Télé-Université du Québec

Gestion de l'évolution d'une ontologie : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie

Proposition de recherche doctorale en informatique cognitive
(DIC 9410)

Delia Codruta ROGOZAN

Directeurs de Recherche

Directeur : Gilbert PAQUETTE
Codirecteur : Richard HOTTE

Résumé

Si les travaux sur les ontologies, leur construction et leur utilisation, ont déjà une dizaine d'années, ceux sur l'évolution de l'ontologie sont bien plus récents et leur poursuite est indispensable pour que les ontologies servent toute application du Web sémantique. Nos objectifs de recherche se situent alors au cœur même des recherches actuelles sur les ontologies étant donné qu'un des plus importants aspects concerne la gestion consistante de l'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué sur le Web. Nous envisageons alors notre contribution à deux niveaux. Au niveau conceptuel nous proposons une méthodologie qui (1) unifie les approches méthodologiques de l'évolution de l'ontologie, actuellement presque inexistantes et (2) intègre des éléments méthodologiques novateurs, principalement en ce qui concerne la gestion des changements entre les versions de l'ontologie et l'opérationnalisation de l'ontologie, une fois évoluée. Au niveau technologique nous proposons deux systèmes informatiques pour supporter des éléments méthodologiques essentiels dans l'évolution de l'ontologie. Ces deux systèmes, OntoAnalyseur et UKIsModificateur, possèdent des fonctionnalités innovantes, capables d'effectuer une analyse sémantique du processus d'évolution et d'assurer un référencement sémantique évolutif des objets du Web sémantique.

TABLE DES MATIÈRES

Introduction

L'organisation de notre proposition doctorale est la suivante. Dans le **chapitre I,** nous discutons des éléments de la problématique. Nous les abordons par la présentation de nos travaux antérieurs qui ont mis en évidence la nécessité de faire évoluer l'ontologie utilisée comme armature de construction d'un système de téléapprentissage par projet. Ensuite nous exposons le contexte général de notre recherche, le Web sémantique, ainsi que le contexte particulier, le Web sémantique comme infrastructure ontologique pour supporter les systèmes de téléapprentissage. Ceci nous conduit vers l'identification de notre problématique de recherche, à savoir **comment supporter l'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué sur le Web sémantique**. Après nous être interrogés sur les problèmes soulevés par cette évolution, nous concluons ce chapitre en présentant nos objectifs de recherche, leur originalité, ainsi que la composante cognitive et informatique de notre projet de recherche.

Dans le **chapitre II**, nous faisons l'état de la question sur les approches méthodologiques d'évolution de l'ontologie tout en démontrant qu'elles n'arrivent pas à constituer une méthodologie complète. Nous énonçons alors notre *objectif conceptuel du projet de recherche*, soit l'intégration de ces approches dans une méthodologie unifiée décrivant un processus complet d'évolution d'une ontologie. Nous présentons ensuite une ébauche de cette méthodologie et concluons par une analyse des outils ontologiques, le but étant ici de mettre en évidence les besoins technologiques ressentis pour l'évolution. La conclusion qui se dégage de cette analyse est qu'actuellement peu d'outils offrent des fonctionnalités pour supporter l'évolution de l'ontologie, notamment pour supporter l'analyse des changements ontologiques ou pour supporter l'opérationnalisation consistante de la nouvelle version de l'ontologie évolutive.

Ainsi, dans le **chapitre III** nous explicitons les *objectifs spécifiques de la recherche*. Le premier objectif est de développer une méthode d'analyse de la relation entre les versions ontologiques composée de : (1) l'identification des changements exécutés pour passer d'une version de l'ontologie à une autre, (2) la découverte de la relation sémantique entre des entités ontologiques appartenant aux différentes versions, et (3) l'analyse de la compatibilité entre les versions. Le deuxième objectif est de développer une méthode pour assurer un référencement sémantique évolutif réalisé par la modification et la mise à jour des liens de références des objets référencés par une ontologie évolutive. Pour chaque objectif nous faisons l'état de la question à partir des travaux connexes et nous mettons en évidence notre contribution originale.

La méthodologie retenue pour atteindre les objectifs de notre recherche, présentée d'ailleurs dans le **chapitre IV**, repose sur la recherche-développement et utilise la notion de conception itérative empruntée au génie logiciel. Après avoir détaillé chaque itération de la démarche de recherche, nous concluons avec un retour sur la problématique, les objectifs et les contributions envisagées, ce qui constitue d'ailleurs la conclusion finale de notre proposition de recherche doctorale.

Chapitre I : Éléments de la problématique

1. Évolution de l'ontologie pour un système d'apprentissage par projet à distance

Avant d'exposer la problématique et les objectifs de notre projet de recherche, il nous semble adéquat de présenter ce qui constitue la genèse de cette problématique. Ainsi, nos premières recherches ont porté sur la conception d'un système d'apprentissage pour supporter des apprenants universitaires dans la réalisation, à distance, des projets d'apprentissage collectifs. La pédagogie par projet joue un rôle important dans la formation en ligne (Bellamy, Grant, Cooper, Borovoy, et Adams, 1994; Donzelini, 1999; Fayolle, Jacquet, et Fouquet, 2001; Fung, 1996; Morgan, 1994; Thomas, 2000). Toutefois, la conception des systèmes d'apprentissage fondés sur cette pédagogie pose aux concepteurs des questions particulières que nous avons analysées dans leurs aspects collectif, dynamique et informationnel (Rogozan, Abdulrab, et Hotte, 2001). Nous avons proposé alors dans (Rogozan, 2003b; Rogozan, Hotte et Abdulrab, 2002) la modélisation d'un *Espace technologique de Réalisation des Projets d'Apprentissage à Distance* (ERPAD). ERPAD est composé d'un *espace dynamique de collaboration (EDC),* créé par les groupes d'apprenants qui peuvent interagir ponctuellement grâce à leurs actions communes, dirigées par des buts proches (Leontiev, 1984), et d'un *espace dynamique de documentation* (EDD), créé par la capitalisation et l'utilisation des produits de projets.

ERPAD est un *Système d'Apprentissage*[1] (Paquette, 2002a) qui supporte les apprenants distants dans l'atteinte d'un certain niveau de compétence par la réalisation des divers projets d'apprentissage. ERPAD offre deux services : (1) mettre en contact des groupes d'apprenants en identifiant leurs actions communes et (2) fournir aux groupes des ressources techniques et documentaires ciblées sur leurs actions. Pour cela, le système d'apprentissage ERPAD utilise une ontologie du domaine d'apprentissage[2] et une ontologie de tâche[3] et il demande aux apprenants de référencer leurs actions ainsi que les ressources produites pendant la démarche de projet en fonction des concepts des ontologies (figure 1). Le référencement des actions et des ressources à un même espace ontologique constitue un processus essentiel pour la gestion de l'EDD et son enrichissement ainsi que pour la construction émergente de l'EDC (Rogozan, Paquette, et Hotte, 2003). De plus, ce référencement facilite la compréhension des apprentissages qui se développent dans

[1] Paquette (2002a) définit le Système d'Apprentissage (SA) comme un concept intégrateur qui regroupe trois composantes principales : (1) le devis du SA, qui définit principalement le modèle de connaissances et le modèle pédagogique, (2) les matériels pédagogiques et les documents réalisés à partir de ce devis et (3) les environnements qui soutiennent la diffusion des apprentissages visées par le SA.

[2] L'ontologie du domaine d'apprentissage réfère au modèle de connaissances explicitant les connaissances à acquérir par les apprenants et leurs relations.

[3] L'ontologie de tâche réfère au modèle pédagogique explicitant les activités à réaliser par les apprenants afin d'acquérir les connaissances visées et leur relations.

l'ERPAD, les références sémantiques étant des moyens par lesquels les apprenants explicitent leurs buts et leurs connaissances, ce qui les rend saisissables pour un observateur externe.

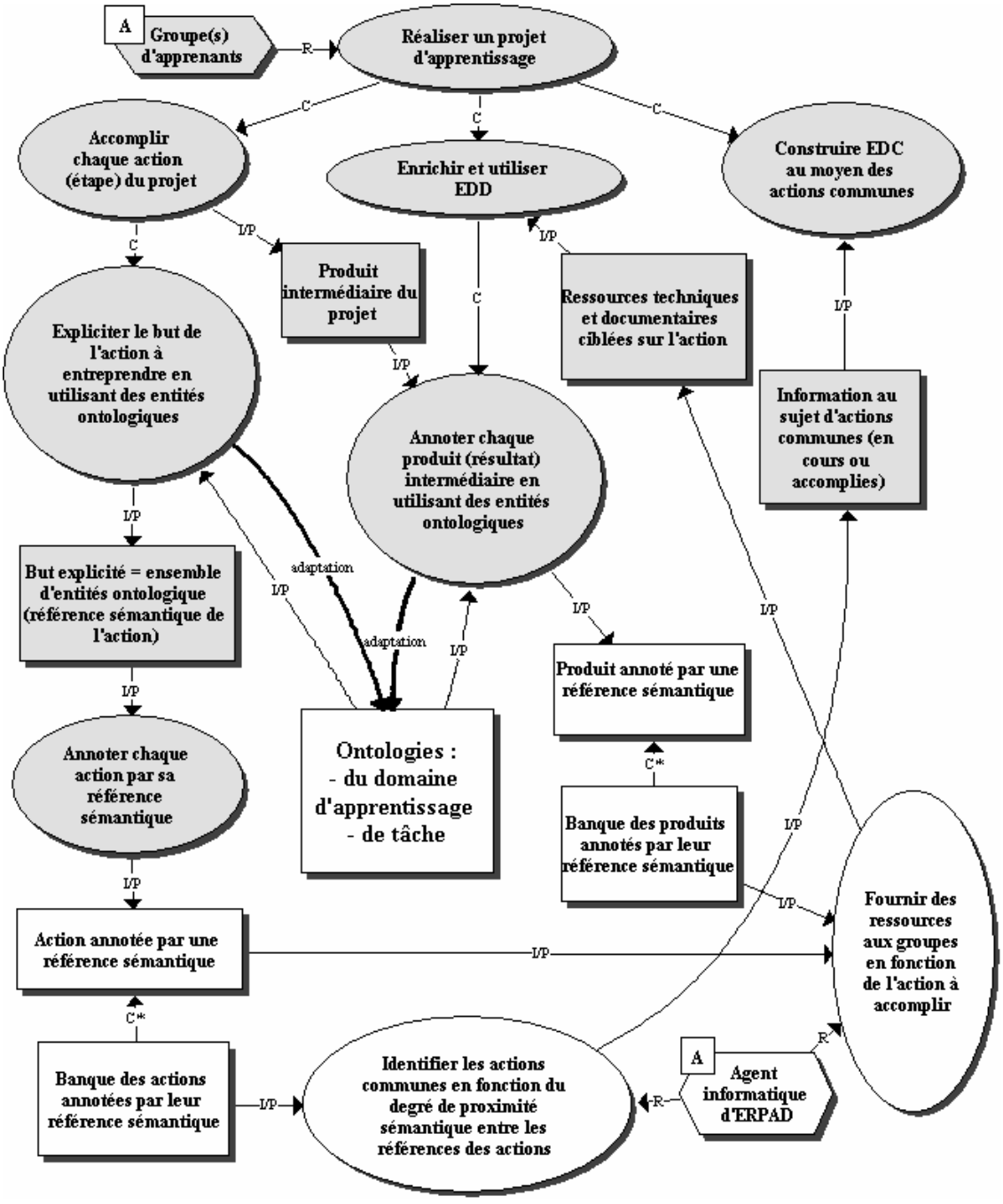


Figure 1. Utilisation des ontologies pour la construction de l'EDC et de l'EDD

Légende : Le formalisme s'interprète comme suit : l'ellipse représente une procédure, le rectangle un objet ou concept, le hexagone ayant le signe A représente un acteur qui régit (lien R) la procédure sous-jacente. Le lien C est un lien de Composition, I/P de Intrant/Produit.

Cependant, ERPAD ne peut pas être entièrement prédéfini par les concepteurs, car l'apprentissage par projet est une démarche constructive et flexible, sans une structure pédagogique complètement préétablie (Lebrun, 2002; Thomas, 2000). C'est une démarche pendant laquelle les apprenants créent leur propre compréhension du monde et des informations véhiculées dans ce monde (Finkle et Torp, 1994; Penuel et Means, 2000). Dans ce contexte, les ontologies vues comme des structures statiques des connaissances ne peuvent pas rendre compte de la démarche de projet. Il est nécessaire alors d'offrir aux apprenants la possibilité d'adapter, pendant leur démarche de projet, l'ontologie du domaine d'apprentissage et celle de tâche (Rogozan et Paquette, 2003). En conclusion, l'évolution de l'ontologie est une condition *sine qua non* pour la conception appropriée de l'ERPAD, celui-ci devant premièrement être capable de gérer cette évolution dans un environnement dynamique, multi-acteurs et distribué sur le Web.

2. Contexte de recherche du projet de doctorat

L'association entre les ontologies et le Web, connue actuellement sous le nom de « Web sémantique » dont l'expression est dû à Berners-Lee et *al.* (2001), représente notre contexte général de recherche. Étant donné que tout système d'apprentissage en ligne qui utilise des ontologies est un système du Web sémantique (Anderson et Whitelock, 2004), notre contexte particulier est alors l'utilisation du Web sémantique comme infrastructure pour des systèmes d'apprentissages distribués.

2.1 Contexte général : Web sémantique et ontologies

2.1.1 Notion de l'ontologie

Les ontologies sont apparues au début des années 90 à la suite des démarches d'acquisition des connaissances pour les systèmes à base des connaissances (SBC). Ces démarches proposaient de dégager les objets du domaine, leur signification et leur contenu, des objets du raisonnement décrivant les règles heuristiques d'utilisation des objets du domaine, le but étant de faciliter la construction des SBC en permettant la réutilisation des composants génériques (Van Heijst, Schreiber, et Wielinga, 1997).

En reprenant les définitions de Borst (1997), de Gruber (1995) et celle de Guarino et Giaretta (1995), nous considérons l'ontologie comme étant *la spécification, plus ou moins formelle, rendant compte partiellement, mais explicitement d'une conceptualisation partagée*, c'est-à-dire acceptée à l'intérieur d'une communauté. Cette conceptualisation est une théorie logique permettant la représentation d'un domaine de discours en terme de (Gomez-Perez, 1999) :

- *concepts* organisés taxonomiquement à l'aide des relations de subsumption *(is-a)* spécifiant des liens de généralisation qui permettent l'héritage entre les concepts;
- *relations* représentant des types d'interactions entre les concepts;

- *axiomes* explicitant des énoncés conceptuels toujours vrais dans le contexte de l'ontologie et utilisés pour contrôler la signification des concepts ou des relations;
- *instances*, utilisées pour représenter des entités concrètes du domaine.

