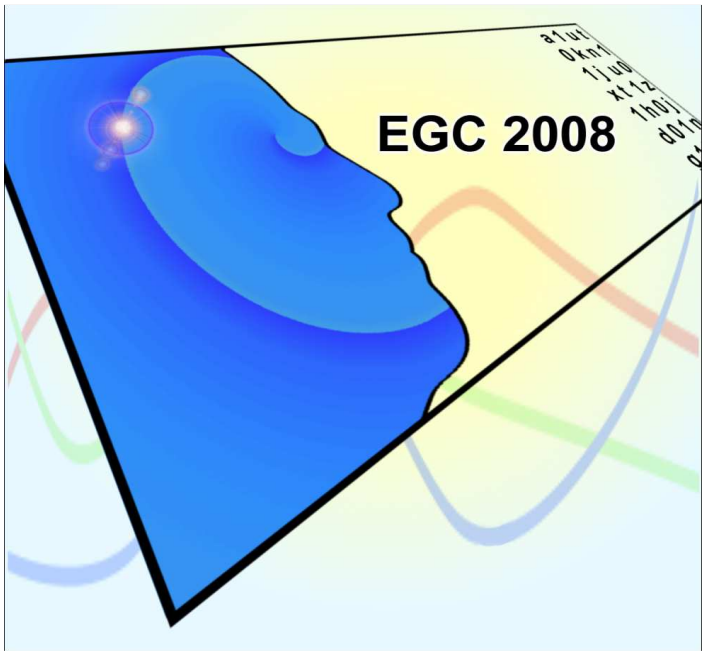
8èmes Journées Francophones
Extraction et Gestion des Connaissances
Sophia Antipolis 29 janvier 2008

Atelier



Modélisation des connaissances

Organisateurs :

- Henri Briand (Univ. de Nantes)
- Fabien Gandon (INRIA, Sophia Antipolis)
- Fabien Picarougne (Univ. de Nantes)

Responsables des Ateliers EGC :

Alzennyr Da Silva (INRIA, Rocquencourt)
Alice Marascu (INRIA, Sophia Antipolis)
Florent Masseglia (INRIA, Sophia Antipolis)

http://www-sop.inria.fr/axis/egc08

EGC

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

Atelier Modélisation des Connaissances

EGC'2008

INRIA Sophia Antipolis, 29 janvier 2008

Gestion sémantique de contenu d'entreprise collaboratif orienté métiers : un retour d'expérience sur le développement d'un prototype dédié au domaine « Qualité, Hygiène, Sécurité et Environnement »

Christophe Thovex*, Francky Trichet**
* Aker Yards France – avenue Bourdelle
44600 – SAINT NAZAIRE
christophe.thovex@akeryards.fr / cthovex@free.fr

** LINA – Laboratoire d'Informatique de Nantes Atlantique (FRE-CNRS 2729)
Equipe Connaissance et Décisions(COD) – Université de Nantes
Polytech'Nantes - La Chantrerie - rue Christian Pauc BP 50609
francky.trichet@univ-nantes.fr

Résumé : Avec l'ouverture des formats documentaires bureautiques tels que ODF ou Open XML, la gestion de contenu textuel d'entreprise évolue de plus en plus vers une problématique de gestion de connaissances. L'intégration aux Systèmes d'Information industriels de contenus semi-structurés transforme les documents d'entreprises en base de données interopérables. Basés sur XML, les nouveaux formats de données bureautiques s'ouvrent au traitement sémantique de l'information. Pour l'industrie, l'enjeu consiste à modeler une architecture logicielle incorporant l'intelligence métiers du contenu documentaire à la gestion des connaissances. Les développements opérationnels sont encore peu nombreux à intégrer les composants du poste utilisateur final, tel que communément rencontrés. Si quelques secteurs se sont dotés de systèmes dédiés à la gestion sémantique de connaissances (*e.g.* presse ou hautes technologies industrielles), le plus souvent ces systèmes restent encore isolés au contexte bureautique le plus répandu, à savoir Open Office, Star Office et MS Office. *A fortiori*, l'interopérabilité avec les puits d'information structurée présents dans les SI reste très limitée. C'est pourtant là un important levier de rationalisation et d'amélioration de la valeur productive des SI. En développant l'interopérabilité, c'est le capital humain qui est appelé à se développer, par la facilité à collaborer. Le domaine « Qualité, Hygiène, Sécurité et Environnement » (QHSE) représente à l'heure actuelle un vecteur de progrès majeur pour l'industrie européenne. L'atelier prototype baptisé « *Quality Semantic Environment* » (QSE) permet de dresser un retour d'expérience sur le processus d'intégration d'une dimension sémantique pour la gestion de connaissances métiers QHSE, dans un système de gestion de contenu d'entreprise banalisé, dédié à l'industrie navale.

Mots clés : gestion sémantique de contenu d'entreprise, ontologie multilingue et cross-lingues, apprentissage, navigation dans les connaissances, OpenXML, SKOS, SPARQL.

1 Introduction

Aker Yards SA est une industrie navale française du groupe scandinave Aker Yards. Elle regroupe de nombreux corps de métiers, utilisateurs multilingues du système d'information. Le besoin en outils logiciels collaboratifs y est important et dans une architecture complexe où le poste utilisateur final forme un tronc commun applicatif, la gestion de contenu bureautique d'entreprise est un élément fort, basé sur les fonctions du protocole WebDAV.

Un système de gestion de contenu d'entreprise devient collaboratif dès lors qu'il permet à des groupes de travail de collaborer en partageant l'information autour d'un thème (activité, projet). Le système offre pour cela des fonctions de gestion documentaire (cf. WebDav), recherche d'information et de communication (forums, wikis, pages web modifiables). Dans le cas présent, l'environnement propose également des *servlets* réutilisables et des processus collaboratifs (*workflows*) applicables au divers types de contenu - documents et formulaires. Rationnellement, la solution intégrée Microsoft® Office SharePoint Services 2007 - MOSS 2007[1] a été retenue comme socle technologique, pour la capacité productive immédiate qu'offre son intégration avec les autres composants du poste utilisateur. Environnement collaboratif, MOSS 2007 exploite Open XML, format documentaire ouvert observant les recommandations W3C XML, XSD et XSLT. Approuvé par l'ECMA pour présentation à l'ISO, Open XML est reconnu trop complexe pour définir une norme comme Open Document Format (ODF – ISO28300-2006). Il pourrait s'imposer en standard de fait.

SharePoint Server (SPS) 2007, serveur de contenu d'entreprise, implémente des mécanismes avancés de contrôle d'accès, check-in/check-out, versioning, audits (trace) et *workflows* pour la gestion électronique documentaire en réseau (mode Web). Il intègre et des services multilingues d'indexation et recherche traitant les ressources, internes ou externes au système, au niveau syntaxique. Il ne traite pas l'information bureautique (données non structurées, semi-structurées et structurées) au niveau sémantique.

Ceci lève une problématique de *fracture*, entre le besoin pragmatique des utilisateurs « métiers » et les fonctions syntaxiques de l'environnement collaboratif. La dimension sémantique, chainon abstrait manquant entre l'utilisateur et l'information électronique est un enjeu global pour l'entreprise. L'introduire, via MOSS, dans le SI favorise la collaboration entre corps de métiers et nationalités. On fournit un retour d'expérience sur la collaboration cross-lingue autour de connaissances métiers et l'apprentissage assisté de leur vocabulaire.

Le projet est agile (*Dynamic Systems Development Method*), et développé en mode itératif. Sur le thème « Qualité, Hygiène, Sécurité, Environnement » (QHSE), un prototype opérationnel sémantique, baptisé S-QHSE, a été développé à partir d'une application à base de connaissances existante : « QHSE ». L'application regroupe environ 22.000 fichiers, 14.000 dossiers, soit 4.7 Go. Les technologies J2EE, VB .NET et C# permettent l'extraction du contenu, sa conversion en Open XML puis la mise en œuvre d'un composant Web Part ASP .NET dédié à l'interrogation sémantique. Le Web Part exploite une ontologie correspondant, suivant la définition consensuelle de T. Gruber [Gruber, 1993], à une « *spécification formelle et explicite d'une conceptualisation partagée* ». Les ontologies, au cœur de nombreuses applications dédiées à la manipulation informatisée des connaissances, permettent d'une part, de partager un vocabulaire conceptuel commun (par exemple orienté

[1] Microsoft Office SharePoint Server 2007 est l'outil de portail et de travail collaboratif de Microsoft. Basé sur le Framework .NET, il sert à gérer et présenter le contenu documentaire d'entreprise.

métier) et d'autre part, de mener différents types de raisonnement sur des assertions des domaines de connaissances modélisés.

Parmi les langages sémantiques W3C, *Simple Knowledge Organization System* - SKOS a été retenu pour son adéquation avec la problématique. Son niveau d'abstraction intermédiaire entre RDF-RDFS et OWL facilite l'accès aux taxonomies et ontologies légères. Les recommandations SKOS mettent en avant le traitement multilingue de concepts organisés. SKOS autorise l'administration par l'entreprise elle-même, selon les besoins qu'elle éprouve, de la sémantique métiers présente dans son SI.

Pour le prototypage, le contenu d'information a été intégralement extrait et converti afin de calibrer un jeu d'essais restreint et pertinent. Certains documents Open XML ont été chargés au sein d'une bibliothèque SPS adaptée. Les métadonnées associées (base de données) à ces documents ont été embarquées (fusion). Une ontologie d'application a été élaborée via une approche ad-hoc, puis enrichie d'un fragment d'ontologie de domaine. Un composant Web Part implémente un modèle simple d'interrogation sémantique basé sur une ontologie, depuis un client Web (http). Par un lot d'exemples satisfaits, ce prototype démontre la validité d'une approche de recherche sémantique d'informations cross-lingues appliquées au domaine de l'industrie navale, et spécifiquement au domaine « Qualité, Hygiène, Sécurité, Environnement ». La recherche d'information s'avère d'une importance cruciale étant donné le caractère sensible de la construction navale, l'abondance et la fréquence d'apparition des réglementations internationales relatives à ce domaine.

Pour l'industrie navale européenne, la sémantique est nécessaire à l'organisation productive de référentiels métiers cross-lingues. Le prototype opérationnel S-QHSE ouvre des perspectives sur le domaine de la recherche collaborative aux SI collaboratifs d'entreprise. Ce travail peut symboliser les prémisses d'un projet ambitieux visant à fédérer les cinq grands constructeurs européens autour du développement d'une plateforme commune, pour la gestion sémantique de documents et connaissances propres à l'industrie navale.

La suite de cet article est structurée comme suit. La section 2 présente le modèle ontologique initial (sous forme de bribes) implémenté en SKOS[2], puis expose brièvement comment ce modèle a évolué pour satisfaire un exemple démonstratif. La section 3 relate la conception du prototype opérationnel intégrant l'exploitation du modèle ontologique dans l'environnement collaboratif. Enfin, la section 4 introduit quelques scénarios de recherche d'informations permettant d'éclairer l'aspect valorisant du composant en termes de collaboration élargie (multi métiers et multilingue) pour l'industrie navale.

2 Modèle ontologique et SKOS

Le corpus identifié pour l'étape de construction de l'ontologie est constitué de documents textuels (principalement des fichiers .doc et .pdf) et classeurs (principalement des fichiers .xls), organisés en hiérarchie de dossiers (3 niveaux maximum). Ce corpus découle de l'application « QSE », associant des documents bureautiques avec des métadonnées contenues dans une base de données. Ces métadonnées sont statiques (toujours les mêmes) et prennent des valeurs finies (un ensemble de valeurs par métadonnées). A un document sont associées toutes les métadonnées. Elles constituent un treillis de concepts (relations

[2] http://www.w3.org/TR/swbp-skos-core-spec/

"horizontales" et "verticales") spécifiques à l'application ou plus génériques, inhérents à l'industrie navale, distribués en deux ensembles, qualifiant les connaissances contenues dans les documents.

L'analyse des métadonnées permet d'en construire une liste, non-exhaustive dans le prototype, mais qui pourrait le devenir dans une version livrable, implémentée en SKOS et respectant les relations unissant les concepts. Deux ontologies légères [Gomez-Perez & al. 2004], *i.e.* des ontologies composées uniquement de hiérarchies de concepts et dénuées de toutes relations et axiomes du domaine, sont ainsi connectées au sein d'un même fichier de définition SKOS.

2.1 Ontologies initiales et représentation (SKOS)

Pour satisfaire le prototype, deux ontologies légères ont été identifiées. La première regroupe les concepts strictement propres à l'application. C'est une ontologie dite d'application [Staab et Studer 2004]. La seconde regroupe des concepts présents, mais non spécifiques à l'application. C'est, conceptuellement, l'intersection de l'ontologie d'application avec une potentielle ontologie de domaine pour l'industrie navale, encore indéfinie. Cette bribe d'ontologie de domaine représente un sous-domaine.

Pour isoler deux ontologies, SKOS utilise la notion de *ConceptScheme*. Un concept SKOS est associé un *ConceptScheme*, et peut aussi être qualifié de *TopLevelConcept* (concept racine, n'ayant aucun concept parent dans le *ConceptScheme*).

De la sorte, deux *ConceptScheme* ont été définis, respectivement et initialement nommés SQHE et QSE (deux bribes ontologiques).

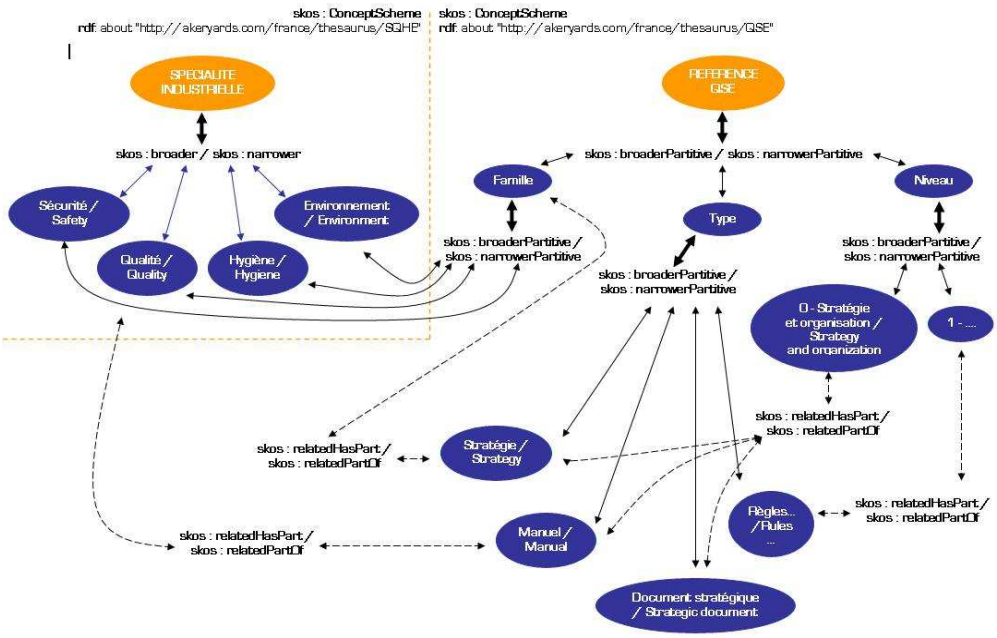


Figure 1 : Exemple de graphe sémantique des métadonnées SQHE/QSE

La figure 1 présente ces deux *ConceptScheme.* Différents aspects conceptuels et techniques sont à remarquer : (1) le rattachement des concepts à un espace de noms (*ConceptScheme/InScheme*), (2) les "top-level concepts" (précisés en couleur en orange - skos:HasTopConcept), (3) les propriétés de la classe *skos:Concept (prefLabel* : nom du concept ; *definition* : résumé du concept ; *scopeNote* : portée du concept - utilisation avec présence de mots clés ; étiquettes de langue au standard W3C (@en, @fr,…), (4) les relations entre concepts : *broader/narrower* : hiérarchie large (SKOS Core) ; *broaderPartitive/narrowerPartitive* : hiérarchie stricte (SKOS Extensions) ; *relatedHasPart/relatedPartOf* : association de concepts (SKOS Extensions). D'autres propriétés sont notamment dédiées à l'administration (suivi d'évolutions).

Notons que pour être valide, une relation entre deux concepts implique la relation inverse (A broader B => B narrower A), hormis pour la relation *skos:HasTopConcept*, permettant d'éviter les références cycliques : (A => B => C => A). Grâce aux *ConceptScheme*, les concepts <Sécurité>, <Qualité>, <Hygiène> et <Environnement> héritent simultanément de <SPECIALITE INDUSTRIELLE> et de <Famille>.

Cet exemple, sans être exhaustif, confirme l'adéquation de SKOS pour la représentation des ontologies légères d'application, le plus souvent liées à un ou plusieurs domaines. Les principales possibilités de modélisation en SKOS suffisent à traiter un grand nombre de cas d'utilisation. Pour traiter l'application « QHSE », SKOS offre le vocabulaire nécessaire et suffisant. L'efficacité opérationnelle de l'ontologie dépend, en général, du pragmatisme de sa conception, en adéquation à un usage concret (profondeur, simplicité, richesse). Pour concevoir une ontologie métiers, on peut recommander d'en connaître les cas d'utilisation.

2.2 Evolutivité et démonstration par l'exemple

Afin de valider l'intérêt industriel du prototype, on s'impose un exemple démonstratif :

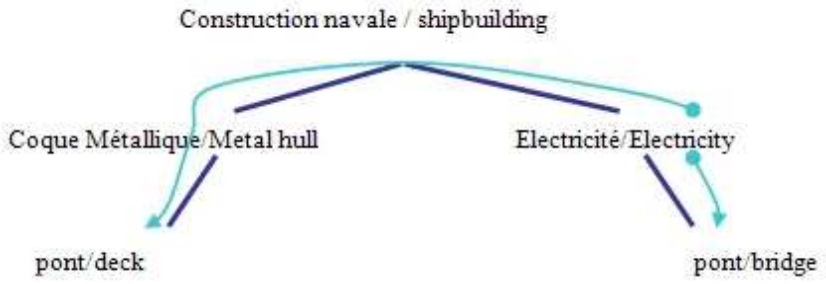


Figure 2 : Exemple démonstratif de hiérarchie de concepts

Dans l'exemple de hiérarchie QSE présenté en Figure 1, l'extrait de l'ontologie d'application comporte des concepts génériques du domaine industriel. Ce sont les éléments du *ConceptScheme* SQHE (en haut à gauche de la figure 1).

Pour satisfaire l'exemple démonstratif (Figure 2), il est obligatoire d'enrichir le sous–domaine initial où les concepts composant l'exemple sont absents. En implémentant l'exemple donné, la bribe d'ontologie de domaine (construction navale) s'élargit avec le sous-domaine qu'elle englobe.

Pour passer du sous-domaine initial (ci-dessus) au sous-domaine suffisant pour démontrer le prototype par l'exemple, *i.e.* respecter l'engagement ontologique minimal [Bachimont, 2004], différents choix de modélisation se sont avérés nécessaires à prendre en compte (cf. figure 3) :

A – Introduction d'un nouveau TopLevelConcept (maître), permettant l'appariement éventuel du sous-domaine au domaine générique de l'ontologie (« clé étrangère ») ;

B – Introduction d'un concept intermédiaire pour regrouper les quatre concepts Qualité, Hygiène, Sécurité, environnement ;

C – Intégration des concepts manquants à la nouvelle ontologie ;

D – Le *ConceptScheme* de la nouvelle ontologie est renommé « INDNAV ».

Ces opérations ne sont pas sans rappeler les principes de modélisation objet (*i.e.* UML). Une similitude avec l'administration de modèles relationnels est aussi décelable (NDLA : normalisation, optimisation). Ceci invite à réfléchir sur l'existence d'un méta-méta-modèle [Pitrat, 1990] réutilisable pour unifier les trois ontologies potentielles relatives à (1) l'application « QHSE », (2) le domaine QHSE et (3) le domaine de l'industrie navale.

