considérée à plusieurs niveaux. On peut souhaiter réutiliser un ensemble de connaissances préexistantes dans la mémoire de projet lors de la conception d'une nouvelle application. On peut également vouloir visualiser une partie des connaissances pour une tâche donnée ou encore dans le cadre d'un "concept métier" particulier, ce qui peut s'avérer très utile dans le cadre d'une conception collaborative par exemple. En effet, la réutilisation d'une sous partie d'un ensemble de connaissances permet de diminuer le niveau de complexité du travail effectué en éliminant un ensemble de détails non pertinents. La visualisation des connaissances va de pair avec la notion de granularité. Il n'est pas toujours nécessaire de visualiser un ensemble de connaissances en affichant tous les éléments jusqu'au plus petit niveau de détail. On peut vouloir rester à un niveau macroscopique. Cet aspect est à prendre réellement en compte.

Pour réutiliser les connaissances, il est nécessaire de délimiter les contours de "blocs de connaissances" qui deviendront des entités réutilisables. Cette phase du processus est particulièrement complexe. Comment placer correctement les limites pour s'assurer de disposer des connaissances nécessaires et suffisantes pour le problème courant sans pour autant *tout* sélectionner ?

La notion de *point de vue* sur les connaissance semble pouvoir apporter une réponse à cette question. Les points de vue révèlent un sous ensemble des connaissances éclairées sous un angle particulier.

1.2.3 Interopérabilité

Les outils cibles représentent tous les connaissances qu'ils gèrent dans des formats spécifiques, ce qui rend leur interopérabilité très difficile. Un des objectifs serait de disposer d'un format de description des différentes sources de connaissances facilitant le partage et l'échange de connaissances entre les applications. Ici, nous ne nous plaçons plus à un niveau général de capitalisation des connaissances mais à un niveau applicatif ou les connaissances sont déjà mises en œuvre dans les systèmes.

En ce qui concerne la format d'échange des données, de nombreuses solutions ont été envisagées jusqu'à aujourd'hui. La plus simple semble être l'utilisation d'un format d'échange classique tel que le XML[3] couplé au développement d'interfaces de conversions XML/format propriétaire pour rester compatible avec les applications ne disposant pas de moyens de traiter le XML nativement. Il existe également d'autres formats normalisés pour l'échange des bases de connaissances tels que Step ou Express. Ce sont des standards internationaux recommandés par de nombreux experts mais qui ont montré leurs limites dans le cadre des applications utilisées par EADS. Ainsi, cette aspect de la gestion des connaissances reste problématique.

Conclusion

Dans ce travail, nous avons choisi de ne pas traiter l'aspect interopérabilité. Nous nous focalisons sur trois des aspects présentés ci-dessus. Dans un premier temps, nous proposons un environnement de capitalisation des connaissances pour la gestion de mémoires de projets. Nous présentons également l'ensemble des ontologies des types de connaissances présentes dans cet environnement. Enfin, nous décrivons des outils pour les manipuler : les points de vue.

[3] eXtended Markup Language

Chapitre 2

État de l'art

La problématique de la gestion des connaissances se trouve à la croisée de plusieurs disciplines du domaine de l'intelligence artificielle. Par conséquent, le travail de recherche bibliographique a occupé une place cruciale dans le déroulement travail.

L'état de l'art que nous proposons ci-dessous a pour ambition de synthétiser les résultats de cette étude bibliographique et également d'introduire et d'exposer les concepts principaux qui seront manipulés dans la suite de ce rapport. Dans un premier temps, nous nous intéresserons aux raisonnements à partir de règles et à partir de cas. En second lieu, nous parlerons d'ingénierie des connaissances et de représentation des connaissances, deux aspects principaux de la gestion des connaissances. Pour réaliser cette étude, nous nous sommes inspirés de l'ouvrage [DCG+00] qui présente une synthèse des méthodes et outils principaux utilisés dans la gestion des connaissances.

2.1 Les systèmes à base de connaissances

L'ingénierie des connaissances étudie les concepts et les méthodes permettant de modéliser et d'acquérir des connaissances. Son objectif est de faciliter la conception et la réalisation de systèmes informatiques "intelligents" que l'on appelle les *Systèmes à Base de Connaissances (SBC)*. Ce domaine de recherche est pluridisciplinaire. Il fait appel à la logique pour la construction de modèles formels, à la linguistique pour la formulation des connaissances et évidement à l'informatique pour la mise en application des modèles définis.

Un système à base de connaissances est un programme capable de raisonner pour résoudre un certain problème en s'aidant de connaissances relatives au domaine d'application considéré. Les systèmes à base de connaissances comportent une fonction de *représentation des connaissances* associée à une fonction de manipulation des connaissances représentées appelée *raisonnement*.

L'acquisition des connaissances est un processus complexe et se présente souvent comme un goulot d'étranglement lors de la mise en œuvre des systèmes à base de connaissances. L'implantation de ce type de systèmes est donc un processus fastidieux qui demande des compétences techniques spécifiques et souvent beaucoup de temps. De plus, une fois mis en service, les systèmes sont souvent lents et il est très difficile de maintenir les gros volumes de connaissances qu'ils contiennent, si bien qu'ils risquent de devenir obsolètes ou non conformes à la réalité.

Le raisonnement à partir de cas, que nous présenterons ci-dessous, apporte des solutions partielles aux problématiques évoquées ici. En particulier, il répond au problème de l'élicitation des connaissances puisque ces dernières ne sont plus présentes dans le système *a priori* mais sont introduites au fur et à mesure grâce à un processus d'apprentissage.

Dans cette section, nous allons présenter les deux catégories de systèmes à base de connaissances que nous manipulerons par la suite : les systèmes à base de règles et les systèmes de raisonnement à partir de cas.

2.1.1 Systèmes à base de règles

2.1.1.1 Présentation

Les systèmes à base de règles, aussi appelés *systèmes experts*, furent très en vogue dans les années 1980. Leur apparente simplicité de fonctionnement leur a permis de devenir rapidement opérationnels.

Les systèmes experts sont des outils de raisonnement efficaces dans les domaines ou les connaissances sont éparses et peu formalisées. Ils sont souvent conçus pour résoudre des problèmes de classification ou de décision (diagnostic médical par exemple).

Un système expert est constitué d'une base de règles, d'une base de faits et d'un moteur d'inférence (cf. figure 2.1).

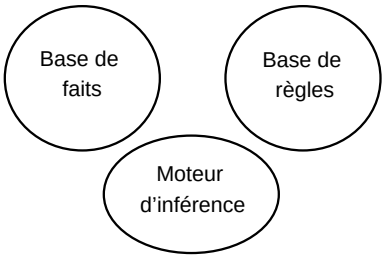

Fig. 2.1 – Schéma de principe d'un système expert

La base de faits contient, pour une situation donnée, des faits avérés ou à établir. Elle est mise à jour après chaque déclenchement de règle. Toute mise à jour implique le déclenchement d'un processus de vérification des incohérences de la base.

La base de règles contient la connaissance du domaine nécessaire au support du raisonnement. Une règle est une unité de connaissance de la forme : "SI Conditions Alors Actions [coefficient]". *Conditions* et *Actions* désignent respectivement un ensemble de conditions et un ensemble d'actions. *Coefficient* permet de définir une priorité pour la règle. L'ensemble des règles forme la base de connaissances ou *base de règles*.

La plupart des règles représentent des connaissances du domaine mais il existent également des règles qui matérialisent des *méta connaissances*. Les méta connaissances sont des connaissances sur les connaissances. Elles peuvent porter sur l'intérêt d'une connaissance, sa priorité, sa véracité, sa précision etc. Les méta connaissances permettent souvent de mettre en place une stratégie de choix lorsque plusieurs règles sont applicables.

Le moteur d'inférence dirige le processus de raisonnement et se charge de mettre à jour la base de faits en fonction de l'évolution du problème. Dans un système expert, le processus de raisonnement consiste dans un premier temps à constituer l'ensemble des règles déclenchables. Ensuite, il est nécessaire de choisir les règles à déclencher puis de les déclencher et d'observer les résultats. Si l'objectif du raisonnement n'est pas encore atteint, il faut réitérer le processus.

Il est nécessaire de définir une stratégie générale pour le choix des règles à déclencher. Il existe plusieurs types de chaînages applicables dans les systèmes experts : chaînage avant, chaînage arrière, chaînage mixte. Leur choix dépend de l'objectif de l'application.

Bien qu'ayant connu une époque glorieuse, les systèmes experts ont rapidement été mis à l'écart à cause des nombreux inconvénients qu'ils présentaient. En effet, l'explosion rapide de la taille de la base de règles les rendaient difficilement maintenables, il devenait de plus en plus dur de vérifier la cohérence des règles entre elles, les erreurs dans les règles étaient indétectables et enfin, il était impossible d'assurer la complétude de l'expertise.

Pourtant, les performances des systèmes experts sont satisfaisantes dans certains domaines. C'est pourquoi ils sont toujours utilisés mais qu'il sont généralement complétés par d'autres systèmes, comme le raisonnement à partir de cas par exemple.

2.1.2 Le raisonnement à partir de cas

Le raisonnement à partir de cas (RÀPC), où "Case-Based Reasoning en anglais" (CBR), est un paradigme de résolution de problème qui résout un problème en s'appuyant sur des solutions connues de problèmes passés jugés similaires à celui étudié.