Toutes ces *entités ontologiques*, c'est-à-dire les concepts, les relations, les axiomes et les instances, doivent être définis explicitement à l'aide d'un langage ayant une sémantique plus ou moins formelle, le degré de formalisation dépendant des agents qui utilisent l'ontologie (tableau 1).

Tableau 1. Sémantique du langage de représentation des ontologies

Sémantique du langage de représentation en fonction des agents qui utilisent les ontologies d'après Gruninger et Lee (2002) et Uschold et King (1995)
- *ontologie utilisée entre les agents logiciels* : ontologie rigoureusement formelle, exprimée par un langage ayant une sémantique formelle et exécutable informatiquement.
- *ontologie utilisée entre les individus et les agents logiciels* : ontologie rigoureusement formelle, exprimée par un langage ayant une sémantique formelle.
- *ontologie utilisée entre les individus* : ontologie informelle, mais cohérente, exprimée en langage naturel ou sous une forme structurée et restrictive du langage naturel.

Ainsi, étant donné que la conceptualisation est spécifiée d'une manière précise et formelle pour la plupart du temps, l'ontologie ne peut pas assumer la richesse interprétative du domaine conceptualisé et ne le fait donc que partiellement. Cet écart entre la conceptualisation et la spécification explicite et formelle est dû aux problèmes de calculabilité dans les agents logiciels, étant décrit par Guarino et Giaretta (1995) comme *l'engagement ontologique* que le concepteur doit assumer pour passer de la conceptualisation à la spécification formelle d'une ontologie.

2.1.2 Ontologies pour le Web sémantique

Le Web actuel est essentiellement syntaxique, la structure des ressources étant bien définie, mais leur contenu restant inaccessible aux traitements machines, seuls les humains étant capables de l'interpréter. Le Web sémantique a alors l'ambition de lever cette difficulté en associant aux ressources du Web des entités ontologiques comme références sémantiques, ce qui permettra aux différents agents logiciels d'accéder et d'exploiter directement le contenu des ressources et de raisonner dessus (Berners-Lee et *al.*, 2001). Ce référencement sémantique peut aussi résoudre les problèmes d'interprétation des ressources informationnelles provenant des applications hétérogènes et reparties (Wache et *al.*, 2001) et de permettre ainsi à ces applications d'être intégrées sémantiquement (Uschold et Gruninger, 2002).

2.1.2.1 Architecture du Web sémantique

L'architecture du Web sémantique (figure 2) repose sur une hiérarchie des langages d'assertion[4] et de description d'ontologies ainsi que sur un ensemble de services pour l'accès aux ressources au moyen de leurs références sémantiques, pour gérer l'évolution et le versionnage des ontologies (WebOnt, 2004b), pour l'utilisation des moteurs d'inférences capables effectuer des raisonnements complexes ainsi que des services pour la vérification de la validité sémantique de ces raisonnements (Oberle, Volz et Staab, 2004).

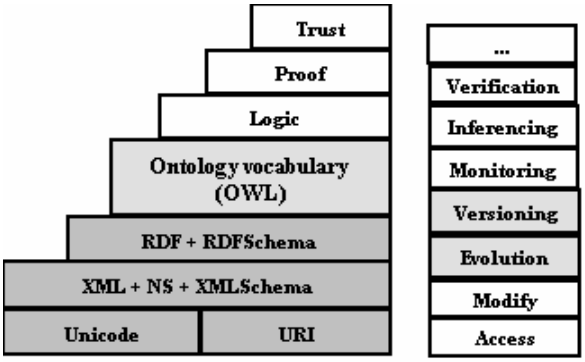


Figure 2. Architecture du Web sémantique adapté de Berners-Lee (2000) et d'Oberle et *al.* (2004)

Le W3C (*World Wide Web Consortium*) propose une hiérarchie des langages pour le Web sémantique dont seulement les couches inférieures[5] sont stabilisées à l'heure actuelle (Berners-Lee, 2000). Cette architecture repose sur URI, (*Uniform Resource Identifier*) qui permet l'attribution d'un identifiant unique à des ressources. XML (e*Extensible Markup Language*) est le langage de base qui procure une syntaxe aux documents structurés, mais il ne dispose d'aucune sémantique permettant de décrire la signification de ces documents. Le XML peut être vu comme la couche de transport syntaxique du Web sémantique, tous les autres langages étant exprimables et échangeables dans la syntaxe XML. Le langage RDF (*Resource Description Framework*) est un langage formel ayant une sémantique simplifiée permettant d'exprimer des relations entre les ressources du Web en utilisant des triplets de la forme *sujet-prédicat-objet*. Au RDF s'ajoute RDFS (*RDFschema*) qui permet de déclarer des classes de ressources RDF avec une sémantique pour définir des hiérarchies de généralisation de classes (*rdfs:subClassOf*). RDFS permet aussi de déclarer des propriétés, définies comme des relations binaires entre les classes, en précisant leur domaine d'applicabilité (*rdfs:domain*) et leur domaine de valeurs (*rdfs:range*) ainsi que leur niveau de généralisation dans la hiérarchie de propriétés *(rdfs:subPropertyOf*).

[4] Les assertions affirment l'existence des relations entre des objets. Elles sont donc adaptées à l'expression des références que l'on veut associer aux ressources du Web.

[5] Par la suite, nous discutons que de couches inférieures de la hiérarchie des langages. Toutefois, nous précisons que le langage *Logic* vise à décrire l'ensemble des déductions qu'on peut faire à partir d'une collection des données, le langage *Proof* vise à décrire les étapes de raisonnement nécessaires pour arriver à une conclusion à partir des faits, et dans la couche *Trust* il est possible d'affirmer des valeurs de confiance par rapport aux raisonnements effectués.

Cependant, l'extension à RDFS ne fournit que des mécanismes primitifs pour spécifier les classes et les propriétés et n'intègre non plus des capacités de raisonnement. Pour munir les langages du Web d'une sémantique formelle permettant la représentation des ontologies, le RDF et RDFS ont été enrichis par l'apport du langage ontologique OWL (*Ontology Web Language*) accepté comme standard par le W3C en février 2004 (WebOnt, 2001). En plus des caractéristiques du RDFS, le OWL permet de définir des classes plus complexes en utilisant des constructeurs provenant de la logique de description (*intersectionOf, unionOf, complementOf*), des restrictions et des axiomes des classes (*equivalentClass, disjointWith, oneOf*) ainsi que des propriétés équivalentes, inverses, symétriques ou transitives. Le langage OWL possède trois sous-langages d'expressivité croissante : (1) le langage de base OWL-LITE pour la description des hiérarchies de classification et des contraintes simples, (2) OWL-DL pour une description d'une expressivité maximale tout en prenant en compte la logique de description afin d'offrir des propriétés de calcul nécessaires aux systèmes de raisonnement, (3) OWL-Full pour la meilleure expressivité dans la description des ontologies, mais sans offrir aucune garantie de calcul.

2.1.2.2 Ontologies OWL du Web sémantique

Sans revenir sur la définition de l'ontologie discutée dans une des sections précédentes, il est clair que, dans le contexte du Web sémantique, la formalisation des ontologies est nécessaire pour que les agents logiciels puissent manipuler la sémantique des ressources du Web. En conséquence, toute ontologie du Web sémantique devrait être spécifiée dans un langage ayant une sémantique formelle riche et consistante : le OWL. Une *ontologie OWL* est une collection d'informations incluant des descriptions de classes, des propriétés et des instances (WebOnt, 2004a). Une classe décrit d'une façon formelle un *concept*[6] abstrait du domaine de l'ontologie. Une *instance* est un membre de l'extension d'un concept (classe), cette extension étant définie comme l'ensemble des tous les individus concrets appartenant à ce concept. Une *propriété* est une relation binaire qui énonce des faits généraux au sujet des concepts ou des faits spécifiques au sujet des instances. Une ontologie est complètement définie quand les concepts sont structurés par leurs propriétés et que ces propriétés sont combinées dans des *axiomes* permettant d'effectuer des inférences sur le contenu de l'ontologie.

2.2 Contexte particulier : Web sémantique, infrastructure ontologique pour les systèmes d'apprentissage en ligne

Restreindre le Web sémantique à son architecture serait limitatif tant que ce sont les applications développées sur celle-ci qui font vivre cette vision et qui sont ainsi la preuve du concept. Sans vouloir être exhaustive, les principales applications du Web sémantique sont dans les domaines du commerce

[6] Dans ce document nous utiliserons le terme concept plutôt que le terme classe.

électronique, des portails et mémoires d'entreprise, du traitement automatique des langues, de la traduction automatique, de la recherche d'information et, également, dans le domaine de la formation en ligne.

Dans la formation en ligne, le Web sémantique se présente comme une infrastructure prometteuse pour la mise en place des systèmes d'apprentissage distribués qui soient flexibles, adaptables aux besoins des utilisateurs et réutilisables dans plusieurs contextes d'apprentissage (Anderson et Whitelock, 2004; Koper, 2004; Stojanovic, Staab, et Studer, 2001; Stutt et Motta, 2004). En utilisant les ontologies, le Web sémantique facilite l'interopérabilité et la gestion des ressources pédagogiques provenant des systèmes hétérogènes et distribués ainsi que la réutilisation des modèles d'enseignement fondés sur des ontologies partagées par diverses communautés. Dû au référencement sémantique des ressources pédagogiques ainsi qu'à la possibilité d'effectuer un appariement sémantique entre ces références et diverses requêtes, le Web sémantique accroît la capacité des moteurs de recherche d'offrir des réponses adaptées aux besoins des utilisateurs. Enfin, il peut supporter la création dynamique des espaces de télétravail personnalisés étant donné qu'il permet d'apparier les préférences, les objectifs ou les buts des acteurs avec des éléments pédagogiques distribués (p.ex. des objets pédagogiques, des activités de télétravail ou des facilitateurs de l'apprentissage), provenant de divers systèmes d'apprentissage.

Particulièrement, l'espace de réalisation des projets d'apprentissage à distance (ERPAD) est un système d'apprentissage sur le Web sémantique, ce système étant composé d'un ensemble d'éléments pédagogiques combinés pour former un tout cohérent. Par *éléments pédagogiques*, nous comprenons les activités (actions) pédagogiques, les ressources utilisées ou produites pendant ces activités ainsi que les acteurs intervenant dans l'acte d'apprentissage. Le référencement sémantique des éléments pédagogiques à un espace commun des connaissances crée alors un environnement de téléapprentissage où la combinaison physique des éléments composants est accompagnée d'une agrégation de leur sens (Paquette et Rosca, 2002), ce qui permet l'agrégation sémantique de l'ERPAD.

3. Problématique et objectifs de recherche

3.1 Identification de la problématique de recherche

Un des aspects les plus importants de l'utilisation des ontologies renvoie à leur caractère évolutif, et cela tant pour les ontologies du Web sémantique que pour les ontologies utilisées dans un système d'apprentissage du Web sémantique.

Les ontologies utilisées comme référentiels sémantiques dans les systèmes d'apprentissage doivent évoluer. À mesure que l'apprentissage se concrétise, les connaissances des acteurs évoluent ce qui demande la modification de l'ontologie si les acteurs considèrent qu'elle ne satisfait plus ses fonctions de référentiel sémantique. De plus, vue comme système commun de référencement, l'ontologie devrait

permettre un feedback évolutif au sens que le référencement de nouvelles ressources et activités pourrait demander la modification de leur référentiel commun afin de prendre en compte de nouvelles caractéristiques. Enfin, dans un système d'apprentissage de type ERPAD, la modification du référentiel sémantique est une nécessité due au caractère dynamique et non prédéfini de l'apprentissage par projet.

Les ontologies du Web sémantique doivent évoluer aussi (Charlet, Bachimont, et Troncy, 2003; Ding, Fensel, Klein, Omelayenko, et Schulten, 2004; Euzenat et all., 2001; Heflin et Hendler, 2000; Klein et Fensel, 2001; Laublet, Reynaud, et Charlet, 2002; Maedche, Motik, et Stojanovic, 2003; McGuinness, 2000; Noy et Klein, 2003a; Stojanovic et Motik, 2002; Studer, 2003). Le Web sémantique est un environnement dynamique, multi-acteurs et distribué. Les ontologies du Web sémantique ne peuvent pas alors être pensées comme des produits, qu'une fois achevés, reste stables par la suite. Les motifs sont multiples, par exemple le domaine de définition peut changer, ce qui nécessite l'évolution de l'ontologie pour rendre compte des changements, ou la réutilisation de l'ontologie pour des tâches différentes demande sa modification, ou encore la conceptualisation de l'ontologie peut changer étant donné qu'elle est construite continuellement pendant le processus d'échange d'informations et des significations entre les acteurs.

Bien que très récents dans la communauté scientifique (apparus au début 2000), les travaux sur l'évolution de l'ontologie sont essentiels pour le Web sémantique (OntoWeb, 2002a; Web_sémantique, 2001; WebOnt, 2004b). Ces travaux constituent alors le pas suivant à faire dans les recherches sur les ontologies, leurs buts devant être de développer des méthodes et des outils supportant le processus d'évolution d'une manière consistante :

> « Although evolution over time is an essential requirement for successful application of ontologies, methods and tools to support this complex task completely are missing » (Stojanovic, Maedche, Motik, et Stojanovic, 2002).

La conclusion est la suivante. Les ontologies du Web sémantique doivent évoluer, les ontologies utilisées dans un système d'apprentissage sur le Web, particulièrement dans un système de type ERAPD, doivent elles aussi évoluer, des méthodes et des outils doivent être conçus pour assurer l'évolution des ontologies. Cela nous amène à identifier notre problématique de recherche[7], à savoir : ***comment supporter l'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué sur le Web sémantique ?***

[7] Il s'agit de la problématique spécifique au projet de doctorat, la solution que nous lui apporterons étant essentielle pour la construction d'un système d'apprentissage de type ERPAD.

3.2 Réflexions sur la problématique de recherche

L'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué, soulève, à notre avis, des questionnements importants :

1. *Comment faciliter la spécification des changements à apporter à l'ontologie dans une manière complète mais conviviale ?*

Assurer l'évolution de l'ontologie dans un environnement dynamique où les acteurs ne sont pas tous des experts dans la construction de l'ontologie, demande la possibilité de spécifier les changements d'une manière simple et compréhensible (Noy et Klein, 2003b; Stojanovic, Maedche et *al.*, 2002). Par exemple, pour déplacer un concept dans la hiérarchie des concepts il est plus approprié d'utiliser une opération de changement complexe de type Déplacer_Entité qu'une suite des opérations élémentaires Effacer_Concept et Ajouter_Concept ayant le même résultat.