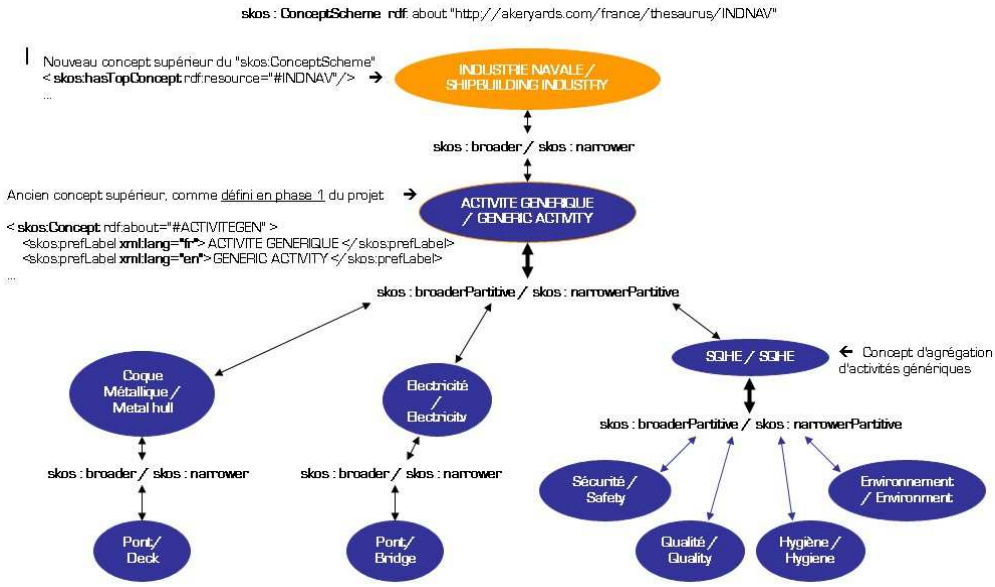


Figure 3 :Ontologie de sous-domaine appariée au domaine et à l'application

3 Architecture et prototype opérationnel S-QHSE

Au sein de l'environnement MOSS 2007, *SharePoint Server 2007* est doté d'un système d'indexation-recherche syntaxique. Il peut étendre la recherche aux ressources de l'entreprise. Après conversion au format Open XML, les documents peuvent embarquer les métadonnées.

Le système peut indexer les serveurs Web (application J2EE, Intranet, Extranet), de messagerie (Exchange et Lotus Domino), les serveurs de fichiers (disques réseau) et sources de données externes (bases relationnelles, annuaires). Selon la configuration, l'occupation des ressources matérielles peut toutefois atteindre un seuil critique.

Le moteur de recherche SPS 2007 s'oppose à *Google Search Enterprise*. Il favorise la confidentialité des informations internes, mais en comparaison, la pertinence de ses résultats semble moindre, faute d'exploiter le potentiel offert par Open XML. Le système SPS de recherche dans les textes reconnaît les mots de plusieurs langues en corrigeant l'orthographe quand le taux d'erreur est faible. Il ne traite pas de synonymes ou de vocables de sens proche des mots clés de recherche. A fortiori, il ne traduit pas en fonction du contexte, comme dans l'exemple démonstratif présenté en figure 2.

SPS 2007 est assimilable à un serveur WebDav applicatif (accès concurrents, versions, audits, *workflows* et ASP .NET). Il s'adosse à une base de données relationnelle et offre une interface graphique versatile, fortement paramétrable. Une « bibliothèque » SharePoint est ainsi rapidement élaborée au sein d'un site spécifique « QHSE » (pages dynamiques – ASP.NET), prêt à recevoir documents et métadonnées.

Les services intégrés d'indexation/recherche syntaxique traitent le contenu présent dans cette bibliothèque. Un composant applicatif standard est présent sur les pages du site, permettant de lancer une recherche à partir de mots-clés saisis (*Web Part de recherche*). La démarche consiste à produire un système respectant cette architecture et ajoutant des capacités de recherche sémantique sur le contenu de la bibliothèque.

Bien que d'une expressivité moindre, SKOS offre une facilité de mise en œuvre supérieure à OWL, tout en permettant l'alignement de concepts (exact ou flou) d'une ontologie à l'autre. Après analyse, le triplet SKOS-SPARQL-ASP.NET a été adopté comme socle de base pour définir l'architecture adéquate.

Une définition pour QHSE fut initiée avec *ThManager v2.0*, gestionnaire graphique de thesaurus SKOS intéressant, puis affinée en mode texte dans *NotePad2* (gratuiciel) pour pallier les limitations fonctionnelles de l'outil spécifique. L'outil *SkosValidator* v1.02 (jena) a été utilisé pour éprouver la validité de l'ontologie. Par itérations successives (création-validation-interrogation), un modèle révélateur satisfaisant a été obtenu.

SPARQL[3] a été exploité dans le cycle de développement itératif pour estimer la pertinence de l'ontologie (engagement minimal, complétude, rigidité). Pour intégrer le couple SKOS-SPARQL dans l'architecture cible, l'API SPARQL v1.0 (06/2007) de J. Tauberer (librairie .NET) a été utilisée. A partir de ce composant, un Web Part a été développé en C#, utilisant l'API SPARQL et les API MOSS 2007. Facultativement, *SmartPart* v1.2, un Web Part encapsulant du code ASP .NET. a été déployé afin d'obtenir une architecture .NET comparable à Java : machine virtuelle et code précompilé (*Common Language Runtime* - CLR).D'après A. Polleres [Polleres 2006], les possibilités de SPARQL vont jusqu'à la construction à la volée de systèmes de règles volatiles.

La fonction sémantique a été finalement conçue en amont de la recherche syntaxique indexée, intégrée à l'environnement qu'elle interface:

1. Un composant WebPart .NET reçoit les mots clés de l'utilisateur dans SPS 2007 ;
2. Il extrait de l'ontologie SKOS, les concepts clés en correspondance sémantique ;
3. Il fournit la liste de mots clés affinée au système de recherche standard ;
4. La recherche syntaxique standard présente les résultats à l'utilisateur.

[3] http://www.w3.org/TR/rdf-sparql-query/

3.1 Conception générale

Ce système doit favoriser les synergies collaboratives entre des ressources humaines hétérogènes. En guidant l'utilisateur dans les connaissances spécifiques aux métiers, le composant aide l'utilisateur novice à trouver les documents appréhendés et soulage l'effort d'experts utilisant le système d'information.

L'architecture du prototype opérationnel sémantique peut être schématisée par un diagramme de déploiement UML (cf. Figure 4). Le déploiement des composants montre comment, sans remettre en cause l'environnement global et en optimisant le coût de développement (contraintes industrielles), il a été aisé d'ajouter une dimension sémantique au service de recherche (indexation syntaxique) initial.

Un Web Part "Recherche sémantique", aux pages serveurs du site, est présenté à l'utilisateur comme alternative de recherche. Le contrôle utilisateur dans lequel ce composant retourne les résultats d'interrogation de l'ontologie (SPARQL) est dynamiquement lié au contrôle utilisateur du Web Part "recherche" standard (indexation syntaxique). L'ontologie SKOS est stockée sur l'environnement SPS 2007, dans un espace réservé ou directement dans la bibliothèque de l'application QHSE (autorisations restreintes).

Si l'utilisateur saisit ses mots clés dans le contrôle associé du composant "Recherche", le lot de résultats (pointeurs de ressources) est syntaxiquement lié à la saisie. Dans l'alternative, si les mots clés sont saisis dans le composant "Recherche sémantique", celui-ci interroge l'ontologie, puis retourne les mots saisis complétés de concepts sémantiquement liés (vocables multilingues) dans les contrôles "mots clés de recherche" (cf. zones bleu marine de la figure 4). L'utilisateur bénéficie d'une *assistance sémantique à la recherche syntaxique*.

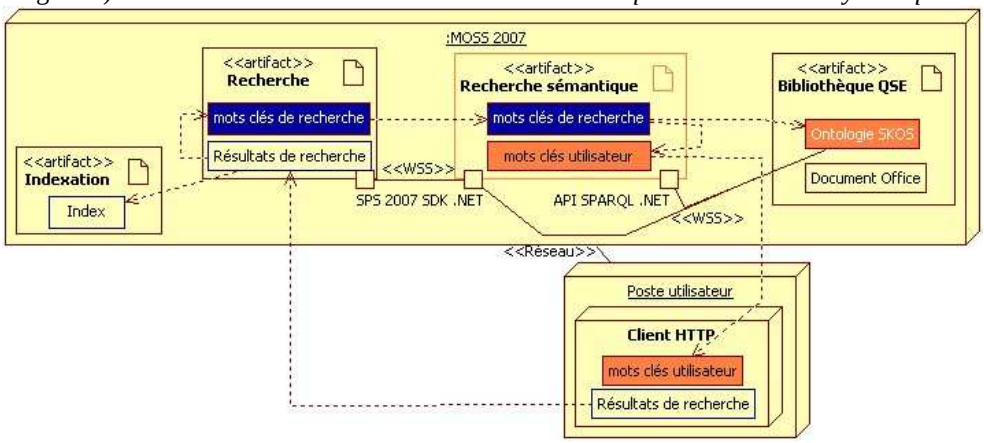


Figure 4 : Diagramme de déploiement de la recherche sémantique QSE

Dans le cas où un seul mot clé est fourni par l'utilisateur, l'interrogation sémantique retourne les correspondances multilingues qu'elle trouve dans les propriétés *skos:definition* des concepts. L'utilisateur n'ayant fourni qu'un mot clé, il se voit ainsi proposer une série d'expressions multilingues lui permettant de cibler judicieusement une nouvelle recherche, plus précise, pour peu qu'il connaisse au moins l'une des langues proposées. Etant donné que chaque nouvelle recherche retourne de nouvelles informations sémantiquement consistantes, l'utilisateur peut progresser en pertinence dans sa saisie, par une forme d'apprentissage pseudo-automatique.

3.2 Conception détaillée

L'application industrielle finale, qui sera développée à partir du prototype courant, devra déployer un lot de requêtes types, rationnalisées de façon à naviguer dans les concordances sémantiques les plus courantes. Dans sa version actuelle, le prototype implémente les notions de généralisation/spécialisation, d'association simple et de subsomption sur deux générations. Il gère l'héritage multiple mais n'exploite pas les associations composées. Pour en faire un produit livrable, il est nécessaire de développer et d'ajuster ses capacités d'interrogation. La gestion de cardinalités basiques pourra être expérimentée, le code actuel ne distinguant pas les relations larges ou strictes SKOS (*skos:xxx, skos:xxxpartitive*). Ceci autoriserait le choix de critères "tous" ou "certains" en regard des mots clés saisis. Le nombre de champs de saisie des mots clés pourra aussi être augmenté, si cela s'avère pragmatiquement intéressant.

Outre le développement du Web Part, on anticipera les méthodes d'administration d'ontologies pour accéder au niveau de traitement sémantique industriel de l'information.

SKOS semble suffisamment élaboré pour industrialiser le prototype. Les dernières recommandations W3C pour SPARQL (RFC-12/11/2007) intègrent le traitement d'expressions régulières, fonctions externes et données géographiques (SIG). L'adéquation au besoin guide le modèle.

3.3 Codage et intégration du Web Part d'interrogation sémantique

Le composant d'interrogation sémantique actuel prend en entrée les critères de recherche de l'utilisateur (mots-clés), extrait de l'ontologie leurs correspondances pertinentes (concepts-clés) et déclenche une recherche syntaxique sur l'ensemble des vocables obtenus (mots utilisateur si aucune correspondance n'est trouvée). Il peut laisser le choix d'employer ou non les concepts-clés éventuellement proposés, pour affiner la recherche.

L'interface étant intégrée à l'application dans SPS 2007, il a été décidé d'implémenter un Web Part .NET avec Visual Studio .NET. Sous l'environnement serveur (SPS 2007, WSS 3.0), les composants suivants ont été utilisés :*Visual Studio 2006 extensions for WSS 3.0*, *WSS Software Development Kit* (WSS SDK), *API SPARQL .NET*, *Smart Part v1.2*. *Smart Part* est un Web Part conteneur de contrôles utilisateurs. Il facilite le développement de Web Parts SPS 2007 en permettant d'intégrer dans une page ASP des contrôles utilisateurs, développés en mode graphique avec Visual Studio .NET. Pour l'heure, l'environnement de développement intégré de Microsoft® n'offre aucun module de programmation graphique des Web Parts. *Smart Part*, développé et diffusé gratuitement, offre une alternative à ce manque.

Le développement itératif permet de constater au plus tôt les limitations des ressources techniques. Les librairies de liens dynamiques (.dll) doivent impérativement comporter un *nom fort* pour être exécutées sur SPS 2007 (serveur), l'API SPARQL devra donc être recompilée pour s'intégrer au serveur. Ceci ne gêne pas le développement du composant et sa validation en mode web en dehors de SPS (serveur IIS).

Une autre limitation a été éprouvée avec l'API SPARQL, qui n'implémente pas encore toute la recommandation. Le signe '@', délimiteur d'étiquette de langage normalisé XML, accepté par le processeur SPARQL est refusé une fois la requête intégrée en C# (i.e. skos:prefLabel "QUALITE"@fr). De ce fait, les interrogations ont été codées en algorithmes dans le prototype opérationnel QSE. L'API SPARQL propose des classes et méthodes

permettant de développer rapidement, néanmoins par ces manques, la charge de développement pour le programmeur est multipliée - de l'ordre d'un facteur trois. Le code final reste de taille réduite, environ 500 Instructions/Lignes Source. A ce stade, nous disposons de :

- un fichier SKOS (deux ontologies) prototypant la sémantique correspondant à l'application et à un exemple démonstratif ;
- un environnement de qualification (MOSS 2007 et documents Open XML) ;
- un Web Part exploitant SPARQL pour interroger les ontologies SKOS.

4 Test et qualification

Cette section décline quelques scénarios de recherche d'informations permettant de tester et qualifier le prototype opérationnel S-QHSE et, par la même, de montrer explicitement la valeur productive du composant en termes de recherche collaborative cross-lingue. Les concepteurs ont imaginé ces scénarios, basés sur une expérience de plusieurs années des outils collaboratifs de l'entreprise, au contact d'utilisateurs métiers.

Les captures d'écran de la figure 5 illustrent comment le système opérationnel satisfait l'exemple donné en figure 2. Le Web Part doit retourner intelligemment les concepts multilingues associés aux expressions saisies par l'utilisateur. C'est par la pertinence des résultats retournés sémantiquement que l'on peut, à faible coût, améliorer la pertinence des résultats de recherche syntaxique.

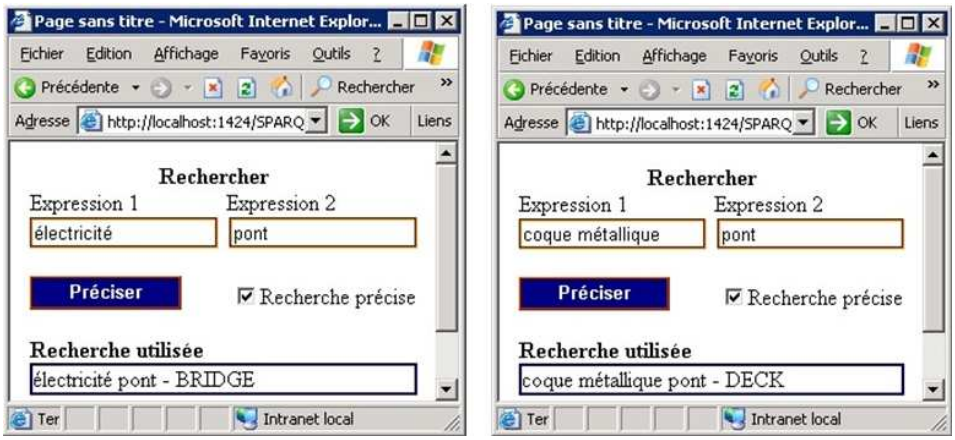


Figure 5 : Traitement sémantique multilingue de l'exemple démonstratif.

Dans cet exemple, pour le mot français "pont", l'utilisateur précise la recherche avec les mots liés aux métiers. Pour les métiers de l'électricité, on précise la recherche syntaxique multilingue avec le mot "bridge", sémantiquement proche (pont électrique / electrical bridge). Pour les métiers liés à la coque métallique, c'est le mot "deck" qui convient le mieux (distance sémantique). Ce premier exemple montre que le composant est doté de capacités de gestion polysémique multilingue, en distinguant les synonymes en plusieurs langues. L'exemple démonstratif donné est satisfait. Qu'en est-il d'autres cas d'utilisation ?

Scénario 1 : Collaboration cross-lingue

Un chaudronnier non francophone oriente sa recherche sur un mot anglais représentatif d'une expression complète : "metal hull", traduction de "coque métallique". Il saisit simplement "hull". Travaillant sur le site français, il précise sa recherche par le mot "pont", son collègue français et non anglophone lui ayant indiqué ce terme. Le Web Part propose d'utiliser les critères de recherche syntaxique "hull", "pont" et "**DECK**". Ainsi, en plus de produire des résultats multilingues, le composant est apte à recevoir des informations cross-lingues ; le chaudronnier trouvera les documents anglophones contenant ses critères .

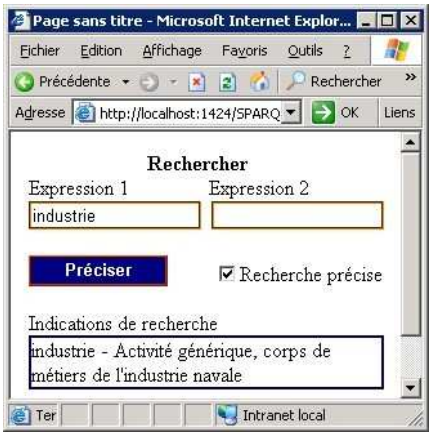
Notons que si l'utilisateur avait précisé "electricity" au lieu de "hull", le composant aurait choisi les mots "electricity", "pont" et "**BRIDGE**". Les résultats de recherche sont donc bien conditionnés par la sémantique induite par l'utilisateur, affranchie de la barrière de la langue.

Scénario 2 : Polysémie et multilinguisme

Un électricien francophone et peu familier des SI, saisirait "Electricité", ou "électricité", ou bien encore "electricité", voire "electricite".
Ici, l'utilisateur reçoit l'information francophone "Pont électrique", correspondant à la langue de son choix sans, ou presque, discrimination orthographique.
S'il avait saisi le morphème - plus petit groupe de syllabes signifiant - "élec" ou "elec", il aurait reçu un lot d'informations vagues en rapport à plusieurs concepts et langues, peu pertinentes, car dénuée de correspondance sémantique précise (critère saisi insuffisant pour déterminer un langage et un concept).
Les indications renvoyées auraient été multilingues et polysémiques, comme "marine electronics" ou "systèmes électriques".

La tolérance à l'erreur syntaxique favorise les utilisateurs éloignés du monde informatique.

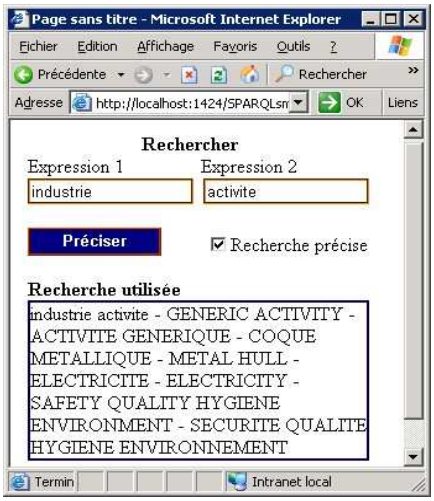
Scénario 4a : Approche concomitante



Invitons maintenant un ouvrier intérimaire polyvalent. Initialement, il ne connaît que le concept d'industrie. Pour cet exemple qui serait aussi démonstratif avec un anglophone, le polyvalent est francophone. Il précise sa recherche avec le seul mot "industrie".
Il reçoit en retour la définition des concepts concomitants dans la langue de son mot clé, sauf homonymie cross-lingue - ici, un seul concept <Activité générique>. L'ouvrier temporaire polyvalent, dispose désormais d'informations suffisantes pour cibler sa recherche en saisissant "activité" (expression 2 – figure suivante).

L'utilisateur novice dans les métiers de l'entreprise est favorisé par l'approche concomitante.