Dans cette section, nous allons présenter en quelques mots l'origine du raisonnement à partir de cas ainsi que ses caractéristiques principales. Nous évoquerons également les enjeux de la recherche dans ce domaine au travers de quelques systèmes existants. La présentation théorique du RÀPC qui est effectuée ici s'inspire fortement de l'excellente introduction au RÀPC proposée par Agnar Aamodt et Enric Plaza [AP94].

2.1.2.1 Origines du raisonnement à partir de cas

Historique

L'intérêt porté au raisonnement à partir de cas est né de deux motivations distinctes mais qui se sont rapidement rejointes. La première motivation, liée au sciences cognitives, peut être vue comme le désir d'imiter au mieux le raisonnement humain. La seconde motivation, quant à elle, est liée au domaine de l'intelligence artificielle. Elle se présente comme une réponse au besoin de méthodes d'intelligence artificielle plus efficaces.

Il est légitime de s'interroger sur l'utilité, et plus particulièrement sur l'efficacité, du raisonnement à partir de cas. Dans le but d'apporter un premier élément de réponse, nous pouvons observer certains faits de la vie courante. D'une part, les individus se trouvent souvent confrontés à des problèmes similaires à ceux qu'ils ont déjà rencontrés. D'autre part, il est fréquent que des problèmes similaires aient des solutions similaires. Notons enfin que d'un point de vue technique, il est souvent plus facile de représenter une expérience concrète que la théorie qui régit le déroulement de cette expérience (s'il est possible de la formaliser). C'est notamment en s'appuyant sur ces observations qu'est né le raisonnement à partir de cas en tant que technique d'intelligence artificielle.

Le raisonnement à partir de cas trouve ses origines dans les travaux sur la mémoire dynamique de Roger Schank [Sch82] mais également dans de nombreuses études psychologiques qui ont apporté des preuves empiriques du rôle prédominant des expériences passées dans la résolution de problème chez l'humain.

Le premier système que l'on peut qualifier de système de raisonnement à partir de cas est CYRUS, outil développé par Janet Kolodner en 1983 à l'université de Yale. Basé sur le modèle

de mémoire dynamique de Schank, le système CYRUS fonctionnait essentiellement selon un processus de questions/réponses. Le modèle de mémoire de cas développé pour ce système a servi de base à de nombreux autres systèmes de raisonnement à partir de cas comme, entre autres, CHEF et CASEY.

Spécificités du ràpc

En règle générale, les méthodes de résolution de problèmes en intelligence artificielle utilisent les connaissances relatives à un domaine spécifique. Elles doivent par conséquent tenter de trouver les solutions potentielles dans un espace de recherche immense.

Le raisonnement à partir de cas se distingue ici des autres approches, dites *traditionnelles* dans le sens où il ne nécessite pas une connaissance exhaustive du domaine pour résoudre les problèmes puisque la plus grande partie des connaissances utiles est contenue dans les cas. Ces connaissances spécifiques correspondent à des expériences de résolution de problèmes passées et gardées en mémoire.

Les systèmes de raisonnement à partir de cas sont également des systèmes évolutifs puisque, comme nous le verrons dans la suite, ils acquièrent de l'expérience au fur et à mesure qu'ils résolvent les problèmes.

On distingue principalement deux grandes catégories de systèmes de raisonnement à partir de cas : les systèmes voués à la résolution de problèmes, qui apportent une solution nouvelle, et les systèmes interprétatifs qui se limitent à la remémoration de cas passés dans le but d'évaluer ou de justifier des solutions existantes.

Les systèmes de raisonnement à partir de cas travaillent avec de nombreux types de problèmes : planification, synthèse, diagnostic, aide à la décision, etc. et ce dans des domaines aussi variés que la médecine, les disciplines juridiques, la sécurité routière, l'ingénierie mécanique et bien d'autres encore.

Mais les systèmes de raisonnement à partir de cas ont également leurs limitations. Du fait même de leur nature, chaque système ne peut répondre qu'à une catégorie restreinte de problèmes. Cette limitation fait partie des axes de recherche actifs, notamment dans l'équipe CEXAS de l'axe 1 du LIRIS : "Données, Documents , Connaissances".

2.1.2.2 Principe de fonctionnement

Comme nous l'avons déjà évoqué, le raisonnement à partir de cas est une approche de résolution de problèmes qui s'appuie sur une mémoire d'expériences de résolutions de problèmes passés, appelés cas sources, pour résoudre de nouveaux problèmes appelés cas cibles. Les solutions des cas sources sont retrouvées et réutilisées lors d'un cycle de raisonnement décomposé en cinq étapes : élaboration d'un nouveau cas, remémoration des cas passés pertinents, réutilisation du ou des cas sources, révision de la solution obtenue et enfin, apprentissage de cette solution en vue d'une réutilisation future. Ce cycle est présenté à la figure 2.2.

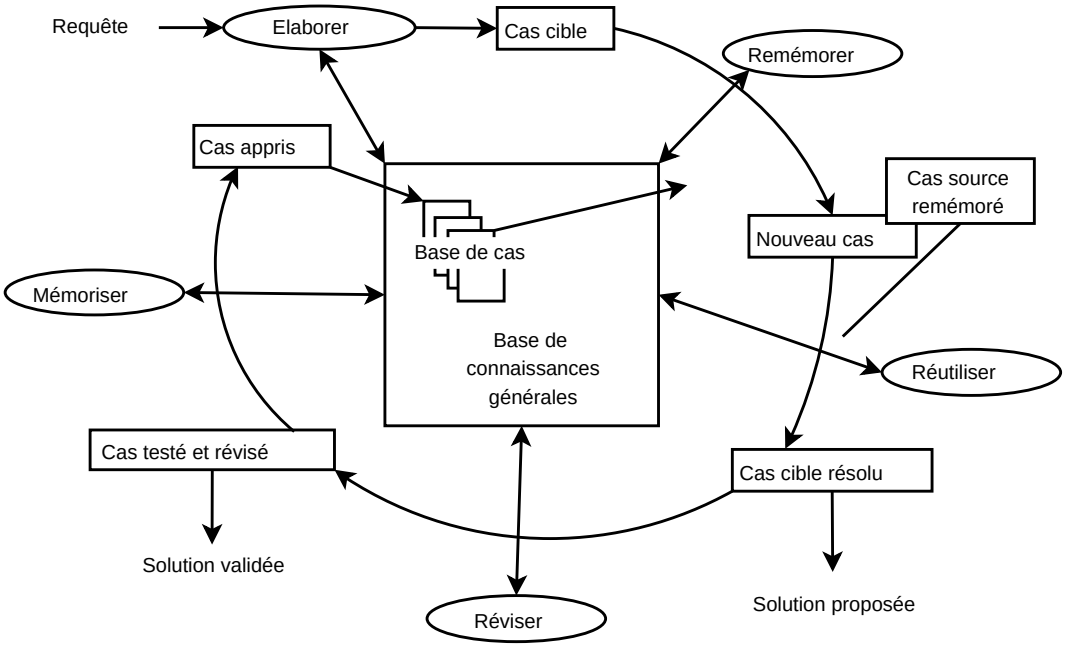

FIG. 2.2 – Le cycle du RàPC

Tout système de raisonnement à partir de cas comporte une base de cas (ou mémoire de cas). Un cas représente une expérience de résolution de problème. Il est constitué d'un problème (P), d'une solution (S) et d'un raisonnement (R) qui permet de conduire du problème à la solution. Ce que nous appelons P et S sont en fait des *descriptions* du problème et de la solution. Nous utilisons ici les notations de [Fuc97]. Par exemple, dans le domaine médical, un problème peut être "la description de l'état d'un patient" et la solution serait alors "un diagnostic correspondant".

Les cinq étapes du cycle de raisonnement à partir de cas, présentées ci-dessous, existent dans la majorité des systèmes, néanmoins, le rôle qu'elles jouent est plus ou moins important en fonction de l'objectif de l'application.

- **Élaboration :** les données concernant la situation de problème courante sont collectées afin de constituer un nouveau cas (non résolu). Ce cas est généralement appelé *cas cible*.
- **Remémoration :** les cas sources de la mémoire de cas sont comparés au cas cible. On cherche ici à évaluer leur similarité avec le cas courant.
- **Réutilisation :** le ou les cas sources remémorés à l'étape précédente sont utilisés pour résoudre le problème cible.
- **Révision :** la solution proposée par le système est vérifié (en général par l'utilisateur) qui décide ou non de sa validité. Ce dernier peut éventuellement corriger la solution si celle-ci ne lui convient pas.
- **Apprentissage :** la nouvelle solution obtenue est apprise par le système en vue d'une réutilisation future. Notons que certains systèmes permettent également d'apprendre des situations d'échec (aucune solution n'a été trouvée à l'étape précédente).

2.1.2.3 Thèmes de recherche en raisonnement à partir de cas

Les étapes du RÀPC font intervenir des mécanismes divers et ont des ramifications dans différents domaines de l'intelligence artificielle. Ce sont des mode de raisonnement complexes. Les problématiques de recherche sont donc nombreuses. Nous pouvons distinguer certains axes de recherche principaux.