2. *Comment gérer la spécification collective des changements par des acteurs distribués ?*

Considérons la situation où plusieurs acteurs distants tentent de modifier la même ontologie. Dans ce cas, les changements effectués dans une partie de l'ontologie se propagent vers les autres parties de l'ontologie, ce qui peut produire des situations conflictuelles (p.ex. un acteur à l'intention d'effacer un concept utilisé par un autre acteur). Cette situation devient encore plus complexe si les acteurs sont autorisés à pouvoir accepter, refuser ou défier les changements des autres, étant donné que la cohérence de l'ontologie doit être maintenue (Euzenat, 1995; Sunagawa, Kozaki, Kitamura, et Mizoguchi, 2003).

3. *Comment préserver l'accès et l'interprétation des objets au moyen de leur référencement à une ontologie évolutive ?*

Le changement d'une ontologie utilisée comme système de référencement peut produire des pertes des coordonnées sémantiques, c'est-à-dire des références d'objets établies par rapport à un certain référentiel sémantique. En conséquence, les objets utilisant ces références ne sont plus accessibles au moyen de l'ontologie modifiée (Heflin, 2001). Soit un objet pédagogique 'Cahier de Physique – 01' qui utilise le concept 'loi de Newton' comme référence sémantique. Après l'application d'un changement fusionnant les concepts 'loi de Newton' et 'loi de Pascal' dans un seul concept 'lois de la mécanique', cet objet n'est plus accessible au moyen de l'ontologie ainsi modifiée, par exemple, pour une recherche de type « trouver les objets pédagogiques décrivant les lois de la mécanique (concept dans l'ontologie évoluée)».

4. *Comment conserver l'interopérabilité sémantique entre l'ontologie évoluée et des autres ontologies locales qui en dépendent ?*

Toute application du Web sémantique demande un accès continu à des ressources provenant des sources d'informations hétérogènes, autonomes et distribuées. Du fait que les ontologies fournissent une description formelle de la sémantique de ces ressources, elles peuvent résoudre leur problèmes d'interprétation (p.ex. équivalence, incompatibilité) et de permettre ainsi aux sources d'informations d'être interopérables. Dans cette situation, le changement d'une ontologie peut avoir des conséquences négatives sur la préservation de l'interopérabilité avec les autres ontologies locales (Maedche, Motik, et Stojanovic, 2003).

5. *Comment et quand informer les utilisateurs des changements apportés à l'ontologie?*

Deux scénarios sont possibles pour l'évolution de l'ontologie : (a) l'ontologie a été changée silencieusement, aucune information concernant son évolution n'étant fournie aux utilisateurs, (b) l'ontologie a été changée visiblement, des informations sur les changements exécutés étant fournies aux utilisateurs. Tant que les changements apportés à l'ontologie sont des extensions monotoniques, par exemple l'ajout des entités ontologiques, ou des changements qui n'affectent pas la conceptualisation d'une entité ontologique, le premier scénario peut être utilisé étant donné que l'accès aux objets référencés par des entités de l'ontologie modifiée n'est pas affecté, non plus l'interopérabilité de cette ontologie. Tout autre type des changements, par exemple effacer, fusionner ou diviser des entités ontologiques, peut invalider l'interopérabilité et l'accès aux objets référencés ce qui demande l'utilisation du deuxième scénario afin de rendre explicite les possibles incompatibilités (Heflin et Hendler, 2000; Klein et Fensel, 2001).

3.3 Objectifs de recherche

En tenant compte des questionnements présentés ci-dessus, nous pouvons affirmer qu'assurer la gestion de l'évolution de l'ontologie d'une manière pertinente constitue un vrai défi que nous envisageons de relever, du moins partiellement, en nous proposant comme objectifs de recherche de concevoir :

- une méthodologie d'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué sur le Web sémantique ;
- une méthode pour assurer l'analyse et la gestion des changements exécutés pour faire évoluer une ontologie et un outil qui implémente cette méthode ;
- une méthode pour assurer un référencement sémantique évolutif des objets référencés et un outil qui implémente cette méthode. Par *objet référencé* nous comprenons tout élément, que ce soit des ressources du Web ou des éléments pédagogiques d'un système d'apprentissage, qui utilise des entités ontologiques comme références sémantiques.

3.4 Originalité de la problématique et des objectifs proposés

Si les travaux sur les ontologies, leur construction et leur utilisation, ont déjà une dizaine d'années, ceux sur l'évolution de l'ontologie sont bien plus récents et leur poursuite est indispensable pour que les ontologies servent le Web sémantique et tout système d'apprentissage du Web sémantique. Nos objectifs de recherche se situent au cœur même des recherches actuelles sur l'évolution de l'ontologie étant donné que « *the management of changes is the key issue in support for evolving ontologies* » (Ding et *al.*, 2004).

Nous envisageons alors notre contribution à deux niveaux.

- au niveau conceptuel nous proposons une méthodologie qui (1) unifie les approches méthodologiques de l'évolution de l'ontologie, actuellement presque inexistantes et (2) intègre des éléments méthodologiques novateurs, principalement en ce qui concerne la gestion des changements ontologiques et l'opérationnalisation de l'ontologie, une fois évoluée.
- au niveau technologique nous proposons deux systèmes informatiques pour supporter des éléments méthodologiques essentiels pour l'évolution de l'ontologie. Ces deux systèmes, OntoAnalyseur et UKIsModificateur, possèdent des fonctionnalités innovantes, capables d'effectuer une analyse sémantique du processus d'évolution et d'assurer un référencement sémantique évolutif.

Avant de revenir sur les éléments de problématique pour mieux les expliciter, nous présentons la composante cognitive et la composante informatique de notre projet de recherche et nous faisons le point de travaux antérieurs en justifiant la pertinence de nos idées par rapport à ces travaux (chapitre II et III).

3.5 Composante cognitive et informatique du projet de recherche

Notre projet de recherche s'intègre bien dans le domaine de l'informatique cognitive, notamment dans le domaine de la représentation *évolutive* des connaissances.

Afin de représenter d'une manière pertinente un état du monde il faudrait, en plus d'énoncer l'ensemble des connaissances du monde qui peuvent être représentées, de trouver des moyens et d'énoncer des principes pour incorporer des nouvelles connaissances ou d'effectuer des révisions de celles déjà existantes (Nejdl et Banagl, 1994; Shoham et Steve, 1994). En ce sens, Ermine et *al.* (1996) notent l'importance d'un modèle d'évolution de systèmes des connaissances, de leur progression dans le temps, de la naissance des connaissances jusqu'à leur disparition. Notre travail de recherche sur l'évolution de l'ontologie, vue comme la spécification d'une conceptualisation de l'état du monde, se veut alors une réponse à des questionnements soulevés auparavant par ces chercheurs ainsi que par ceux oeuvrant dans le domaine de la gestion des connaissances (Charlet, Zacklad, Kassel, et Bourigault, 2000).

L'ontologie est représentée à l'aide d'un langage ayant une sémantique formelle, ce langage étant un formalisme de représentation des connaissances permettant de préciser exactement quelles significations associer aux entités ontologiques en fonction des significations de symboles qu'elles contient (p.ex. symboles de prédicats ou d'opérateurs logiques) et de la manière dont ces symboles sont assemblés (Bachimont, 2000). Revenant à l'évolution, un formalisme de représentation devrait alors intégrer des moyens représentatifs pour exprimer la nature évolutive de l'ontologie, ce que nous mettons d'ailleurs en évidence dans notre projet de recherche en développant aussi un modèle pour un tel formalisme.

Afin de permettre l'exploitation informatique des modèles conceptuels que nous proposons pour la gestion de l'évolution de l'ontologie, nous envisageons à développer (1) le système OntoAnalyseur qui implémente une méthode semi-automatique d'identification de changements ontologiques et d'analyse de la relation sémantique entre les versions ontologiques et (2) le système UKIsModificateur qui implémente une méthode semi-automatique pour assurer la modification appropriée et la mise à jour des références sémantiques d'objets référencés.

Chapitre II. Évolution de l'ontologie : méthodologies, méthodes et outils

Une méthodologie est une succession, compréhensible et intégrée, des méthodes et des techniques qui créent ensemble un système théorique explicatif pour l'accomplissement d'une tâche cognitivement complexe (IEEE, 1995). Une méthode est un processus ordonné utilisé pour l'ingénierie d'un produit ou pour exécuter un service (IEEE, 1990). Une technique est une procédure utilisée pour atteindre un objectif donné (IEEE, 1995).

Actuellement, il existe une vaste palette d'approches pour la construction des ontologies (OntoWeb, 2002b). La plupart de ces approches décrivent la construction d'une ontologie à partir de zéro, par exemple la méthode Cyc, SENSUS ou KACTUS, la méthode d'Uschold et King ou celle de Grüninger et Fox, les méthodologies METHONTOLOGIE et On-To-Knoweldge. D'autres approches décrivent la construction d'une ontologie par la fusion des ontologies différentes, par exemple les méthodes FCA-Merge et PROMPT ou la méthodologie MOMIS. Cependant, seulement METHONTOLOGY (Fernandez, Gomez-Perez, et Juristo, 1997) et On-To-Knowledge (Sure, Staab, et Studer, 2004) considèrent le processus d'évolution dans le cycle de vie d'une ontologie, mais elles ne fournissent ni méthodes, ni recommandations pour le supporter.

Pourtant, du point de vue des ontologies, seront cruciales pour le Web sémantique les méthodes et les outils contribuant à gérer l'évolution de l'ontologie dans un environnement multi-acteurs, distribué et dynamique. Dans ce chapitre, nous faisons alors l'état de l'art des approches méthodologiques pour assurer l'évolution de l'ontologie et nous mettons en évidence leurs limites respectives en proposant l'ébauche d'une méthodologie complète. À partir d'une analyse des principaux outils de construction d'ontologie, nous identifions ensuite des besoins technologiques pour supporter l'évolution de l'ontologie d'une manière consistante.

1. Méthodologies et méthodes pour supporter l'évolution de l'ontologie : l'état de la question

Plusieurs recherches mettent en évidence l'importance majeure de l'évolution de l'ontologie ainsi que le manque presque total des approches pour gérer cette évolution (Charlet et *al.*, 2003; OntoWeb, 2002b; WebOnt, 2004b). Actuellement, seulement deux approches traitent de l'évolution de l'ontologie sur le plan méthodologique : (1) l'approche développée par l'équipe de l'AIFB (Institute of Applied Informatics and Formal Description Methods) de l'université de Karlsruhe, et (2) l'approche développée par l'équipe du département IMSE (Information Management and Software Engineering) de l'université d'Amsterdam.

1.1 Méthodologie de l'AIFB pour supporter l'évolution de l'ontologie

L'évolution de l'ontologie est définie par Maedche et *al.* (2003), Maedche, Motik, Stojanovic, Studer et Volz, (2003), Stojanovic et *al.* (2002), Stojanovic et Motik (2002), Stojanovic et Stojanovic (2002), comme étant :

> *Définition 1 :* la modification appropriée de l'ontologie et la propagation consistante des changements dans les artefacts dépendants, c'est-à-dire dans les objets référencés, les ontologies dépendantes, et les applications logicielles utilisant l'ontologie.

Pour supporter l'évolution de l'ontologie, les auteurs proposent alors une méthodologie composée de cinq étapes principales.

Représentation des changements. Cette étape vise l'édition des changements élémentaires ou complexes. Un ***changement élémentaire*** est un changement primitif et non décomposable, les changements typiques étant l'ajout, l'effacement ou la modification des entités ontologiques. Un ***changement complexe*** est composé des plusieurs changements élémentaires qui forment ensemble une seule entité logique, par exemple le déplacement, la fusion ou la séparation des entités ontologiques. Klein et Noy (2003) mettent en évidence l'avantage d'utiliser des changements complexes : (1) ils sont plus facilement utilisables et compréhensibles car leur intention est explicite, contrairement à une suite de changements élémentaires ayant le même résultat, et (2) ils permettent l'évolution de l'ontologie avec moins de pertes des données[8].

Sémantique des changements. L'ontologie doit évoluer d'un état consistant vers un autre état consistant, c'est-à-dire l'état où les contraintes du modèle ontologique sont respectées. Afin de résoudre les inconsistances introduites par les changements, d'autres changements additionnels peuvent être nécessaires[9], la tâche de cette étape étant alors de permettre la résolution de tous les changements additionnels d'une manière systématique.

Implémentation. Cette étape vise l'exécution des changements, une fois approuvés par les utilisateurs.

Propagation des changements. Le but de l'étape de la propagation des changements est de modifier automatiquement les instances et les ontologies dépendantes afin de préserver leur consistance avec l'ontologie évoluée. Pour cela, Maedche et *al.* (2003) proposent une approche de modification des ontologies dépendantes[10] par l'application récursive du processus d'évolution en fonction des changements appliqués à l'ontologie évoluée.

---

[8] Déplacer_Concept préserve les instances, contrairement à la suite des changements Effacer et Ajouter_Concept.
[9] Sachant que chaque propriété doit avoir minimum un concept comme domaine et comme co-domaine, l'effacement d'un concept étant le seul domaine d'une propriété demande soit d'effacer la propriété elle-même, soit de définir un autre concept comme domaine de cette propriété.
[10] L'ontologie dépendante est celle qui utilise une partie de l'ontologie évolutive dans sa structure ontologique.

Validation. Les utilisateurs évaluent le résultat de l'évolution et recommencent le processus, si nécessaire.

Limites de la méthodologie de l'AIFB

Premièrement, les auteurs ne proposent aucune étape d'analyse des effets des changements sur la relation de compatibilité entre l'ontologie évoluée et les artefacts dépendants. Ceci est une limite importante étant donné que l'évolution de l'ontologie peut provoquer la dégradation du référencement sémantique des objets ou la dégradation de l'interopérabilité avec d'autres ontologies ou encore celle du comportement des systèmes fondés sur l'ontologie (Heflin, Hendler, et Luke, 1999; Stuckenschmidt et Klein, 2003a). En ce sens, l'étape de propagation des changements, proposée par les auteurs, est assez unidirectionnelle vue qu'elle vise seulement la modification des ontologies dépendantes afin de préserver leur consistance avec l'ontologie évoluée. Deuxièmement, les auteurs développent un processus d'évolution qui ne prend pas en compte la gestion des versions multiples. Ceci peut s'avérer convenable dans un environnement contrôlable où l'utilisation de l'ontologie peut être totalement prédéfinie. Toutefois, prédéfinir au complet l'utilisation d'une ontologie n'est pas une tâche possible sur le Web sémantique et, sans savoir comment une ontologie est utilisée, on peut difficilement propager systématiquement les changements dans les artefacts dépendants.

