HAL
open science

COBRA : Une plate-forme de RàPC basée sur des ontologies

Amjad Abou Assali, Dominique Lenne, Bruno Debray, Sébastien Bouchet

To cite this version:

Amjad Abou Assali, Dominique Lenne, Bruno Debray, Sébastien Bouchet. COBRA : Une plate-forme de RàPC basée sur des ontologies. IC 2009, May 2009, Hammamet, Tunisie. pp.277-288. hal-00377373

HAL Id: hal-00377373

https://hal.science/hal-00377373v1

Submitted on 21 Apr 2009

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

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

COBRA : Une plate-forme de RàPC basée sur des ontologies

Amjad Abou Assali[1], Dominique Lenne[1], Bruno Debray[2] et Sébastien Bouchet[2]

[1] Université de Technologie de Compiègne, CNRS
HEUDIASYC
{aabouass,dominique.lenne}@utc.fr

[2] INERIS[‡]
{bruno.debray,sebastien.bouchet}@ineris.fr

Résumé :
Cet article présente un projet en cours qui a pour objectif de développer une plate-forme de RàPC pour le diagnostic basée sur des ontologies, appelée COBRA. Cette plate-forme est constituée de deux parties principales : les modèles de connaissances décrits par des ontologies, et les processus de raisonnement. Nous travaillons actuellement sur la défaillance des barrières de sécurité installées sur des sites industriels. Cependant, notre objectif est de rendre la plate-forme générique et indépendante du domaine d'application. Nous affirmons que, pour mieux exploiter les avantages des ontologies dans les systèmes de RàPC, il est important de pouvoir utiliser n'importe quel concept dans la description des cas. Ainsi, COBRA permet de définir les attributs de chaque cas dynamiquement au moment de l'exécution, ce qui conduit à une base de cas hétérogène. Dans cet article, nous présentons l'architecture de la plate-forme, les modèles de connaissances, les processus principaux, ainsi que les problèmes rencontrés en travaillant avec des cas hétérogènes.

Mots-clés : Raisonnement à partir de cas, Ontologie, Base de cas hétérogène.

1 Introduction

La gestion des risques est devenue une préoccupation importante sur la plupart des sites industriels. Pour réduire les risques potentiels, des barrières de sécurité sont proposées par des experts. Toutefois, ces barrières peuvent échouer à assurer la fonction de sécurité pour laquelle elles ont été installées, et des accidents peuvent se produire. Dans ce contexte, un expert industriel intervient pour diagnostiquer le dysfonctionnement des barrières. Dans un premier temps, il essaie de se remémorer des expériences de dysfonctionnement observées dans des situations similaires. L'hypothèse que l'expert

[‡]Institut National de l'Environnement industriel et des RISques.

fait est que “si une barrière n’a pas bien fonctionné dans une autre situation similaire, il est fortement probable qu’elle ne fonctionne pas, dans la situation actuelle, *pour des raisons similaires*”. Quand l’expert avance dans son diagnostic, il cherche parfois à obtenir plus d’informations sur la situation actuelle pour pouvoir trouver la bonne cause de défaillance. Pour simuler l’activité de l’expert, nous utilisons une approche de RàPC (Riesbeck & Schank, 1989) conversationnelle.

Le RàPC est une approche de résolution de problèmes. Il a pour objectif de résoudre un nouveau problème, appelé *problème cible*, à l’aide d’un ensemble de problèmes déjà résolus, appelés *problèmes sources*. Les approches *knowledge-intensive* du RàPC (KI-CBR) sont celles pour lesquelles les connaissances du domaine jouent un rôle fondamental (et pas uniquement la base de cas). Dans notre travail, nous suivons une approche knowledge-intensive et conversationnelle car les cas sont enrichis au fur et à mesure que l’expert avance dans son analyse.

Nous travaillons actuellement sur la défaillance des barrières de sécurité installées sur des sites industriels, en particulier les capteurs de gaz. Ces capteurs déclenchent une alarme lorsqu’il y a une fuite de certains gaz quelque part dans le site. Dans ce contexte, un cas représente un diagnostic de la défaillance d’un capteur de gaz dans un environnement donné.

Nous présentons, dans cet article, COBRA (Conversational Ontology-based CBR for Risk Analysis), une plate-forme de KI-CBR indépendante du domaine d’application. Cette plate-forme est basée sur des modèles de connaissances représentés par des ontologies pour décrire le domaine et les cas. Nous affirmons qu’il est important de ne pas prédéfinir la structure (les attributs) des cas dans les systèmes de KI-CBR, mais de permettre à l’utilisateur de décrire ses cas par n’importe quel concept de l’ontologie de domaine. Par conséquent, nous obtenons une base de cas hétérogène où les cas peuvent être décrits par différents attributs. En outre, nous présentons les processus d’authoring et de remémoration des cas, ainsi que les mesures de similarité utilisées pour pallier les problèmes liés à l’hétérogénéité des cas.