- **Représentation des connaissances :** la problématique principale consiste ici à chercher ce que doit contenir un cas. Comment organiser un cas pour assurer au système des performances satisfaisantes en termes de temps d'exécution, d'espace mémoire et surtout de pertinence des réponses ?
- **Méthodes de recherche :** il s'agit ici de définir des méthodes objectives d'évaluation de la similarité entre les cas. La similarité est un aspect qui mérite qu'on lui porte beaucoup d'attention. Il est important de représenter explicitement les connaissances de similarité au même titre que les autres connaissances car elles sont réellement porteuses de sémantique. C'est un aspect capital lorsque l'on cherche à faire de la représentation des connaissances pour le raisonnement à partir de cas, et pourtant, ce n'est pas toujours pris en compte dans les systèmes actuels.
- **Réutilisation des cas :** cet axe de recherche correspond à l'étude de l'étape d'adaptation dans le raisonnement à partir de cas. Les enjeux soulevés par cette étape sont essentiels. C'est de la qualité de l'adaptation que vont dépendre la qualité et la pertinence des solutions obtenues par les systèmes. Un axe de recherche actuel porte sur le lien étroit existant entre l'étape de remémoration et l'étape d'adaptation, conduisant à l'idée de remémoration guidée par les critères d'adaptabilité. Une partie du travail présenté dans la suite du document abordera ce problème du point de vue de la modélisation des connaissances.
- **Révision :** cet aspect de la recherche se concentre sur les méthodes permettant de "réparer" les solutions proposées par le systèmes lorsqu'elles ne sont pas satisfaisantes ou bien sur les techniques permettant "d'apprendre" ces solutions si elles le sont.
- **Apprentissage :** dans cet axe, on essaie de retenir uniquement les parties pertinentes des nouveaux cas pour éviter de surcharger inutilement le système.

Nous nous intéressons particulièrement aux trois premiers axes car ils ont un lien très étroit avec la problématique de gestion des connaissances. Dans le paragraphe suivant, nous verrons que les systèmes utilisent différentes méthodes pour représenter les cas.

Environnements pour le ràpc

Différentes communautés de recherche en raisonnement à partir de cas ont tenté de répondre à certaines problématiques évoquées ci-dessus et ont expérimenté leurs théories dans des systèmes concrets. Nous allons en présenter certains ici.

Dans CREEK [Aam91], Agnar Aamodt et son équipe proposent un modèle générique de raisonnement basé sur les mécanismes des réseaux sémantiques. La partie RÀPC du système s'appuie sur les *frame*. Il en est de même dans RESYN/RÀPC [Lie97], [DEMN98], système d'aide à la conception de plans de synthèse en chimie organique.

Le système ROCADE [Fuc97] a été développé dans le même esprit que CREEK. Il expérimente les systèmes de représentation des connaissances par objets pour les différents raisonnements et pour le RÀPC en particulier.

CBR*Tools [JT98] se présente également comme un framework, mais les cas sont également représentés comme des objets logiciels. Contrairement à Rocade et Creek, il est spécifiquement dédié au ràpc.

Il est vrai qu'historiquement, l'objet occupe une place prédominante dans la représentation des connaissances [FK85], mais il est parfaitement envisageable d'utiliser d'autres moyens pour accomplir les même choses. Huajun Chen et Zhaohui Wu [CW03] proposent une approche basée sur le RDF[1] pour faire du raisonnement à partir de cas dans le Web Sémantique. Il est également possible de travailler avec des logiques de description [SV98] ou des graphes sémantiques [MC96].

Cette étude nous permet de constater qu'il existe presque autant de modèles de connaissances que d'environnements de mise en œuvre de ces connaissances. Par conséquent, les connaissances sont difficiles à partager entre les différents systèmes et, a fortiori, elles sont peu réutilisables. Nous cherchons à fournir une approche unifiée de la modélisation des connaissances pour palier à cela.

Dans cette partie, nous nous sommes attardés sur l'aspect "représentation des connaissances" dans le raisonnement à partir de cas car c'est un des points clés de notre étude. Dans la suite, nous proposerons un modèle de représentation des connaissances incluant entre autres les connaissances de similarité et d'adaptation. Le modèle que nous présenterons s'appuie sur les formalismes de représentations exposés ici et cherche à les étendre en améliorant les aspects *réutilisation* des connaissances et *interopérabilité* des systèmes.

2.2 Mise en œuvre des systèmes à base de connaissances

Pour mettre au point des systèmes à base de connaissance, il est nécessaire de disposer d'outils de modélisation et d'acquisition des connaissances. Ces outils peuvent être classés en deux catégories principales correspondant aux deux niveaux définis par Newell en 1982 : le niveau connaissance et le niveau symbole.

D'après l'hypothèse du niveau de connaissance, la représentation d'une part et les connaissances d'autre part doivent être clairement distinguées. Dès lors, les concepteurs de systèmes à base de connaissances cherchent à séparer les spécifications des connaissances de leur implantation opérationnelle dans un système. Ce processus revient à créer un modèle intermédiaire de *représentation des connaissances* entre le monde réel et le monde informatique.

Au niveau connaissance, on trouve essentiellement des méthodes d'acquisition et de modélisation des connaissances. Au niveau symbole se trouvent les environnements de mise en œuvre des méthodes et modèles décrits au niveau précédent. Nous présenterons ces deux aspects dans les parties suivantes.

2.2.1 Le niveau "connaissance"

Un modèle est une abstraction qui permet de réduire la complexité d'un système en se focalisant sur certains aspects, en fonction de certains buts à atteindre. Proposer des méthodes de modélisation des connaissances est essentiel. Ces méthodes pourront guider les experts lors de la conception de systèmes à base de connaissances ainsi que lors de l'acquisition des

[1] RDF : Resource Description Framework

connaissances. De plus, les modèles proposés se présentent en général comme un *langage commun* aux experts du domaine et aux concepteurs des applications.

Dans cette partie, nous allons présenter succinctement quelques méthodologies de modélisation des connaissances.

2.2.1.1 ComMet

ComMet (COMponential METhodology) a été défini par Luc Steels en 1990. ComMet repose sur l'idée que la connaissance dans un système expert traditionnel peut facilement être décomposée en des composants distincts et indépendants [Ste90]. Luc Steels définit trois types de composants différents : les tâches, les méthodes et modèles.

Les modèles représentent une description abstraite de la réalité (modèle du domaine, modèle des cas). Les tâches sont des actions qui doivent être réalisées. Une tâche utilise des modèles *source* et a un impact sur les modèles *destination*. Les méthodes sont des algorithmes (succession d'actions) qui spécifient comment les tâches doivent être accomplies.

D'un point de vue pratique, nous pouvons noter que l'un des objectifs principaux de ComMet était de permettre la réutilisabilité des composants ainsi que l'accessibilité du système aux non experts du domaine, ce qui rendait l'outil particulièrement attrayant.

2.2.1.2 KADS, CommonKADS

Mise au point en 1985 dans le cadre d'un grand projet européen, la méthodologie KADS [WVdVSA92], [WVdVSA92], subit de profonds changements en 1992 et devient CommonKADS [FvH94], [Mar94]. CommonKADS est donc le résultat de douze ans de recherche. Cette méthodologie est actuellement vue comme le standard européen pour la modélisation des connaissances.

CommonKADS est une méthode visant à aider à la modélisation des connaissances d'un ensemble d'experts dans le but de réaliser des systèmes d'aide à la décision basés sur les connaissances.

Cet outil s'appuie sur un ensemble de modèles ayant des fonctions distinctes :

Le modèle d'expertise : il modélise les processus par lesquels un expert trouve une solution à un problème.

Le modèle d'organisation : il décrit l'organisation dans laquelle le système sera utilisé.

Le modèle des tâches : il décrit les diverses tâches exécutées dans l'environnement de fonctionnement du système.

Le modèle agent : il décrit les attributions des agents qui exécutent les tâches.

Le modèle communication : il décrit les interactions entre les agents et fournit les mécanismes pour améliorer la communication.

Le modèle de conception : il établit le lien entre les modèles conceptuels et leurs implantations informatiques.

2.2.1.3 MOKA

La méthodologie MOKA (Methodology and tools Oriented to Knowledge-based engineering Applications) [MOK98] propose un atelier complet (démarche, modèles de représentation et outils) dédié au développement d'applications d'ingénierie à base de connaissances.

Cette méthodologie définit un cycle de vie qui identifie les rôles, les activités, etc. dans une application à base de connaissances. Elle utilise des modèles pour représenter les connaissances. Ces connaissances sont divisées en deux catégories : *product knowledge* (connaissances portant sur les entités physiques conçues) et *process knowledge* (connaissances sur les étapes de conception du produit).

MOKA définit son propre langage de modélisation : MML (MOKA Modelling Language) [MOK00]. Ce langage se présente comme une extension d'UML.

Notons qu'EADS CCR étaient l'un des leaders de ce projet européen, qui implique également de nombreux partenaires industriels tels que l'aérospatiale, British Aerospace, Daimler-Chrysler et PSA [MOK99].

Comme dans les modèles présentés ici, dans le système que nous proposons, nous nous appuyons sur des recueils d'expertises pour capturer les connaissances. Nous ne cherchons pas à capitaliser les *méthodes spécifiques* de résolution de problèmes car elles sont situées dans les applications cibles. À l'inverse, nous souhaitons capitaliser les connaissances "générales" issues des recueils d'expertises. Les modèles de connaissances que nous proposons sont représentés sous formes d'ontologies. Nous adoptons ici la définition de Gruber pour les ontologies : une ontologie est "un accord sur une conceptualisation partagée par une communauté".

2.2.2 Le niveau "symbole"

Au niveau symbole, on trouve les éléments qui permettent de mettre en œuvre ce qui a été défini au niveau connaissance. On distingue d'une part les langages de représentation qui permettent d'implanter les connaissances dans les systèmes informatiques et d'autre part les outils (libres ou commerciaux) comme KaïDara ou JRules qui offrent un ensemble complet de fonctionnalités. Ici, nous nous intéresserons uniquement aux langages de représentation des connaissances.