1.2 Méthodologie de l'IMSE pour supporter le versionnage de l'ontologie

L'évolution de l'ontologie est définie par Klein (2002a, 2002b), Klein, Ding, Fensel et Omelayenko (2002), Klein et Noy (2003), Noy et Klein (2003a), Noy et Musen (2003a) comme étant :

> *Définition 2* : la capacité de gérer les changements de l'ontologie et leurs effets en créant et en maintenant différentes versions d'une ontologie. Cette capacité consiste à identifier et à différencier les versions, à modifier les versions, à spécifier des relations qui rendent explicites les changements effectués entre les versions et à utiliser des mécanismes d'accès pour les artefacts dépendants, c'est-à-dire les objets référencés, les ontologies et les applications dépendantes.

Contrairement à la première définition qui considère uniquement l'ontologie évoluée, l'accès aux artefacts s'effectuant par cette ontologie, la deuxième définition considère l'utilisation de plusieurs versions de l'ontologie, l'accès aux artefacts s'effectuant au moyen de ces versions multiples. Les auteurs utilisent alors le terme *versionnage* pour décrire leur approche et proposent deux éléments fondamentaux pour une méthodologie de versionnage de l'ontologie :

- un modèle d'analyse de la relation entre deux versions d'une ontologie. Ce modèle permet (1) d'expliciter ce qui a été changé dans la définition des entités ontologiques, (2) de spécifier la relation sémantique entre les entités ontologiques dont la définition a été changée d'une version à

une autre (p.ex. entités équivalentes ou conceptuellement différentes), (3) de décrire les changements par un ensemble de métadonnées spécifiant la date, l'auteur et le but de chaque changement, (4) de décrire le contexte où les changements sont valides.

- une technique d'identification des versions sur le Web ayant deux principes de base : (1) un changement dans la définition d'une entité ontologique produit une nouvelle version, ayant un nouveau URI (*Uniform Resource Identifier*)[11], tant qu'un changement dans l'annotation textuelle d'une entité produit seulement un nouveau fichier avec un nouveau URL (*Uniform Resource Locator*) et (2) la forme d'URI indique si la version est compatible avec la version antérieure.

Limites de l'approche versionnage

Les auteurs ne proposent pas une approche pour supporter le processus d'évolution de l'ontologie, mais plutôt pour supporter la gestion des versions d'une ontologie après son évolution. Ils fournissent alors un modèle d'analyse de la relation entre les versions de l'ontologie, mais sans se préoccuper de la gestion de l'accès aux artefacts dépendants (i.e. objets référencés, ontologies, applications) au moyen de versions de l'ontologie. De plus, les auteurs ne développent aucun cadre fonctionnel pour intégrer la totalité des éléments méthodologiques qu'ils proposent.

1.3 Autres éléments méthodologiques pour l'évolution de l'ontologie

La recherche actuelle sur les méthodologies d'évolution ne se résume pas toutefois qu'à l'approche de l'AIFB et celle de l'IMSE, d'autres auteurs proposant des éléments méthodologiques à cet effet.

Heflin et Hendler (2000) et Heflin et *al.* (1999) développent SHOE, un langage fondé sur HTML, qui offre des primitives pour la gestion des versions multiples, en permettant d'associer à chaque version ontologique un identifiant unique ainsi qu'une balise « *Backward-Compatible_With* » spécifiant les versions compatibles ou rétrocompatibles. Oliver, Shahar, Musen, et Shortliffe (1999) définissent un modèle conceptuel, CONCORDIA, pour la gestion des changements d'une terminologie médicale. Ce modèle associe à chaque concept un identifiant unique, les concepts pouvant ensuite être seulement retirés et non pas physiquement effacés. Ainsi, pour chaque concept, le modèle CONCORDIA est capable de garder la trace de tous ses concepts-parent ou enfants retirés en utilisant leur identifiant. Enfin, McGuinness (2000) fournit des recommandations théoriques pour garantir un processus d'évolution de l'ontologie avec un minimum d'erreurs en utilisant des techniques de fusion pour mettre en évidence ces erreurs.

Du point de vue de l'évolution de l'ontologie dans un environnement multi-acteurs, Pinto et Martins (2002), et Pinto, Staab, et Tempich (2004) proposent un modèle pour gérer la négociation des changements

[11] Un URI permet d'attribuer un identifiant unique à une ressource sur le Web. Ainsi, un URI peut être un URL, avec la caractéristique qu'il permet d'identifier une ressource sur le Web, avec certitude et de manière unique.

provenant des acteurs distants qui tentent de modifier une ontologie partagée. Dans ce modèle, chaque utilisateur modifie l'ontologie à partir de son poste individuel, en construisant ainsi sa propre ontologie locale. Un comité scientifique analyse ensuite les ontologies locales pour identifier les changements à introduire dans l'ontologie partagée. Une approche similaire est développée par Sunagawa et *al.* (2003), la différence étant que les acteurs distants modifient l'ontologie partagée sans aucun intermédiaire. Ainsi, pour assurer la gestion de la dépendance entre les changements effectués par les différents acteurs, ceux-ci peuvent choisir d'accepter les changements des autres, de réduire la dépendance par rapport aux changements des autres ou de rompre la dépendance.

2. Ébauche d'une méthodologie complète

Les approches présentées dans les sections précédentes sont actuellement les seules ayant des indications méthodologiques pour supporter l'évolution de l'ontologie, mais elles ne fournissent ni une méthodologie complète ni un cadre intégrateur pour les éléments qu'elles proposent. Nous énonçons alors notre ***objectif conceptuel de recherche*** qui consiste à intégrer ces approches dans une méthodologie unifiée décrivant un processus complet d'évolution de l'ontologie dans un contexte multi-acteurs, incluant la gestion des versions multiples.

2.1 Évolution de l'ontologie – notre définition

Notre objectif d'unifier les approches méthodologiques dans un cadre cohérent, requiert une nouvelle définition de l'évolution de l'ontologie, une définition qui englobe les deux autres précédentes.

Définition 1 (Rôle de l'ontologie). Le service assuré par l'ontologie, le but de son utilisation. Par exemple, dans notre contexte particulier de recherche, l'ontologie a le rôle de référentiel sémantique commun pour les éléments pédagogiques qui composent un système d'apprentissage.

Définition 2 (Consistance de l'ontologie). Une ontologie est consistante si les contraintes concernant le modèle et les axiomes de cette ontologie sont respectées.

Définition 3 (Évolution de l'ontologie). Le processus de changement de l'ontologie en fonction des modifications dans son domaine, dans sa conceptualisation ou dans son usage. Ce processus se traduit en réalité par l'exécution d'un ou plusieurs changements ontologiques pour passer d'une version de l'ontologie (V_N) à une version nouvelle (V_{N+1}), tout en préservant la consistance et les rôles de l'ontologie.

Définition 4 (Changements ontologiques). Les changements exécutés pour passer de V_N à V_{N+1}. Nous nommons changement ontologique tout changement dans la définition des entités d'une ontologie, sans considérer le changement des annotations textuelles de ces entités, ainsi que tout changement dans la structure taxonomique de l'ontologie.

Définition 5 (Développeur de l'ontologie). Tout acteur impliqué dans le processus d'évolution de l'ontologie.

2.2 Méthodologie d'évolution de l'ontologie

Nous avons conçu la méthodologie d'évolution de l'ontologie comme un processus en neuf étapes principales, ce processus étant accompli par les développeurs à l'aide d'un atelier de support à l'évolution de l'ontologie (figure 3). Dans cette section, nous présentons brièvement ces étapes, les ayant explicitées davantage dans (Rogozan, 2004; Rogozan, Paquette, et Rosca, à paraître en 2004).

Étape 1. Accéder, identifier et télécharger la version de l'ontologie à modifier. Les ontologies se trouvent généralement stockées dans des librairies distribuées sur le Web qui fournissent des mécanismes d'emmagasinage, permettant un accès constant aux ontologies ainsi que la possibilité de naviguer dans les ontologies elles-mêmes (Ding et Fensel, 2001). Dans cette étape, les développeurs doivent accéder à l'ontologie à modifier et le télécharger sur leur poste de travail. Nous notons cette ontologie avec $\mathbf{V_N}$ et nous la nommons *version précédente de l'ontologie*.

Étape 2. Identifier les changements. Les développeurs identifient les changements à apporter à l'ontologie d'une manière descendante, c'est-à-dire à partir des modifications dans le domaine de définition ou d'utilisation de l'ontologie, ou ascendante, c'est-à-dire à partir de l'analyse de l'ontologie elle-même.

Étape 3. Éditer les changements. Dans cette étape, les développeurs éditent les changements identifiés précédemment. En ce qui concerne les fonctionnalités pour permettre l'édition des changements, Klein (2002b) propose une taxonomie des changements élémentaires spécifiant l'ajout, l'effacement et la modification des entités ontologiques définies à l'aide de OWL-LITE. Nous envisageons alors de prolonger cette taxonomie à OWL-DL et de développer aussi une taxonomie des changements complexes précisant, par exemple, le déplacement, la fusion ou la séparation des entités d'une ontologie OWL-DL. Pour chaque changement dans l'ontologie, il est possible de générer différents changements additionnels conduisant chacun à des états finaux différentes (Stojanovic, Maedche et *al.*, 2002). Par exemple, si un concept à l'intérieur de la hiérarchie des concepts est supprimé, ses sous-concepts peuvent être, soit supprimés, soit rattachés au concept-parent, soit rattachés au concept-racine de la hiérarchie. Le résultat de l'étape d'édition consiste alors dans une séquence des changements à apporter à l'ontologie, chaque changement ayant la forme d'un couple de type (changement édité, scénario de changements additionnels). De plus, chaque changement devrait être annoté en utilisant un ensemble de métadonnées décrivant l'auteur, le but du changement, ainsi que le rôle sémantique de l'entité ontologique étant le sujet du changement (Klein, 2002a; Oliver et *al.*, 1999).

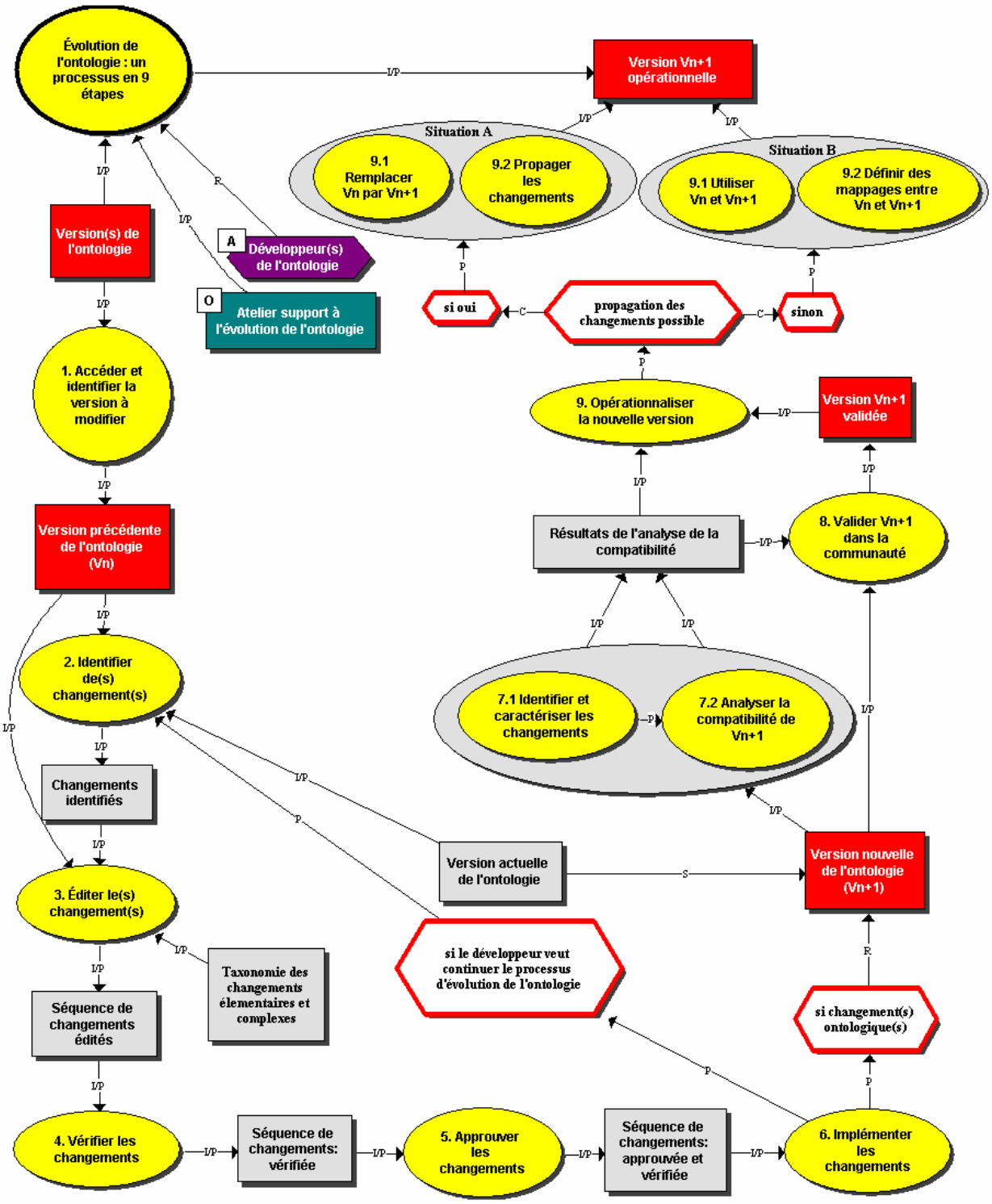


Figure 3. Modèle général de la méthodologie d'évolution de l'ontologie

Légende : Le formalisme de modélisation MOT(Paquette, 2002b) s'interprète comme suit : l'ellipse représente une procédure, le rectangle un objet ou concept, le hexagone une règle, le hexagone ayant le signe A représente un acteur qui régit la procédure sous-jacente et le hexagone avec le signe O un outil qui supporte cette procédure. Le lien C est un lien de Composition, I/P de Intrant/Produit, P de Précédence.

Étape 4. Vérifier les changements. Cette étape vise la vérification des effets des changements sur la consistance de l'ontologie. Si des changements invalident les contraintes de l'ontologie, alors les développeurs doivent être informés et une méthode pour les supporter dans la résolution d'inconsistances doit leur être fournie (cf. § 1.1, sémantique de changements). Pour les changements qui invalident les axiomes de l'ontologie, les développeurs peuvent aussi modifier directement les axiomes invalidés afin de contourner les inconsistances.

Étape 5. Approuver les changements. Dans cette étape, la séquence des changements est validée conjointement par les développeurs de l'ontologie. Si cette validation n'est pas possible dû au caractère distribué de l'environnement d'occurrence du processus d'évolution, alors des mécanismes de gestion des conflits doivent être prévus (cf. § 1.3. Pinto et *al.*, 2004; Sunagawa et *al.*, 2003).