2 État de l’art

L’intégration des connaissances génériques du domaine d’application dans les systèmes de KI-CBR a été un aspect important dans plusieurs projets. Dans l’architecture de CREEK (Aamodt, 1994), nous trouvons un couplage assez fort entre les connaissances des cas et celles du domaine. Ainsi, les cas sont immergés dans un modèle générique du domaine représenté par un réseau sémantique. Fuchs & Mille (2005) ont proposé une modélisation du RàPC au niveau connaissance. Ils ont distingué quatre modèles de connaissance : 1) le modèle conceptuel du domaine décrivant les concepts utilisés pour décrire l’ontologie du domaine indépendamment du raisonnement ; 2) le modèle de cas qui sépare le cas en problème, solution, et trace de raisonnement ; 3) les modèles de tâches de raisonnement qui comprennent un modèle de spécification et un autre de décomposition de tâches ; 4) et les modèles supports du raisonnement. D’Aquin *et al.* (2006) ont travaillé sur l’intégration du RàPC dans le Web sémantique. Pour cela, ils ont proposé une extension de OWL (Ontology Web Language) permettant de représenter les connaissances d’adaptation du RàPC. L’expression des connaissances

du domaine et des cas en OWL leur a permis de rajouter au système de RàPC les capacités de raisonnement propres à OWL en exploitant, par exemple, la subsumption et l'instanciation. Dìaz-Agudo & Gonzàlez-Calero (2000) ont proposé une architecture indépendante du domaine qui aide à l'intégration d'ontologies dans les applications de RàPC. Leur approche consiste à construire des systèmes intégrés qui combinent des connaissances spécifiques aux cas avec des modèles génériques des connaissances du domaine. Ils ont présenté CBROnto (Dìaz-Agudo & Gonzàlez-Calero, 2002), une ontologie de tâche/méthode qui fournit le vocabulaire nécessaire pour décrire les éléments impliqués dans les processus de RàPC, et qui permet également d'intégrer différentes ontologies de domaine. CBROnto a été réutilisée plus tard par jCOLIBRI, un framework orienté-objet (en JAVA) assez puissant pour la construction de systèmes de RàPC (Recio-Garcìa *et al.*, 2006; Dìaz-Agudo *et al.*, 2007). jCOLIBRI sépare la gestion des bases de cas en deux aspects : la persistance et l'organisation en mémoire, ce qui permet d'avoir différents supports de stockage de cas (fichiers text/XML, ontologie, *etc.*) accessibles via des connecteurs spécifiques. Toutefois, jCOLIBRI ne permet pas le traitement de bases de cas dynamiques et hétérogènes, ce qui nous a amené à développer une couche supplémentaire pour pallier ce manque.

3 Architecture de COBRA

Plusieurs architectures des systèmes de RàPC ont été proposées dans la littérature. Ces architectures partagent plus ou moins les mêmes composantes. Aamodt & Plaza (1994) ont présenté le cycle fameux de RàPC constitué des processus : REMÉMORER, ADAPTER, RÉVISER, et MÉMORISER. Un cinquième processus, ÉLABORER, a été distingué plus tard (voir Renaud *et al.* (2007)). En 2002, Lamontagne & Lapalme ont présenté une vue globale où l'on trouve les processus hors-ligne/en-ligne, et les connaissances ("knowledge containers") permettant de préserver et exploiter les expériences passées. Comme le montre la figure 1, COBRA est basé sur ces architectures et est composé de deux parties principales :

- *Processus* : cette partie est constituée d'un processus hors-ligne, *authoring des cas*, et de six processus de raisonnement : ÉLABORER, REMÉMORER, DIAGNOSTIQUER, ENRICHIR, VALIDER et MÉMORISER. Dans les systèmes conversationnels de RàPC pour le diagnostic, il est important de distinguer deux processus fondamentaux, *diagnostiquer* et *enrichir*. Dans la phase de "diagnostic", le système essaie d'identifier les causes de défaillance à partir des cas similaires. Si aucune cause n'est trouvée, ou si le diagnostic proposé par le système n'a pas été validé par l'utilisateur, le système demande à l'utilisateur d'enrichir la description du cas cible pour chercher de nouveaux cas similaires. Ce cycle se reproduit jusqu'à ce qu'un bon diagnostic soit proposé par le système, ou qu'aucune solution ne puisse plus être trouvée.
- *Connaissances* : dans les systèmes de RàPC, on distingue quatre catégories de connaissances ("knowledge containers") : vocabulaire, base de cas, mesures de similarité, et connaissances d'adaptation. Dans notre système, nous retrouvons les trois premières catégories, mais nous n'avons pas de règles d'adaptation pour le moment.