Les langages de représentation des connaissances sont essentiels pour que les systèmes informatiques puissent manipuler les connaissances. Ces langages doivent avoir un fort pouvoir d'expression, être compréhensible et être accessible par la système. Il en existe plusieurs types : langages à objets, logiques de description (qui s'appuient sur des concepts empruntés à l'objet), graphes conceptuels, etc. Dans cette partie, nous allons nous intéresser aux deux premiers.

2.2.2.1 Représentation des connaissances à objets

La notion d'objet est souvent utilisée en intelligence artificielle pour décrire et organiser les connaissances [NL93], [KLW95], [LN99]. Elle trouve ses origines dans les travaux de Minsky (1975) sur les réseaux sémantiques et sur les frames ainsi que dans les travaux de Schank (1982) sur les scripts. Les frames représentent des prototypes d'objets alors que les scripts s'attachent plutôt à décrire des comportements.

Les objets peuvent être regroupés en classes décrivant toutes les propriétés communes des

objets. L'ensemble des classes d'objets est muni d'une relation d'ordre partiel appelée relation de *spécialisation* permettant de construire une hiérarchie de ces classes. Il existe une relation d'héritage entre les classes dans la hiérarchie.

Dans cette approche, les attributs des objets (frames) sont décrits à l'aide de facettes. Les attributs décrivent les différents aspects des objets. Ils peuvent exprimer soit des propriétés intrinsèques des objets soit des relations entre les différents objets. Les facettes, quant à elles, indiquent les contraintes et les propriétés des attributs. Elles peuvent être déclaratives ou procédurales. Une facette déclarative contient les informations définissant l'attribut. Une facette procédurale décrit des procédures qui sont déclenchées par réflexe lors d'un accès à l'attribut.

Le modèle de représentation des connaissances par objets est très structuré et permet donc de représenter et d'organiser facilement les connaissances.

2.2.2.2 Les logiques de description

Les logiques de description, parfois appelées logiques terminologiques, forment une famille de langages de représentation des connaissance. Elle servent de support au raisonnement terminologique. Elles trouvent leurs origines dans les travaux de Brachman sur la représentation des connaissances et notamment dans le système KL-ONE et ses nombreux descendants. À titre indicatif, le système Classic, descendant direct de KL-ONE, est utilisé dans de nombreuses application et est devenu une référence dans le domaine. Une solide introduction aux logiques de descriptions peut être trouvée dans [Nap97].

Les logiques de description permettent de représenter les connaissances d'un domaine à l'aide de *descriptions* qui peuvent être des concepts, des rôles ou bien des individus. Les concepts sont des classes d'individus tandis que les rôles représentent les relations binaires entre les classes. L'ensemble d'individus est appelé *extension* du concept. Les concepts et les rôles possèdent une description structurée. En logique de descriptions, une base de connaissance est donc composée d'une hiérarchie de concepts et d'une éventuelle hiérarchie de rôles.

Les concepts et les rôles sont organisés en hiérarchie grâce à une relation spécifique appelée relation de subsomption. Nous pouvons définir la relation de subsomption comme suit.

> Un concept C subsume un concept D si C est plus général que D au sens où l'ensemble d'individus représenté par C contient l'ensemble d'individus représenté par D.

La classification et l'instanciation sont deux opérations qui sont à la base du raisonnement sur les descriptions que nous avons appelé "raisonnement terminologique". La classification permet de déterminer les positions des concepts et des rôles dans les hiérarchies. L'instanciation permet de retrouver les concepts dont un individu est susceptible d'être une instance. Le raisonnement par classification est devenu une méthode d'inférence essentielle dans le domaine de l'intelligence artificielle.

De nombreux systèmes s'appuient sur les représentations à objets pour décrire les connaissances qu'ils manipulent. C'est également le cas de l'outil Protégé que nous allons utiliser pour mettre en œuvre notre application. C'est pourquoi une présentation de ces élements était nécessaire. Dans le chapitre suivant nous présenterons Protégé plus spécifiquement.

Conclusion

Nous avons présenté dans cet état de l'art certaines méthodes et certains outils pour la gestion des connaissances. De nombreux outils permettent d'acquérir et de modéliser les connaissances pour des applications particulières, mais rares sont ceux qui permettent de capitaliser ces connaissances entre plusieurs applications dans le cadre d'un projet par exemple. Nous nous sommes inspirés des travaux sur les mémoires de projet [MRC99], [MCR99] pour proposer un outil de gestion des connaissances qui permette de capitaliser en une même mémoire de projet des connaissances provenant à la fois de différents domaines (mécanique, aérodynamiques, etc.) et liés à différentes méthodes de raisonnement (raisonnement à partir de cas, raisonnement à base de règles, etc.).

Chapitre 3

Un environnement de capitalisation des connaissances

Dans ce chapitre, nous proposons des éléments pour répondre aux besoins en gestion des connaissances évoqués dans le premier chapitre. Nous cherchons un moyen de représenter les connaissances utilisées par les différents systèmes de manière unifiée. Nous souhaitons, à long terme, obtenir une méthode qui permette d'assurer la cohérence des connaissances manipulées et qui offre également des facilités pour réutiliser certains *blocs* de connaissances d'une application à l'autre. Nous avons deux besoins principaux : proposer un environnement de capitalisation des connaissances pour la gestion des mémoires de projets et expérimenter l'utilisation de Protégé[1] pour la mise en œuvre de cet environnement.

Nous allons présenter la façon dont nous avons abordé ces objectifs. Dans un premier temps, nous allons exposer l'architecture de ce système de capitalisation des connaissances. Nous montrerons qu'il permet de représenter des connaissances, de les manipuler, de raisonner sur ces connaissances, de les exporter dans différents formats et enfin d'effectuer des opérations de validation ou d'invalidation de certaines hypothèses. Nous nous intéresserons ensuite à la partie "représentation des connaissances". Nous expliciterons les trois modèles de connaissances principaux que nous avons défini : le modèle des connaissances du domaine, le modèle des règles et le modèle du raisonnement à partir de cas. Enfin, nous proposerons différentes déclinaisons de la notion de *point de vue* permettant de manipuler aisément les connaissances.

3.1 Architecture du système

Pour répondre à nos trois problématiques que sont la capitalisation des connaissances, la réutilisation des connaissances et l'interopérabilité des applications, nous avons choisi d'articuler notre application autour du système Protégé. Bien que nous n'ayons pas traité l'aspect interopérabilité, nous l'avons pris en compte lors de la conception de l'environnement.

Protégé est un outil de gestion des connaissances développé à l'université de Stanford. Il permet essentiellement de définir des ontologies. Grâce à Protégé, il est possible de définir des *classes*, des relations entre ces classes et des instances des différentes classes. Il permet également d'effectuer des traitements sur les modèles définis et il offre enfin des possibilités d'import et d'export des connaissances dans différents formats (XML, UML, RDF). Un rapide aperçu des fonctionnalités de Protégé est donnée dans [Knu03]. Protégé offre de nombreuses fonctionnalités que nous souhaitons utiliser dans les différentes parties de notre système.

La figure 3.1 présente l'architecture de notre système et la façon dont ses différents composants s'articulent avec Protégé. Nous allons détailler chacun de ces aspects dans les sections suivantes.

[1] http ://protege.stanford.edu

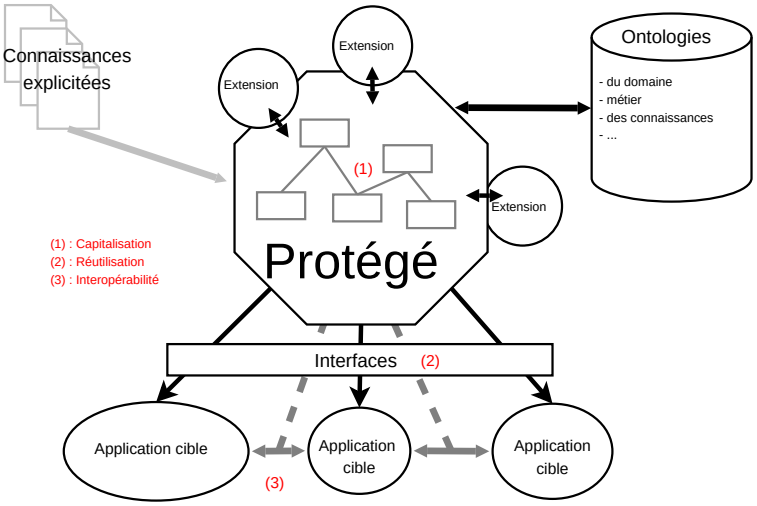

FIG. 3.1 – Architecture du système - *Sur cette figure, nous distinguons bien nos trois problématiques : la capitalisation des connaissances au cœur de Protégé avec la définition des modèles de connaissances et des différentes ontologies ; la réutilisation des connaissances dans les applications cibles et enfin, l'intéropérabilité entre les application avec l'utilisation combinée des interfaces de Protégé et des modèles de connaissances.*

3.1.1 Définition des ontologies

L'utilité première de Protégé est de permettre la représentation d'ontologies, nous allons donc l'utiliser pour représenter les ontologies correspondant aux différentes typologies de connaissances manipulées. Ces ontologies peuvent être de plusieurs types :

- Ontologies des connaissances du domaine.
- Ontologies métier, par exemple : mécanique, électricité, aérodynamiques, etc.
- Ontologies tâches : aménagement du cockpit, création d'un équipement de contrôle, etc.
- Ontologies des connaissances comme les connaissances relatives au raisonnement à partir de règles ou au raisonnement à partir de cas.