Étape 6. Implémenter les changements. Cette étape vise l'implémentation des changements avec une préservation de la trace et des métadonnées associées, par exemple dans un journal de changements, et avec une gestion de l'identification des versions, sachant que seulement les changements ontologiques produisent une ***nouvelle version de l'ontologie***, que nous notons avec $\mathbf{V_{N+1}}$, ayant un nouveau URI (cf. § 1.2). Si besoin, les développeurs peuvent retourner maintenant à l'étape 2 du processus, l'ontologie à modifier étant maintenant la nouvelle version obtenue à la fin de l'étape 6.

Étape 7. Identifier les changements et analyser la relation de compatibilité entre V_N et V_{N+1}. Le processus d'évolution peut causer des incompatibilités pouvant provoquer l'altération des rôles de l'ontologie (cf. chapitre I § 3.1). Afin d'identifier ces incompatibilités, le but de cette étape est de (1) identifier les changements exécutés pour passer de V_N à V_{N+1}, (2) caractériser leurs effets sur la relation sémantique entre les entités ontologiques appartenant à V_N et celles appartenant à V_{N+1}[12] et (3) analyser les effets des changements sur la compatibilité de V_{N+1} du point de vue de la préservation des rôles de l'ontologie[13].

Étape 8. Valider V_{N+1} dans la communauté. Dans cette étape, les développeurs approuvent ou désapprouvent collectivement la nouvelle version de l'ontologie avant de la rendre opérationnelle.

Étape 9. Opérationnaliser V_{N+1}. En fonction des résultats de l'analyse de la relation de compatibilité entre V_N et V_{N+1}, le problème le plus difficile est maintenant de préserver les rôles de l'ontologie pour la nouvelle version V_{N+1}. La préservation des rôles de l'ontologie peut se faire selon deux types de situation :

[12] Soit un changement qui modifie le nom d'une entité ontologique sans modifier sa définition ontologique. Dans ce cas, la relation entre cette entité en V_N et l'entité en V_{N+1} est une relation d'identité. Pour plus d'exemples, voir chapitre III, paragraphe 1.2.3.

[13] Soit un changement qui efface un concept utilisé comme référence sémantique par un objet. Dans ce cas l'objet n'est plus accessible au moyen de V_{N+1}. En conséquence, le changement est incompatible, car le rôle de l'ontologie n'est pas entièrement préservé.

- *Situation A* : quand la modification et la mise à jour des artefacts dépendants (*i.e.* objets référencés, ontologies et applications dépendantes) en fonction des changements exécutés pour passer de V_N à V_{N+1} peuvent se réaliser, alors uniquement la nouvelle version de l'ontologie peut être utilisée par la suite pour accéder ou interpréter ces artefacts. Nous empruntons à Stojanovic, Stojanovic, et Handschuh (2002) le terme « propagation des changements » pour désigner cette situation.
- *Situation B* : quand la modification et la mise à jour des artefacts dépendants par rapport à la nouvelle version de l'ontologie ne peuvent pas se réaliser, alors il faudrait définir des mappages entre les versions de l'ontologie afin de permettre l'accès et l'interprétation des artefacts par les deux versions. Les mappages inter-ontologies (Wache et *al.*, 2001), dans notre cas entre les versions de la même ontologie, sont des règles spécifiant des liens de correspondance sémantique (p.ex. équivalence, intersection ou incompatibilité) entre des entités ontologiques appartenant aux versions différentes.

En définissant cette ébauche de la méthodologie, notre but principal est d'identifier les étapes essentielles à prendre en compte pour supporter la gestion de l'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué. Nous sommes conscients que la réalisation de chaque étape demande une méthode spécifique, chaque méthode pouvant être considérée comme un sujet à débattre lors d'un projet de doctorat. Nous n'avons pas alors l'ambition de développer des méthodes nouvelles ou d'adapter celles existantes pour permettre l'accomplissement de chacune des étapes, mais plutôt d'analyser l'existant en fonction du cadre conceptuel fourni par cette méthodologie. Ainsi, dans le paragraphe suivant nous analysons les fonctionnalités fournies actuellement par des outils de construction d'ontologie par rapport à chaque étape du processus d'évolution, le but étant ici d'identifier les besoins technologiques ressentis. Avant de présenter cette analyse, nous précisons cependant que nous envisageons de perfectionner cette esquisse méthodologique unifiée en partant de notre propre expérience des problèmes rencontrés lors de la mise en place d'une méthode pour assurer l'opérationnalisation d'une ontologie évolutive utilisée comme référentiel sémantique d'un système d'apprentissage (cf. chapitre 3 et 4).

3. Outils pour supporter la méthodologie d'évolution de l'ontologie

Nombreux outils de construction d'ontologies ont été développés ces dernières années (OntoWeb, 2002c). Notre analyse des fonctionnalités logicielles porte ainsi que sur ceux qui sont le plus connus et le plus utilisés actuellement.

KAON[14] (Karlshrue Ontology and Semantic Web framework) est actuellement la seule architecture informatique conçue pour permettre la construction et l'évolution des ontologies (Oberle et *al.*, 2004), les

[14]Disponible sur le site http://kaon.semanticweb.org

autres outils analysés étant dédiés principalement à la construction des ontologies. **OntoEdit**[15] (Sure et *al.*, 2002), développé par l'AIFB de Karlsruhe, a le mérite de s'appuyer sur une réflexion méthodologique significative, la méthodologie On-To-Knowledge. **PROTÉGÉ-2000**[16] (Noy, Fergerson, et Musen, 2000), développé par l'équipe SMI (Stanford Medical Informatics) de l'université de Stanford, est un environnement graphique de construction d'ontologies. **OILEd**[17] (Bechhofer, Horrocks, Goble, et Stevens, 2001), développé à l'université de Manchester, est dédié essentiellement à la construction des petites ontologies dans le langage de représentation OIL (Ontology Inference Layer) qui est un précurseur du langage OWL (Ontology Web Langage). **WebODE**[18] (Arpirez, Corcho, Fernandez-Lopez, et Gomez-Perez, 2001), développée par l'équipe de LAI (Laboratoire d'Intelligence Artificielle de la polytechnique de Madrid), est une plate-forme de conception d'ontologies fonctionnant en ligne qui implémente la méthodologie METHONTOLOGY.

Dans le tableau 2, nous analysons ces outils selon les fonctionnalités qui sont nécessaires pour supporter les neuf étapes du processus d'évolution de l'ontologie.

Tableau 2. Analyse des outils ontologiques selon le processus d'évolution de l'ontologie

Étape	Fonctionnalité	KAON Tool suite	OntoE.	Protégé 2000	OILEd	WebOde
Support méthodologique		Evolution process	On-To-Knowledge	Non	Non	Methontology
1	Librairie des (versions) d'ontologies	S/P	S/P	S/P	S/P	S/P
3	Fonctionnalités pour l'édition des changements élémentaires	Oui	Oui	Oui	Oui	Oui
	Fonctionnalités pour l'édition des changements complexes	S/P	S/P	Non	Non	Non
	Ensemble des stratégies d'évolution selon le type du changement	S/P	Non	Non	Non	Non
	Annotation des changements par un ensemble des métadonnées	S/P	S/P	S/P	S/P	S/P
4	Vérification de la consistance de l'ontologie	Oui	Oui (par OntoBroker)	Ouil (par PAL, FaCT)	Oui (par FaCT)	Oui (par OntoClean)
	Aide à la résolution des inconsistances introduites par les changements	Oui	Non	Non	Non	Non

[15] Une version de démonstration est disponible sur le site d'Ontoprise, la société qui développe cet outil en collaboration avec l'AIFB de Karlsruhe.
[16] Disponible sur les site http://protege.stanford.edu/index.shtml
[17] Disponible sur le site http://www.w3.org/2001/sw/WebOnt/
[18] Disponible sur le site http://delicias.dia.fi.upm.es/webODE/

5	Support au travail collectif avec gestion des conflits	S/P	S/P (par SWAP)	Non	Non	S/P
7	Journal de changements	S/P	S/P	S/P	Non	Non
	Identification des changements entre les versions	S/P	Non	Non	Non	Non
	Caractériser la relation sémantique entre les entités de V_N et celles de V_{N+1}	Non	Non	Non	Non	Non
	Analyse des effets des changements sur la compatibilité des versions	Non	Non	Non	Non	Non
9	Propagation des changements dans les objets référencés par l'ontologie	Non	Non	Non	Non	Non
	Propagation des changements dans les ontologies dépendantes	S/P	Non	Non	Non	Non
	Mappage (mise en correspondance) entre les versions ontologiques	Non	Non	S/P par AchorPrompt	Non	Non
	Gestion de l'accès aux objets référencés des versions multiples	Non	Non	Non	Non	Non

Légende : « OUI » - support complet, « S/P » - Support partiel, « Non » - aucun support

L'analyse d'outils ontologiques a mis en évidence plusieurs aspects, comme nous le montrons ci-après :

- *Étape 1* : Les outils fournissent uniquement des fonctionnalités pour accéder et pour stocker les ontologies sous la forme des fichiers, dans une base de données.
- *Étape 3* : Tous les outils possèdent des fonctionnalités pour éditer les changements élémentaires, mais seulement KAON et OntoEdit ont des fonctionnalités pour éditer un type de changement complexe (*Déplacer_Entité*). De plus, la plupart des outils offrent une seule possibilité pour résoudre les changements, par exemple le changement *Effacer_Concept* qui efface aussi tous les sous-concepts. Seulement KAON autorise les utilisateurs à adapter la façon dans laquelle certains changements seront résolus. En ce qui concerne l'annotation des changements, les outils permettent seulement la description textuelle d'une entité ontologique, mais ils ne donnent pas la possibilité d'annoter les changements en utilisant un ensemble de metadonnées spécifiques.
- *Étape 4* : Tous les outils assurent la vérification de la consistance de l'ontologie soit par l'outil lui-même, soit en utilisant les services d'un raisonneur comme FaCT (Horrocks, Sattler, et Tobies, 1999), qui utilise la logique de description pour vérifier la cohérence de l'ontologie, comme OntoBroker (Oberle, Wenke, Volz, et Staab, 2003), qui est un moteur d'inférence basé sur la logique de frames, ou comme OntoClean (Guarino et Welty, 2002), qui permet l'évaluation des taxonomies selon des notions comme

rigidité, unité, identité. Cependant, seulement KAON possède la capacité de résoudre un certain nombre des inconsistances trouvées, les autres outils ne proposant aucun support à cet effet.

- *Étape 5* : Seulement KAON et WebOde permettent aux utilisateurs de modifier la même ontologie en leur fournissant un mécanisme de synchronisation permettant la gestion des situations conflictuelles. Quant à OntoEdit, il utilise un plugiciel qui implémente le modèle de développement collaboratif défini par Pinto et *al.* (2004) (cf § 1.3).
- *Étape 7* : La plupart des outils possèdent un journal des changements, mais seul KAON est capable d'effectuer une identification des changements élémentaires. De plus, les outils ne fournissent aucun support pour l'analyse des effets des changements sur la relation entre les versions de l'ontologie.
- *Étape 9* : Les outils ne proposent aucun support pour la propagation des changements dans les objets référencés. Seul KAON fournit un support partiel pour la propagation des changements dans les ontologies dépendantes, mais il ne supporte pas la modification des mappages entre les ontologies afin de préserver leur interopérabilité avec l'ontologie évoluée. D'ailleurs, seulement Protégé-2000 peut supporter le mappage entre deux versions de l'ontologie au moyen du plugiciel AnchorPROMPT (Noy et Musen, 2002) qui est un outil graphique capable d'assister les utilisateurs dans l'identification des similarités entre différentes ontologies. Quant à la gestion de l'accès aux objets référencés par des versions multiples, les outils ne supportent pas cette tâche.

L'analyse des principaux outils a mis en évidence le manque des fonctionnalités importantes pour assurer le processus d'évolution de l'ontologie. La conclusion qui se dégage de cette analyse est alors la suivante : tant que les premières étapes du processus d'évolution sont plus ou moins supportées par les outils ontologiques, l'étape d'identification des changements et d'analyse des effets des changements sur la relation, sémantique et de compatibilité, entre les versions de l'ontologie (i.e. étape 7) et l'étape d'opérationnalisation de la nouvelle version (i.e. étape 9) ne sont aucunement supportées par les outils ontologiques. Toutefois, ces deux étapes sont essentielles pour assurer la préservation des rôles de l'ontologie, faute de quoi il se peut que la nouvelle version ne soit pas utilisable. Dans le chapitre suivant, nous abordons alors en profondeur les deux étapes en question.

Chapitre III : Opérationnaliser l'évolution de l'ontologie

Dans le chapitre précédent, nous avons mis en évidence le manque d'outils ayant des fonctionnalités spécifiques pour assurer l'identification des changements ontologiques ainsi que pour assurer une analyse des effets des ces changements sur la relation entre les versions d'une ontologie. Cependant, ces aspects sont essentiels dans le processus d'évolution de l'ontologie. Sans savoir de quelle manière les entités ontologiques ont été changées, à la suite de quels types de changements, ayant quels types d'effet, la préservation des rôles de l'ontologie, par exemple celui de l'ontologie utilisée comme référentiel sémantique d'objets du Web, est difficilement réalisable, sinon impossible :

> « The evolution of ontologies causes operability problems, which will hamper the effective reuse. The problem is even worse, because there are dependencies between data sources and the ontologies. Therefore, a method is needed to handle revisions of ontologies and the impact on existing Web sources. (…) There are a lot of things that are not yet done. We think that the most important flaw is the lack of a detailed analysis of the effect of specific changes on the interpretation of data.» (Klein et Fensel, 2001).

Afin de supporter l'analyse des changements effectués pour passer d'une version de l'ontologie à une autre ainsi que la préservation des rôles de l'ontologie qui, dans notre contexte particulier de recherche, signifie la préservation de l'utilisation d'une ontologie évolutive comme référentiel sémantique des éléments pédagogiques d'un système d'apprentissage, nous avons défini deux ***objectifs spécifiques du projet de doctorat*** :

1. une méthode d'identification des changements ontologiques et d'analyse des effets des changements sur l'interprétation des entités ontologiques ainsi que sur la relation de compatibilité entre les versions d'une ontologie. Cette méthode sera implémentée dans un outil informatique que nous nommons *OntoAnalyseur*.
2. une méthode pour assurer un référencement sémantique évolutif réalisé par la modification et la mise à jour des liens de références, c'est-à-dire le lien qui relie un objet à sa référence dans un espace ontologique, pour les objets référencés à l'aide des entités d'une ontologie évolutive. Cette méthode sera implémentée dans un outil informatique que nous nommons *UKIsModificateur.*

1. Identification des changements et analyse des effets des changements **(objectif spécifique 1)**

Dans cette section, nous abordons le premier objectif spécifique du projet de doctorat. Nous commençons par discuter les approches existantes concernant l'identification et l'analyse des effets des changements

ontologiques et, après avoir mis en évidence leurs limites respectives, nous proposons notre méthode supportée par OntoAnalyseur.