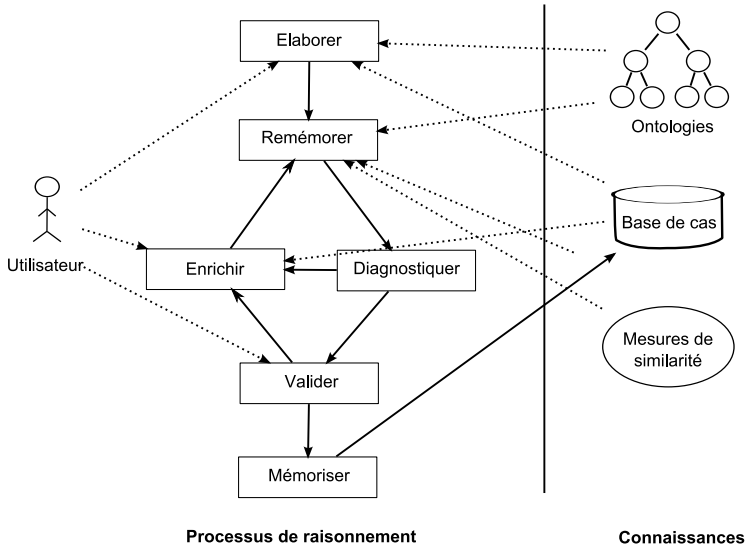

FIG. 1 – Architecture de COBRA.

Pour décrire le vocabulaire et la base de cas, le système se base sur deux modèles de connaissances : les modèles de domaine et de cas.

3.1 Modèle de domaine

Ce modèle représente les connaissances du domaine sous forme d'une ontologie. Dans les systèmes de KI-CBR, les ontologies jouent un rôle important (Recio-Garcìa *et al.*, 2006) comme vocabulaire pour décrire les cas, comme structure de connaissances où les cas sont localisés, et comme source de connaissance permettant le raisonnement sémantique dans les méthodes de calcul de similarité.

Dans ce travail, nous utilisons l'ontologie noyau développée dans (Abou Assali *et al.*, 2008), qui contient des concepts génériques sur la sécurité industrielle tels que : Événement, Phénomène dangereux, Explosion, Effet, *etc.* Une autre ontologie de domaine a été élaborée dont les concepts sont des spécialisations de l'ontologie noyau. Elle décrit le domaine des barrières de sécurité, le domaine d'application actuel (voir FIG. 2). Les ontologies sont décrites en OWL Lite, et elles ont été développées par plusieurs experts de l'INERIS avec l'aide d'un expert en ontologie.

3.2 Modèle de cas

Un cas dans notre système est un cas de diagnostic. Il contient trois parties principales : la partie *description*, qui décrit le contexte dans lequel a été réalisé le diagnostic, la partie *mode de défaillance*, et la partie *causes*. Prenons, par exemple, la défaillance de capteurs de gaz. La partie description peut contenir : le gaz à mesurer, la technologie du capteur utilisé, le seuil de la concentration d'alarme (*i.e.* à quelle concentration

du gaz le capteur doit déclencher une alarme), *etc.* Le mode de défaillance peut être : une fausse alarme, une absence de détection de gaz, *etc.* Enfin, la cause de défaillance peut être : la technologie du capteur est inadaptée pour le gaz à mesurer, un mauvais calibrage du capteur, *etc.* Un cas est généralement décrit par un couple (*problème*, *solution*). Selon notre modèle, les parties *description* et *mode de défaillance* correspondent à la partie *problème*, et la partie *causes* correspond à la partie *solution*.

Pour améliorer la communication entre la base de cas et le modèle de domaine, notre modèle de cas est représenté à l'aide d'une ontologie qui intègre le modèle de domaine. Nous nous inspirons en cela de l'approche utilisée dans jCOLIBRI. Cette ontologie contient les concepts racines suivants (FIG. 2) :

- CBR-CASE qui subsume les concepts représentant les différents types de cas qui peuvent exister dans le système.
- CBR-DESCRIPTION qui subsume les concepts représentant les parties d'un cas (mode de défaillance, cause, *etc.*).
- CBR-INDEX qui permet d'intégrer les concepts du modèle de domaine.