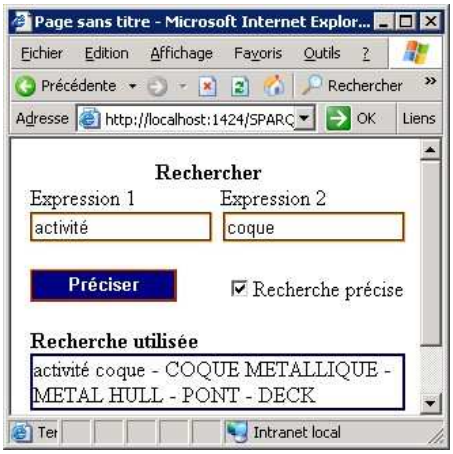
Scénario 4b : Concomitance et subsomption



Avec les mots "industrie" et "activité" ("activite"), l'utilisateur novice dans le domaine d'activité reçoit une nouvelle liste précise de mots clés utilisables. Cette liste est composée des noms de concepts en subsomption avec les nouvelles expressions saisies. Cela signifie qu'il trouve aussi des correspondances s'étendant sous les concepts concomitants (2 niveaux). Les critères proposés sont multilingues et relativement nombreux, puisque dans l'ontologie, <Industrie navale> est un concept de haut niveau (modèle causal).
L'utilisateur novice peut à nouveau, comme précédemment, préciser sa recherche à partir de la proposition qu'il reçoit en reconnaissant parmi les expressions un concept l'intéressant.

L'utilisateur est assisté pour spécialiser sa recherche.

Scénario 4c : Apprentissage des connaissances métiers par la navigation



Les mots "activité" et "coque" retournent une liste de mots utilisables pour préciser plus finement la recherche syntaxique standard SPS 2007.
L'utilisateur, qu'il soit novice ou aguerri, est assisté dans sa démarche de recherche. Partant d'une notion large ou déjà pointue, il est guidé dans l'usage et l'apprentissage de nouvelles connaissances métiers cross-lingues (*i.e.* "*coque* - PONT - DECK").
Note : avec "indus" et "hull", l'utilisateur reçoit "*indus hull* - *PONT* - *DECK*".

Pour divers rôles et compétences, tout corps de métiers utilisant l'application S-QHSE y trouve une médiation entre le vocabulaire utilisateur et celui de l'entreprise.

Nous n'avons pas illustré ici de cas précis orienté sur les métadonnées de l'ontologie d'application. Les résultats obtenus sont de même ordre et prendraient tout leur intérêt avec une ontologie d'application exhaustive. Le prototype opérationnel S-QHSE assume pleinement le rôle de médiation entre le vocabulaire utilisateur et le vocabulaire métier de l'entreprise [Delahousse 2007].

5 Conclusions et perspectives

SKOS, surcouche intermédiaire entre RDF et OWL, est conçu pour modéliser une alternative à la puissance et à la complexité de OWL. Il prodigue un accès simplifié aux grands thèmes sémantiques (classification, terminologies, taxonomies), et facilite la représentation d'**ontologies légères multilingues**.

Le travail d'industrialisation apporte une dimension supplémentaire à la conception de SI sémantiques. On peut l'envisager sur un plan élargi au niveau du poste utilisateur final ou plus localement restreint aux applications le nécessitant, selon la stratégie choisie. La disponibilité du contexte industriel est un sérieux atout. Une étude expérimentale menée dans un contexte d'application industrielle est irremplaçable pour l'approche du besoin réel. L'entreprise, forte des données recueillies par ce biais, peut alors envisager une approche économique du projet pour choisir ou non d' investir et pérenniser l'initiative.

Concevoir un SI industriel sémantique performant implique autant la compréhension pragmatique des métiers utilisateurs qu'une connaissance évolutive des nouvelles technologies. La figure concluant cet article (ci-après) illustre une perspective possible de ces travaux sur le développement d'un SI collaboratif et sémantique de l'industrie navale.

Par ce retour d'expérience, il nous paraît maintenant plus simple d'appréhender la problématique d'intégration et de valeur productive du poste utilisateur final induite par le traitement sémantique de l'information "métiers". La démarche brièvement exposée ici fait écho, dans le monde de l'entreprise, aux systèmes de recherche collaborative sur Internet (e.g. "Searchius"). Ces travaux d'intégration d'une dimension sémantique sur un socle technologique "standard de faits", ont ouvert une voie sécurisée entre des grands thèmes de recherche NTIC et l'industrie navale européenne. Ils sont un pas de plus vers la création de valeurs, dans la *société de connaissances* où les deux activités s'inscrivent désormais.

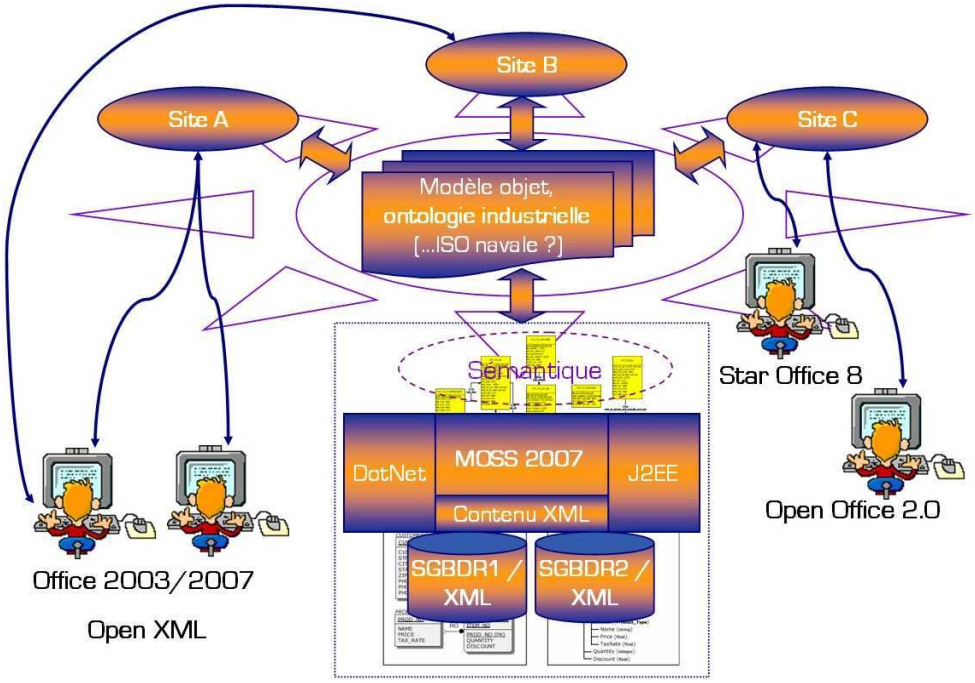


Figure 6 : SI collaboratif et sémantique navale industrielle de référence.

Références

Bachimont, B. (2004). *Arts et Sciences du numérique : ingénierie des connaissances et critique de la raison computationnelle.* Habilitation à Diriger des Recherches. Université de Technologie de Compiègne.

Delahousse, J. (2007). *L'extension des capacités des moteurs de recherche par l'utilisation de terminologies métiers ou comment rendre les moteurs de recherche plus efficaces.* http://mondeca.wordpress.com

Staab, S. and Studer, R. (2004). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer. ISBN:3-540-40834-7. http://dblp.uni-trier.de

Gómez-Pérez, A. and Fernández-Lopez, M. and Corcho, O. (2004). *Ontological Engineering With Examples from the Areas of Knowledge Management*. Expert systems (Computer science). Springer, ISBN 1852335513

Pitrat, J. (1990). *Métaconnaissances - futur de l'intelligence artificielle*. Paris: Hermès.

Gruber, T. (1993). *Toward principles for the design of ontologies used for knowledge sharing*. In N. Guarino et R. Poli (Eds.), Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands. Kluwer Academic Publishers.

Polleres, A. (2006) SPARQL Rules ! (Grupo de Inteligencia Artificial – Universidad Rey Juan Carlos). http://www.w3.org/TR/rdf-sparql-query/

Conception d'une ontologie pour la modélisation de comportements anormaux dans le cadre de la détection d'intrusion

Laurent Deroche*, Henri Briand*, Rémi Lehn**

*Ecole polytechnique de l'université de Nantes, la Chantrerie, rue Christian Pauc,
BP 50609, 44306 Nantes Cedex3,
laurent.deroche@univ-nantes.fr, henri.briand@polytech.univ-nantes.fr
**LINA, 2, rue de la Houssinière, BP 92208, 44322 Nantes Cedex 03,
Equipe COnnaissance et Décision (COD)
remi@fc.univ-nantes.fr

Résumé. Le calcul autonomique oriente aujourd'hui de nombreux travaux de recherche portant sur la gestion autonome des réseaux, et développe ainsi le concept de réseaux autonomiques. Parmi les objectifs d'auto-gestion du calcul autonomique, la sécurité, et plus particulièrement l'auto-protection des systèmes, nécessite des réponses proportionnées à des contraintes particulières afin de détecter des fonctionnements anormaux sur des réseaux hétérogènes et des systèmes de plus en plus distribués.

1 Introduction

L'autonomie dans les réseaux représente une nouvelle fonctionnalité essentielle qui est à l'origine de la future génération des réseaux autonomiques, réseaux qui auront la capacité de réagir à des évènements ou stimulis apparaissant dans leurs environnements. Les réseaux autonomiques reprennent les concepts d'auto-gestion édictés dans le calcul autonomique (IBM, 2004), ils doivent donc être capable de se configurer automatiquement, de chercher en permanence à améliorer leurs performances, de détecter, de diagnostiquer, de subvenir à des dysfonctionnements, et de se protéger d'attaques malveillantes. Sur ce dernier point il est nécessaire aujourd'hui de répondre à des problématiques de sécurité liées au nombre croissant d'éléments logiciels et matériels interagissant et à l'émergence de réseaux dynamiques et décentralisés. Pour cela il est possible de concevoir une architecture multi-agents reposant sur une base de connaissances représentée sous la forme d'une ontologie. L'ontologie peut être utilisée afin de définir une modélisation conceptuelle de haut niveau de cette connaissance sur laquelle s'appuieront les agents qui, grâce à leur caractéristiques intrinsèques, pourront répondre de manière appropriée aux stimulis de l'environnement. L'utilisation d'une telle architecture se révèle alors propice au développement de systèmes de détection d'intrusion pour les réseaux autonomiques. Il est en effet possible de représenter de manière sémantique les tentatives d'intrusions et ainsi grâce aux capacités particulières des agents, de prévenir et de faire avorter

ces tentatives. Cet article se propose donc de présenter une utilisation possible d'une ontologie dans le cadre de la détection d'intrusion.

2 Etat de l'art

Dans cette section est faite une présentation du calcul autonomique, de ses objectifs, de ses évolutions possibles, de son application aux réseaux, ainsi que des caractéristiques des ontologies.

2.1 Le calcul autonomique

Le calcul autonomique est une initiative d'IBM lancée en 2001 avec la publication d'un manifeste (Horn, 2001) désignant la complexité croissante des systèmes comme un frein au progrès des technologies de l'information. Dans ce manifeste Paul Horn fait le constat qu'au taux de croissance actuelle il n'y aurait bientôt plus de personnes qualifiés pour maintenir les systèmes informatiques du monde. En effet, les progrès importants réalisés dans le domaine des technologies de l'information ont considérablement complexifié l'informatique, ce qui a entraîné une nécessité de faire évoluer les caractéristiques et les capacités des éléments permettant de gérer ces ressources et leurs interconnexions. L'objectif de cette initiative est de simplifier la gestion des infrastructures en automatisant des processus et en injectant du savoir-faire directement au cœur des systèmes, les rendant ainsi plus autonomes. L'analogie peut être faite avec le système nerveux autonome qui permet à l'ensemble des fonctions vitales de notre corps (respiratoires, digestives et cardio-vasculaires) de fonctionner et de s'autoréguler sans que nous en ayons conscience, et sans que cela ne fasse intervenir de réflexion. Un système peut être appréhendé comme un ensemble de ressources informatique liées ensemble afin d'effectuer un certains nombre de fonctions préalablement définies. Un système peut donc être un serveur, une application, un équipement réseau, etc. Un tel système doit respecter un certain nombres de règles afin de pouvoir être considéré comme autonomique. Ces caractéristiques ont été définies en huit points clés :

1. le système autonomique doit disposer d'une connaissance détaillée de ses composants.
2. le système autonomique doit pouvoir se configurer et se reconfigurer sous l'effet de conditions variables et imprévisibles.
3. le système autonomique doit optimiser son fonctionnement de façon continue, afin de satisfaire les objectifs de haut niveau.
4. le système autonomique doit être capable de se restaurer suite à un dysfonctionnement, sans altération des données, ni défaut de traitement.
5. le système autonomique doit être capable de détecter, d'identifier et de se protéger afin de maintenir l'intégrité et la sécurité globale du système.
6. le système autonomique doit disposer d'une connaissance suffisante de son environnement afin de faciliter son adaptation.
7. le système autonomique doit pouvoir fonctionner dans un environnement hétérogène et implémenter des standards ouverts, ce ne peut donc pas être une solution propriétaire.
8. le système autonomique doit cacher sa complexité à l'utilisateur.

Pour la réalisation de ces objectifs, les systèmes autonomiques disposent d'une connaissance détaillée de leurs états internes et de leur environnement grâce à une supervision continue des changements pouvant intervenir. Cette supervision est effectuée grâce à une boucle de contrôle fermée qui va permettre la collecte d'évènements et donc déclencher les actions adéquates.

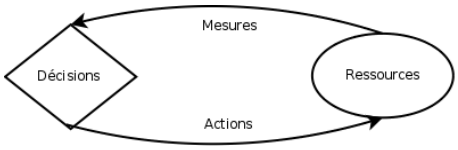


FIG. 1 – *Boucle de contrôle*

Le calcul autonomique vise donc à rendre des services performants et intelligents à des utilisateurs. Ces systèmes autonomiques doivent anticiper les besoins et permettre aux utilisateurs de se concentrer sur ce qu'ils souhaitent accomplir plutôt que sur la manière dont ils devraient configurer le système pour pouvoir y arriver. L'objectif final de l'évolution vers l'autonomie est d'aboutir à un système totalement autonome. L'évolution vers l'autonomie totale peut être décomposée en cinq niveaux :

Niveau 1 (de base, *basic*) : il s'agit de la gestion traditionnelle, il y a de nombreuses sources de données systèmes générées. Ce niveau nécessite de nombreux personnels qualifiés pour l'administration et la maintenance. A ce niveau toutes les opérations sont effectuées manuellement.

Niveau 2 (gérée, *managed*) : les donées sont consolidées au travers d'outils de gestion. Les personnels analysent et prennent des décisions. A ce niveau on note une amélioration de la productivité et meilleure connaissance du système.

Niveau 3 (prédictive, *predictive*) : un système de contrôle permet une corrélation des informations et recommande des actions à prendre. Les personnels approuvent les recommandations et initient les actions à prendre. A ce niveau les prises de décision sont plus efficaces et les compétences nécessaires pour l'administrattion sont réduites.

Niveau 4 (adaptative, *adaptive*) : un système de contrôle permet une corrélation des informations et prend les actions adaptées. A ce niveau les interactions humaines sont minimales.

Niveau 5 (autonomique, *autonomic*) : les composants sont gérés dynamiquement par des politiques de haut niveau.

Pour atteindre ce but les acteurs de l'industrie doivent avoir une approche évolutionnaire et apporter des améliorations aux systèmes en place afin de ne pas avoir à remplacer l'intégralité des environnements liés aux technologies de l'information (Ganek et Corbit, 2002).

2.2 Rendre le réseau autonome

La gestion des réseaux est une activité traditionnellement contrôlée de manière manuelle par un ou plusieurs administrateurs qui sont en charge de la configuration, de la maintenance

et de la sécurisation des environnements. Ces dernières années la convergence des services a changé la vue traditionnelle du réseau composé d'équipements homogènes pour celle d'un réseau englobant une multitude de technologies différentes (mobile/fixe, statique/ad-hoc, etc.), d'équipements hétérogènes et de services divers (temps-réel, qualité de service, etc.). Il apparaît alors que l'administration par l'homme de ces technologies devient de plus en plus complexe au vu du nombre de paramètres de configuration et de réglages que cela impose. Le concept de gestion autonome semble alors une solution à l'administration actuelle des réseaux afin qu'il puisse s'autogérer pour remédier à la complexité croissante et aux coûts excessifs de leur gestion. Pour rendre le réseau plus autonome plusieurs aspects importants sont à prendre en compte. Tout d'abord, il est nécessaire d'en décentraliser le contrôle et la gestion afin de permettre à ces composants d'être plus indépendants en terme de prise de décisions. Les entités composant le réseau doivent être capable de percevoir leur environnement et de réagir de manière adaptée aux évènements s'y produisant. Ces entités doivent être non seulement réactives mais également proactives, c'est à dire qu'elles doivent avoir des buts à atteindre qui dirigent leurs choix et leurs actions pour pouvoir mener une stratégie à long terme. De plus il est nécessaire que ces entités puissent coopérer ensemble tout en réalisant leurs objectifs propres, c'est à dire par exemple, pouvoir garantir une qualité de service de bout en bout à travers des réseaux hétérogènes. Pour réaliser ces buts une entité doit réaliser un certain nombre de plans de manière autonome, elle doit pour cela être capable de s'auto évaluer afin de pouvoir les modifier le cas échéant (dans le cadre d'une optimisation de ces performances).

2.3 Les ontologies

Une ontologie est une spécification explicite d'une conceptualisation d'un domaine (Gandon, 2006). Une conceptualisation rend compte du sens des termes, et la spécification explicite fait des ontologies des objets syntaxiques. Lors de cette conceptualisation seules les propriétés essentielles sont définies car les propriétés étant en nombre indéfinies, on ne peut obtenir qu'une caractérisation partielle ou approximative du domaine. Selon Grueber (1993), plusieurs critères sont à prendre en compte lors sa conception :

- La clarté : une ontologie doit fournir le sens attendu de termes définis, de manière indépendante du contexte. Lorsque c'est possible la définition doit être complète et documentée en langage naturel.
- La cohérence : l'ontologie ne doit autoriser que les inférences qui sont en accord avec les définitions afin de ne pas créer de contradictions.
- L'extensibilité : une ontologie doit être conçue afin d'anticiper l'utilisation du vocabulaire partagé. C'est à dire que l'ontologie doit être capable de définir de nouveaux termes pour des usages spéciaux, basés sur le vocabulaire existant, et ne nécessitant pas de modifications des définitions existantes.
- Une déformation d'encodage minimale : une déformation d'encodage apparaît lorsque les choix d'une représentation sont fait uniquement pour une commodité d'implémentation. Ceci doit être évité car les agents partageant la connaissance peuvent être implémentés dans différents systèmes de représentation et styles de représentation.
- Un engagement ontologique minimal : une ontologie doit faire aussi peu d'affirmations que possible sur ce qui a été modélisé afin de pouvoir instancier et spécialiser l'ontologie suivant les besoins.

Le but fondamentale d'une ontologie est d'améliorer la communication entre humains, entre humains et ordinateurs, entre ordinateurs et donc finalement d'aller vers un vocabulaire standardisé. Les ontologies peuvent être utilisées comme des composants de base pour construire un système autonomique. Elles peuvent être utilisées pour exécuter des analyses automatiques et continues basées sur des politiques de haut niveau définies pour détecter des menaces ou encore des dysfonctionnements. L'introduction d'une ontologie dans un système d'information (SI) vise à améliorer :

- Spécification : acquisition des connaissances pour aider à l'analyse des besoins et à la définition des spécifications.
- Réutilisation : composant réutilisable partagé par plusieurs logiciels.
- Fiabilité : réduire les coûts de maintenance en automatisant des vérifications de cohérence.
- Interopérabilité : format d'échange standardisé pour la coopération entre différents SI.

Il existe de nombreux langages permettant la conception d'ontologies. Parmi ceux-ci on distingue les langages basés sur la logique du premier ordre, comme NOTATION3, N-Triples et RDF (*Resource Description Framework*) , qui représentent les connaissances sous forme d'assertion (sujet, prédicat, objet), et les langages basés sur les logiques de description comme OWL (*Web Ontology Language*).