1.1 Identification et analyse des effets des changements ontologiques : l'état de la question

1.1.1 Identification des changements ontologiques

Les outils capables d'identifier des changements ontologiques sont actuellement plutôt inexistants. En conséquence, nous avons identifié que trois outils ayant quelques fonctionnalités à cet effet (Kaon, OntoView et PromptDiff).

KAON (Maedche, Motik, et Stojanovic, 2003) enregistre dans un journal tous les changements exécutés pendant l'évolution de l'ontologie. Dans le journal, les changements sont décrits à l'aide des concepts d'une ontologie *o:ChangeLog* qui spécifie seulement des changements élémentaires de type *o:AddEnitity* ou *o:DeleteEntity* ou encore *o:ModifyEntity*. En conséquence, la trace des changements fournie par le journal ne peut pas rendre compte des changements complexes, par exemple fusionner ou séparer des entités ontologiques, ce qui est une limitation[19] importante sachant que le but est d'analyser les effets de changements sur la compatibilité entre les versions de l'ontologie ou sur la relation sémantique entre leurs entités. De plus, KAON enregistre les changements dans l'ordre de leur exécution, ceci nécessitant une analyse supplémentaire qui consiste, par exemple, à regrouper les changements en fonction de leur type ou en fonction des entités sur lesquelles ils agissent.

Pour identifier des changements ontologiques, *OntoView* (Klein, 2002b; Klein, Fensel, Kiryakov, et Ognyanov, 2002) compare les versions d'une ontologie au niveau structurel, c'est-à-dire au niveau de définitions de concepts et propriétés, afin de découvrir les définitions modifiées d'une version à l'autre. Pour chaque définition modifiée, OntoView produit ensuite une liste de changements considérés comme étant nécessaires pour passer de la définition de l'entité ontologique en V_N à la définition de cette entité en V_{N+1}. Ce type d'identification, contrairement à celui utilisé dans KAON, ne nécessite aucune information supplémentaire au sujet du processus d'évolution de l'ontologie. Ceci est un grand avantage dans le contexte du Web sémantique où il est difficile de prévoir un journal des changements pour chaque version de l'ontologie. Toutefois, de même comme KAON, OntoView est capable d'identifier uniquement des changements élémentaires de type ajout ou effacement des entités ontologiques. Une autre caractéristique d'OntoView est qu'il permet aux utilisateurs de préciser le fait qu'un changement a modifié la signification

[19] Pour un exemple simple, considérons la séquence des changements : Effacer_Concept_ID='Professeur', Effacer_Concept_ID='Tuteur', Ajouter_Concept_ID='Facilitateur'. Un système pourrait-il déduire, seulement en analysant la trace des changements fournie par le journal, qu'il s'agit de fusionner les concepts 'Formateur' et 'Tuteur' dans un nouveau concept 'Facilitateur' et qu'en conséquence, la nouvelle version est rétrocompatible avec la version initiale ? À notre avis, la réponse est non.

d'une entité ontologique, cette entité étant alors conceptuellement différente de celle de V_N, ou qu'un changement a seulement enrichi la signification d'une entité ontologique, cette entité restant alors identique avec celle de V_N. Néanmoins, il ne possède aucune fonctionnalité pour caractériser la relation sémantique entre les entités ontologiques de V_N et celles de V_{N+1} en termes d'équivalence, d'inclusion ou encore de spécification/généralisation.

PromptDiff (Noy et Musen, 2002, 2003b) effectue lui aussi une comparaison entre les versions d'une ontologie sans avoir besoin d'une trace du processus d'évolution. En mettant en correspondance les entités ontologiques d'une version avec celles d'une autre version, PromptDiff indique seulement si les entités ont été modifiées ou non. Nous ne pouvons pas alors parler d'une technique d'identification des changements étant donné que l'outil ne fournit aucune information au sujet des changements ontologiques. Cette tâche revient à l'humain, l'outil ne fait qu'attirer l'attention des utilisateurs sur les entités ontologiques modifiées.

1.1.2 Analyse des effets des changements ontologiques

À notre connaissance, il n'existe actuellement ni des études théoriques complètes, ni des outils capables d'analyser les effets de changements sur la relation sémantique entre des entités ontologiques ou sur la compatibilité entre les versions de l'ontologie. Seuls deux groupes de chercheurs abordent ce sujet, mais ils restent assez brefs dans leur analyse.

Stuckenschmidt et Klein (2003a, 2003b) proposent une analyse des effets des changements sur la classification des concepts dans la hiérarchie, en indiquant principalement les situations où les concepts deviennent plus spécialisés ou plus généralisés sous l'effet des changements qui ajoutent, effacent ou modifient des entités ontologiques. Toutefois, les auteurs ne fournissent aucune analyse des effets des changements sur la compatibilité entre les différentes versions d'une ontologie. De plus, ils analysent que des changements élémentaires ce qui est insuffisant pour une interprétation pertinente de la modification conceptuelle introduite dans la nouvelle version de l'ontologie[20].

Heflin et Hendler (2000) et Heflin et *al.* (1999) analysent les effets de changements qui ajoutent ou effacent des entités ontologiques. Les auteurs mettent ainsi en évidence le fait que tout changement de type Effacer_Entité peut produire des inconsistances conduisant à une altération de l'accès aux objets référencés par V_{N+1} ou de leur interprétation, et que tout changement de type Ajouter_Entité n'affecte ni l'interprétation, ni l'accès aux objets référencés et, dans ce cas, V_{N+1} est rétrocompatible avec V_N. Bien que les auteurs abordent les effets de changements sur la compatibilité de V_{N+1} et sur la préservation de l'accès aux objets référencés, cette analyse est incomplète étant donné qu'elle abordent uniquement les

[20] Les développeurs peuvent utiliser des changements élémentaires pour exprimer des modifications complexes (voir note 19) mais cela reste implicite, donc inaccessible à un système informatique de raisonnement.

changements élémentaires et qu'elle ne distingue non plus entre le changement d'un concept de celui d'une propriété, sachant que l'effet produit par un ou par l'autre peut être différent (cf. § 1.2.3).

1.2 OntoAnalyseur : une solution complète

La conclusion qu'on peut tirer de la discussion ci-dessus, est qu'actuellement aucun outil ne permet ni l'identification des changements complexes, ni l'analyse des effets des changements sur la relation entre les versions de l'ontologie. En conséquence, nous proposons une méthode d'analyse qui, implémentée à OntoAnalyseur, devrait lui permettre de fournir les services suivants :

1. identifier les changements élémentaires et complexes exécutés pour passer de V_N à V_{N+1} (cf. § 1.2.1).
2. caractériser la relation sémantique entre les entités ontologiques appartenant à V_N et celles appartenant à V_{N+1} (cf. § 1.2.2).
3. analyser le type de compatibilité de V_{N+1} et identifier les changements totalement compatibles, rétrocompatibles et incompatibles (cf. § 1.2.3).

Dans la section suivante, notre but est de présenter les techniques utilisées par OntoAnalyseur pour fournir les services mentionnés ci-dessus, de les illustrer brièvement ainsi que de mettre en évidence les travaux qui restent à faire ou les obstacles qui restent à franchir.

1.2.1 Identification des changements par OntoAnalyseur

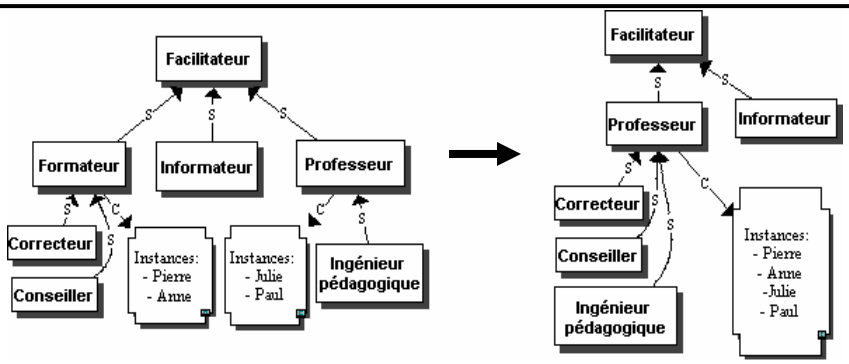
Étant donné le caractère dynamique et distribué du Web sémantique, il est indispensable de considérer deux situations pour l'identification des changements ontologiques : (1) évolution de l'ontologie sans trace des changements et (2) évolution de l'ontologie avec trace des changements. Étant donné que l'ensemble des changements possibles dépend du modèle ontologique sous-jacent, nous envisageons de permettre à OntoAnalyser d'identifier les changements à partir des ontologies formalisées en OWL-DL.

1.2.1.1 Identification des changements sans trace d'évolution

L'identification des changements doit s'effectuer seulement en fonction des versions V_N et V_{N+1}, aucune trace des changements n'étant fournie à OntoAnalyseur. L'obstacle à franchir est alors de permettre à cet outil d'identifier des changements complexes, par exemple fusionner, séparer ou déplacer des entités ontologiques, en plus des changements élémentaires.

Une solution peut être fournie par les techniques de fusion d'ontologies, par exemple ONION (Calvanese, Giacomo, et Lenzerini, 2001) ou PROMPT (Noy, 2004; Noy et Musen, 2000, 2003b). Ces techniques identifient des similarités entre ontologies différentes, en se basant sur un ensemble de règles heuristiques, par exemple deux entités ayant le même type, la même place dans la hiérarchie et le même nom peuvent être considérées comme proches sémantiquement, ou deux entités ayant le même ensemble

des instances peuvent être considérées comme équivalentes. Nous faisons l'hypothèse que l'application des techniques de fusion peut s'avérer efficace à l'identification des changements complexes, dans la mesure où nous envisageons de les appliquer à des versions d'une même ontologie, donc portant sur le même domaine des connaissances. Pour illustrer notre propos, nous présentons une règle[21] d'identification d'un changement complexe qui fusionne les concepts *'Professeur'* et *'Tuteur'* dans un seul concept *'Professeur'* (figure 4). Nous avons développé cette règle d'identification en adaptant les deux règles heuristiques présentées comme exemples pour les techniques de fusion.



Règle 1 : Identification du changement « Fusionner_Concepts dans un concept ayant le même nom qu'un des concepts fusionnés »
Soit B, un concept de V_N et non de V_{N+1} et soit $I_{B(N)}$, l'ensemble des instances de B dans V_N Soit $SC_{B(N)}$, l'ensemble des sous-concepts de B dans V_N Soit A, un concept de V_N et de V_{N+1} et soit $I_{A(N+1)}$, l'ensemble des instances de A dans V_{N+1} Soit $SC_{A(N+1)}$, l'ensemble des sous-concepts de A dans V_{N+1}
1 : Pour tout B
2 : Si $I_{B(N)}$ est inclus dans $I_{A(N+1)}$ 3 : Si $SC_{B(N)}$ est inclus dans $SC_{A(N+1)}$
4 : Alors B est fusionné avec A pendant l'évolution de V_N à V_{N+1} et le résultat dans V_{N+1} est A

Figure 4. Règle heuristique d'identification d'un changement de type Fusionner_Concepts

Afin de démontrer la validité de notre hypothèse, nous envisageons de définir des règles pour un certain nombre de changements complexes : (1) fusion des concepts ou des propriétés, (2) séparation d'un concept ou d'une propriété, (3) déplacement d'un concept, d'une propriété ou d'une instance (déplacer transversalement, c'est-à-dire sur le même niveau taxonomique, et orthogonalement, c'est-à-dire sur des niveaux taxonomiques différents).

1.2.1.2 Identification des changements avec trace d'évolution intégrée

Dans ce cas, la trace des changements est accessible à OntoAnalyseur. Nous proposons une manière innovante de rendre accessible cette trace : les changements exécutés pour passer de V_N à V_{N+1} sont intégrés explicitement dans la version V_{N+1} représentée en OWL. Nous illustrons notre propos dans la figure

[21] Nous présentons cette règle à titre explicatif, nous ne certifions pas sa complétude.

5. Soit dans V_N, les deux concepts *'Formateur'* et *'Professeur'* représentés en langage OWL. Ces concepts sont fusionnés pour obtenir le concept *'Professeur'*. Dans la version V_{N+1}, l'information concernant ce changement apparaît directement dans la définition du concept *'Professeur'* où : la primitive *OntologyChangeResult* indique que le concept *'Professeur'* est le résultat du changement *Merge_Classes* appliqué aux concepts *'Facilitateur'* et *'Professeur'* de la version ayant l'URI *ActeursTA_3*.

Version V_N ayant l'URI *ActeursTA_3*	**Version V_{N+1} ayant l'URI** *ActeursTA_4*
<owl:Class rdf:ID="Formateur"/> <rdfs:subClassOf rdf:resource="#Facilitateur" /> </owl:Class> <owl:Class rdf:ID="Professeur"/> <rdfs:subClassOf rdf:resource="#Facilitateur" /> </owl:Class>	<owl:Class rdf:ID="Professeur"/> <rdfs:subClassOf rdf:resource="#Facilitateur" /> <oc:OntologyChangeResult/> <oc:Merge_Classes/> <oc:class rdf:resource="ActeursTA_3#Facilitateur" /> <oc:class rdf:resource="ActeursTA_3#Professeur" /> </oc:Merge_Classes/> </oc:OntologyChangeResult/> </owl:Class>

Figure 5. Exemple de trace des changements intégrée dans V_{N+1}

Comme nous avons montré dans l'exemple ci-dessus, nous obtenons la nouvelle version de l'ontologie avec toutes les informations concernant l'évolution de V_N à V_{N+1} déjà intégrées dans la version V_{N+1}, ce qui permet une identification facile des changements élémentaires et complexes. En la comparant avec la trace fournie par un journal des changements, cette trace intégrée est plus intéressante étant donné qu'elle permet d'ordonnancer les changements en fonction des entités sur lesquelles s'appliquent et d'expliciter ainsi, de manière formelle, l'intention du développeur ayant accompli un certain changement. De plus, la trace intégrée des changements ne nécessite aucun stockage séparé des versions de l'ontologie, ceci étant un avantage considérable dans un environnement distribué comme le Web sémantique.

La principale question posée par ce type d'identification est de savoir de quelle manière représenter la trace d'évolution à l'intérieur de V_{N+1}. Nous envisageons alors de proposer un formalisme, que nous nommons OC (*OntologyChange*), permettant l'intégration des informations de changements élémentaires et complexes dans une version ontologique, tel que nous l'avons illustrée dans la figure 5. Ce formalisme se veut une extension de OWL, car un langage de représentation des ontologies du Web doit pouvoir s'accommoder aux changements ontologiques (WebOnt, 2004b).