Dans cette étude, nous nous sommes focalisés sur le dernier type d'ontologies. Notre objectif étant d'enrichir la mémoire de projet avec de nouvelles connaissances, nous avons besoin de modèles génériques qui seront ensuite déclinés par spécialisation pour les différentes applications. Cette observation nous conduit à la définition de *niveaux de modélisation*. Nous distinguons trois niveaux de modélisation : le niveau *générique*, le niveau *application* et le niveau *instance*. Les modèles qui sont décrits au niveau générique sont valables pour toutes les applications. Les modèles de niveau application correspondent à une spécialisation des modèles génériques pour répondre aux besoins d'applications particulières. Un modèle de niveau instance est issu d'un modèle de niveau application et contient les connaissances d'une application particulière. Les modèles de niveau instance sont vraiment fonctionnels et servent de support de raisonnement dans des cas concrets.

Dans ce travail, nous ne nous intéresserons qu'au niveau de modélisation générique. Néanmoins, les modèles que nous proposons s'inspirent d'applications concrètes et seront validés sur des échantillons issus des systèmes réels.

3.1.2 Protégé comme outil de validation

Protégé ne propose pas seulement un environnement de représentation des ontologies, il offre également de nombreuses extensions permettant d'effectuer des opérations diverses sur ces ontologies. Dans notre système, nous souhaitons pouvoir utiliser ces extensions pour valider différentes sous parties des connaissances représentées. Nous allons présenter un bref aperçu des possibilités offertes par les extensions de Protégé.

- Visualisation des ontologies grâce à des extensions permettant un affichage graphique (graphes, représentation UML, etc.) comme Ontoviz[2] où TGViz.
- Expression de contraintes grâce au langage PAL (Protégé Axiom Language) et à l'extension EZPal qui permet de manipuler très facilement les primitives de ce langage.
- Validation des règles avec des outils comme le système à base de règles CLIPS et l'extension CLIPSTab qui permet un interfaçage très simple avec cet outil puissant.
- Mise en œuvre de facettes procédurales en Java avec l'extension JCalls. Cette opportunité permettrai notamment de palier à l'absence de mécanisme natif de gestion des facettes procédurales dans Protégé.

3.1.3 Protégé et export de modèles de connaissances

Protégé propose également des possibilités d'import et d'export des modèles de connaissances dans différents formats tels que le RDF, l'UML, le XML. Il est donc envisageable d'utiliser ces fonctionnalités pour échanger des modèles de connaissances avec des applications cibles externes.

Notons enfin que Protégé offre à l'utilisateur expérimenté la possibilité de développer ses propres extensions. Par conséquent, nous ne sommes pas limités aux outils offerts par le logiciel.

L'architecture que nous proposons permet de répondre aux trois sous parties de la problématique à laquelle nous sommes confrontés : la capitalisation des connaissances est effectuée au cœur de Protégé grâce à la mise en place de modèles de connaissances sous forme d'ontologies ; la réutilisation est rendue possible par les nombreuses possibilités d'interfaces disponibles ; l'interopérabilité des applications, quant à elle, est permise grâce à la combinaisons des interfaces qui permettent de faire de l'import/export de données et des modèles de connaissances qui permettent d'avoir une structure des connaissances commune aux différentes applications.

Nous avons proposé ici une architecture complète pour la capitalisation des connaissances mais nous n'en avons formalisé qu'une sous partie : la modélisation des connaissances au niveau générique. C'est cette partie que nous allons détailler dans la section suivante.

3.2 Capitalisation des connaissances

Dans cette section, nous allons présenter les modèles sur lesquels nous avons travaillé. Rappelons que ces modèles sont définis à un niveau générique, c'est-à-dire qu'ils sont indépendants de toute application. Nous allons présenter brièvement le modèles des connaissances du domaine puis nous nous intéresserons plus particulièrement aux modèles du raisonnement à partir de règles et du raisonnement à partir de cas.

[2]Toutes les extensions évoquées sont disponibles à l'adresse : http ://protege.stanford.edu/plugins.html

3.2.1 Les connaissances du domaine

Les connaissances du domaine sont, par définition, spécifiques au domaine d'application étudié. Par conséquent, il n'y a que peu de choses à définir au niveau du modèle générique. Nous ne donnons que les outils pour représenter ces connaissances dans les autres modèles.

Les connaissances du domaine sont définies grâce à des ensembles de descripteurs, un descripteur étant défini comme un couple attribut-valeur. Dans ce modèle, nous donnons également les outils, i.e un ensemble de relations, pour organiser les concepts. Ces relations sont nombreuses : spécialisation, composition, agrégation, etc. Dans la figure 3.2, nous donnons un exemple de la représentation des connaissances du domaine sous Protégé.

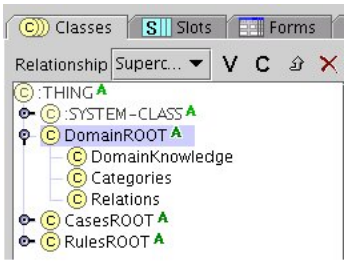

FIG. 3.2 – Modèle des connaissances du domaine, représentation sous Protégé - *Nous voyons sur cette capture d'écran que les concepts que nous avons évoqués sont représentés sous forme de "classes" au sens de Protégé.*

3.2.2 Le modèle de règles

Le modèle de règles que nous avons représenté sous Protégé s'appuie sur le schéma suivant (figure 3.3). Dans le modèle générique, nous définissons les principaux éléments constituant les règles ainsi que des concepts qui pourront ensuite être spécialisés pour représenter les tâches de raisonnement.

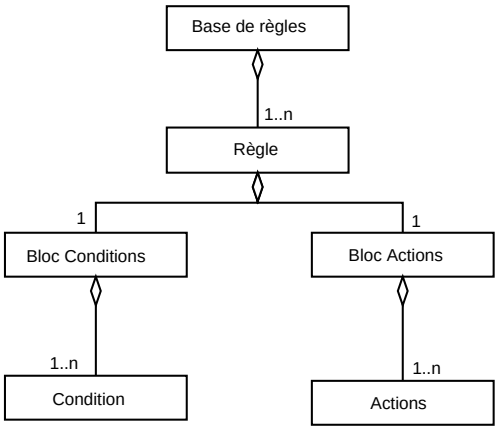

FIG. 3.3 – Modèle des connaissances de type règles - *Ce diagramme UML propose un modèle minimal d'une base de règles. Il s'inspire du modèle utilisé par EADS. Pour plus de clarté, nous n'avons pas fait figurer les tâches de raisonnement.*

3.2.3 Le modèle de cas

Le modèle des cas est le modèle le plus complexe, nous allons le présenter en détail. La figure 3.4 représente la modélisation UML d'une base de cas telle que nous l'avons représenté dans Protégé.

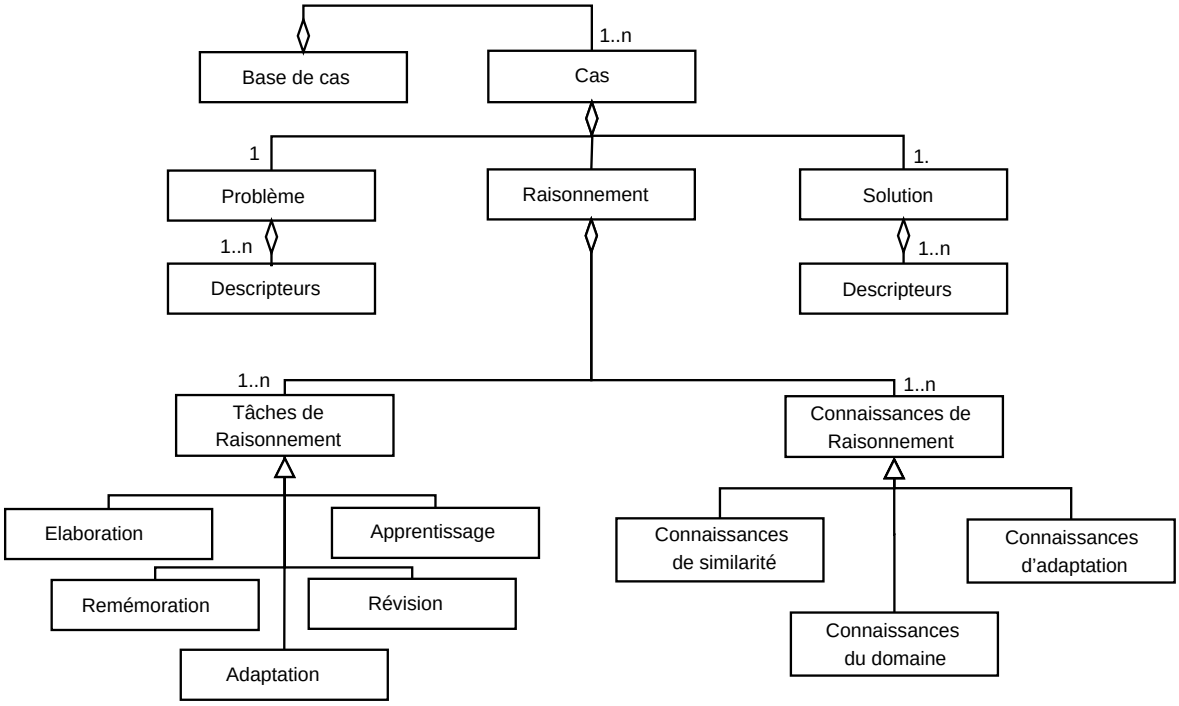

FIG. 3.4 – Modèle des connaissances de type cas - *Ce diagramme UML présente la modélisation d'une base de cas telle que nous l'avons représenté sous Protégé. Notons la présence des tâches de raisonnement ainsi que des connaissances de raisonnement qui seront détaillées plus loin.*

Pour décrire la structure des cas, nous reprendrons les notations définies dans [Fuc97]. Une base de cas est constituée de plusieurs cas. Un cas est composé de trois éléments principaux : le problème, la solution et le raisonnement conduisant du problème à la solution. Le problème et la solution sont composés de descripteurs (couples attribut-valeur). Au niveau générique, il est impossible de donner plus de détails sur ces aspects puisqu'ils dépendent du domaine.