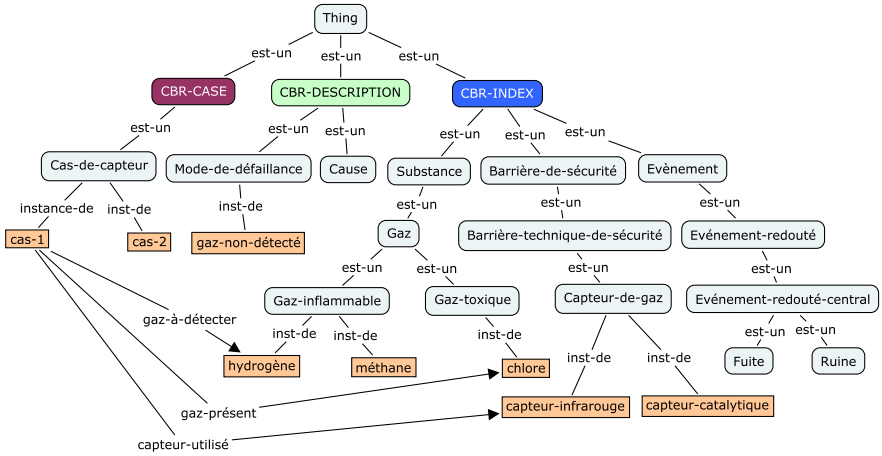

FIG. 2 – Modèle de cas.

Les cas sont alors représentés par des instances de l'ontologie, et ils ont donc deux types d'attributs :

- des attributs simples correspondant à des propriétés *data-type* de l'ontologie qui prennent des valeurs simples, *i.e.* string, int, float, *etc.*
- des attributs complexes correspondant à des instances de l'ontologie.

4 Processus de RàPC

Nous décrivons dans la suite les premiers processus de notre cycle de RàPC : authoring des cas, élaboration d'un cas cible (d'une requête), et remémoration de cas.

4.1 Authoring des cas

Pour initialiser les bases de cas, différents moyens peuvent être envisagés (Yang *et al.*, 2008). Dans notre travail, des experts du domaine ont été sollicités, dans un premier temps, pour construire des cas de diagnostic de la défaillance de capteurs de gaz à partir de leur expérience. Puis, des cas ont été rajoutés à partir des documentations existantes. Les cas sont décrits par des concepts du modèle de domaine. Toutefois, nous nous sommes rendu compte que nos cas ne partagent pas toujours la même structure, autrement dit, les mêmes attributs. Pour cela, deux objectifs ont émergé :

- Tout d'abord, les experts ne doivent pas être limités à un certain nombre d'attributs pour décrire leur expérience. Ils doivent pouvoir utiliser tout concept du modèle de domaine dont ils ont besoin. Cela conduit donc à une base de cas hétérogène, ce qui complique la phase de remémoration de cas.
- Comme les cas peuvent être décrits par différents attributs, il est utile d'aider l'expert durant l'authoring de son expérience. Pour ce faire, le système lui montre une liste des concepts les plus utilisés dans les cas similaires ordonnancés suivant leur importance. La base de cas est alors exploitée non seulement dans la phase de remémoration, mais aussi dans la phase d'authoring des cas.

Le processus d'élaboration de cas cibles est similaire. Par contre, des poids peuvent être associés aux attributs d'un cas cible pour être pris en compte dans la phase de remémoration (voir la section suivante).

4.2 Remémoration de cas

Dans cette phase, des mesures de similarité sont utilisées pour récupérer les cas similaires à un cas cible. En général, avec les structures orientées-objet des cas, les mesures de similarité suivent le principe "local-global" (Bergmann & Stahl, 1998; Richter, 2008) qui dit : "le but est de déterminer la similarité entre deux objets, un objet représentant le cas source (ou une partie du cas) et un autre objet représentant le cas cible (ou une partie du cas). Cette similarité est appelée la similarité globale, et est calculée d'une manière récursive ; *i.e.* pour chaque attribut simple, une mesure de similarité *locale* détermine la similarité entre les deux valeurs d'attribut. En revanche, pour chaque attribut complexe, une mesure de similarité *globale* est utilisée. Enfin, les valeurs des similarités locales et globales sont agrégées, d'une manière récursive, pour donner la similarité globale des deux objets comparés".

D'un point de vue ontologique, le calcul de similarité entre deux concepts de l'ontologie peut être divisé en deux composantes (Bergmann & Stahl, 1998; Recio-Garcìa *et al.*, 2006) : *une similarité basée-concept* (ou similarité intra-classe) qui dépend de l'emplacement des concepts dans l'ontologie, et *une similarité basée-slot* (ou similarité inter-classe) qui dépend des valeurs des attributs communs des objets comparés.