3 Approche architecturale

Cette section présente les objectifs définis pour rendre la détection d'intrusion autonome ainsi qu'une ébauche de l'ontologie.

3.1 Objectifs

L'objectif de ces recherches est de définir et de proposer une solution afin de concevoir une auto-protection pour les technologies liées aux systèmes d'informations et plus particulièrement dans le cadre de la détection d'intrusion. Un système de détection d'intrusion ou IDS (Intrusion Detection System) est un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une action de prévention sur les risques d'intrusion. Traditionnellement deux méthodes distinguent les différentes solutions proposées sur le marché, l'approche basée sur la connaissance et l'approche comportementale. Dans notre approche nous nous proposons de définir une architecture permettant détecter des tentives d'intrusion connues en nous basant sur leurs signatures, et également d'anticiper ces intrusions en détectant des comportements anormaux, ces actions devant être gérées de manière autonome par le système. Notre auto-protection doit donc permettre à l'entité autonome de se défendre contre des attaques accidentelles et délibérées, et doit de plus éviter les intrusions et les accès non autorisés. Pour réaliser ces objectifs l'entité autonome doit intégrer un ensemble de fonctionnalités :

- Le système doit être autonome, il ne doit pas nécessiter l'intervention d'un opérateur pour éradiquer une menace.
- Le système doit être reconfigurable, afin de s'adapter et de lutter efficacement contre les menaces éventuelles.

– Le système d'auto-protection doit lui même être protégé, afin de ne pas pouvoir être détourné de ses fonctions.
– Le système doit pouvoir s'appuyer sur une base de connaissances, afin de pouvoir différencier les comportements normaux et anormaux. De plus la gestion de cette base doit être dynamique afin de suivre les évolutions du système.

Une telle architecture va permettre une configuration et une reconfiguration plus aisée des modules de sécurité, et va également faciliter la mise en place de nouvelles stratégies de sécurité.

3.2 Définition de l'ontologie

Notre ontologie modélise le domaine de la détection d'intrusion réseau. Le langage utilisé est le langage OWL-DL (*Web Ontology Language - Description logic*) et l'outil logiciel utilisé pour la modélisation est Protégé dans sa version 3.4 beta. OWL est une recommandation du World Wide Web Consortium (W3C, 2004) basée sur les langages OIL (*Ontology Inference Layer*) et DAML + OIL , permettant de définir et d'instancier des ontologies. Une ontologie OWL est un graphe RDF et peut donc s'écrire avec des formes syntaxiques différentes. La signification d'une ontologie OWL est uniquement déterminée par le graphe RDF, l'utilisation de formes syntaxiques RDF/XML est admise tant que cela produit le même ensemble sous-jacent de triplets RDF. Un triplet RDF est une association sujet, prédicat, objet. Le sujet représente la ressource (URI, *Uniform Resource Locator*) à décrire, le prédicat un type de propriété applicable à cette ressource, et l'objet représente une donnée ou une autre ressource. La direction de l'arc est toujours pointée vers l'objet.

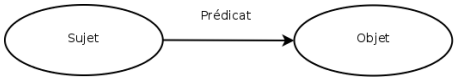


FIG. 2 – *Le triplet RDF*

Une ontologie OWL peut contenir des descriptions de classes, de leurs propriétés et de leurs instances. La sémantique formelle OWL indique comment déduire ses conséquences logiques, c'est à dire comment déduire les faits non présents dans l'ontologie mais qui découlent de la sémantique. Une ontologie OWL est composée d'éléments de base comme des classes, des propriétés, des instances de classes et les relations entre ces instances. OWL se distingue du couple RDF/RDFS du fait qu'il intègre en plus des outils de comparaison des classes et des propriétés (équivalence, contraire, symétrie, etc). Le choix de OWL-DL pour notre modélisation a été guidé par le fait qu'il permet une forte expressivité tout en garantisssant la complétude des raisonnements (toutes les inférences sont calculables). De plus, de nombreux outils ont été développés afin de fournir des systèmes de raisonnement performants supportant les ontologies définies grâce à OWL-DL.
La FIG. 3 ci-dessous présente graphiquement la partie haute de notre ontologie, les ellipses représentant les classes et les sous-classes, et les arcs orientés les relations d'héritage et les propriétés d'objets. Dans cette partie haute nous avons défini les classes *HostSystem*, *HostComponent* et *Intrusion* (les classes *Router* et *Personal Computer* sont ici à titre d'exemples

de sous-classes possibles de *HostSystem*). Il existe une relation *is victim of* entre la classe *HostSystem* et la classe *Intrusion* et une autre relation *has components* entre *HostSystem* et *HostComponent*.

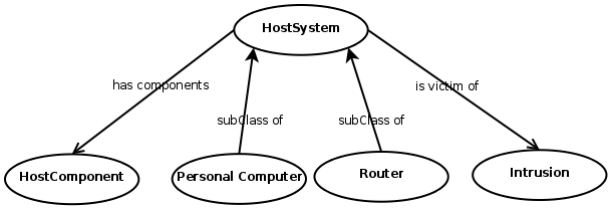


FIG. 3 – *Partie haute de l'ontologie*

La FIG. 4 décrit la classe *HostComponent* et ses sous-classes. La classe *NetworkComponent* est composée de sous-classes décrivant les différentes interfaces réseaux. La classe *Protocol* permet la description de protocoles (IP, TCP, UDP). Il est à noter que les propriétés des sous-classes *Ip, Tcp,* et *Udp* ne représentent pas les caractéristiques des entêtes de ces protocoles, mais des valeurs relatives par exemple au nombre de connexions TCP simultanées autorisées ou encore aux versions du protocole IP autorisées. La classe *AgentComponent* permet la description des agents logiciels locaux et distants, en prenant en compte des caractéristiques comme le type de l'agent ou encore son GUID (*Globally Unique IDentifier*), ceci afin que l'agent est une bonne vision de son environnement et que cela facilite les communications inter-agents. Cette classe possède également une relation avec la classe *Intrusion* (non décrite sur la figure) et une relation avec la classe *Connections*.

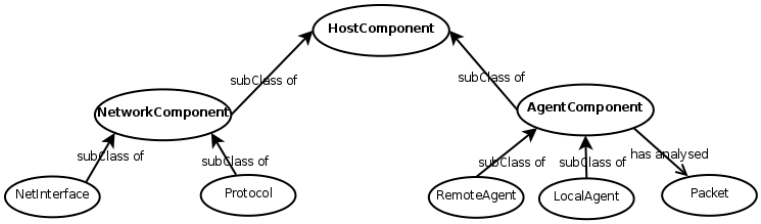


FIG. 4 – *Classe HostComponent*

La FIG. 5 suivante décrit la classe *Connections* et ses sous-classes. La classe *Connections* permet de créer des instances de paquets contenant des informations liées aux entêtes IP, TCP et UDP. La classe *IpV4Connections* est composée de propriétés liées à l'entête IP (IP source, IP destination, informations de fragmentation, etc.), la classe *TcpConnections* est composée de propriétés liées à l'entête TCP (bits de contrôle, numéro d'acquittement, port TCP de destination, etc.), et la classe *UdpConnections* contient les propriétés liées à l'entête UDP (port UDP de destination et port UDP source). Dans le cadre du fonctionnement de notre système, les informations sont recueillies en temps réel sur l'interface réseau surveillée. Par exemple lors

d'une tentative de déni de service comme l'inondation de SYN (*SynFlood*), de nombreuses connexions semi-ouvertes sont établies afin de saturer de manière rapide les ressources systèmes. Notre programme en positionnant la valeur booléenne *HalfOpenExceed* de la classe *Connections* à *True* lorsque qu'un seuil préalablement défini est dépassé, permet à l'ontologie d'inférer qu'il s'agit d'une signature de type *SynFlood*.

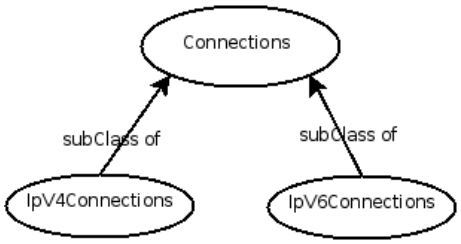


FIG. 5 – *Classe Connections*

La FIG. 6 décrit la classe *Intrusion* et ses différentes sous-classes et relations. Elle permet la définition des différents types de menaces existantes, la sous-classe *DoS* (*Deny of Service*) en étant un exemple. Elle possède une relation avec la classe *Intruder* (permettant la description des éventuels intrus) et avec la classe *Signatures*. Elle possède des propriétés liées à l'heure de la tentative d'intrusion et aux contre-mesures possibles.

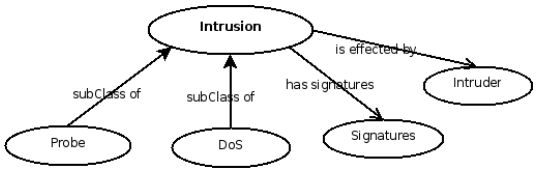


FIG. 6 – *Classe Intrusion*

La FIG. 7 décrit la classe *Signatures* et ses différentes sous-classes. La classe *TrafficSignatures* caractérise le type de traffic généré par l'exécution d'une attaque. La classe *ProcessSignature* contient les propriétés d'exécution d'un programme et notamment son impact sur les ressources locales.

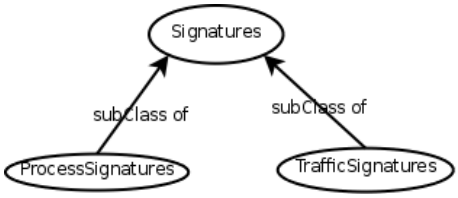


FIG. 7 – *Classe Signatures*

4 Travaux connexes

Durant les années 1990, les menaces comme les virus et les vers sont devenus un problème très préocupant pour la communauté informatique. Une des réponses proposées était le développement d'un système immunitaire artificiel inspiré du système immunitaire naturel (Burgess, 1998). Ce système sera précurseur de ce que sera le calcul autonomique et son architecture possède des aspects très interessant pour le développement d'éléments autonomes sécurisés. Le stockage distribué sécurisé ou SDS (*Secure Distributed Storage)* est une solution pour déployer de manière efficace des informations importantes sur de nombreux serveurs distincts de manière à augmenter la disponibilité et la fiabilité, et pour mettre en place des mécanismes d'auto-correction et d'auto-protection (Garay et al., 1997). Dans le cadre de la détection d'intrusion des travaux ont été menés pour concevoir une architecture basée sur l'utilisation d'agents autonomes (Balsubramananiyan et al., 1998), qui consiste à faire travailler collectivement de multiples entités indépendantes. Plusieurs travaux de conception d'ontologies dans le cadre de la détection d'intrusion ont été menés. Undercoffer et al. (2003) ont défini une ontologie centrée sur la cible à l'aide du language DAML (*Darpa Agent Markup Language*). Cette ontologie permet une modélisation du domaine des attaques informatiques et facilite le processus de raisonnement pour pouvoir détecter et pallier aux intrusions malveillantes. Sa modélisation définit trois catégories, le composant système visé, les moyens mis en oeuvre, les conséquences de l'attaque, et la localisation de l'attaquant (local ou distant). Les travaux de Mandujano (2005) ont permis la définition d'un système multi-agents basé sur une ontologie centrée sur l'attaquant, la modélisation ayant été réalisée en OWL. Cette architecture doit permettre non pas de protéger les ressources locales d'un système mais de prévenir l'utilisation de ces ressources pour compromettre des systèmes voisins. on peut également citer les travaux de Hung et Liu (2006), qui ont défini une ontologie centrée sur l'utilisateur.

5 Perspectives et conclusions

Dans cet article nous pésentons un système d'auto-protection se basant sur une définition sémantique des tentatives d'intrusion. Une ontologie permet la modélisation des comportements connus en utilisant des signatures caractéristiques de certaines attaques, mais l'objectif principal est de pouvoir adapter le comportement général du système par l'apprentissage de nouvelles attaques afin de pouvoir réagir aux attaques non connues. Les ontologies doivent permettre d'atteindre cet objectif grâce à leurs capacités à inférer de nouveaux comportements. De plus les propriétés de réutilisabilité et d'interopérabilité de ces ontologies sont essentielles afin de s'adapter aux problématiques de distribution des systèmes et d'évolution permanente des architectures réseaux.

Références

Balsubramananiyan, J., J. Garcia-Fernandez, D. Isacoff, E. Spafford, et D. Zamboni (1998). An architecture for intrusion detection using autonomous agents. *http://citeseer.ist.psu.edu/balasubramaniyan98architecture.html*.

Burgess, M. (1998). Computer immunology. *http://www.usenix.org/event/lisa98/full_papers/burgess/burgess.pdf*.

Gandon, F. (2006). Ontologies informatiques. *http://interstices.info/display.jsp?id=c_17672*.

Ganek, A. G. et T. A. Corbit (2002). The dawning of the autonomic computing era. *http://www.ibm.com/developerworks/autonomic/library/ac-summary/ac-ganek.html?\S_TACT=105AGX09&S_CMP=EDU*.

Garay, A., R. Gennaro, C. Jutla, et T. Rabin (1997). Secure distributed storage and retrieval. *http://citeseer.ist.psu.edu/garay97secure.html*.

Grueber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. *http://www.cise.ufl.edu/~jhammer/classes/6930/XML-FA02/papers/gruber93ontology.pdf*.

Horn, P. (2001). Autonomic computing : Ibm's perspective on the state of information technology. *http://www.research.ibm.com/autonomic/manifesto/*.

Hung, S. et S. Liu (2006). A user-centric intrusion detection system by using ontology approach. *http://www.atlantis-press.com/publications/aisr/jcis-06/index_jcis.html?http%3A//www.atlantis-press.com/php/paper-details.php%3Fid%3D118*.

IBM (2004). An architectural blueprint for autonomic computing. *http://www-03.ibm.com/autonomic/pdfs/ACBP2_2004-10-04.pdf*.

Mandujano, S. (2005). An ontology-supported outbound intrusion detection system. *http://citeseer.ist.psu.edu/mandujano05ontologysupported.html*.

Undercoffer, J., J. Pinkston, J. Anupam, et T. Finin (2003). A target-centric ontology for intrusion detection. *http://www.cs.vu.nl/~heiner/IJCAI-03/Papers/Undercoffer.pdf*.

W3C (2004). Owl web ontology language reference. *http://www.w3.org/TR/owl-ref/*.

Summary

The autonomic computing directs today numerous research on self-management of networks, and expands the concept of autonomic networks. Among the objectives of self-managing, security, and more particularly systems self-protection, requires proportionate responses to particular constraints to detect abnormal operations across heterogeneous networks and systems more and more distributed.

Ontologies pour l'analyse des risques industriels

Amjad Abou Assali*, Dominique Lenne*
Bruno Debray**

*Université de Technologie de Compiègne, CNRS
HEUDIASYC
{aabouass, dominique.lenne}@hds.utc.fr,
**INERIS
bruno.debray@ineris.fr

Résumé. L'analyse de risques est un processus visant à décrire les scénarios conduisant à des phénomènes dangereux et à des accidents potentiels sur une installation industrielle. Pour réaliser une analyse de risques, un expert dispose de nombreuses ressources (études de dangers, connaissances expertes, etc.) qui sont cependant souvent difficiles à exploiter parce qu'elles ne sont pas suffisamment structurées ni formalisées. Ce papier présente un projet en cours de réalisation qui vise à développer un système d'aide à l'analyse de risques. Notre approche passe par trois stades principaux. Le premier consiste à développer une base de connaissances de la sécurité industrielle reposant sur des ontologies. Le deuxième consiste à indexer les ressources en utilisant les ontologies développées, ce qui a donné lieu à la réalisation d'un système de recherche d'information. Vu la nature des connaissances dont disposent les entreprises industrielles, le troisième stade consiste à développer un système de raisonnement à partir de cas (RàPC) basé sur les ontologies développées. L'intérêt de ce dernier stade est d'étudier le lien entre les systèmes de RàPC et les ontologies en vue d'améliorer la sécurité industrielle.

1 Introduction

La gestion des risques est un processus visant à identifier, analyser et réduire au maximum le risque ou le maintenir dans des limites acceptables (Debray et al., 2006). L'une des étapes clés de la gestion des risques est l'analyse de risques. Ce papier présente un travail en cours de réalisation dans le cadre du projet KMGR (Knowledge Management pour la Gestion des Risques), mené en partenariat avec l'Institut National de l'Environnement industriel et des RISques (INERIS). L'INERIS dispose de nombreuses sources de connaissances (études de dangers, base d'accidents, essais, etc.) auxquelles l'expert peut faire référence afin de réaliser une analyse de risques ; i.e. l'expert est souvent amené à chercher des informations diverses sur la sécurité industrielle, mais aussi sur les études de dangers antérieures qui peuvent être utiles pour l'analyse actuelle.

Afin d'aider l'expert à réaliser des analyses de risques industriels, ce travail a pour premier objectif de développer une base de connaissances de la sécurité industrielle reposant sur des

ontologies (section 2). Nous présentons, dans ce papier, les ontologies développées, ainsi que la méthodologie et les outils proposés (les cartes heuristiques) pour faciliter le développement d'ontologies par les experts industriels.

Le travail vise après à développer un système d'indexation des ressources (documents, outils, etc.) utiles pour l'analyse de risques (section 3, page 5). Ce système permet d'indexer ces ressources par des concepts des ontologies développées, et de produire un index en RDF[1].

Nous présentons ensuite notre système de recherche d'information (section 4, page 6) qui interroge l'index par le biais de requêtes SPARQL[2], et navigue dans les ontologies afin de répondre à la requête de l'utilisateur, et de le guider en lui proposant des concepts intéressants qui aident à diriger voire affiner sa recherche.

Enfin, nous concevons un système de raisonnement à partir de cas (section 5, page 7) ayant pour objectif d'aider l'expert à réaliser des analyses des risques industriels en raisonnant sur d'autres analyses et expériences préalablement produites. Les ontologies développées sont intégrées à ce système afin d'étudier l'apport de cette intégration aux systèmes d'aide à l'analyse des risques industriels.

2 Développement d'ontologies

2.1 Ontologies développées

Dans un premier temps, ce travail a pour objectif de développer une base de connaissances de la sécurité industrielle reposant sur plusieurs ontologies. En classifiant ces ontologies selon leurs niveaux de spécificité (figure 1), nous avons :

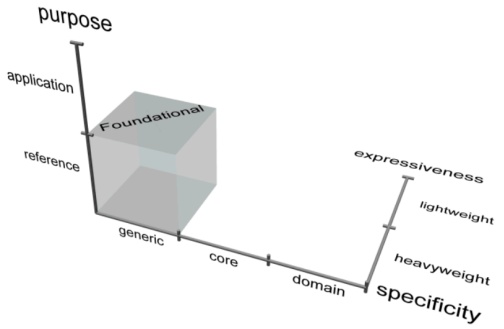


FIG. 1 – *Dimensions de classification d'ontologies.*

1. une ontologie noyau (core ontology) de la sécurité industrielle contenant les concepts généraux du domaine de la sécurité industrielle (e.g. phénomène dangereux, explosion, effet, etc.).
2. une ontologie du domaine du GPL (Gaz de Pétrole Liquéfié). Cette ontologie contient des concepts concernant le GPL (e.g. réservoir de stockage, vanne pneumatique, etc.), et qui sont des spécialisations d'autres concepts de l'ontologie noyau.

[1]http ://www.w3.org/RDF/
[2]http ://www.w3.org/TR/rdf-sparql-query/

3. une ontologie de domaine de l'analyse de risques décrivant les concepts utilisés pour réaliser une analyse de risques. Pour définir ces concepts, nous nous sommes basés sur la méthode ARAMIS[3] d'analyse de risques (Debray et al., 2006; Trommeter et al., 2006). Du coup, on trouve ici des concepts comme, par exemple : Barrière de sécurité, Évènement redouté central, etc. Par ailleurs, ces concepts sont également des spécialisations d'autres concepts de l'ontologie noyau.
4. enfin, une ontologie décrivant les types de ressources dans l'entreprise (rapport, article, outil, etc.), leurs propriétés (titre, auteur, etc.), et les relations entre ces ressources (voir_aussi, a_pour_référence, etc.). Cette ontologie fait partie de notre système d'indexation.