Le raisonnement est composé de tâches de raisonnement ainsi que de connaissances de raisonnement qui sont mobilisées par les tâches. Les tâches de raisonnement correspondent aux différentes étapes du cycle du raisonnement à partir de cas. Sur la figure 3.4, nous donnons une liste non exhaustive des connaissances de raisonnement existantes. Les connaissances de similarité et d'adaptation qui sont représentées ont une grande importance dans le raisonnement à partir de cas. Il est possible de les modéliser au niveau générique.

Les modèles que nous présentons ci-dessous s'appuient sur les travaux de B. Fuchs, J. Lieber, A. Mille et A. Napoli [FLMN00] sur la remémoration des cas guidée par les critères d'adaptabilité[3].

[3]Ce travail s'appuie également sur les travaux en cours de soumission sur la formalisation de l'adaptation dans le RÀPC réalisés par la même équipe

3.2.3.1 Les connaissances de similarité

Lorsque l'on s'intéresse à la similarité, on considère les parties problèmes du cas source et du cas cible. Nous considérons que les cas sont composés d'un ensemble de descripteurs qui peuvent être structurés au moyen d'entités de type objets. Un descripteur d_i, est un couple attribut-valeur. Nous supposons que nous ne pouvons comparer que les descripteurs ayant les mêmes noms. Pour étudier la similarité entre deux cas, nous allons comparer l'ensemble de leurs descripteurs communs deux à deux. Nous noterons d_i^s les descripteurs de problèmes sources et d_i^t les descripteurs de problème cible.

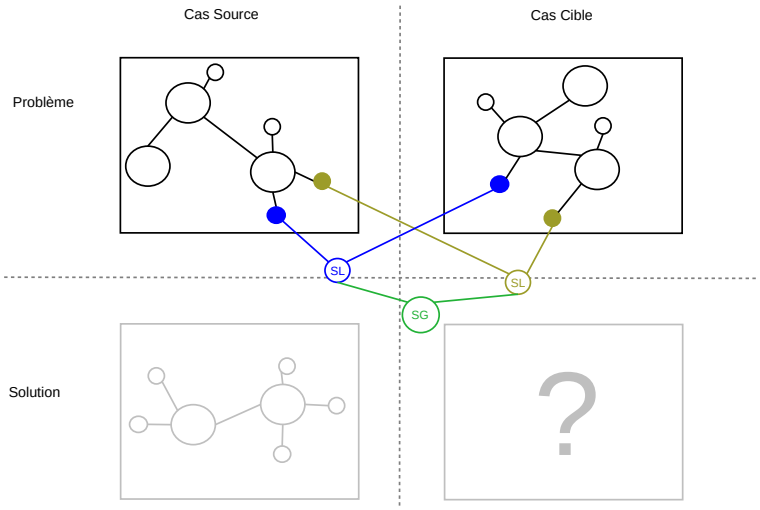

FIG. 3.5 – Calcul de la similarité - *Cette figure illustre le calcul d'une similarité entre deux cas en s'appuyant sur la comparaison de deux descripteurs de problèmes*

Des similarités locales sont calculées pour chaque couple de descripteurs considéré. Une similarité locale correspond à une différence (un écart entre les descripteurs) Δd_i entre les descripteurs pondérée grâce à un opérateur $\otimes_{pb}$ par un poids w_i qui dépend de l'importance du descripteur dans la constitution du cas. L'opérateur de "différence" dépend de l'application considérée, ce n'est pas nécessairement l'opérateur classique puisque tous les descripteurs ne sont pas numériques.

Similarité locale :

$$\Delta d_i \otimes_{pb} w_i$$

Ensuite, les différentes similarités locales sont combinées grâce à une fonction d'agrégation $\bigoplus_i(pb)$ qui dépend du domaine d'étude. Cette combinaison donne alors une valeur de similarité globale qui renseigne sur le degré de similarité entre deux cas.

Similarité globale :

$$\bigoplus_i (pb) \Delta d_i w_i$$

Cette description des similarités fait apparaître différentes notions (similarité globale, similarité locale, etc.) que nous avons représenté sous Protégé sous forme de concepts du modèle générique.

3.2.3.2 Les connaissances d'adaptation

La phase d'adaptation du RÀPC consiste à modifier la solution d'un cas source pour résoudre le problème cible.

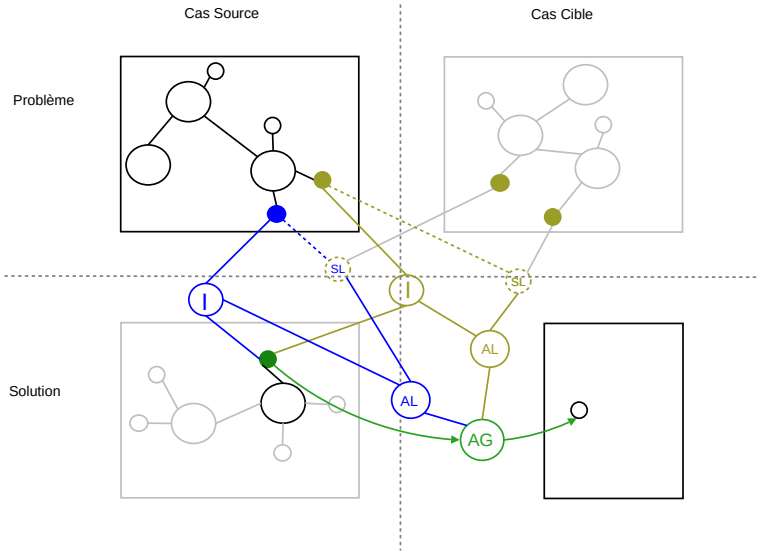

FIG. 3.6 – Calcul de l'adaptation - *Cette figure illustre la phase d'adaptation dans le raisonnement à partir de cas.*

Dans un cas, plusieurs descripteurs de problème peuvent avoir un impact sur un même descripteur de solution. Par exemple, si l'on cherche à calculer le prix de vente d'une automobile, l'âge et le kilométrage ont un impact sur le prix de vente. Cet exemple fait apparaître la notion d'*influence*. Certains descripteurs de problème ont plus d'influence sur un descripteur de solution que d'autres. Par exemple, pour reprendre le cas de la voiture d'occasion, dans le cadre du calcul du prix, le kilométrage de la voiture aura beaucoup plus d'influence que sa couleur. La première étape de l'élaboration d'un opérateur d'adaptation consiste donc à évaluer l'influence de chaque descripteur de problème sur les descripteurs de solution. On note cette fonction d'influence $\mathcal{I}(D_j^s/d_i^s)$.

Dans un deuxième temps, ces fonctions d'influence sont combinées avec les écarts Δd_i calculés précédemment. Nous obtenons alors autant de fonctions d'adaptation locales que de descripteurs de problème source mis en jeu. Ces fonctions sont définies de la sorte : $\Delta_i D_j = \Delta d_i \otimes_{Sol} \mathcal{I}(D_j^s/d_i^s)$.

Les adaptations locales sont ensuite combinées grâce à un opérateur d'agrégation $\bigoplus_i(Sol)$. Cette opérateur donne alors les éléments nécessaires pour adapter la valeur d'un attribut de solution cible D_j^t correspondant à l'attribut de solution source D_j^s étudié.

Bien entendu, il est possible de raffiner chacun des points du modèle que nous venons de présenter pour répondre aux besoins d'applications spécifiques. Cette étape de spécialisation devra se faire au niveau *application* de la modélisation.

3.3 Les points de vue

Une fois l'outil de capitalisation des connaissances mis en œuvre, le problème de la réutilisation se pose. Comment retrouver ce qui a été fait dans le passé sur un sujet donné (la conception d'un moteur d'avion par exemple) ? Comment réutiliser des morceaux de la connaissance capitalisée étroitement liées entre elles ? Comment naviguer dans les connaissances et visualiser des sous ensembles pertinents pour un problème donné ? Afin d'apporter des réponses à ces problèmes, nous proposons d'introduire la notion de *point de vue*.

3.3.1 Introduction

La notion de point de vue ou perspective[4] est couramment utilisée en informatique pour apporter de la modularité aux systèmes. En effet, imaginons qu'il soit possible de décrire des objets en ne s'intéressant qu'aux propriétés qui sont utiles pour un problème donné, et ce, à tout moment. Il devient alors très facile de manipuler les objets lors de l'exécution de tâches cognitives puisque l'on se limite toujours aux connaissances strictement nécessaires. Cette notion a intéressé beaucoup de chercheurs et pourtant, bien que très utile, elle reste très peu formalisée et il existe presque autant de définition des points de vue que d'utilisations qui en sont faites. Dans le domaine de la représentation des connaissances, les points de vue peuvent être considérés comme un moyen de structurer une base de connaissances selon différentes perspectives relatives au domaine d'application. Ils permettent également de prendre en compte les connaissances provenant des différentes disciplines d'un domaine. Il est également possible de les utiliser pour décomposer des problèmes en sous problèmes. Ce concept est déjà utilisé dans le domaine de la représentation orientée objet ainsi que dans le domaine des graphes conceptuels ([RD97]).