4.2.1 Mesures de similarité

Les attributs d'un cas cible n'ont pas toujours la même importance dans le calcul de similarité. Ainsi, il est important de permettre à l'utilisateur d'associer à chaque attribut

un certain poids. Dans notre travail, les poids peuvent être attribués à deux niveaux différents du cas cible :

- Les attributs simples peuvent avoir l'un de trois modes de calcul de similarité :
  - IGNORE : l'attribut n'a pas d'importance.
  - EXACT : qui permet de vérifier l'égalité stricte des valeurs de l'attribut.
  - NUMÉRIQUE : qui est applicable aux attributs numériques seulement. Plus les deux valeurs sont proches l'une de l'autre, plus elles sont similaires.
- Les attributs complexes ont un poids dans l'intervalle [0, 1]. En outre, chaque attribut simple d'un attribut complexe peut avoir l'un des trois modes de calcul (IGNORE, EXACT, NUMÉRIQUE).

Nous allons expliquer, dans la suite, les mesures de similarité utilisées dans COBRA.

Soit $Q = \{q_i : 1 \leq i \leq n, n \in \mathbb{N}^*\}$ un cas cible pour lequel on cherche des cas similaires, où q_i est un attribut simple ou bien complexe, et soit $\Omega = \{C_j : 1 \leq j \leq k, k \in \mathbb{N}^*\}$ la base de cas, où $C_j = \{c_{jl} : 1 \leq l \leq m_j, m_j \in \mathbb{N}^*\}$. La similarité basée-concept, sim_{cpt}, est définie comme suit :

Pour chaque attribut complexe, $q \in Q$ et $c \in C$,

$$sim_{cpt}(q, c) = w_q * \frac{2 * prof(LCS(q, c))}{prof(q) + prof(c)} \tag{1}$$

où w_q est le poids associé à q, $prof$ est la profondeur d'un concept (ou d'une instance) dans l'ontologie, et LCS est le plus petit subsumant commun (Least Common Subsumer) de deux instances. Dans un cas particulier, quand q et c représentent la même instance, nous avons : $prof(LCS(q, c)) = prof(q)$.

La similarité basée-slot, sim_{slt}, est définie comme suit :

$$sim_{slt}(q, c) = \frac{\sum_{s \in CS} sim(q.s, c.s)}{|CS|} \tag{2}$$

où CS est l'ensemble des attributs simples en commun entre q et c (Common Slots), $|CS|$ est sa cardinalité, $q.s$ (ou $c.s$) représente l'attribut simple s de q (ou de c), et $sim(q.s, c.s)$ est la similarité entre ces deux attributs. Pour le moment, nous considérons seulement les deux premiers modes (IGNORE, EXACT), et donc $sim(q.s, c.s)$ est définie comme suit :

$$sim(q.s, c.s) = \begin{cases} 1 & \text{si} \quad (w_{q.s} = exact) \wedge (v_{q.s} = v_{c.s}) \\ 0 & \text{sinon} \end{cases}$$

où $w_{q.s}$ est le mode associé à l'attribut $q.s$, et $v_{q.s}$ est la valeur de cet attribut dans q.

La mesure globale de similarité entre les deux attributs complexes, q et c, est définie par la formule suivante (Zhang *et al.*, 2006) :

$$sim(q, c) = (1 - \alpha) * sim_{cpt}(q, c) + \alpha * sim_{slt}(q, c) \tag{3}$$

où α est un paramètre d'expérience (actuellement, $\alpha = 0.4$).

Pour calculer la similarité basée-concept, chaque attribut complexe du cas cible est comparé à son attribut correspondant d'un autre cas source. Dans les bases de cas *homogènes*, tous les cas partagent la même structure prédéfinie, et donc la correspondance entre les attributs complexes des cas est déjà définie. En revanche, dans les bases de cas *hétérogènes*, cette correspondance n'est pas préalablement définie. En conséquence, avant de calculer la similarité basée-concept, il faut déterminer les attributs complexes correspondants.

4.2.2 Déterminer les attributs correspondants

Pour chaque attribut complexe $q' \in Q$, soit $c' \in C_j$ l'attribut complexe correspondant dans le cas $C_j \in \Omega$. Donc,

$$sim_{cpt}(q', c') = \max_{1 \leq l \leq m_j} (sim_{cpt}(q', c_{jl})) \tag{4}$$

Nous constatons que la prise en compte de la similarité maximale par rapport à un seul cas n'est pas suffisante en tant que telle. Car il se peut que c' soit l'attribut le plus similaire à q' dans le cas C_j alors qu'en réalité q' n'a aucun attribut correspondant dans C_j. Pour cela, il faut également comparer cette similarité avec la similarité maximale que l'on peut obtenir sur l'ensemble des cas, ce qui conduit à la satisfaction de la condition suivante :

$$\frac{sim_{cpt}(q', c')}{\max_{1 \leq j \leq k, 1 \leq l \leq m_j}(sim_{cpt}(q', c_{jl}))} \geq \beta \tag{5}$$

où β est un certain seuil (actuellement, $\beta = 0.7$).