2.2 Méthodologie et outils de développement

Le développement des ontologies est réalisé en lien avec plusieurs experts de l'INERIS, avec l'aide d'un expert du développement d'ontologies. Pourtant, nous avons constaté que les experts du domaine trouvent des difficultés à concevoir des ontologies, et même à travailler directement avec les plates-formes de développement d'ontologies (Protégé, par exemple). Pour cela, nous avions pour enjeu de leur proposer une méthodologie à suivre et des outils de développement plus faciles à manipuler.

Plusieurs méthodologies ont été définies pour développer une ontologie (Sure et al., 2002; Fernandez et al., 1997). Dans ce travail, nous proposons une méthodologie qui s'inspire de la méthodologie METHONTOLOGY (Fernandez et al., 1997; Nagypál, 2005) pour définir les stades de développement suivants (figure 2) :

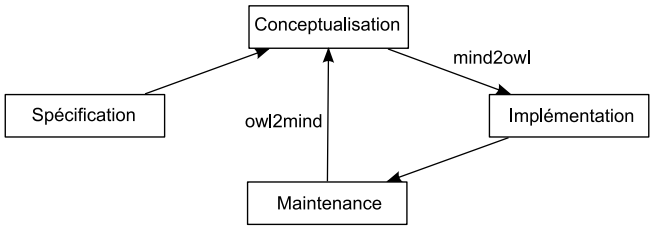


FIG. 2 – *Cycle de développement d'ontologie.*

1. **Spécification** : Ce stade consiste à donner une description générale de l'ontologie à développer. On peut envisager les délivrables suivants :
   (a) un document des *principes de design* d'ontologies (Noy et McGuinness, 2001) précisant la façon de nommer les concepts, ainsi que les divers choix à prendre durant le processus de développement.
   (b) un document de *spécification de l'ontologie* décrivant le but, la portée, et les concepts essentiels (les concepts racines) de l'ontologie. Il est important également de préciser pour chacun de ces concepts ses sources d'informations (rapports, cahiers des charges, etc.), ainsi que les experts qui peuvent travailler dessus.

[3]http ://aramis.jrc.it/

2. **Conceptualisation** : Ce stade consiste à produire le modèle conceptuel de l'ontologie sous forme de glossaire de termes, tableaux, hiérarchie de concepts, etc. Pour cela, nous avons passé par deux étapes :
   (a) La *définition des concepts* : d'abord, nous avons déterminé les concepts à partir d'un glossaire de termes industriels qui existait déjà à l'INERIS. Ensuite, nous avons suivi une méthode semi-automatique (hors du scope de ce papier), mais aussi une méthode manuelle pour extraire les autres concepts intéressants.
   (b) La *hiérarchisation de ces concepts* : vu la difficulté que rencontrent les experts du domaine en travaillant directement avec les plates-formes de développement d'ontologie, nous avons cherché d'autres moyens qui aident à produire le modèle conceptuel de l'ontologie. Plusieurs supports ont été proposés dans la littérature pour produire des modèles conceptuels (UML, cartes heuristiques (Sure et al., 2002), etc.). Nous avons constaté que les cartes heuristiques (mind maps) sont très efficaces pour représenter et visualiser la hiérarchie des concepts. Elles sont, par ailleurs, assez répandues et faciles à manipuler (nous avons utilisé pour cela l'éditeur FreeMind[4]). En revanche, comme les cartes heuristiques ne sont pas destinées à la conceptualisation d'ontologies, nous avons proposé quelques principes qui exploitent les différents éléments de FreeMind (nœud, icône, commentaire, etc.) afin de modéliser les éléments d'ontologie (classe, propriété, instance, etc.) dont on a besoin lors de la conceptualisation (figure 3) :
      – chaque *classe* de l'ontologie est représentée par un *nœud* dans la carte (e.g. Substance dangereuse).
      – pour rajouter une *propriété* au nœud de cette classe, on crée un sous-nœud "Propriété" (avec l'icône illustrée dans la figure 3). Puis, on rajoute les propriétés comme sous-nœuds de ce dernier nœud, tout en marquant le type de chaque propriété : un type simple pour les propriétés datatype (e.g. symbole : string), ou bien le nom d'une autre classe pour les relations (e.g. has_phrase_de_risque : Phrase de risque).
      – chaque *instance* d'une classe est représentée par un *nœud* avec *l'icône* ① (e.g. methanol, chlore).

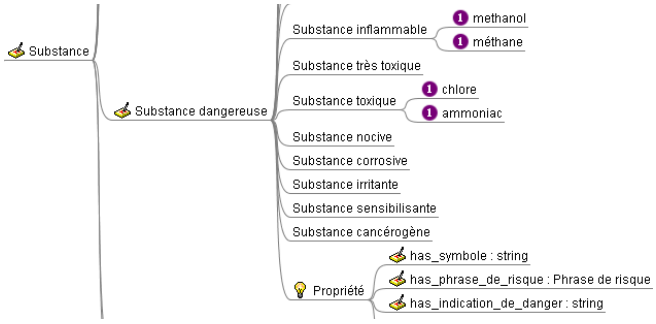


FIG. 3 – *Partie de la carte heuristique de l'ontologie de la sécurité industrielle.*

[4]http ://freemind.sourceforge.net/

3. **Implémentation** : Ce stade consiste à passer du modèle conceptuel à une ontologie implémentée dans l'un des langages d'ontologie (OWL Lite). Ce passage est fait automatiquement grâce à l'outil *mind2owl* que nous avons développé. Cet outil récupère le modèle conceptuel (en XML) réalisé en FreeMind, et le transfère en fichier OWL que l'on peut ouvrir et modifier avec la plate-forme Protégé.
4. **Maintenance** : Ce stade consiste à mettre à jour l'ontologie développée en suivant une méthode centralisée où une seule personne est responsable de la maintenance. Pour ce faire, nous repassons au modèle conceptuel au travers de l'outil *owl2mind* que nous avons développé ; nous modifions le modèle conceptuel ; et enfin, nous reproduisons le modèle implémenté en OWL (figure 2). Notre réflexion concernant ce stade est encore en cours, car il faut prendre en compte plusieurs aspects importants de la maintenance (comme, par exemple, son impact sur les autres systèmes du projet).

3 Système d'indexation

Le système d'indexation développé est composé de deux parties principales (figure 4) : une première partie contenant l'ontologie décrivant les types de ressources à indexer avec ses individus, et une deuxième partie contenant les trois autres ontologies avec leurs individus (Abou Assali et al., 2007).

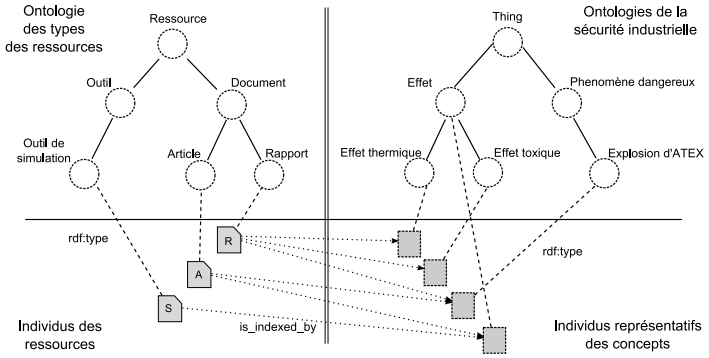


FIG. 4 – *Architecture du système d'indexation.*

Le processus d'indexation proposé se réalise en deux étapes (figure 5) :

- Instanciation. Pour chaque ressource, on crée un individu de la classe correspondante (rapport, article, etc.), et on renseigne les valeurs de ses différents attributs (titre, voir_aussi, etc.).

- Association. Dans cette étape, on associe à chaque individu correspondant à une ressource les concepts des trois ontologies jugés pertinents pour cette ressource. Cette association se fait à travers la relation "*is_indexed_by*" que nous avons définie en tant que propriété objet entre une ressource et un concept industriel (figure 4). À la fin de cette étape, on obtient un fichier RDF représentant l'index que l'on peut interroger par des requêtes SPARQL.

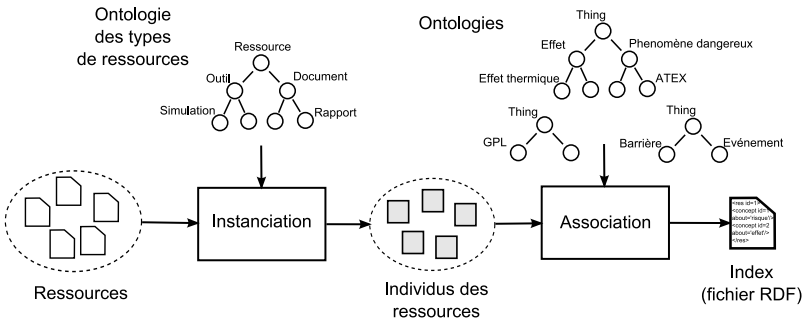


FIG. 5 – *Processus d'indexation.*

4 Système de recherche d'information

Pour faciliter l'exploitation des ressources, nous avons développé un système de recherche d'information basé sur les ontologies et l'index produit. Ce système est composé des trois couches suivantes (figure 6) :

- **Interface** : c'est une application Web (figure 7), où l'on peut saisir une requête en utilisant les concepts des ontologies développées.
- **Outils d'accès aux connaissances** : qui contient les outils nécessaires (Java servlet, et Jena API) permettant d'interroger les sources de connaissances afin de répondre à la requête de l'utilisateur.
- **Sources de connaissances** : qui contient les ressources, l'index, et la base de connaissances.

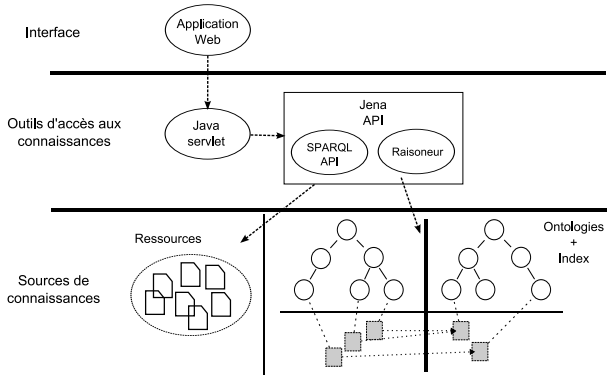


FIG. 6 – *Architecture du système de recherche d'information.*

Pour mieux expliquer les caractéristiques de ce système, nous supposons dans la suite que l'utilisateur a lancé la requête "*effet + explosion*" (voir figure 4). Nous présentons à l'aide de cet exemple les caractéristiques les plus importantes du système :

FIG. 7 – *Système de recherche d'information.*

Propositions. Trois types de propositions se présentent à l'utilisateur afin de le guider dans sa recherche :

1. **propositions lexicales** : ce sont les concepts contenant les termes de la requête. Ici, on trouvera "*Effet*", "*Effet thermique*", et "*Effet toxique*" pour le premier terme, et "*Explosion d'ATEX*" pour le deuxième. Supposons, dans la suite, que l'utilisateur choisisse, parmi ces propositions, "*Effet thermique + Explosion d'ATEX*".
2. **propositions sémantiques** : ce sont les concepts pères et/ou fils des concepts de la requête dans les ontologies, et qui sont liés à certaines ressources. Ici, on trouvera "*Effet + Explosion d'ATEX*" qui sont liés à l'article *A*.
3. **propositions d'affinage** : ce sont les autres concepts liés aux mêmes ressources résultantes, et qui aident à affiner les résultats. Ici, on trouvera "*Effet toxique*".

Interface bilingue. Le système de recherche d'information présente une interface bilingue (française ou anglaise), et nous travaillons actuellement pour développer la recherche bilingue qui permettra à l'utilisateur de saisir ses requêtes en deux langues.

Recherche par acronymes. Dans le domaine de la sécurité industrielle, on trouve beaucoup de concepts connus par leurs acronymes (BLEVE, BOIL-OVER, etc.). Ainsi, nous avons associé ces acronymes à leurs concepts correspondants dans les ontologies afin de permettre à l'utilisateur de les utiliser dans sa recherche.

5 Système de raisonnement à partir de cas

Le raisonnement à partir de cas (RàPC) est un domaine de recherche ayant pour objectif de fournir des méthodes et des outils pour trouver des solutions à de nouveaux problèmes en raisonnant sur des problèmes anciens similaires déjà résolus. Un processus d'adaptation permet ensuite d'adapter les solutions adoptées dans les problèmes anciens à ceux actuels.

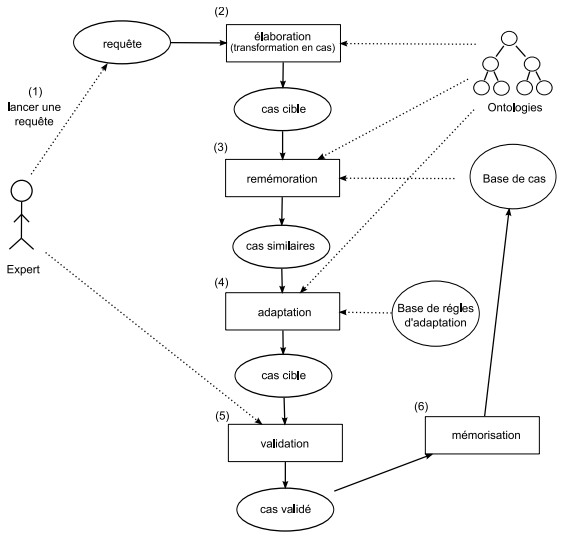


FIG. 8 – *Système de raisonnement à partir de cas.*

Dans la méthode ARAMIS d'analyse de risques (Debray et al., 2006; Trommeter et al., 2006), l'expert a pour mission, dans un premier temps, de décrire les scénarios d'accidents potentiels (l'enchaînement des évènements redoutés) sur une installation industrielle. Ensuite, il doit proposer des barrières (mesures) de sécurité (Ayrault, 2005) ayant pour objectif de réduire au maximum les risques engendrés par les évènements redoutés. La question que se pose l'expert à ce stade est : *quelles sont les barrières de sécurité les plus performantes à mettre en œuvre dans le contexte actuel* ?

Pour répondre à cette problématique, nous avons proposé de développer un système de RàPC qui permet à l'expert de se servir des études d'analyse de risques préalablement produites dans l'analyse actuelle. Ce système devra proposer à l'expert des barrières performantes adaptées de celles qui ont été employées dans des contextes similaires.

Ce système est composé des étapes classiques d'élaboration, de remémoration, d'adaptation, de validation, et de mémorisation (Aamodt et Plaza, 1994; Renaud et al., 2007). Comme on peut le remarquer dans la figure 8, les ontologies développées sont intégrées à notre système afin d'étudier l'apport de cette intégration aux systèmes d'aide à l'analyse des risques industriels. Les ontologies interviennent particulièrement dans les étapes d'élaboration (pour décrire un cas), de remémoration (pour enrichir la recherche de cas similaires), et d'adaptation (pour améliorer la qualité des solutions proposées).

6 Conclusion

Nous avons présenté, dans ce papier, un projet en cours de réalisation pour développer un système d'aide à l'analyse des risques industriels. Ce système comporte une base de connaissances reposant sur des ontologies ; un système d'indexation des ressources importantes pour

la sécurité industrielle ; un système de recherche d'information ; ainsi qu'un système de RàPC basé sur les ontologies développées.

Le développement des ontologies s'inspire de la méthodologie METHONTOLOGY (Fernandez et al., 1997) pour définir les stades de spécification, conceptualisation, implémentation, et maintenance. Pour pallier aux difficultés rencontrées par les experts en utilisant les plates-formes de développement d'ontologie (Protégé, par exemple), nous avons proposé d'utiliser les cartes heuristiques comme moyens pratiques pour produire les modèles conceptuels des ontologies. Nous avons également développé les outils *mind2owl* et *owl2mind* qui permettent de passer du modèle conceptuel au modèle implémenté en OWL, et vice versa.

Nous avons présenté ensuite le système d'indexation basé sur les ontologies développées, et le processus d'indexation permettant de produire un index en RDF interrogeable par le langage de requêtes SPARQL. En s'appuyant sur cet index et sur les ontologies, nous avons développé un système de recherche d'information qui aide l'expert dans sa recherche en lui proposant des concepts intéressants afin de diriger et affiner sa recherche.

Enfin, nous avons présenté le système de RàPC auquel nous avons intégré nos ontologies en vue d'améliorer l'analyse des risques industriels. Ce système est actuellement en cours d'élaboration. Une des difficultés rencontrées est que la plupart des données, concernant la mise en œuvre des barrières de sécurité, sont qualitatives, et du coup, leur exploitation dans un système de RàPC n'est pas immédiate.

Références

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

Abou Assali, A., D. Lenne, et B. Debray (2007). KoMIS : An Ontology-Based Knowledge Management System for Industrial Safety. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA 2007)*, Regensburg, Germany.

Ayrault, N. (2005). Evaluation des dispositifs de prévention et de protection utilisés pour réduire les risques d'accidents majeurs : Evaluation des barrières techniques de sécurité. Rapport Omega 10, INERIS, France.

Debray, B., S. Chaumette, S. Descourière, et V. Trommeter (2006). Méthodes d'Analyse des Risques Générés par une Installation Industrielle. Rapport Omega 7, INERIS, France.

Fernandez, M., A. Gomez-Perez, et N. Juristo (1997). METHONTOLOGY : From Ontological Art towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, USA, pp. 33–40.

Nagypál, G. (2005). Methodology for Building SWS Ontologies in DIP. *DIP deliverable D 3*.

Noy, N. et D. McGuinness (2001). Ontology Development 101 : A Guide to Creating Your First Ontology. Technical Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.

Renaud, J., B. Chebel Morello, B. Fuchs, et J. Lieber (2007). *Raisonnement à Partir de Cas 1 : Conception et Configuration de Produits*, Volume 1. Hermès - Lavoisier.

Riesbeck, C. et R. Schank (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates.

Sure, Y., M. Erdmann, J. Angele, S. Staab, R. Studer, et D. Wenke (2002). OntoEdit : Collaborative Ontology Development for the Semantic Web. *First International Semantic Web Conference (ISWC 2002)*.

Trommeter, V., S. Chaumette, S. Descourrière, V. De-Dianous, F. Prats, B. Debray, et T. Le (2006). Un Outil d'Analyse de Risques. Cahier des charges, INERIS, France.

Summary

This paper presents an ongoing project that aims to develop an assistance system for risk analysis at industrial organisations. First, we present a methodology and tools to develop ontologies by domain experts. Then, we propose an approach to index the organization resources using ontologies, and we present the information retrieval system which is based on the ontologies and the index produced. Finally, we present our case-based reasoning system associated to the ontologies to help the expert realize risk analysis studies using already existing ones.

Gestion des changements d'une ontologie : Résolution d'inconsistances guidée par l'évaluation de la qualité

Rim Djedidi*, Marie-Aude Aufaure*

* Département Informatique, Supélec Campus de Gif
Plateau du Moulon – 3, rue Joliot Curie – 91192 Gif sur Yvette Cedex
Rim.Djedidi@Supelec.fr
Marie-Aude.Aufaure@Supelec.fr

Résumé. L'adaptabilité des ontologies à l'évolution des domaines qu'elles modélisent et à la dynamicité des environnements dans lesquelles elles sont appliquées est un besoin fonctionnel essentiel à prendre en compte dans le cycle d'ingénierie ontologique. Plusieurs problématiques émanent de l'évolution d'ontologie : l'identification des besoins de changement, la spécification des changements, l'application des changements, la traçabilité des changements, la propagation des changements aux artefacts dépendants, etc. Dans cet article, nous proposons une approche de gestion de changements d'ontologie dont l'objectif est d'optimiser et automatiser la validation des changements en assurant la consistance et la qualité de l'ontologie évoluée. La maintenance de la consistance est assurée à travers la proposition d'alternatives de résolution d'inconsistance. Un modèle de qualité est défini et appliqué pour évaluer l'impact des alternatives proposées sur la qualité de l'ontologie et guider la résolution des inconsistances tout en minimisant la dépendance à l'utilisateur.

1 Introduction

Initialement pensées comme une conceptualisation finie d'un domaine de connaissances délimité et stable, les ontologies sont de plus en plus appliquées à des environnements ouverts et dynamiques, les connaissances qu'elles modélisent ainsi que les besoins utilisateurs auxquels elles répondent sont en continuelle évolution. Pour prendre en compte cette dynamicité, les ontologies doivent s'adapter aux changements.