1.2.2 Découverte des relations sémantiques entre entités ontologiques par OntoAnalyseur

La dimension sémantique doit être prise en compte dans la comparaison entre les versions d'une ontologie évolutive (Charlet et *al.*, 2003). Ainsi, pour fournir une caractérisation pertinente de la relation entre deux versions, nous devons considérer plus que la simple relation d'identité ou non identité entre leurs entités

ontologiques (cf. § 1.1.1). Nous proposons alors une technique, qui, implémentée dans OntoAnalyseur, devrait permettre l'identification automatique des relations sémantiques entre les entités ontologiques appartenant à V_N et celles appartenant à V_{N+1}, selon des critères comme : identité, équivalence, inclusion, généralisation, spécialisation ou conceptuellement différentes. Par l'utilisation du terme 'automatique', nous n'affirmons pas qu'OntoAnalyseur puisse déduire les relations sémantiques *correctes et valides*, cela reste sans doute l'apanage des humains. Par contre, nous faisons l'hypothèse qu'OntoAnalyser pourrait identifier des relations sémantiques qui peuvent s'avérer exactes, en utilisant un ensemble de règles heuristiques comme nous le montrons dans le tableau 3.

Tableau 3 : Règles de découverte des relations sémantiques entre entités ontologiques

Règle 1 : Identification de la relation « Inclut_Entité »
Énoncé : Soit A_N et B_N deux concepts appartenant à V_N. Si A_N et B_N ont été fusionnés et le résultat dans V_{N+1} est le concept C_{N+1} alors, C_{N+1} *inclut_entités* A_N et B_N.
Règle 2 : Identification de la relation «Est_ Inclus »
Énoncé : Soit A_N un concept appartenant à V_N. Si A_N est séparé et les résultats dans V_{N+1}sont les concepts B_{N+1} et C_{N+1} alors B_{N+1} *est_inclus* dans A_N et C_{N+1} *est_inclus* dans A_N.
Règle 3 : Identification de la relation «Généralisation »
Énoncé : Soit A_N un concept appartenant à V_N. Si un axiome de superclasse a été effacé de la définition de A_N alors le résultat A_{N+1} dans V_{N+1} *est_une_généralisation* de A_N
Règle 4 : Identification de la relation «Spécialisation »
Énoncé : Soit A_N et B_N des concepts appartenant à V_N où B_N est une superclasse de A_N. Si B_N est déplacé plus bas dans la hiérarchie alors le résultat A_{N+1} dans V_{N+1} *est_une_spécialisation* de A_N

En fonction des types de changements élémentaires et complexes, nous envisageons d'étudier les effets de changements sur la relation sémantique entre les entités ontologiques, de définir un ensemble de règles de découverte et de les implémenter à OntoAnalyseur.

1.2.3 Analyse de la compatibilité par OntoAnalyseur

Heflin (2001) fournit la preuve que les changements qui ajoutent des entités à l'ontologie produisent une version rétrocompatible, tant que les changements qui effacent des entités produisent une version incompatible. Cette conclusion est exacte, mais elle ne prend pas en compte le fait que « *defining the compatibility between different ontology versions becomes a salient issues since there are several dimensions to compatibility* » (Noy et Klein, 2003a). Prenons un exemple. Supposons V_N où les concepts *'Professeur'* et *'Tuteur'* sont déclarés disjoints à l'aide d'un axiome de classe. Dans ce cas, un moteur d'inférence peut conclure que toute instance du concept *'Professeur'* est différente de toute instance du concept '*Tuteur'*. Supposons maintenant V_{N+1}, où cet axiome a été effacé. Dans ce cas, le moteur d'inférence utilisant V_{N+1} ne peut plus arriver à la conclusion ci-dessus, ce qui provoque une altération de son comportement. Par contre, si l'ontologie n'est pas utilisée par un moteur d'inférence, mais elle est utilisée uniquement pour servir comme référentiel des connaissances (concepts), alors l'effacement de

l'axiome en question ne provoque aucune altération de l'accès aux objets référencés au moyen des concepts de l'ontologie.

En conséquence, en fonction des rôles de l'ontologie, nous avons besoin de considérer plusieurs dimensions de l'analyse de la compatibilité. Nous proposons alors une analyse de la compatibilité selon trois dimensions (tableau 4) : (1) la préservation des instances de l'ontologie, (2) la préservation de la structure conceptuelle de l'ontologie, (3) la préservation de l'axiomatisation de l'ontologie.

Tableau 4. Analyse de la compatibilité selon trois dimensions

Type de résultat de l'analyse	Situation d'occurrence pour le type de résultat	Changements provoquant le type de résultat : exemples
Préservation des instances de l'ontologie		
Préservation des instances sans ajout P_{IS}	L'ensemble des instances de V_N est le même que celui de V_{N+1}	- Fusionner ou Séparer_Concept - Déplacer ou Effacer_Propriété
Préservation des instances avec ajout P_{IA}	L'ensemble des instances de V_N est élargi en V_{N+1}	- Ajouter_Instance - Ajouter_Instance de propriété
Non-préservation des instances P_{IN}	L'ensemble des instances de V_N est réduit ou modifié en V_{N+1}	- Effacer_Concept, si le concept a des instances - Effacer_Instance
Préservation de la structure conceptuelle de l'ontologie		
Préservation de la structure conceptuelle sans ajout P_{CS}	L'ensemble des concepts de V_N est le même que celui de V_{N+1}	- Déplacer ou Effacer_Instance - Déplacer ou Effacer_Propriété
Préservation de la structure conceptuelle avec ajout P_{CA}	L'ensemble des concepts de V_N est élargi en V_{N+1}	- Ajouter_Concept - Reclassifier une instance comme un concept
Non-préservation de la structure conceptuelle P_{CN}	L'ensemble des concepts de V_N est réduit ou modifié en V_{N+1}	- Fusionner_Concepts - Séparer_Concept
Préservation de l'axiomatisation de l'ontologie		
Préservation de l'axiomatisation sans ajout P_{AS}	L'ensemble des faits inférés à partir de V_{N+1} est le même que l'ensemble des faits inférés à partir de V_N	- Modifier ou Déplacer_Concept - Déplacer ou Effacer_Instance
Préservation de l'axiomatisation avec ajout P_{AA}	L'ensemble des faits inférés à partir de V_{N+1} est un super ensemble de l'ensemble des faits inférés à partir de V_N	- Ajouter_Propriété, Axiome - Déplacer la propriété (axiome) d'un concept au concept-parent
Non-préservation de l'axiomatisation P_{AN}	L'ensemble des faits inférés à partir de V_{N+1} est réduit ou modifié par rapport à l'ensemble des faits inférés à partir de V_N	- Effacer_Propriété, Axiome - Déplacer la propriété (axiome) d'un concept à son sub-concept

En prenant en compte les trois dimensions de l'analyse de la compatibilité, nous pouvons classifier les changements exécutés pour passer de V_N à V_{N+1} en :

- *changements totalement compatibles* sont les changements qui préservent entièrement les rôles de l'ontologie. Généralement, l'ensemble des changements totalement compatibles, que nous notons C_{ch}, exécutés pour passer de V_N à V_{N+1} peut être calculé de la manière suivante :

$$C_{ch} = \left(P_{IS} \bigcap P_{CS} \bigcap P_{AS}\right)$$

- ***changements rétrocompatibles***. L'ensemble des changements rétrocompatibles, que nous notons BC_{ch} , peut être calculé de la manière suivante :

$$BC_{ch} = \left(P_{IA} \cup P_{CA} \cup P_{AA}\right) - \left(P_{IN} \cup P_{CN} \cup P_{AN}\right).$$

  Les changements rétrocompatibles préservent les rôles de l'ontologie, mais le nouveau contenu, introduit en V_{N+1} au moyen de ces changements, n'est pas utilisé pour élargir les rôles de l'ontologie, par exemple pour référencer les éléments pédagogiques d'un système d'apprentissage en fonction des nouvelles connaissances introduites à l'ontologie.

- ***changements incompatibles*** sont les changements qui ne préservent pas les rôles de l'ontologie. L'ensemble des changements incompatibles, que nous notons IC_{ch} , peut être calculé ainsi :

$$IC_{ch} = \left(P_{IN} \cup P_{CN} \cup P_{AN}\right).$$

Selon les rôles de l'ontologie, il est possible que les trois dimensions d'analyse ne soient pas toutes nécessaires pour analyser la compatibilité de la nouvelle version. Il faudrait alors calculer l'ensemble des changements selon les formules ci-dessus en considérant seulement la ou les dimensions nécessaires. Nous envisageons alors d'implémenter dans OntoAnalyseur les trois procédures du calcul, ainsi qu'une classification des changements ontologiques selon les dimensions d'analyse. En fonction des dimensions d'analyse demandées par la préservation des rôles de l'ontologie, OntoAnalyseur aura la capacité de :

- calculer les ensembles des changements totalement compatibles, incompatibles ou rétrocompatibles.
- identifier le type de compatibilité de V_{N+1}, sachant que (1) s'il existe au moins un changement incompatible alors V_{N+1} est incompatible avec V_N, (2) s'il n'existe aucun changement incompatible, mais il existe au moins un changement rétrocompatible alors V_{N+1} est rétrocompatible avec V_N et (3) s'il n'existe aucun changement incompatible ou rétrocompatible alors V_{N+1} est totalement compatible avec V_N.

2. Référencement sémantique évolutif (objectif spécifique 2)

Rendre utilisable la nouvelle version de l'ontologie ne signifie pas cependant qu'elle soit aussi opérationnelle, c'est-à-dire que les rôles de l'ontologie sont préservés (Ding et *al.*, 2004). Comme nous l'avons montré ci-dessus, tout changement incompatible peut produire des inconsistances conduisant à l'altération des rôles de l'ontologie. Quant aux changements rétrocompatibles, les rôles de l'ontologie sont préservés, mais le nouveau contenu, ajouté à l'ontologie, n'est pas utilisé pour assurer leur élargissement. Dans notre contexte particulier de recherche, l'ontologie est utilisée comme référentiel sémantique pour les éléments pédagogiques qui composent un système d'apprentissage sur le Web sémantique. Pour préserver

ce rôle de l'ontologie, nous proposons une méthode qui propage les changements ontologiques, effectués pour passer de V_N à V_{N+1}, dans les objets référencés par l'ontologie. La propagation des changements dans les objets référencés signifie la modification des liens de références, un lien de référence étant le chemin qui relie un objet à une entité ontologique utilisée comme référence sémantique.

2.1 Référencement sémantique sur le Web : l'état de la question

Actuellement, un problème majeur se pose dans le référencement sémantique des ressources du Web, celle de pouvoir maintenir leurs liens de références à une ontologie évolutive :

> « interlinkage between objects and evolving ontologies need to be managed (…) because incoming information that is to be annotated does not only require some more annotating, but also continuous adaptation to new semantic terminology». (Staab, Maedche, et Handschuh, 2001).

Une gamme assez large d'outils pour permettre le référencement sémantique des objets du Web est proposée à l'heure actuelle. À titre d'exemple, nous mentionnons les outils COHESE (Bechhofer et Goble, 2001) et MnM (Motta, Vargas-Vera, Domingue, Lanzoni, et Ciravegna, 2002) qui facilitent le référencement des pages Web en utilisant des balises HTML construites à partir des ontologies et l'outil SHOE Knowledge Annotator (Heflin, 2001) qui permet d'annoter des pages Web avec des instances d'ontologies décrites avec SHOE, un langage qui ajout au HTML des marqueurs spécifiques. Malgré cela, aucun outil ne possède des fonctionnalités pour la gestion du référencement sémantique en fonction du changement des ontologies (OntoWeb, 2002c). Seul le modèle d'architecture CREAM (Creating RElational, Annotation-based Metadata) propose des recommandations générales à cet effet (Handschuh, Staab, et Maedche, 2001; L. Stojanovic, Stojanovic et *al.*, 2002). D'ailleurs, les outils OntoMat-Annotizer et OntoAnnotate (Staab et *al.*, 2001), qui implémentant tous les deux le modèle CREAM, sont les seuls qui prennent en compte, dans leurs perspectives de développement, la maintenance semi-automatique du référencement. Pourtant, cette fonctionnalité est essentielle dans le domaine du Web sémantique. En conséquence, nous proposons dans le paragraphe suivant une solution pour assurer un référencement sémantique évolutif des objets référencés à l'aide des entités d'une ontologie évolutive.

2.2 UKIsModificateur – solution pour le référencement sémantique évolutif

Nous définissons un *UKI (Uniform Knowledge Identifier)*, comme une chaîne compacte des caractères qui explicite le lien de référence reliant un objet référencé à son repère sémantique dans un espace de connaissances, ce repère étant identifié avec certitude, de manière claire et unique. Pour illustrer, considérons le UKI suivant *"http://www.w3.org/ActeursTA_3.0#Formateur",* où *http://www.w3.org/* donne le chemin d'accès à l'ontologie, *ActeursTA_3.0* donne le nom et la version (3.0) de l'ontologie et *#Formateur*

identifie, par son nom (ID), le concept utilisé comme référence. Dans une ontologie OWL, une entité ontologique peut avoir seulement un ID unique, c'est-à-dire qu'aucune entité de l'ontologie ne peut être identifiée par l'ID utilisé par une autre entité.

Afin de préserver le rôle d'une ontologie évolutive utilisée comme référentiel sémantique, nous proposons une méthode de modification des UKIs des objets référencés, en fonction des changements exécutés pour passer de V_N à V_{N+1}. Cette méthode, implémentée dans UKIsModificateur, lui permettra de fournir les services suivants :

1. Modifier les UKIs des objets pour permettre à ces objets de référer à la version V_{N+1} tout en préservant leur interprétation acquise à travers les références sémantiques (technique de modification des UKIs présentée dans la section 2.2.1).
2. Mise à jour des UKIs (modes de mise à jour des UKIs présentés dans la section 2.2.2).

Nous envisageons de développer une méthode de modification et de mise à jour des UKIs qui soit générique, permettant ainsi à UKIsModificateur d'être utilisé dans plusieurs contextes. Cette généricité est possible étant donné que les changements ontologiques ne sont pas spécifiques à un domaine, au contraire, ils sont des changements transversaux à tous les domaines où l'évolution de l'ontologie s'avère nécessaire. Toutefois, nous précisons que le système UKIsModificateur sera utilisé, dans un premier temps, dans le contexte de l'ontologie utilisée comme référentiel sémantique des systèmes d'apprentissage.