Concrètement, dans de nombreux systèmes, les points de vue sont considérés comme une sélection parmi les caractéristiques d'un objet de celles qui seront pertinentes pour une tâche ou une personne particulière. Les façons d'utiliser les points de vue sont nombreuses : par exemple, il est possible de sélectionner à chaque étape du raisonnement le point de vue dont les conclusions sont jugés les meilleures (c'est le cas dans le système SAXEX) ; il est également possible de confronter des solutions obtenues indépendamment les une des autres dans différents points de vue afin d'aboutir à un consensus, comme cela est fait dans le système IDEM ; enfin, chaque point de vue peut apporter sa propre part à la solution [DLN04].

Certains chercheurs on effectué des tentatives de formalisation de points de vue [AS95] basées sur la logique du premier ordre. Mais ces formalisations sont trop abstraites pour pouvoir être utilisables dans un système avec des objectifs opérationnels.

Nous proposons ici une nouvelle approche formelle des points de vue qui s'inspire fortement des travaux que nous allons présenter dans la section suivante. Cette nouvelle approche a une double ambition : d'une part, elle cherche à faciliter la visualisation des éléments contenus dans les bases de connaissances et d'autre part, elle offre un moyen simple délimiter des blocs de connaissances pour leur réutilisation.

3.3.2 Points de vue et représentation des connaissances

Le système d'aide à la décision en cancérologie Kasimir [DLN02] utilise les points de vue comme un cadre pour la décomposition de la phase d'adaptation dans le raisonnement à partir

[4]Nous utiliserons ici indifféremment les deux appellation

de cas. Les sous problèmes issus de cette décomposition peuvent être résolus indépendamment les uns des autres puisque l'hypothèse de l'indépendance des points de vue est à la base de cette conception.

Dans cette approche, les points de vue permettent également de représenter efficacement les dépendances entre les problèmes et les solutions ainsi que les contraintes qui sont utiles pour guider l'adaptation. Les contraintes peuvent par exemple réduire le nombre de valeurs à considérer.

La conception des points vues proposée ici s'inspire du modèle TROEPS proposé par Olga Marino [Mar93]. TROEPS est l'un des premiers système de représentation des connaissances par objet qui propose un modèle de points de vue. Dans TROEPS, un concept possède un ensemble de points de vue. Un point de vue d'un concept est une hiérarchie des classes. Il existe un mécanisme de passerelle qui permet de représenter le lien entre les différents points de vue d'un même concept. C'est ce mécanisme de passerelle qui est utilisé pour représenter les contraintes dans Kasimir.

Dans Creek [OA98], Pinar Öztürk et Agnar Aamodt proposent d'utiliser les points de vue comme un mécanisme pour sélectionner les zones sur lesquelles il est nécessaire de focaliser son attention lors du raisonnement. En d'autres termes, les points de vue définissent un sous ensemble d'attributs pertinents pour la tâche de raisonnement en cours. Cette définition est guidée par le but à atteindre. Cette approche n'apporte pas de solution directe à la représentation des objets dans le système (tous les attributs d'un objet, quels qu'ils soient, sont stockés de manière identique) mais elle permet de faire une sélection considérable des informations manipulées lors de la phase de résolution de problème.

3.3.3 Une nouvelle approche des points de vue

L'étude des travaux antérieurs nous a permis de voir les différentes utilités des points de vue ainsi que les différentes façons de les concevoir. Cela nous a inspiré la définition de deux types de points de vue aux utilisations différentes. Afin d'illustrer notre explication, nous considérerons que les connaissances sont organisées en un treillis de concepts avec une relation de généralisation/spécialisation (cf. figure 3.7). Cette hypothèse n'est pas une nécessité mais nous la posons pour des raisons pratiques.

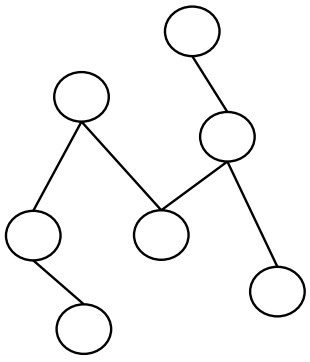

Fig. 3.7 – Un treillis de concepts

3.3.3.1 Les points de vue statiques

Les points de vue statiques sont définis comme un ensemble de concepts reliés par une relation de spécialisation situés sous un concept *général* dans le treillis de concepts. Nous appelons le concept général *racine* du point de vue. Ainsi, les points de vue statiques permettent d'éclairer un sous-ensemble de la hiérarchie de concepts selon un angle particulier.

Les points de vue statiques peuvent se montrer très utiles pour la réutilisation et le partage de blocs de connaissances. Ils permettent de délimiter le contour d'un modèle de connaissances de sorte à en faire une entité réutilisable. En effet, en sélectionnant une sous partie d'un ensemble de concepts, on s'expose toujours au risque d'omettre un concept sans lequel d'autres n'auraient pas de sens. En travaillant sur les intersections de points de vue statiques, il est donc possible de définir des sélections "restrictives" ou "étendues" des concepts à conserver, ce qui apporte une solution au problème précédent. Conformément à ce que nous avons présenté plus haut, nous pouvons utiliser ce type de perspectives pour apprécier la partie RÀPC et la partie règles d'un système hybride indépendamment l'une de l'autre.

Nous avons considéré ici que les concepts étaient organisés grâce à une relation de généralisation/spécialisation, mais nous pourrions également imaginer utiliser d'autres types de relations. Il serait alors possible de définir d'autres types de points de vue statiques.

3.3.3.2 Les points de vue dynamiques

La définition des points de vue dynamiques est orthogonale à la précédente. Comme leur nom l'indique, les points de vue dynamiques sont construits automatiquement à la demande de l'utilisateur d'une application particulière. Ils correspondent à une requête spécifique et peuvent être comparées aux *vues* en base de données.

Les points de vue dynamiques se présentent comme la sélection d'un ensemble de concepts partageant un attribut commun ou étant en relation les uns avec les autres par une relation spécifique. Par exemple, la sélection de l'ensemble des concepts traitant de l'aspect mécanique d'un moteur d'avion constitue une perspective dynamique.

Ce type de perspective a pour but de faciliter la visualisation de la hiérarchie de concept lors de l'utilisation de l'application. Au mécanisme de sélection des concepts, nous ajoutons la notion de *profondeur*. Cette notion permet de définir la granularité de la vision que l'on souhaite avoir de la hiérarchie de concepts. Son fonctionnement est simple, il consiste à limiter le nombre de concepts "fils" dans la hiérarchie de concepts que l'on affiche.

Le système que nous avons présenté ici est loin d'être complet. Nous nous sommes essentiellement focalisés sur les formalisations des modèles de connaissances génériques ainsi que sur la notion de points de vue. Dans le chapitre suivant, nous présenterons les limites de ce système ainsi que les améliorations dont il pourrait faire l'objet.

Chapitre 4

Discussion

L'architecture que nous avons proposée s'articule autour du système de construction d'ontologies Protégé. Le formalisme de représentation des connaissances proposé par Protégé ne fait pas l'objet d'un consensus. Il s'inspire fortement des représentations à objets mais n'en implante pas tous les concepts, comme par exemple les facettes procédurales ou les mécanismes de classification. Ces lacunes s'expliquent par le fait que Protégé n'est pas un système destiné à l'opérationnalisation du raisonnement mais à la capture des connaissances ce qui empêche la représentation d'une partie importante des connaissances à caractère procédural. D'autres aspects mineurs liés à l'interface limitent l'utilisation de Protégé et ralentissent souvent le travail.

Néanmoins, Protégé possède de nombreux avantages qu'il convient de souligner. D'un point de vue ingénierie, les versions stables du logiciel sont utilisables sur toutes les plate-formes. La communauté des utilisateurs et des développeurs est très réactive, ce qui accélère grandement la vitesse de correction des éventuels "bugs" et surtout l'intégration rapide de nouvelles fonctionnalités souvent fort utiles. Mais ce qui est le plus appréciable, ce sont les nombreuses extensions qui augmentent considérablement les fonctionnalités du logiciel et la richesse des types de connaissances que l'on peut potentiellement représenter. Ces arguments nous ont poussé à utiliser Protégé dans le cadre de ce projet, d'autant qu'il a déjà été expérimenté pour d'autres projets de recherche dans l'équipe CEXAS.

En ce qui concerne les modèles de connaissances, les propositions d'ontologies que nous avons réalisées sont restées à un niveau générique bien qu'inspirées par des extraits significatifs des connaissances modélisées par EADS CCR. L'ontologie de cas que nous avons proposée s'est limitée aux connaissances liées à la remémoration et à l'adaptation qui sont les phases centrales du raisonnement. Les connaissances relatives aux tâches d'élaboration, de révision et d'apprentissage n'ont pas été abordées car cela nous mènerait bien au delà de l'objectif de ce travail.

Enfin, les réflexions sur la notion de point de vue sont restées à un état de propositions d'idées qu'il faudra encore définir et formaliser plus rigoureusement. De nombreux points soulevés ici seront à prendre en compte dans les réflexions futures. Il faudra notamment s'interroger sur les différentes perspectives statiques que l'on peut définir ainsi que sur la façon de procéder avec les "intersections" entre les perspectives.