4.2.3 Exemple

Prenons, par exemple, la partie suivante d'une description d'un cas : "Sur un site industriel, un capteur de gaz *infrarouge* a été installé pour détecter le *méthane*. D'autres gaz étaient présents sur le site dont *l'ammoniac*. Le capteur n'a pas bien fonctionné et une explosion s'est produite". Supposons maintenant une requête cherchant les cas où un capteur de gaz (peu importe sa technologie) a été utilisé pour détecter *l'hydrogène*. Nous avons alors :
$C = \{$ capteur-infrarouge, méthane, ammoniac $\}$
$Q = \{$ capteur-de-gaz, hydrogène $\}$

Supposons que $w_{hydrogene} = 1$, afin d'identifier l'attribut correspondant à l'hydrogène (de la requête), nous trouvons (voir FIG. 3, Formules (1), (4)) :

$sim_{cpt}(\text{hydrogène}_Q, \text{capteur-infrarouge}_C) = 0$
$sim_{cpt}(\text{hydrogène}_Q, \text{méthane}_C) = (2 * 3)/8 = 0.75$
$sim_{cpt}(\text{hydrogène}_Q, \text{ammoniac}_C) = 0.5$

Donc, le *méthane* est l'attribut qui correspond mieux à *l'hydrogène* (En ne considérant qu'un seul cas).

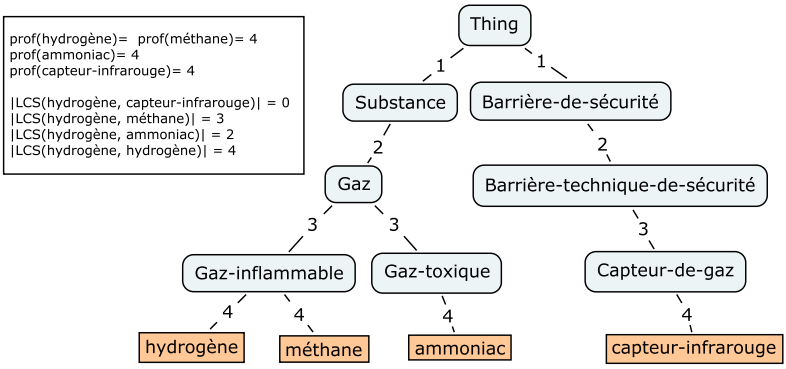

FIG. 3 – Partie du modèle de domaine.

4.2.4 Notion de rôle

Bien que la solution proposée pour déterminer la correspondance entre les attributs complexes soit efficace, elle donne parfois des résultats ambigus quand plusieurs attributs candidats sont proposés. Modifions l'exemple décrit précédemment : "l'un des gaz présents sur le site, en plus de l'ammoniac, était *l'hydrogène*" ; *i.e.* $C =\{$ capteur-infrarouge, méthane, ammoniac, hydrogène $\}$.

En suivant la méthode précédente, nous trouvons que l'hydrogène du cas C est l'attribut correspondant à l'hydrogène de Q. Toutefois, ce n'est pas le résultat souhaité puisque nous cherchons les cas où l'hydrogène était le gaz à détecter et non pas un gaz présent sur le site.

Pour lever cette ambiguïté, nous proposons de rajouter la notion de rôle ; *c'est-à-dire*, pour chaque attribut complexe pouvant jouer différents rôles dans un cas, son rôle doit être précisé dans chaque cas. En effet, le rôle représente dans l'ontologie une relation (de type objet) entre le cas et son attribut (dans cet exemple, $C\rightarrow$ *gaz-à-détecter*$\rightarrow$ méthane).

Pour trouver la correspondance entre les attributs complexes, nous combinons les deux approches : tout d'abord, les attributs ayant le même rôle sont identifiés ; ensuite, on applique la méthode proposée dans la section 4.2.2 aux autres attributs.

5 Plate-forme COBRA

COBRA est une plate-forme de RàPC indépendante du domaine. Elle permet la construction de systèmes de RàPC dont les connaissances sont décrites par des ontologies. La plate-forme est développée en JAVA en tant qu'application basée sur Eclipse[1], grâce au framework RCP[2] (Rich Client Platform). Elle profite ainsi de beaucoup de fonctionnalités offertes par Eclipse.