La gestion des changements d'une ontologie est une tâche cruciale. L'ingénieur d'ontologie doit contrôler tous les effets de changement, les résoudre et évaluer l'impact du changement sur l'ontologie. Il est indispensable de disposer d'une approche méthodologique de gestion des changements qui permet de guider la validation des changements et d'automatiser le processus d'évolution.

Dans cet article, nous présentons une approche de gestion de changements d'ontologies OWL selon laquelle, la validation des changements tient compte aussi bien de la consistance que de la qualité de l'ontologie évoluée.

L'article est structuré comme suit : dans la section 2, nous introduisons les principes de gestion de changements d'ontologie. La section 3 détaille les différentes phases de l'approche proposée. Le modèle d'évaluation de qualité est présenté dans la section 4 et la conclusion et les perspectives de nos travaux dans la section 5.

2 Gestion des Changements d'une ontologie

Pour qu'une ontologie réponde à ses objectifs organisationnels et aux attentes des usagers, il est essentiel qu'elle puisse prendre en compte les besoins de changements. Les motifs de changements sont multiples : adaptation à la dynamicité du domaine, aux patrons d'usage de l'application utilisant l'ontologie ou aux besoins de changements liés à la représentation interne de l'ontologie et sa conceptualisation. L'ontologie évoluée doit répondre aux besoins de changement tout en assurant la consistance des connaissances qu'elles modélisent.

Plusieurs approches d'évolution d'ontologies ont été proposées dans la littérature. Certaines se sont focalisées sur l'identification des besoins de changements (Klein et al., 2002) (Plessers et De Troyer, 2005), (Eder et Wiggiser, 2007), d'autres sur la spécification formelle des changements (Klein, 2004), (Stojanovic et al., 2002) et l'implémentation des changements (Stojanovic et al., 2003). A travers l'étude des besoins de changements et l'environnement d'évolution d'une ontologie, Stojanovic (2004) a proposé un processus complet d'évolution d'ontologies KAON, organisé en 6 phases : identification des changements, représentation des changements, sémantique des changements, propagation des changements aux artefacts dépendants (ontologies et/ou applications référencées par l'ontologie évoluée), implémentation des changements et validation des changements.

2.1 Classification des changements

La classification des changements est étroitement liée au langage de représentation de l'ontologie. Plusieurs classifications ont été proposées dans la littérature. L'idée principale consiste à définir une taxonomie de changement pour un modèle spécifique de représentation d'ontologie.

Maedche et al. (2002) classifient les changements d'une ontologie KAON selon trois niveaux d'abstraction : les changements élémentaires dont la modification porte sur une seule entité de l'ontologie (ex. ajouter une propriété), les changements composites dont la modification porte sur le voisinage direct d'une entité de l'ontologie (ex. généraliser un concept, remonter un concept) et les changements complexes dont la modification porte sur un ensemble arbitraire d'entités de l'ontologie.

Une taxonomie similaire pour les changements d'une ontologie OWL a été proposée dans (Klein, 2004). Deux types d'opérations de changements ont été distingués : les opérations de changement basiques appliquées à une entité de l'ontologie, et les opérations de changement complexes qui représentent des mécanismes de regroupement d'opérations basiques pour former des unités logiques de changement.

Les changements élémentaires et basiques (opérations basiques) sont dérivés directement et de manière exhaustive du modèle de représentation de l'ontologie. Les changements composites et complexes sont infinis.

2.2 Impact et sémantique des changements

Changer le comportement d'un concept ou d'une relation de sorte que de nouvelles règles ou connaissances prennent effet, fait intervenir des connaissances inter-reliées et complexes et nécessitent la définition de mécanismes spécifiant comment les connaissances peuvent être changées et comment maintenir la consistance de ces connaissances. Les changements peuvent avoir plusieurs conséquences particulièrement sur la consistance de l'ontologie. La spécification de la consistance dépend du modèle dans lequel est représentée l'ontologie. Stojanovic et al (2003) définissent la consistance comme suit : « une ontologie O est dite consistante si et seulement si, elle satisfait les contraintes définies par le modèle fondamentale de l'ontologie ».

L'analyse des effets de changements inclue non seulement la vérification de la consistance mais aussi la résolution des inconsistances détectées. Néanmoins, il n'y a pas suffisamment de travaux sur la maintenance de la consistance dans la littérature. La proposition la plus significative est présentée par Stojanovic et al. (Stojanovic et al., 2002) (Stojanovic et al., 2003) (Stojanovic, 2004). Les auteurs spécifient la sémantique des changements d'une ontologie KAON et proposent des stratégies de résolution pour maintenir la consistance de l'ontologie évoluée en se basant sur les contraintes du langage KAON. Plusieurs solutions peuvent être proposées pour résoudre une inconsistance. Les stratégies de résolution sont présentées à l'utilisateur comme des changements additionnels à appliquer conjointement aux changements requis pour maintenir l'ontologie dans un état consistant. L'utilisateur décidera de la stratégie à appliquer en fonction de son expertise du domaine.

2.3 Application des changements

L'application des changements consiste à implémenter physiquement le changement en gardant la traçabilité des modifications effectuées sur l'ontologie.

Pendant l'analyse des changements, les besoins de changement exprimés par l'utilisateur, sont enrichis par des changements dérivés permettant d'assurer la consistance de l'ontologie (et le cas échéant de ses artefacts dépendants). L'application réelle des changements n'est exécutée que si les résultats de leur analyse sont approuvés par l'ingénieur d'ontologie. La séparation de l'analyse des changements de leur application finale constitue l'un des principaux avantages du processus d'évolution proposé dans (Stojanovic, 2004). L'ingénieur d'ontologie peut valider l'ensemble des changements comme il peut les annuler, s'il juge que leur pertinence ne justifie pas leur coût ou si dans un contexte collaboratif, les utilisateurs divergent dans leur appréciation du changement.

La représentation et l'interprétation des informations sur les changements nécessitent une compréhension commune du modèle des changements et du journal des changements. Maedche et al. (2002) ont introduit à ce propos deux notions : l'*ontologie d'évolution* et le *journal d'évolution*. L'ontologie d'évolution définit un modèle des changements applicables à une ontologie de domaine permettant ainsi, de mieux les gérer. Le journal d'évolution décrit l'historique des changements appliqués à l'ontologie de domaine sous forme de séquences chronologiques d'information sur chaque changement. Il renferme des connaissances sur le développement et la maintenance de l'ontologie de domaine. La modélisation des changements (ontologie d'évolution) et de l'historique de leur application (journal d'évolution) facilitent la synchronisation de l'évolution de l'ontologie de domaine. Par ailleurs, si une annulation (ou une reprise) des changements est nécessaire, le journal d'évolution, basé sur

un modèle formel de changements, permet de mieux guider ces opérations (Stojanovic, 2004).

Une fois les changements appliqués, d'autres problèmes peuvent être identifiés et inférer de nouveaux besoins de changements selon un processus d'évolution cyclique.

3 Approche méthodologique de validation de Changements

L'application d'un changement peut altérer la consistance et la qualité de l'ontologie. Toutefois, l'impact des changements sur la consistance et la qualité n'ont pas été pris en compte pour autant dans la littérature. Stojanovic (2004) a proposé un processus global de gestion d'évolution pour les ontologies KAON dans lequel elle spécifie la sémantique des changements et la maintenance de la consistance. Les principes de résolution d'inconsistances ont aussi été adaptés dans (Haase et Stojanovic 2005), aux ontologies OWL. Cependant, à notre connaissance, les effets de changements sur la qualité de l'ontologie n'ont pas été étudiés dans la littérature bien que la notion de qualité d'une ontologie ait fait l'objet de plusieurs recherches (Hartmann et al., 2005) (Brank et al., 2005) (Supekar, 2005) (Yang et al. 2006).

Les inconsistances peuvent être résolues de plusieurs manières. Différentes alternatives de résolution peuvent en effet, être proposées pour un changement. Haase et Stojanovic (2005) présentent un modèle pour la sémantique des changements OWL et introduisent des stratégies de résolution en se basant sur les contraintes du modèle OWL-Lite. Ces stratégies sont présentées à l'utilisateur pour qu'il puisse valider le changement.

Cependant, la gestion des changements ne doit pas se focaliser uniquement sur la consistance mais aussi sur la qualité de l'ontologie. Selon cette perspective, nous proposons une approche complémentaire à ces travaux permettant de minimiser l'intervention de l'utilisateur dans la validation des changements et ce, en évaluant l'impact des alternatives de résolution sur la qualité de l'ontologie pour guider le choix de l'alternative à appliquer. L'approche est conduite à travers les phases suivantes :

1. Détection des inconsistances en vérifiant la consistance de l'ontologie par rapport à la sémantique formelle du modèle qui la décrit ;
2. Proposition d'alternatives de résolution pour les inconsistances détectées ;
3. Evaluation de l'impact des alternatives de résolution sur la qualité de l'ontologie ;
4. Application des changements et validation finale.

3.1 Détection des inconsistances

Cette phase a pour objectif de délimiter la partie inconsistante de l'ontologie. La consistance est vérifiée par rapport à la sémantique formelle du modèle dans lequel est décrite l'ontologie. La définition de la consistance dérivée du modèle de l'ontologie, facilite la caractérisation des inconsistances qui peuvent être causées par un type particulier de changements (les changements les plus fréquents, les changements élémentaires tels que l'ajout d'un concept, l'ajout d'une propriété, etc.).

Dans notre approche de validation de changements, nous nous intéressons à l'évolution des ontologies OWL-DL. Aujourd'hui, les logiques de descriptions DL représentent le modèle fondamental du Web sémantique, particulièrement dans la conception d'ontologies (Horrocks, et al., 2003). OWL-DL restreint l'utilisation du vocabulaire OWL (ex. un élément

ne peut être à la fois classe et instance) et bénéficie des acquis des DL : une sémantique bien définie, des propriétés formelles compréhensibles en termes de complexité et de décidabilité, des algorithmes d'inférence connus et des raisonneurs DL existants qui permettent de vérifier la consistance des ontologies.

La consistance peut être définie comme étant un ensemble de conditions qu'une ontologie doit satisfaire. Les conditions de consistance d'une ontologie OWL peuvent être classées selon trois niveaux de consistance (Haase et Stojanovic, 2005): consistance structurelle, consistance utilisateur et consistance logique.

3.1.1 Consistance structurelle

La consistance structurelle se réfère aux contraintes du langage de l'ontologie, c'est-à-dire l'utilisation de ses constructeurs. Les conditions structurelles de OWL-DL représentent les conditions propres à ses constructeurs et des contraintes sur les axiomes élémentaires ou les combinaisons d'axiomes. Elles proscrivent certains constructeurs et certaines manières d'utiliser les constructeurs (Horrocks, et al., 2003). Les conditions structurelles de OWL-DL sont nombreuses et variées, il est difficile de fixer un ensemble prédéfini d'inconsistances à résoudre. Haase et Stojanovic (2005), ont limité la résolution d'inconsistances structurelle à un ensemble arbitraire de conditions de OWL-Lite. Les conditions structurelles de OWL-DL peuvent être vérifiées par l'emploi de raisonneurs DL tels que Racer, FaCT++, Pellet, etc. La plupart de ces raisonneurs sont assez performants pour ce type de tâche.

3.1.2 Consistance utilisateur

La consistance utilisateur se réfère aux contraintes liées à la modélisation et au contexte applicatif et/ou d'usage de l'ontologie. Ces contraintes sont définies explicitement par l'utilisateur (ingénieur d'ontologie, expert du domaine) et ne relèvent pas du langage lui-même. Haase et Stojanovic (2005) identifient deux types de conditions utilisateur :

- La consistance générique qui permet de préciser des critères qualitatifs de modélisation tels que l'application des méta-propriétés de OntoClean (Guarino et Welty, 2002). Elle est définie par l'ingénieur d'ontologie ;
- La consistance dépendant du domaine qui permet de définir des conditions particulières directement liées au domaine modélisé. Elles relèvent de l'expertise du domaine.

3.1.3 Consistance logique

La consistance logique se réfère à la sémantique formelle de l'ontologie et donc à sa satisfiabilité dans le sens où elle est sémantiquement correcte et ne présente pas de contradictions logiques.

OWL-DL est un langage centré axiome. Les concepts et les rôles possèdent des descriptions structurelles, élaborées à partir d'un certain nombre de constructeurs. Une sémantique est associée à chaque description par l'intermédiaire d'une interprétation du domaine (Horrocks et Patel-Schneider, 2004). La satisfaction de l'ontologie par une interprétation est contrainte par la satisfaction de tous les axiomes de l'ontologie (axiomes de concepts, de propriétés et d'individus). Une ontologie *O* est consistante s'il existe une interprétation *I* qui satisfait *O*. Une interprétation *I* satisfait une Ontologie *O* si elle satisfait tous les axiomes de *O* (l'interprétation de chaque axiome est vraie).

Les contraintes logiques de OWL-DL sont nombreuses. Dans notre approche, nous nous focalisons sur un sous-ensemble de conditions liées aux axiomes de subsomption de concepts et de propriétés, équivalence de concepts et de propriétés, disjonction de concepts, symétrie et transitivité de propriétés, cardinalités et domaine et co-domaine d'une propriété.

Les domaines et co-domaines sont exprimés par des axiomes spécifiant les conditions nécessaires pour qu'une instance puisse être associée à une classe. Les assertions de domaine et co-domaine permettent d'appliquer des inférences sur les individus.

Exemple. Si on a : ObjectProperty(employer domain(Entreprise) range(Personne)) ; et on instancie la propriété *employer* : Individual(UniversitéParisXI value(employer Pierre)) ; alors, on peut inférer que *UniversitéParisXI* est une *Entreprise* et *Pierre* une Personne. L'instanciation des propriétés n'étant pas contrainte en OWL (hypothèse du monde ouvert), les instances *UniversitéParisXI* et *Pierre* peuvent ne pas être déjà assignées à des classes. Cependant, de telles inférences ne sont pas toujours très précises (*Pierre* peut être un Enseignant, un *PersonnelAdministratif*, …) et peuvent parfois causer des inconsistances (*UniversitéParisXI* est une Université, le concept *Université* peut être disjoint du concept *Entreprise*).

3.2 Proposition d'alternatives de résolution d'inconsistances

La résolution d'inconsistances permet de proposer à une ontologie donnée et un changement à appliquer, les alternatives potentielles de résolution des inconsistances causées par ce changement. Ces alternatives représentent des opérations de changements additionnels à appliquer pour maintenir l'ontologie dans un état consistant.

Pour résoudre des inconsistances logiques, Haase et Stojanovic (2005) se basent sur la détermination et suppression des axiomes causant ces inconsistances selon deux approches : l'une générant le nombre minimal de changements à appliquer pour obtenir la sous-ontologie consistante maximale et l'autre permettant de localiser les inconsistances c'est-à-dire de retrouver la sous-ontologie inconsistante minimale. Les axiomes à supprimer sont ensuite présentés à l'utilisateur pour qu'il puisse décider et contrôler l'évolution de l'ontologie.

Dans notre approche, tenant compte du principe de « continuité ontologique » qui stipule qu'une connaissance existante ne peut être infirmée, nous tendons à minimiser les alternatives de suppression d'axiomes en proposant des alternatives de fusion de concepts, de division de concepts, de généralisation, de redistribution d'instances, etc. Ces alternatives sont proposées pour satisfaire un premier noyau de conditions de consistance logique (voir section 3.1.3). Cependant, si une suppression est nécessaire (elle peut d'ailleurs faire l'objet du changement lui-même et non seulement d'une alternative de résolution), elle est appliquée tout en vérifiant son impact sur l'ontologie. Certaines connaissances peuvent en effet, être réfutées à un moment donné, notamment lorsque l'ontologie est en cours de construction.

Exemple. Si *I* une instance du concept *C1*, sous-concept du concept *C*, est assignée au concept *C2*, sous-concept de *C* et disjoint de *C1*, au lieu de supprimer l'axiome de disjonction ou l'axiome d'instanciation de *I* à *C1*, l'inconsistance de disjonction peut être résolue par les alternatives suivantes :

- L'instance *I* est redistribuée et assignée au concept *C* plutôt qu'à ces sous-concepts, ainsi la sémantique de l'instance *I* est gardée même si elle moins précise et l'axiome de disjonction reste vrai ;
- Un nouveau sous-concept *C*3 est ajouté au concept *C*, l'instance *I* lui sera assignée, il représentera les sous-concepts de *C* auxquels l'axiome de disjonction n'est pas appliqué sans que ce dernier ne soit supprimé.

3.3 Evaluation des alternatives de résolution

Plutôt que de présenter les différentes alternatives de résolution à l'ingénieur d'ontologie, nous proposons de guider le choix de l'alternative à appliquer en évaluant l'impact de chacune des alternatives sur la qualité de l'ontologie. Ainsi, la validation des changements est optimisée par l'emploi de techniques d'évaluation permettant de guider la résolution des inconsistances et de minimiser la dépendance à l'utilisateur.

L'impact des différentes résolutions proposées sur la qualité de l'ontologie est évalué sur la base d'un modèle hiérarchique d'évaluation qui tient compte de la structure et de l'usage de l'ontologie à travers plusieurs critères et métriques de qualité (voir section 4). L'alternative qui préserve la qualité de l'ontologie, peut être automatiquement choisie et les changements directement validés et appliqués (phase suivante). L'expert n'interviendra que si toutes les alternatives ont un impact négatif sur la qualité. Il sera guidé dans sa décision, par l'indication de l'alternative la moins coûteuse pour la qualité.

Une autre différence s'ajoute en comparaison avec les travaux de Haase et Stojanovic (2005) : l'emploi de techniques d'évaluation de qualité dans la résolution des inconsistances à travers un modèle d'évaluation basé sur des critères de qualité quantifiables. Dans (Haase et Stojanovic 2005), seuls quelques critères de qualité non-quantifiables, ont été évoqués sous forme de conditions génériques de consistance, définies par l'ingénieur pour assurer une bonne modélisation de l'ontologie tels que les méta-propriétés de la méthodologie OntoClean représentant les notions de rigidité, identité, unité et dépendance (Guarino et Welty, 2002). Par ailleurs, pour minimiser l'impact de suppression d'axiomes sur les ontologies, les auteurs se réfèrent à un ensemble de besoins spécifiques pouvant être définis par l'utilisateur (tels que la minimisation du nombre d'axiomes à supprimer, ou la définition d'un degré de pertinence pour chaque axiome pour guider le choix des axiomes à supprimer). Dans notre approche, de tels besoins peuvent toujours être considérés mais en amont, nous avons défini une couche d'évaluation de l'impact sur la qualité qui tient compte de critères quantifiables et indépendants de l'utilisateur.

3.4 Application des changements

Cette phase correspond à la validation finale du changement nécessaire et de ses changements dérivés tout en assurant la traçabilité des modifications appliquées à l'ontologie. L'analyse des changements et les résultats de maintenance de la consistance de l'ontologie sont appliqués à une version temporaire de l'ontologie qui peut être abandonnée si les changements sont finalement annulés, l'ontologie initiale sera alors préservée. Dans le cas contraire, une nouvelle version d'ontologie est définie et l'historique des changements est sauvegardé.

Notons que l'approche est centrée sur la validation des changements (et non sur tout le processus d'évolution), elle part de l'analyse d'un changement formellement spécifié et vise à l'appliquer tout en assurant la consistance et la qualité de l'ontologie. Par ailleurs, cette

approche est principalement adaptée à un processus d'enrichissement d'une ontologie indépendante et ne tient pas compte dans sa version actuelle, de la dimension distribuée de l'environnement de l'ontologie.

4 Modèle d'évaluation de la qualité d'une ontologie

Dans le contexte d'évolution d'ontologie, les techniques d'évaluation ont été employées pour identifier les besoins de changements utiles à l'ontologie (Haase et Sure, 2005). Les approches d'évaluation peuvent être classées en fonction du niveau d'évaluation : le niveau lexical (Maedche et Staab, 2002), le niveau taxonomique (Gangemi et al., 2005), (Brewster et al., 2004), le niveau usage (Porzel et Malaka, 2004), le niveau données (Patel et al., 2004) ou encore selon des approches multi-critères (Fox et al, 1998) (Burton-Jones et al, 2004) (Lozano-Tello et Gomez-Perez, 2004).