Conclusion

Nous avons proposé une architecture pour la gestion des connaissances dans les systèmes à base de connaissances hybrides. Ce domaine étant très vaste, nous nous sommes focalisés plus particulièrement sur des aspects qui nous semblaient les plus pertinents. Nous nous sommes intéressés en particulier au raisonnement à partir de cas qui est un des principaux thèmes de recherche de l'équipe CEXAS et dont l'aspect représentation est un point délicat. Nous nous sommes également focalisé sur la notion de point de vue, sujet qui reste peu formalisé et difficile à aborder.

Les perspectives de ce travail sont nombreuses. A court terme, nous prévoyons de valider les modèles de connaissances exposés ici en s'appuyant sur des exemples concrets et plus complets extraits des applications de EADS CCR.

A plus long terme, un travail considérable de développement des différents éléments prévus dans notre architecture est nécessaire. Il faudra développer de nouveaux modèles de connaissances et compléter les modèles existants. Il sera également nécessaire de proposer une méthodologie de modélisation et de structuration des connaissances facilitant leur réutilisation et leur manipulation. En effet, la connaissance est constituée d'un réseau très complexe de concepts et de relations entre ces concepts qu'il est difficile de maîtriser. La notion de perspective ou "point de vue" reste à approfondir et à formaliser plus rigoureusement pour l'expérimenter en vue d'offrir un cadre structurant pour la définition des connaissances et leur réutilisation future.

Bibliographie

[Aam91] Agnar Aamodt. *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning*. PhD thesis, University of Trondheim, Norwegian Institute of Technology, Department of Computer Science, 1991.

[AP94] Agnar Aamodt and Enric Plaza. Case-based reasoning : Foundational issues, methodological variations, and system approaches. *AICOM*, 7 :39–59, 1994.

[AS95] Giuseppe Attardi and Maria Simi. A formalisation of viewpoints. *Fundamenta Informaticae*, 23 :149–174, 1995.

[CW03] Huajun Chen and Zhaohui Wu. CaseML : a RDF-Based Case Markup Language for Case-Based Reasoning in Semantic Web. In *WS5 : From Structured Cases to Unstructured Problem Solving Episodes For Experience-Based Assistance at ICCBR'03*, pages 324–333, NTNU, Trondheim, Norway, 23-26 June 2003.

[DCG+00] Rose Dieng, Olivier Corby, Alain Giboin, Joanna Golebiowska, Nada Matta, and Myriam Ribière. *Méthodes et outils pour la gestion des connaissances*. Dunod, 2000.

[DEMN98] Roland Ducournau, Jérôme Euzenat, Gérald Masini, and Amedeo Napoli, editors. *Langages et modèles à objets : état et perspectives de la recherche*, chapter 14. Number 19 in Didactique. INRIA, Rocquencourt (FR), 1998.

[DLN02] Mathieu D'Aquin, Jean Lieber, and Amedeo Napoli. Représentation multi-points de vue des connaissances pour l'adaptation. In Marie-Christine Jaulent, Christel Le Bozec, and Eric Zapletal, editors, *Actes du Xème séminaire français de raisonnement à partir de cas*, pages 23–31, 2002.

[DLN04] Mathieu D'Aquin, Jean Lieber, and Amedeo Napoli. Représentation de points de vue pour le raisonnement à partir de cas. In *LMO*, pages 245–258, 2004.

[FK85] Richard Fikes and Tom Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9) :904–920, 1985.

[FLMN00] Béatrice Fuchs, Jean Lieber, Alain Mille, and Amedeo Napoli. An Algorithm for Adaptation in Case-Based Reasoning. In *Proceedings of the 14th European Conference on Artificial Intelligence - ECAI 2000*, pages 45–49, Berlin, 20-25 august 2000. IOS Press.

[Fuc97] Béatrice Fuchs. *Représentation des connaissances pour le raisonnement à partir de cas : le système Rocade*. PhD thesis, Université Jean Monnet, Saint-Etienne, 1997.

[FvH94] Dieter Fensel and Frank van Harmelen. A Comparison of Languages which Operationalize and Formalize KADS Models of Expertise. *The Knowledge Engineering Review*, 9 :105–146, 1994.

[JT98] Michel Jaczynski and Brigitte Trousse. An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners. In *6Th German Workshop on Case-Based Reasoning*, Berlin, Germany, 1998.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. Technical Report TR-90-003, Journal ACM, 1995. 42 :741-843.

[Knu03] Holger Knublauch. An AI tool for the real world - Knowledge modeling with Protégé. Web, June 2003.

[Lie97] Jean Lieber. *Raisonnement à partir de cas et classification hiérarchique. Application à la planification de synthèse en chimie organique.* PhD thesis, Université Henri Poincaré - Nancy 1, 1997.

[LN99] Jean Lieber and Amedeo Napoli. Raisonnement à partir de cas et résolution de problèmes dans une représentation par objets. *Revue d'intelligence artificielle*, 13 :9–35, 1999.

[Mar93] Olga Marino. *Raisonnement classificatoire dans une representation a objets multi-points de vue.* PhD thesis, LIFIA / IMAG / INRIA Rhône-Alpes, Université Joseph Fourier - Grenoble I., 1993.

[Mar94] Philippe Martin. La Methodologie d'acquisition des connaissances KADS et les explications. Technical report, INRIA - Sophia Antipolis, 1994.

[MC96] Marie-Laure Mugnier and Michel Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1) :7–56, 1996.

[MCR99] Nada Matta, Olivier Corby, and Myriam Ribiere. Méthodes de capitalisation de mémoire de projet. Technical report, INRIA - Equipe ACACIA, 1999.

[MOK98] Consortium MOKA. State of the Art Study of Current Methodologies. Technical report, MOKA, 1998.

[MOK99] Consortium MOKA. Report on Testing the MOKA Methodology (Phase I) (D3.2). Technical report, MOKA, 1999.

[MOK00] Consortium MOKA. MOKA Modelling Language MML-WG Definition. Technical report, MOKA, 2000.

[MRC99] Nada Matta, Myriam Ribiere, and Olivier Corby. Définition d'un modèle de mémoire de projet. Technical report, INRIA - Equipe ACACIA, 1999.

[Nap97] Amedeo Napoli. Une introduction aux logiques de description. Technical report, INRIA, 1997.

[NL93] Amedeo Napoli and Claude Laurenço. Représentations à objets et classification. Conception d'un système d'aide à la planification de synthèses organiques. *Revue d'intelligence artificielle*, 7(2) :175–221, 1993.

[OA98] Pinar Oztürk and Agnar Aamodt. A context model for knowledge-intensive case-based reasoning. *International Journal of Human Computer Studies*, 48 :331–355, 1998.

[RD97] Myriam Ribière and Rose Dieng. Introduction of Viewpoints in Conceptual Graph Formalism. In *ICCS 1997*, pages 168–192, 1997.

[Sch82] Roger Schank. *Dynamic memory ; a theory of reminding and learning in computers and people.* Cambridge University Press, 1982.

[Ste90] Luc Steels. Components of expertise. *AI Magazine*, 11(2) :30–49, 1990.

[SV98] Sylvie Salotti and Véronique Ventos. Study and Formalization of a Case-Based Reasoning System Using a Description Logic. In *EWCBR*, pages 286–297, 1998.

[WVdVSA92] Bob Wielinga, Walter Van de Velde, Guus Schreiber, and Hans Akkermans. The KADS Knowledge Modelling Approach. In *2nd Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 23–42, Hatoyama, Saitama, Japan, 1992. Hitachi, Advanced Research Laboratory.

Gestion des connaissances pour des systèmes à base de connaissances hybrides

Résumé

La gestion des connaissance est aujourd'hui un enjeu capital pour les sociétés manipulant de plus en plus de "savoir faire". Le groupe EADS CCR, qui développe des systèmes à base de connaissances hybrides souhaite pouvoir capitaliser les connaissances dans une mémoire de projet pour pouvoir les réutiliser plus facilement dans différentes applications cibles. Nous proposons une architecture centrée sur l'application Protégé pour la capitalisation et la réutilisation des connaissances ainsi que pour l'interopérabilité des applications. Nous décrivons les différentes composantes de cette architecture puis nous formalisons les modèles de connaissances de deux systèmes à base de connaissances spécifiques : les systèmes à base de règles et les systèmes de raisonnement à partir de cas. Enfin, nous introduisons la notion de *points de vue* pour faciliter la manipulation et la visualisation des connaissances dans notre architecture.

Mots clés : Gestion des connaissances, Systèmes à Base de Connaissances, Mémoires de projet, Raisonnement à Partir de Cas, Systèmes à Base de Règles, Modèles de Connaissances, Points de vue.

Abstract

Knowledge management is today of crucial importance to companies increasingly using knowledge. EADS CCR who are developing Knowledge Based Systems from hybrid sources, would like to capitalize data within a project memory to be able to re-use them more easily in various target applications. We offer a framework centred on the Protégé application for the capitalization and re-use of knowledge as well as interoperability of applications. We describe the different components of this framework, then we formulate knowledge models of two systems based on specific information : Rule-Based Systems and Case-Based Reasoning Systems. Finally, we introduce the notion of *viewpoints* to facilitate the handling and visualisation of knowledge within our framework.

Keywords : Knowledge Management, Knowledge Based Systems, Project Memories, Case-Based Reasoning, Rule-Based Systems, Knowledge models, Viewpoints.