[1] http ://www.eclipse.org/

[2] http ://wiki.eclipse.org/index.php/Rich_Client_Platform

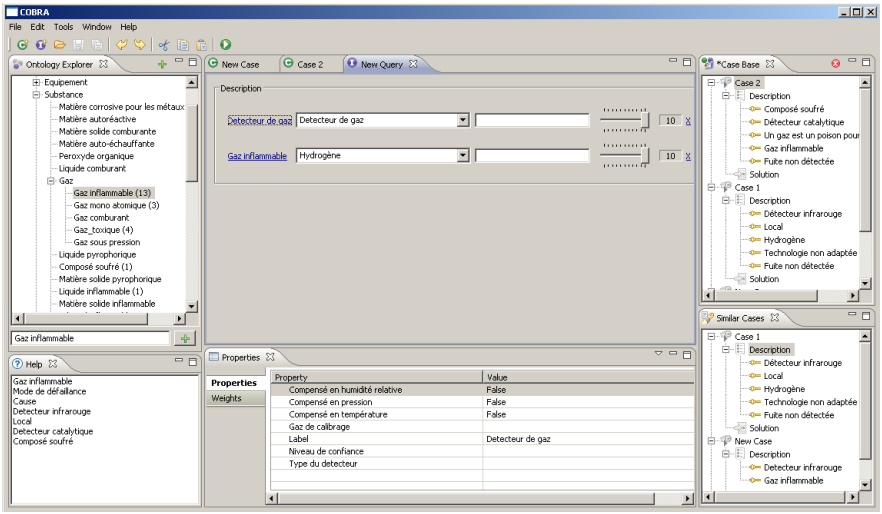

FIG. 4 – La plate-forme COBRA.

La plate-forme contient les onglets (viewers) principaux suivants (FIG. 4) : l'explorateur d'ontologie (en haut à gauche) qui montre les concepts de l'ontologie de domaine, la base de cas (en haut à droite), l'onglet des cas similaires par rapport à un cas cible, l'onglet d'aide qui propose à l'utilisateur les concepts candidats et les plus utilisés dans la base de cas, et l'onglet des propriétés permettant de modifier les valeurs de l'instance sélectionnée (d'un cas ou d'un attribut de cas).

L'authoring des cas se fait en créant un nouveau cas, puis en renseignant ses attributs simples et/ou en y ajoutant des attributs complexes (instances de l'ontologie). L'utilisateur peut choisir parmi les instances présentes, ou il peut créer et ajouter ses propres instances, ce qui permet d'enrichir la base de connaissances. Ensuite, l'utilisateur peut associer (ou créer) des rôles aux attributs complexes si nécessaire.

Comme le montre la figure 5, la plate-forme est basée sur jCOLIBRI, et nous avons étendu cette API par une couche qui permet le traitement de bases de cas dynamiques et hétérogènes. Cette couche contient notre propre connecteur d'ontologie ainsi que le module de calcul de similarité. Grâce à cette architecture, le développement d'un nouveau système de RàPC se fait en fournissant l'ontologie de domaine, et en paramétrant des fichiers XML de configuration.

6 Conclusion et perspectives

Nous avons présenté dans cet article notre plate-forme COBRA, une plate-forme de KI-CBR indépendante du domaine. Nous travaillons actuellement sur le domaine des barrières de sécurité, en particulier les capteurs de gaz, installées sur des sites industriels. Dans ce contexte, COBRA sera utilisé pour répondre à deux objectifs :

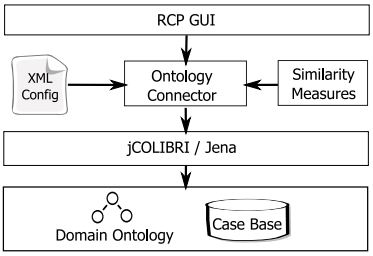

FIG. 5 – Les composantes de COBRA.

– Capitaliser les connaissances autour de la défaillance des capteurs de gaz.
– Fournir une aide aux experts et aux ingénieurs de sécurité pour pouvoir diagnostiquer les causes de défaillance des capteurs dans des conditions industrielles.

COBRA permet aux cas d'avoir différentes structures, et donc de travailler avec des bases de cas hétérogènes. Pour développer COBRA, nous nous sommes appuyés sur l'API jCOLIBRI. Cette API ne permet pas le traitement de bases de cas dynamiques et hétérogènes, car les attributs de cas doivent être définis auparavant (dans le code source) avant de lancer la plate-forme. Nous avons donc gardé les aspects intéressants de jCOLIBRI, et ajouté une couche supplémentaire afin de pallier ce manque.