Pour évaluer l'impact des changements sur la qualité de l'ontologie, nous avons défini un modèle hiérarchique centré sur deux aspects principaux de qualité : la structure et l'usage de l'ontologie. Chaque aspect est décrit à travers un ensemble de critères et chaque critère est évalué par un ensemble de métrique (Djedidi et al., 2007). Certains critères peuvent être contradictoires : les changements qui augmentent la complexité de l'ontologie améliorent la cohésion de sa structure. Au début du processus de gestion des changements, l'utilisateur procède à la pondération des critères du modèle en attribuant un poids à chacun des critères en fonction de son importance par rapport au domaine modélisé et l'application utilisant l'ontologie.

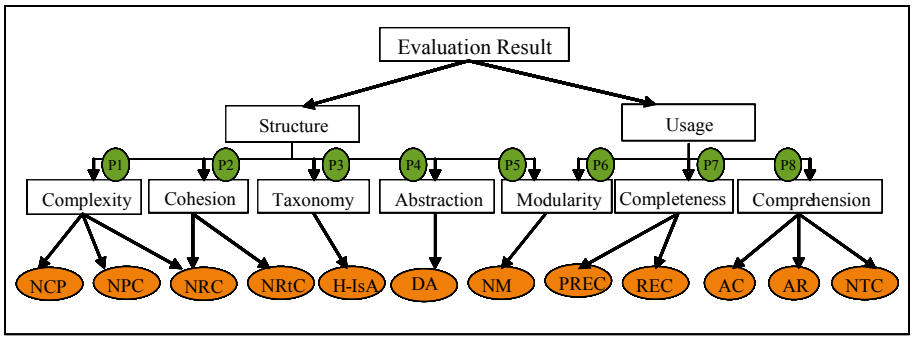


FIG. 1 – *Modèle d'évaluation de la qualité d'une ontologie.*

La première étape d'évaluation consiste à calculer les valeurs de chaque métrique puis d'évaluer chaque critère en comparant sa valeur d'évaluation avec la dernière évaluation effectuée (valeur avant l'application du changement, récupérée dans l'historique des modifications). Les critères d'évaluation considérés sont la complexité, la cohésion, la taxonomie, l'abstraction, la modularité, la complétude et la compréhension. Les métriques proposées sont NCP (nombre moyen de chemins de la racine vers un concept), NPC (le nombre moyen de propriétés par concept), NRC (le nombre moyen de relations par concept), NRtC (Le nombre de concepts racines : de hiérarchies), H-is (le ratio entre le nombre de relation is-a et les relations sémantiques), DA (la profondeur moyenne d'une hiérarchie), NM (le nombre de

modules disjoints formant l'ontologie), Prec (la précision), Rec (le rappel), AC (le pourcentage de concepts annotés), AR (le pourcentage de relation annotées) et NTC (Le nombre moyen de termes utilisés pour identifié un concept).

Notons que le modèle d'évaluation a été défini indépendamment du modèle de représentation de l'ontologie et ce pour qu'il soit applicable dans un contexte plus général et pour différents modèles d'ontologies.

5 Conclusion et perspectives

Dans cet article, nous proposons une approche de gestion de changements d'ontologie dont l'objectif est d'optimiser et automatiser la validation des changements tout en assurant la consistance et la qualité de l'ontologie évoluée.

La maintenance de la consistance est assurée à travers la proposition d'alternatives de résolution d'inconsistance. Un modèle de qualité est défini et appliqué pour évaluer l'impact des alternatives proposées sur la qualité et guider la résolution des inconsistances tout en minimisant la dépendance à l'utilisateur. L'alternative préservant la qualité est choisie automatiquement. Elle représente les changements complémentaires à appliquer pour maintenir la consistance de l'ontologie et sa qualité. Les changements peuvent alors être validés directement. Par contre, si toutes les alternatives ont un effet négatif sur la qualité, l'intervention de l'expert est sollicitée. Le résultat d'évaluation offre une information complémentaire à l'expertise de l'utilisateur qui le guidera dans sa décision en lui indiquant l'alternative la moins coûteuse et l'aidera aussi dans la justification des coûts des changements par rapport à leur pertinence.

Actuellement, nous travaillons sur le développement d'un système de gestion de changements d'ontologie OWL implémentant cette approche (un premier prototype a déjà été élaboré) et sur la validation de l'approche en l'appliquant pour enrichir une ontologie OWL de Pneumologie.

Références

Brank, J., Grobelnik, M. et Mladenic, D (2005*). A Survey of Ontology Evaluation Techniques*, Proceedings of the Conference on Data Mining and Data Warehouses SiKDD'05, Ljubljana, Slovenia.

Brewster, C., Alani, H., Sasmahapatra, S. et Wilks, Y(2004). *Data driven ontology evaluation*, Proceedings of International Conference on Language Resources and Evaluation, Lisbon.

Burton-Jones, A., Storey, V.C., Sugumaram, V. et Ahluwalia, P (2004*). A semiotic metrics suite for assessing the quality of ontologies*, Data and Knowledge Engineering.

Djedidi R., Abboute H., et Marie-Aude Aufaure (2007). *Evolution d'ontologie : validation des changements basée sur l'évaluation*, Proceedings 1ères journées francophones sur les ontologies, Sousse, Tunisie.

Eder J. et Wiggisser K (2007). *Change Detection in Ontologies Using DAG Comparison*, In CAiSE Proceedings, LNCS, pp. 21—35, Springer edition

Fox, M.S., Barbuceanu, M., Gruninger, M. et Lin, J (1998). *An organization ontology for enterprise modelling*, Simulating organizations, MIT Press.

Gangemi, A., Catenacci, C., Ciaramita, M., Gil, R.et Lehmann, J (2005). *Ontology evaluation: A review of methods and an integrated model for the quality diagnostic task*, Technical Report available at http://www.loa-cnr.it/Publications.html

Guarino N. et C. Welty (2002). *Evaluating Ontological Decisions with OntoClean*, In Communication of the ACM, 45(2), pp 61-65.

Haase P. et Stojanovic L (2005). *Consistent Evolution of OWL Ontologies*, In European Conference on Semantic Web ECSW'05.

Haase, P., Sure, Y (2005). *Incremental Ontology Evolution – Evaluation*, Institut AIFB, University of Karlsruhe.

Hartmann J., Spyns P., Giboin A., Maynard D., Cuel R., Suarez-Figueroa M.C., et Sure Y (2005). *Methods for ontology evaluation*, Knowledgeweb D1.2.3 deliverable.

Horrocks I. et Patel-Schneider P. F(2004). *Reducing OWL Entailment to Description Logic Satisfiability*. Journal of Web Semantics, 1(4), (2004)

Horrocks I., Patel-Schneider P. F., et van Harmelen F (2003). *From SHIQ and RDF to OWL: The Making of a Web Ontology Language* . Journal of Web Semantics, 1(1).

Klein, M (2004). *Change Management for Distributed Ontologies*. PhD Thesis, Dutch Graduate School for Information and Knowledge Systems, SIKS Dissertation Series No. 2004-11.

Klein M., Fensel D., Kiryakov A, et Ognyanov D (2002). *Ontology versioning and change detection on the Web*, in 13th International Conference on Knowledge Engineering and Knowledge Management EKAW02.

Lozano-Tello, A. et Gomez-Perez, A (2004). *Ontometric: A method to choose the appropriate ontology*, Journal of Database Management, Vol. 15, N°2 (2004) 1-18.

Maedche, A., Stojanovic, L., Studer, R. et Volz, R (2002). *Managing multiple ontologies and ontology evolution in OntoLogging*. In Proceedings of the Conference on Intelligent Information Processing (IIP-2002), pp.51—63, Montreal, Canada.

Maedche, A. et Staab, A (2002). *Measuring similarity between ontologies*. Proceedings of the European Conference on Knowledge Acquisition and Management – EKAW Madrid, Spain, LNCS/LNAI 2473, Springer (2002) 251-263.

Plessers P, De Troyer O (2005). *Ontology Change Detection using a Version Log*, In Proceedings of the International Semantic Web Conference ISWC'05.

Porzel, R. et Malaka, R (2004). *A task-based approach for ontology evaluation*, Proceedings of ECAI 2004 Workshop on Ontology Learning and Population.

Stojanovic L (2004). *Methods and Tools for Ontology Evolution*. Phd thesis, University of Karlsruhe.

Stojanovic, L., Maedche, A., Motik, B., Stojanovic, N (2002). *User-driven ontology evolution management*, In European Conference on Knowledge Engineering. and Management 285-300.

Stojanovic L., Maedche M., Stojanovic N. and Studer R (2003). *Ontology evolution as reconfiguration-design problem solving*, In K-CAP03 Proceedings, ACM pp.162—171.

Supekar, K (2006). *A peer-review approach for ontology evaluation*, Proceedings of the 8th International. Protégé Conference, Madrid, Spain.

Yang, Z., Zhan, D., Ye, C (2006). *Evaluation Metrics for Ontology Complexity and Evaluation Analysis*, IEEE International Conference on e-Business Engineering, ICEBE06.

Summary

Adapting ontologies to the modelized domain evolution and to their dynamic environment is an essential functional requirement to take into account in ontology engineering cycle. Several problems emanate from ontology evolution: capturing change requirement, change specification, change application, change traceability, change propagation to dependant artefacts, etc. In this paper, we propose an ontology change management approach. The purpose of the approach is to optimize and automate change validation while maintaining consistency and quality of the evolved ontology. Consistency is maintained by proposing inconsistency resolution alternatives. A quality model is defined and applied to evaluate the impact of the proposed alternatives on ontology quality and guide inconsistency and thus, to minimize user dependency.

Utilisation de taxonomies d'items pour la fouille de règles d'association

Jérôme David*, Régis Gras**,
Julien Blanchard**, Fabrice Guillet**, Henri Briand**

*INRIA Rhône-Alpes - Equipe EXMO
jerome.david-at-inrialpes.fr
**Polytech'Nantes - Equipe COD - LINA FRE CNRS 2729
{julien.blanchard,fabrice.guillet,henri.briand}@univ-nantes.fr

Résumé. A partir de l'approche des règles d'association généralisée et des approches de réduction de redondance, nous proposons dans ce papier d'étudier les atouts et les limites de ces deux approches. A partir de cette étude, nous proposons une méthode fusionnant et généralisant ces deux approches. Pour cela, nous donnons notre définition de la redondance basée à la fois sur l'utilisation de taxonomies entre items (règles d'association généralisées) et sur l'inclusion entre itemsets (règles min-max). Nous introduisons également un principe permettant de comparer les valeurs de qualité attendue et observée d'une règle redondante afin de déterminer si elle est potentiellement intéressante.

1 Introduction

En ECD, la fouille de règles d'association (Agrawal et al., 1993) est une technique populaire produisant des connaissances de la forme « Si *prémisse* alors *conclusion* », notées *prémisse* → *conclusion*. Dans une règle d'association, la prémisse et la conclusion sont des propositions portant sur les attributs d'une table. Une règle *prémisse* → *conclusion* représente une tendance implicative entre l'ensemble des enregistrements vérifiant la prémisse vers l'ensemble des enregistrements vérifiant la conclusion.

Un des principaux atouts des règles d'association est qu'elles représentent des connaissances de manière simple et explicite. Cependant, une de leurs principales limites concerne la quantité de règles générées par les algorithmes tels que Apriori (Agrawal et al., 1993; Agrawal et Srikant, 1994) qui croît de manière exponentielle en fonction du nombre d'attributs considérés (Blanchard, 2005). Afin de faire face aux quantités impressionnantes de règles générées, il est nécessaire de filtrer les ensembles de règles afin de ne retenir que les plus intéressantes (selon certains points de vue). Une première approche consiste à utiliser des mesures d'intérêt afin de vérifier la qualité implicative des règles et d'étudier de nombreuses caractéristiques telles que la nouveauté, la significativité, la surprise, la non-trivialité, etc. (Ceglar et Roddick, 2006). Une seconde approche s'appuie sur des contraintes qui peuvent être appliquées sur les items ou itemsets.

Parmi cette dernière approche, de nombreuses méthodes ont été exposées dont celles traitant de la réduction de la redondance (Lehn, 2000), (Zaki, 2000), (Pasquier et al., 2005) et des

règles d'association généralisées (Srikant et Agrawal, 1995), (Srikant et Agrawal, 1997), (Han et Fu, 1999).

L'objectif de ce papier est d'unir les approches basées sur les règles d'association généralisées et les principes de réduction de redondance afin de proposer une méthode générale de fouille de règles non-redondantes à partir de bases de données où les attributs peuvent être organisés en taxonomies.

2 Rappels, définitions et notations

Nous considérons un ensemble E composé de n individus décrits par un ensemble I de variables booléennes (items) au moyen de la relation binaire $\delta \subseteq I \times E$. L'ensemble des individus associés à un item $i \in I$ est noté $\delta(i)$. Par extension, si l'on considère un ensemble $a \subseteq I$ de variables booléennes, appelé itemset, l'ensemble des individus $x \in E$ associés par δ à l'itemset a est noté $A = \bigcap_{i \in a} \delta(i)$. La cardinalité de A est notée n_a.

Définition. Une règle d'association est un couple de variables noté $a \rightarrow b$ où a et b sont des itemsets disjoints, appelés respectivement prémisse et conclusion.

Une règle d'association représente une tendance implicative entre l'ensemble des individus associés à a vers l'ensemble des individus associés à b. Elle peut se lire de la manière suivante : « Si un individu vérifie a alors il vérifie généralement b ». Les exemples d'une règle, notés $A \cap B$ ($|A \cap B| = n_{ab}$), représentent les individus associés à la fois à a et b. Les contre-exemples de la règle, notés $A \cap \overline{B}$ ($|A \cap \overline{B}| = n_{a\bar{b}}$), sont les individus associés à a mais pas à b. Une règle est d'autant meilleure qu'elle admet beaucoup d'exemples et peu de contre-exemples. Une règle n'ayant que des exemples est une implication logique.

2.1 Support et confiance

Deux mesures sont utilisées usuellement lors de la fouille de règles d'association, il s'agit du support et de la confiance. Le support représente la fréquence d'individus respectant la règle dans l'ensemble étudié (Agrawal et al., 1993) :

$$support(a \rightarrow b) = \frac{n_{ab}}{n}$$

La confiance permet quant à elle de vérifier la validité de la règle. Elle représente la proportion d'individus associés à la prémisse qui sont également associés à la conclusion (Agrawal et al., 1993) :

$$confiance(a \rightarrow b) = \frac{n_{ab}}{n_a}$$

2.2 Notion de généralité d'un itemset

Lorsque que l'on considère la présence d'une ou plusieurs taxonomies sur les items d'une base de données, la relation de généralisation/spécialisation entre ensembles de variables booléennes (ou itemsets) intervient à deux niveaux distincts. Premièrement, cette relation de généralisation/spécialisation peut être naturellement issue de l'information taxonomique. Par

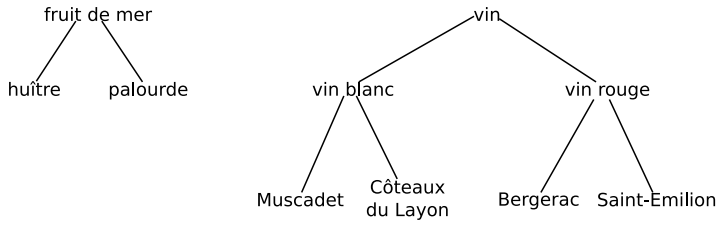


FIG. 1 – *Exemple de taxonomies sur des items issus d'une base de données*

exemple en s'appuyant sur la figure 1, la prémisse de la règle "fruits de mer → vin blanc" est plus générale que celle de la règle "huîtres → vin blanc", puisque l'item *fruits de mer* est plus général que l'item *huîtres*. Cette relation de généralisation/spécialisation peut être également présente sous forme d'une inclusion entre les itemsets constituant les prémisses et conclusions. Par exemple, la conclusion de la règle "*huîtres → vin blanc*" sera plus générale que celle de la règle "*huîtres → vin blanc ∧ citron*" puisque $\{vin\ blanc\} \subset \{vin\ blanc, citron\}$.

A partir des constats précédents, nous utilisons la définition suivante de la notion de généralité entre itemsets :

Définition. Un item $i \in I$ est plus général qu'un item $j \in I$ sur E, noté $j \leq i$, si et seulement si $\delta(j) \subseteq \delta(i)$. Par extension, un itemset $a \subseteq I$ est plus général qu'un itemset $b \subseteq I$ sur E, noté $b \leq a$, si et seulement si $B \subseteq A$.

Une des propriétés découlant de cette notion de généralité entre itemsets est que si un itemset b est plus général qu'un itemset a (noté $a \leq b$) alors la règle $a \rightarrow b$ est une implication logique et a donc une confiance $confiance(a \rightarrow b) = 1$.

3 Approches de réduction des règles basées sur la généralisation

3.1 Règles d'association généralisées

La première approche traitant de la généralité entre item est celle des règles d'association généralisées, introduite par Srikant et Agrawal (1995). La fouille de règles d'association généralisées prend en considération la présence de taxonomies sur les items et vise à extraire des règles générales en terme de support.

Définition. Une règle $a \rightarrow b$ sera plus générale en terme de support qu'une règle $a' \rightarrow b'$ si $a' \leq a$ et $b' \leq b$.

Etant donnée la présence possible d'une relation d'ordre sur l'ensemble des items du jeu de données, la notion de règle d'association est généralisée pour prendre en compte la relation d'ordre :

Définition. Une règle d'association généralisée $a \rightarrow b$ est une règle d'association pour laquelle il n'existe pas d'item de la conclusion qui soit plus général qu'un item de la prémisse. Formellement, $a \rightarrow b$ sera une règle d'association généralisée si l'assertion $\forall a_i \in a,\ \nexists b_j \in b,\ a_i \leq b_j$ est vérifiée (Srikant et Agrawal, 1995).

Exemple. La règle *huître* $\wedge\ muscadet \rightarrow$ *vin blanc* n'est pas une règle généralisée puisque muscadet est plus spécifique que vin blanc. Son interprétation « si l'on achète des huîtres et du muscadet, alors on achète du vin blanc » est triviale (le muscadet est du vin blanc). Par conséquent, cette seconde règle est inutile vis-à-vis de la première. Par contre, la règle *huître* $\wedge$ *vin blanc* $\rightarrow muscadet$ est une règle généralisée. Elle peut être interprétée « si l'on achète des huîtres et du vin blanc alors ce vin blanc sera sûrement du muscadet ».

Dans l'approche proposée par Srikant et Agrawal (1995), la notion de redondance est définie à partir de la notion de généralité (en terme de support). Pour cela, il est proposé de sélectionner les règles qui sont soit les plus générales, ou ayant des valeurs de confiance et de support au moins plus de R fois supérieures aux valeurs que l'on aurait eues en suivant une réduction des exemples et contres-exemples proportionnelle à celle des effectifs des prémisses et conclusion.

Pour cela, les notions d'ancêtre d'un itemset, d'ancêtre d'une règle et de support et confiance attendus sont formalisées.

La notion d'ancêtre d'un itemset z est définie comme étant un itemset plus général que z ayant le même nombre d'items :

Définition. Un itemset $\widehat{z}$ est un ancêtre de l'itemset z si les contraintes suivantes sont vérifiées :

- $|z| = |\widehat{z}|$: les deux ensembles z et $\widehat{z}$ contiennent le même nombre d'items.
- $\widehat{z} \neq z$: les deux ensembles z et $\widehat{z}$ sont différents.
- $\forall z_i \in \widehat{z} - z,\ \exists z_j \in z,\ z_j \leq z_i$: chaque item de $\widehat{z}$ possède un item plus spécifique ou identique dans z.

Exemple. Les itemsets $\{$*huître*, *vin blanc*$\}$ et $\{$*fruit de mer*, *vin blanc*$\}$ sont des ancêtres de l'itemset $\{$*huître*, $muscadet\}$. Cependant, l'itemset $\{$*vin blanc*$\}$, même s'il est plus général que $\{$*huître*, $muscadet\}$, n'est pas un de ses ancêtres, puisqu'ils ne contiennent pas les mêmes nombres d'éléments.