L'hétérogénéité des cas complique la détermination de la correspondance entre les attributs complexes (d'un cas cible et d'autres cas sources). Dans cet article, nous avons proposé deux approches complémentaires pour résoudre ce problème basées sur les calculs de similarité et sur la définition des rôles des attributs dans leurs cas.

Actuellement, nous sommes en train de développer les autres processus de RàPC et d'étudier le lien qu'il peuvent avoir avec l'ontologie de domaine. Parallèlement, nous rajoutons de nouveaux cas (sur la défaillance de capteurs de gaz) à la base de cas. Nous allons également évaluer ce travail auprès d'experts de l'INERIS à deux niveaux :

– Le premier niveau concerne l'architecture proposée de RàPC, par exemple : à quel point la structure des cas et les processus de raisonnement sont-ils proches de l'activité réelle de l'expert ? Quels sont les concepts à rajouter à l'ontologie de domaine pour pouvoir décrire les nouveaux cas ? L'expert trouve-t-il les propositions d'aide intéressantes ?
– Le deuxième niveau concerne les solutions données par le système. Après l'ajout d'un certain nombre de cas, les experts doivent vérifier la qualité du diagnostic proposé par le système par rapport à certains cas cibles.

Références

AAMODT A. (1994). Explanation-Driven Case-Based Reasoning. *Lecture Notes In Computer Science*, p. 274–274.

AAMODT A. & PLAZA E. (1994). Case-Based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, **7**(1), 39–59.

ABOU ASSALI A., LENNE D. & DEBRAY B. (2008). Ontology development for industrial risk analysis. In *IEEE International Conference on Information & Communication Technologies : from Theory to Applications (ICTTA'08)*, Damascus, Syria.

BERGMANN R. & STAHL A. (1998). Similarity measures for object-oriented case representations. In *Proceedings of the European Workshop on Case-Based Reasoning, EWCBR'98*.

D'AQUIN M., LIEBER J. & NAPOLI A. (2006). *Artificial Intelligence : Methodology, Systems, and Applications*, volume Volume 4183/2006 of *Lecture Notes in Computer Science*, chapter Case-Based Reasoning Within Semantic Web Technologies. Springer Berlin / Heidelberg.

DÌAZ-AGUDO B. & GONZÀLEZ-CALERO P. (2000). An architecture for knowledge intensive CBR systems. *Advances in Case-Based Reasoning–(EWCBR'00). Springer-Verlag, Berlin Heidelberg New York*.

DÌAZ-AGUDO B. & GONZÀLEZ-CALERO P. (2002). CBROnto : a task/method ontology for CBR. *Procs. of the 15th International FLAIRS*, **2**, 101–106.

DÌAZ-AGUDO B., GONZÀLEZ-CALERO P., RECIO-GARCÌA J. & SÀNCHEZ-RUIZ-GRANADOS A. (2007). Building CBR systems with jcolibri. *Science of Computer Programming*, **69**(1-3), 68–75. Special issue on Experimental Software and Toolkits.

FUCHS B. & MILLE A. (2005). Une modélisation au niveau connaissance du raisonnement à partir de cas. In L'HARMATTAN, Ed., *Ingénierie des connaissances*.

LAMONTAGNE L. & LAPALME G. (2002). Raisonnement à base de cas textuels : Etat de l'art et perspectives. *Revue d'intelligence artificielle*, **16**(3), 339–366.

RECIO-GARCÌA J., DÌAZ-AGUDO B., GONZÀLEZ-CALERO P. & SANCHEZ A. (2006). Ontology based CBR with jCOLIBRI. *Applications and Innovations in Intelligent Systems*, **14**, 149–162.

RENAUD J., MORELLO B., FUCHS B. & LIEBER J. (2007). Raisonnement à Partir de Cas 1 : conception et configuration de produits. *Hermes-Lavoisier, February*, **1**.

RICHTER M. (2008). *Case-Based Reasoning on Images and Signals*, volume 73/2008 of *Studies in Computational Intelligence*, chapter Similarity, p. 25–90. Springer Berlin / Heidelberg.

RIESBECK C. & SCHANK R. (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates.

YANG C., FARLEY B. & ORCHARD B. (2008). Automated Case Creation and Management for Diagnostic CBR Systems. *Applied Intelligence : The International Journal of Artificial Intelligence, Neural Networks and Complex Problem-Solving Technologies*, **28**(1), 17–28.

ZHANG K., TANG J., HONG M., LI J. & WEI W. (2006). Weighted Ontology-Based Search Exploiting Semantic Similarity. *Lecture Notes In Computer Science*, **3841**, 498.