A partir de la définition de la notion d'ancêtre d'un itemset, la notion d'ancêtre d'une règle d'association généralisée peut être définie :

Définition. Les ancêtres d'une règle $a \rightarrow b$ sont les règles de la forme $\widehat{a} \rightarrow \widehat{b}$, $\widehat{a} \rightarrow b$, $a \rightarrow \widehat{b}$. Une règle $\widehat{a} \rightarrow \widehat{b}$ est un ancêtre proche de la règle $a \rightarrow b$ si il n'existe pas de règle $a' \rightarrow b'$ telle que $a' \rightarrow b'$ est un ancêtre de $a \rightarrow b$ et $\widehat{a} \rightarrow \widehat{b}$ est un ancêtre de $a' \rightarrow b'$.

Exemple. La règle *fruit de mer* $\rightarrow$ *vin* est un ancêtre de la règle *huître* $\rightarrow$ *muscadet* (car *huître* $\leq$ *fruit de mer* et *muscadet* $\leq$ *vin*). Cependant, elle n'est pas son ancêtre proche car la règle *fruit de mer* $\rightarrow$ *vin blanc*, dont *fruit de mer* $\rightarrow$ *vin* est l'ancêtre, est également ancêtre

de *huître* $\rightarrow$ *muscadet*. Par contre, la règle *fruit de mer* $\rightarrow$ *vin blanc* est l'ancêtre proche de *huître* $\rightarrow$ *muscadet*.

La valeur attendue du support d'une règle $a \rightarrow b$ pour un ancêtre $\widehat{a} \rightarrow \widehat{b}$ est :

$$E_{\widehat{a}\cup\widehat{b}}(a \cup b) = \frac{\prod_{x \in a\cup b - \widehat{a}\cup\widehat{b}} support(x)}{\prod_{\widehat{x} \in \widehat{a}\cup\widehat{b} - a\cup b} support(\widehat{x})} \times support(\widehat{a} \cup \widehat{b}) \quad (1)$$

Exemple. Considérons les supports suivants : $support($*fruit de mer*$) = 0,6$, $support($*huître*$) = 0,3$, $support($*vin blanc*$) = 0,8$, $support($*muscadet*$) = 0,4$, $support(\{$*fruit de mer*, *vin blanc*$\}) = 0,4$ et $support(\{$*huître*$, muscadet\}) = 0,3$, et les deux règles *fruit de mer* $\rightarrow$ *vin blanc* et *huître* $\rightarrow$ *muscadet*. Le support attendu de l'itemset $\{$*huître*$, muscadet\}$ par rapport à l'itemset $\{$*fruit de mer*, *vin blanc*$\}$ est $E_{\{\textit{fruit de mer,vin blanc}\}}(\{$*huître*$, muscadet\}) = \frac{support(\textit{huître}) \times support(muscadet)}{support(\textit{fruit de mer}) \times support(\textit{vin blanc})} \times support(\{$*fruit de mer*, *vin blanc*$\}) = 0,1$.

De manière similaire, la valeur attendue de confiance d'une règle $a \rightarrow b$ pour un ancêtre $\widehat{a} \rightarrow \widehat{b}$ est :

$$E_{\widehat{a}\rightarrow\widehat{b}}(a \rightarrow b) = \frac{\prod_{b_i \in b - \widehat{b}} support(b_i)}{\prod_{\widehat{b_i} \in \widehat{b} - b} support(\widehat{b_i})} \times confiance(\widehat{a} \rightarrow \widehat{b})$$

Exemple. A partir de l'exemple précédent, la confiance attendue de la règle *huître* $\rightarrow muscadet$ par rapport à la règle *fruit de mer* $\rightarrow$ *vin blanc* est $E_{\textit{fruit de mer}\rightarrow muscadet}($*huître* $\rightarrow muscadet) = \frac{support(\textit{muscadet})}{support(\textit{vin blanc})} \times confiance($*fruit de mer* $\rightarrow$ *vin blanc*$) = 1/3$ ($confiance($*fruit de mer* $\rightarrow$ *vin blanc*$) = support(\{$*fruit de mer*, *vin blanc*$\})/support($*fruit de mer*$) = 2/3$).

3.1.1 Règles R-intéressantes (Srikant et Agrawal, 1995)

Une règle $a \rightarrow b$ sera R-intéressante vis-à-vis d'un de ses ancêtres $\widehat{a} \rightarrow \widehat{b}$ si au moins l'une des deux conditions suivantes est respectée :

- $support(a \rightarrow b) = R \times E_{\widehat{a}\cup\widehat{b}}(a \cup b)$
- $confiance(a \rightarrow b) = R \times E_{\widehat{a}\rightarrow\widehat{b}}(a \rightarrow b)$

Intuitivement, ces deux critères signifient qu'une règle $a \rightarrow b$ sera R-intéressante (vis-à-vis d'un ancêtre $\widehat{a} \rightarrow \widehat{b}$) si les valeurs de support ou de confiance de $a \rightarrow b$ est R fois celles attendues étant donné $\widehat{a} \rightarrow \widehat{b}$.

Etant donné un ensemble de règles S et un seuil minimum d'intérêt R, une règle $a \rightarrow b$ sera **intéressante** dans S si elle n'a pas d'ancêtre ou si elle est **R-intéressante** vis-à-vis de ses ancêtres proches. Une règle $a \rightarrow b$ sera partiellement intéressante dans S si elle n'a pas d'ancêtre ou si elle est R-intéressante pour au moins un de ses ancêtres proches.

3.1.2 Règles redondantes et inutiles (Han et Fu, 1999)

A partir du modèle des règles d'association généralisées, Han et Fu (1999) ont proposé une autre définition de la redondance. Cette définition de la redondance n'est pas basée sur un critère de gain de confiance. Dans ce cas, une règle sera redondante par rapport à un de ses

ancêtres si elle n'a pas une valeur de confiance très différente (au niveau d'un seuil α). Une règle $a \rightarrow b$ sera redondante vis-à-vis d'un de ses ancêtres $\widehat{a} \rightarrow \widehat{b}$ si :

$$E_{\widehat{a}\rightarrow\widehat{b}}(a \rightarrow b) - \alpha \leq confiance(a \rightarrow b) \leq E_{\widehat{a}\rightarrow\widehat{b}}(a \rightarrow b) + \alpha$$

Dans (Han et Fu, 1999), il est également dit qu'une règle ayant une prémisse plus spécifique qu'une autre est inutile si elle n'a pas confiance très différente (au niveau d'un seuil β). Formellement, une règle $c \rightarrow b$ est inutile s'il existe une règle $a \rightarrow b$ vérifiant les deux conditions suivantes :

- $c \subset a$
- $confiance(a \rightarrow b) - \beta \leq confiance(c \rightarrow b) \leq confiance(a \rightarrow b) + \beta$

3.1.3 Avantages et limites

Les deux approches présentées considèrent qu'une règle $a' \rightarrow b'$ n'est pas intéressante (resp. redondante) par rapport à une règle $a \rightarrow b$, avec $a' \leq a$ et $b' \leq b$, si elle n'apporte pas un certain gain de qualité (resp. si elle n'a pas une valeur de confiance très différente). L'avantage principal de ce type d'approche réside dans la comparaison des valeurs de qualité (limitées au support et à la confiance) obtenues par une règle par rapport à celles attendues (sous hypothèse de réduction indépendante et proportionnelle des cardinalités des ensembles d'individus respectant la prémisse et de ceux respectant la conclusion).

La première limite de cette approche concerne une prise en compte partielle de la généralisaton/spécialisation d'itemsets. En effet, la définition d'ancêtre d'un itemset se restreint à la généralisation d'une ou plusieurs de leurs items et ne prend pas en compte l'inclusion de leurs ensembles d'items.

La deuxième limite concerne l'utilisation de la définition d'ancêtre d'une règle pour réduire la quantité de règles extraites. Par exemple, si l'on considère les règles *huître* $\rightarrow muscadet$ et *huître* $\rightarrow$ *vin blanc*, la règle *huître* $\rightarrow muscadet$, qui a un ancêtre *huître* $\rightarrow$ *vin blanc*, sera considérée comme redondante si elle ne répond pas aux critères de qualité utilisés. Cependant, en toute logique, on devrait considérer l'inverse car *huître* $\rightarrow$ *vin blanc* peut être déduite à partir de *huître* $\rightarrow muscadet$ étant donné que le muscadet est un vin blanc. De plus, en utilisant ce type d'approche on ne peut pas détecter que *huître* $\rightarrow$ *vin blanc* est redondante par rapport à *fruit de mer* $\rightarrow muscadet$ car aucune des deux règles n'est un ancêtre de l'autre.

3.2 Réduction des redondances

Pasquier et al. (2005) proposent un autre moyen de réduire la redondance en s'appuyant sur les treillis de galois (Barbut et Monjardet, 1970) pour l'extraction des règles. Leur définition de la redondance n'est pas la même que celle de Han et Fu (1999), et le principe suivi n'est pas celui des règles généralisées (Srikant et Agrawal, 1995). En effet, dans ce cas, la présence de taxonomies entre items n'est pas considérée et la définition de la redondance se base sur des propriétés logiques et non sur la notion de généralité en terme de support.

Pour illustrer cette notion de redondance, Pasquier et al. (2005) prennent un exemple de règles extraites d'une base de données mycologique bien connue : la base de données MUSHROOMS. Sur cet exemple, présenté table 1, 9 règles possèdent les mêmes valeurs de confiance et de support et ont toutes l'item « lames libres » dans la prémisse. Parmi cet ensemble de

1- lames libres → comestible
2- lames libres → comestible ∧ voile partiel
3- lames libres → comestible ∧ voile blanc
4- lames libres → comestible ∧ voile partiel ∧ voile blanc
5- lames libres ∧ voile partiel → comestible
6- lames libres ∧ voile partiel → comestible ∧ voile blanc
7- lames libres ∧ voile blanc → comestible
8- lames libres ∧ voile blanc → comestible ∧ voile partiel
9- lames libres ∧ voile partiel ∧ voile blanc → comestible

TAB. 1 – *Règles redondantes dans une base de données mycologique*

règles, seulement la numéro 4 est intéressante car les autres lui sont redondantes. En effet, la règle numéro 4 a la même confiance et le même support que les autres règles, et de plus, elle possède la prémisse minimale et la conclusion maximale parmi les 9 règles.

A partir de cette notion de redondance, une règle est appelée **règle min-max**, parmi un ensemble de règles S, si elle est non-redondante (c.-à-d. qu'il n'existe pas de règles dans S ayant une prémisse incluse dans la sienne et de conclusion dont la sienne est un sous-ensemble) ou si sa confiance et son support sont égaux à ses règles redondantes.

Définition. Une règle d'association $a \rightarrow b$ de S est une **règle min-max** si il n'existe pas de règle $a' \rightarrow b'$ ($\neq a \rightarrow b$) appartenant à S vérifiant toutes les conditions suivantes :
- $a \subseteq a'$ et $b' \subseteq b$
- $confiance(a \rightarrow b) = confiance(a' \rightarrow b')$
- $support(a \rightarrow b) = support(a' \rightarrow b')$

3.2.1 Avantages et limites

Cette approche a le mérite, par rapport à celle de Srikant et Agrawal (1995), de proposer une réduction des règles extraites sur la base d'une sémantique plus pertinente : « une règle r est redondante par rapport à une règle r' si r a une prémisse plus spécifique ou une conclusion plus générale que r' ». Cependant, elle ne prend pas en compte une éventuelle taxonomie (relation de généralisation/spécialisation) sur les items.

Une autre limite de cette approche concerne la définition de la règle min-max. Une règle r qui est redondante par rapport à une règle r' sera tout de même conservée si leurs valeurs de support ou de confiance diffèrent. Cependant, lorsque l'on spécialise la conclusion ou que l'on généralise la prémisse, il y a de grande chances que les règles r et r' aient des valeurs de qualité (support et confiance) différentes sans pour autant remettre en cause la redondance et l'inutilité de r par rapport à r'. Par exemple, considérons les règles $fumer \rightarrow cancer$, $fumer \wedge homme \rightarrow cancer$ et $fumer \wedge femme \rightarrow cancer$ ayant toutes les trois une confiance de 50%. Les règles $fumer \wedge homme \rightarrow cancer$ et $fumer \wedge femme \rightarrow cancer$ auront un support de moitié par rapport à $fumer \rightarrow cancer$ (en considérant qu'il y a autant de femmes que d'hommes). En suivant le critère min-max, les trois règles sont conservées (leurs valeurs de support diffèrent) alors que les deux dernières sont redondantes par rapport

à $fumer \rightarrow cancer$ et n'apportent aucune information. Si l'on considère en plus la règle $fumer \rightarrow cancer \wedge tousser$ avec une confiance de 45%, les deux règles $fumer \rightarrow cancer \wedge tousser$ et $fumer \rightarrow cancer$ seront conservées (parce qu'elles ont des confiances différentes) alors que la première règle permet de déduire la seconde et la seconde (même si elle a une confiance supérieure) n'apporte pas vraiment d'information supplémentaire.

4 Combinaison des deux approches

Nous avons étudié deux principes de réduction du nombre de règles. Chaque approche présente des avantages pour une fouille de règles non-redondante s'appuyant sur des taxonomies d'items : la première permet de prendre en compte des taxonomies sur les items ; la deuxième utilise un critère de réduction de redondance plus adapté que la première approche.

A partir des deux approches, nous dirons qu'une règle r est redondante par rapport à une règle r' si r' a une prémisse plus générale et une conclusion plus spécifique que r. Cette définition généralise à la fois l'inclusion entre itemset et l'utilisation d'une taxonomie sur les items en utilisant la définition 2.2. Ainsi, formellement, $a \rightarrow b$ est redondante par rapport à une règle $a' \rightarrow b'$ ($\neq a \rightarrow b$) si $a \leq a'$ et $b' \leq b$.

Exemple. En reprenant la taxonomie sur les items présentée figure 1 et un autre item « *citron* », considérons les règles *huître* $\rightarrow$ *vin blanc* et *fruit de mer* $\rightarrow$ *muscadet* $\wedge$ *citron*. La première règle est redondante par rapport à la seconde, puisque d'une part, *huître* $\leq$ *fruit de mer*, et d'autre part, $\{$*muscadet*, *citron*$\} \leq$ *vin blanc*.

Même si une règle est redondante (selon notre définition) par rapport à une autre, elle peut apporter toutefois une information supplémentaire. En effet, elle peut avoir, proportionnellement, un nombre de contre-exemples étonnamment plus faible (ou plus élevé) que celui de la règle pour laquelle elle est redondante. Par exemple, considérons la règle $fumer \rightarrow cancer$ (avec un support de 4% et une confiance de 50%) et une de ses règles redondantes $fumer \wedge boire \rightarrow cancer$ (avec un support de 2% et une confiance de 75%). A la vue de ces valeurs de confiance, nous pouvons aisément observer que le fait de boire et de fumer augmente le risque de cancer par rapport au fait de fumer uniquement ($+50\%$ de chances).

Afin de prendre en compte de telles situations, il nous semble nécessaire de définir un critère permettant de juger du gain (resp. de la perte) de qualité d'une règle redondante par rapport à la règle qui la généralise. En reprenant le principe de support attendu (Srikant et Agrawal, 1997), nous proposons de définir la cardinalité attendue d'un itemset a par rapport à son généralisant a' ($a \leq a'$). Ainsi, nous définissons la cardinalité attendue de A, notée $E_{a'}(n_a)$ (sous hypothèse d'indépendance entre A et A') par :

$$E_{a'}(n_a) = \frac{\prod_{x \in a - a'} n_x}{\prod_{x' \in a' - a} n_{x'}} \times n_{a'} \tag{2}$$

Ensuite, à partir des cardinalités attendues, nous pouvons calculer la valeur de qualité attendue pour une règle r redondante par rapport à r'. Soit $m(n_a, n_b, n_{ab}, n)$ une mesure d'intérêt de règles d'association. Soit $a \rightarrow b$ une règle et $a' \rightarrow b'$ ($\neq a \rightarrow b$) une de ses règles redondantes ($a \leq a'$ et $b' \leq b$). Nous dirons que la règle redondante $a \rightarrow b$ est inutile si :

$$m(a \rightarrow b) = m(n_a, n_b, n_{ab}, n) \simeq m(E_{a'}(n_a), E_{b'}(n_b), E_{a'b'}(n_{a'b'}), n) \tag{3}$$

Dans le cas de règles d'association généralisées, ce principe appliqué aux mesures de support et confiance donne les mêmes mesures de support et de confiance attendues proposées par Srikant et Agrawal (1997). Ce principe a cependant l'avantage de généraliser l'approche à :

- tout type de généralisation : soit en utilisant une (ou des) taxonomie(s) entre items, soit en se basant sur l'inclusion entre itemset.
- tout type de mesures : en nous basant sur les cardinalités attendues, cela permet d'appliquer le principe à n'importe quelle mesure de qualité.

5 Conclusion

A partir des approches de réduction du volume des règles extraites par les algorithmes de fouille, nous avons proposé une approche généralisant de réduction de la redondance. En nous basant sur les travaux menés d'un coté par Srikant et Agrawal (1997) sur les règles d'association généralisées et, d'un autre coté, par Pasquier et al. (2005) sur la réduction de la redondance (règles min-max), nous avons proposé une fusion de ces deux approches afin de pouvoir faire de la fouille de règles non-redondantes s'appuyant sur d'éventuelles taxonomies entre items. Nous avons, pour cela, donné une définition de la redondance prenant en compte d'une part la relation de généralisation/spécialisation entre items et d'autre part la relation d'inclusion entre itemsets. Nous avons également introduit un principe permettant de calculer la valeur de qualité attendue d'une règle redondante afin de la comparer avec la valeur de qualité réellement obtenue. Ce principe a le mérite d'être applicable aux mesures classiques telles que la confiance ou le support mais également utilisable avec n'importe quelle autre mesure.

Ce travail n'est encore qu'au stade de proposition, il faudrait tester ce principe sur des cas réels de bases de données et mettre au point les algorithmes nécessaires à la détection des redondances, en utilisant, par exemple, le calcul des couvertures minimales (Lehn, 2000).

Références

Agrawal, R., T. Imielinski, et A. Swami (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216. ACM Press.

Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, et C. Zaniolo (Eds.), *Proceedings of 20th International Conference on Very Large Data Bases (VLDB 94)*, pp. 487–499. Morgan Kaufmann.

Barbut, M. et B. Monjardet (1970). *Ordres et classifications : algÃÍbre et combinatoire*. Hachette.

Blanchard, J. (2005). *Un système de visualisation pour l'extraction, l'évaluation, et l'exploration interactives des règles d'association*. Ph. D. thesis, Université de Nantes.

Ceglar, A. et J. F. Roddick (2006). Association mining. *ACM Computing Surveys 38*(2), 5.

Han, J. et Y. Fu (1999). Mining multiple-level association rules in large databases. *IEEE Transactions on Knowledge and Data Engineering 11*(5), 798–805.

Lehn, R. (2000). *Un système interactif de visualisation et de fouille de règles pour l'extraction de connaissances dans les bases de données*. Thèse de doctorat, Université de Nantes.

Pasquier, N., R. Taouil, Y. Bastide, G. Stumme, et L. Lakhal (2005). Generating a condensed representation for association rules. *Journal of Intelligent Information Systems 24*(1), 29–60.

Srikant, R. et R. Agrawal (1995). Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB 95)*, pp. 407–419.

Srikant, R. et R. Agrawal (1997). Mining generalized association rules. *Future Generation Computer Systems 13*(2-3), 161–180.

Zaki, M. J. (2000). Generating non-redundant association rules. In *Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 00)*, New York, NY, USA, pp. 34–43. ACM.

Summary

In this paper, we present the assets and the limits of the generalized association rule model and of the approaches for reducing redundancy. Then, we propose to merge and generalize this two approaches. For this, we give a definition of redundancy which both take into account taxonomies between items (generalized association rules) and inclusion between itemsets (min-max rules). We also introduce a principle allowing to compare expected and observed quality values of a redundant rule in order to assess its potential interest.