2.2.1 Modification des UKIs par UKIsModificateur

Nous faisons l'hypothèse, qu'étant donné la richesse des informations fournies par OntoAnalyseur, la modification des UKIs pourrait se faire d'une manière pertinente, comme nous le montrons par la suite. En considérant les changements exécutés pour passer de V_N à V_{N+1} (1er service d'OntoAnalyseur) ainsi que leurs effets sur la compatibilité de V_{N+1} (3ième service d'OntoAnalyseur), nous pouvons rencontrer trois situations concernant la modification des UKIs. Pour illustrer ces situations, considérons l'exemple d'une banque d'éléments pédagogiques référencés à l'aide des concepts d'une ontologie, où la seule dimension nécessaire pour l'analyse de la compatibilité est la préservation de la structure conceptuelle de la nouvelle version de l'ontologie. Les trois situations sont les suivantes :

1) Pour les changements incompatibles qui n'affectent pas les UKIs, car ceux-ci référent à des entités ontologiques non touchées par ces changements, le UKIsModificateur modifie seulement le nom[22] de la version. Par exemple, pour le UKI *"http://www.w3.org/ActeursTA_3.0#Formateur",* le nom *ActeursTA_3.0* de la version V_N est remplacé par le nom *ActeursTA_4.0* de la version V_{N+1}.

[22] Ce nom est bien l'URI de la version. Pour alléger la compréhension du texte, nous avons cependant préféré de remplacer l'URI par le nom de la version.

2) Pour les changements incompatibles qui affectent les UKIs, le UKIsModificateur modifie le nom de la version, mais aussi le repère sémantique à l'intérieur de V_{N+1}. Prenons l'exemple d'un changement qui fusionne les concepts *'Formater'* et *'Professeur'* afin d'obtenir le concept *'Professeur'* dans la version *ActeursTA_4.0*. La relation sémantique entre les entités ontologiques (2ième service d'OntoAnalyseur) indique alors que le concept *'Professeur'* de la version *ActeursTA_4.0* inclut les concepts *'Professeur'* et *'Formateur'* de la version *ActeursTA_3.0*. En conséquence, la modification des UKIs peut se faire comme nous montrons ci-dessus.

UKIs référant à V_N	**UKIs modifiés, référant à V_{N+1}**
http://www.w3.org/ActeursTA_3.0#Formateur	*http://www.w3.org/ActeursTA_4.0#Professeur*
http://www.w3.org/ActeursTA_3.0#Professeur	*http://www.w3.org/ActeursTA_4.0#Professeur*

3) Pour les changements compatibles et ceux rétrocompatibles, les UKIs des objets ne sont pas affectés. Dans ce cas, le UKIsModificateur change seulement le nom de la version dans les UKIs. Toutefois, pour prendre en compte les nouvelles connaissances introduites dans la version V_{N+1} par les changements rétrocompatibles il faudrait définir, au besoin, de nouveaux UKIs.

Afin de mettre au point la technique de modification des UKIs, le principal obstacle à franchir est celui de savoir de quelle façon modifier les UKIs atteints par des changements incompatibles. Nous envisageons alors de définir un ensemble de règles qui permettront la modification des UKIs en fonction des informations fournies par OntoAnalyseur. Une autre question est celle de savoir de quelle manière UKIsModificateur aura accès aux UKIs à modifier, c'est-à-dire il va aller chercher les UKIs ou ceux-ci lui seront fournis par les clients externes utilisant l'ontologie comme référentiel sémantique ? À une première vue, nous préférions la deuxième solution, mais cela reste à discuter.

2.2.2 Modes de mise à jour des UKIs par UKIsModificateur

Dans la section précédente, nous avons présenté la technique de modification des UKIs en fonction des informations fournies par OntoAnalyseur. Par contre, il est possible que certains changements incompatibles ne puissent pas être propagés automatiquement dans les UKIs, ou encore, même si la propagation est possible il se peut que sa pertinence puisse être questionnée par les utilisateurs. De plus, les connaissances nouvelles, introduites par les changements rétrocompatibles, ne peuvent pas être propagées dans le UKIs sans l'intervention des utilisateurs. Nous proposons alors trois modes de mise à jour des UKIs, à être supportés par UKIs Modificateur :

1) *Mise à jour automatique*. Ce mode est utilisé pour les changements incompatibles qui n'affectent pas les UKIs ainsi que pour les changements compatibles et ceux rétrocompatibles. Quant aux changements incompatibles qui affectent les UKIs, la mise à jour automatique dépend du degré de pertinence de la modification proposée : si le degré est élevé, par exemple pour le changement qui

fusionne deux concepts, alors la mise à jour peut se faire automatiquement, sinon il est nécessaire de faire appel aux utilisateurs pour valider la modification proposée pour un ou plusieurs UKIs.

2) *Mise à jour semi-automatique*. Ce mode est utilisé quand les modifications, proposées pour les UKIs affectés par des changements incompatibles, possèdent un degré de pertinence discutable. Considérons par exemple, un changement qui efface un concept se trouvant au milieu d'une hiérarchie des concepts. En faisant l'hypothèse que le concept-parent englobe la signification de ses concepts-enfant, un UKI peut être modifié pour référer au concept-parent du concept effacé. Toutefois, pour garantir la pertinence de cette modification, l'approbation de l'utilisateur est nécessaire. Dans ce cas, la mise à jour est faite automatiquement par UKIsModificateur, mais seulement après avoir obtenu l'approbation de l'utilisateur.

3) *Mise à jour manuelle avec conseils*. Ce mode est utilisé pour tout autre type de situation que celles présentées ci-dessus : (a) pour prendre en compte les nouvelles connaissances introduites par des changements rétrocompatibles, (b) pour les changements incompatibles dont la modification des UKIs ne peut pas être automatisée (p.ex. effacer une hiérarchie entière des concepts) et (c) pour les changements incompatibles dont la modification des UKIs selon le mode 2 n'a pas été approuvée par les utilisateurs. Dans ce mode, les UKIs sont modifiés par les utilisateurs qui reçoivent un support au niveau de l'interface, par exemple en leur signalant visuellement les changements rétrocompatibles et incompatibles, mais aussi des conseils les guidant dans leur tâche.

Pour mettre au point les modes de mise à jour, il faudrait nous questionner premièrement sur le degré de pertinence des modifications des UKIs affectés par des changements incompatibles, et deuxièmement sur l'aide à apporter aux utilisateurs pour leur faciliter la modification manuelle des UKIs ainsi que la compréhension ultérieure des modifications apportées, par exemple en associant aux UKIs des métadonnées explicitant ces modifications.

Dans ce chapitre, nous avons discuté les objectifs spécifiques de notre projet de recherche ainsi que des solutions pour atteindre ces objectifs. Dans le chapitre suivant, nous allons donc expliquer comment nous envisageons de développer et valider les solutions proposées.

Chapitre IV : Méthodologie de recherche

La méthodologie retenue pour atteindre nos objectifs de recherche repose sur la recherche-développement (Van der Maren, 1996). De plus, nous empruntons au génie logiciel la technique de conception itérative, connue sous le nom de RUP (*Rational Unified Process*) (Larman, 2002), afin de nous permettre de définir une démarche de recherche itérative, axée sur la croissance et l'affinement successif des solutions conceptuelles proposées et des composants informatiques au moyen des itérations multiples.

1. Organisation de la démarche de recherche-développement

Dans l'intention d'atteindre nos objectifs, nous utilisons une démarche de recherche-développement itérative. Pour chaque itération nous choisissons un sous-objectif spécifique, par exemple l'identification des changements complexes, auquel correspond une des solutions générales déjà élaborées, par exemple l'identification avec trace d'évolution intégrée, le but étant de concevoir, implémenter informatiquement et tester rapidement une solution partielle. Commencer avec une petite réalisation avant que les spécifications soient toutes finalisées ou que la totalité de la conception ait été pensée, permet d'obtenir un feed-back rapide ce qui peut (1) enrichir les solutions conceptuelles à développer ou affiner celles déjà implémentées, et (2) apporter la preuve que la conception et l'implémentation partielles sont sur la bonne voie. Cette approche nous semble particulièrement adéquate à nos objectifs de recherche qui visent à développer un ensemble de techniques séparables, mais qui dépendent en cascade une de l'autre (OntoAnalyseur et UKIs Modificateur). Dans la figure 6, nous présentons la démarche de recherche-développement que nous envisageons d'appliquer et dans la figure 7, nous présentons le schéma général d'une itération.

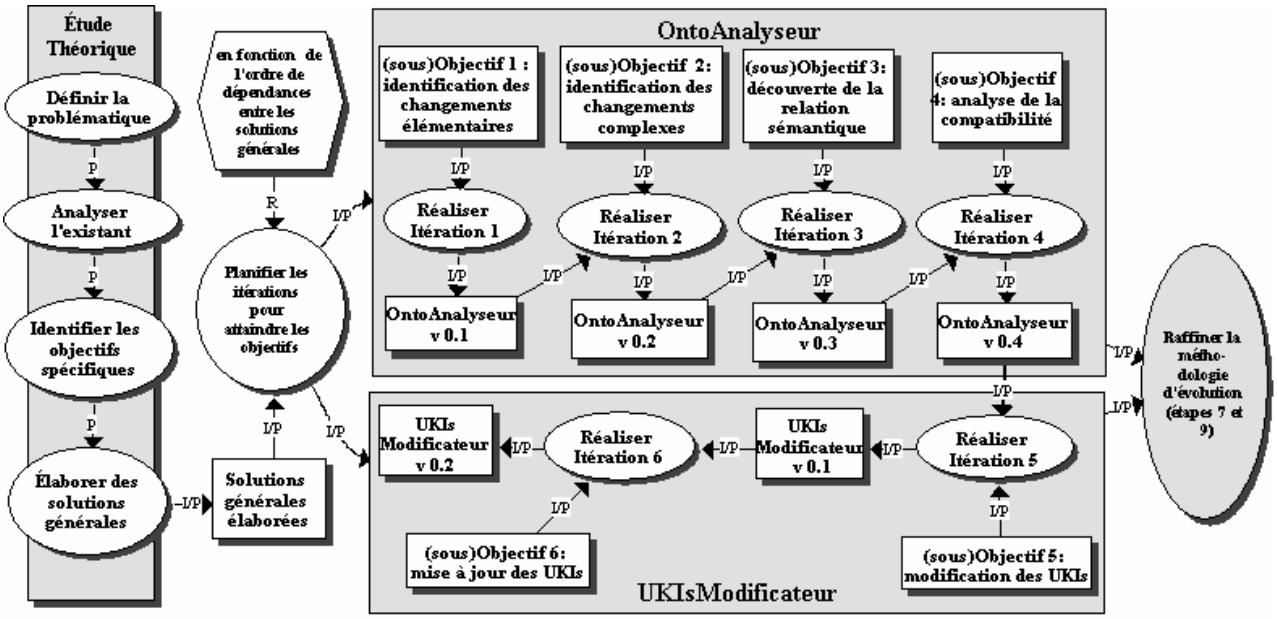


Figure 6. Organisation de la recherche du projet de doctorat (pour la légende, voir celle de la figure 1)

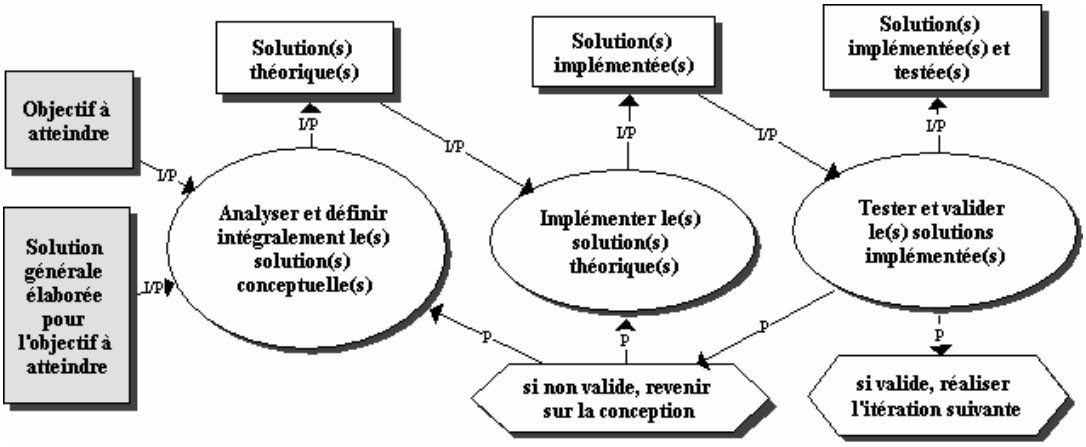


Figure 7. Schéma d'une itération

En conclusion, nous utilisons une démarche de recherche où la phase d'analyse et de définition des solutions conceptuelles, celle de concrétisation de ces solutions par une implémentation informatique ainsi que la phase de test fonctionnel et de validation de la conception partielle, se trouvent imbriqués dans une série d'itérations construction/feed-back/adaptation[23]. Dans les sections suivantes, nous décrivons alors chaque itération de la démarche de recherche en commençant avec les itérations pour concevoir OntoAnalyseur et en finissant avec celles pour concevoir UKIsModificateur.

1.1 Description des itérations pour OntoAnalyseur

Itération 1. Identification des changements élémentaires	
Résultat attendu : identification des changements élémentaires *Ajouter, Effacer, Renommer_Entité*	
Identification sans trace d'évolution	**Identification avec trace d'évolution intégrée**
Analyse et définition des solutions conceptuelles	
- analyser les règles d'identification des changements élémentaires utilisées par OntoView - adapter et définir des règles d'identification des changements élémentaires appliqués aux ontologies OWL-DL	- définir des primitives du formalisme OC (*Ontology Change*) pour la déclaration des changements élémentaires appliqués aux ontologies OWL-DL - définir la modalité d'intégration de ces primitives dans la version V_{N+1} d'une ontologie OWL-DL
Implémentation dans OntoAnalyseur → OntoAnalyseur V 0.1	
- implémenter les règles d'identification des changements élémentaires	- définir la manière dont OntoAnalyseur analyse la trace d'évolution OC pour identifier des changements élémentaires - implémenter la manière d'analyse à OntoAnalyseur 0.1
Instruments d'implémentation (langage de programmation, technique d'implémentation, architecture informatique) à définir	
Test fonctionnel et validation (phase commune)	
- test fonctionnel de l'implémentation - validation de la conception (effectuée par le concepteur du système) : (1) considérer une version OWL-DL de V_N et la modifier pour obtenir V_{N+1}, (2) décrire les changements effectués, (3) les comparer ensuite avec ceux identifiés par OntoAnalyseur et (4) recommencer avec une autre version jusqu'à obtenir une validation satisfaisante. **Participants et Instruments** : il n'y pas besoin ni des utilisateurs ni des instruments de validation spécifiques pour l'itération 1.	

[23] À partir des itérations, nous envisageons aussi d'enrichir la méthodologie d'évolution de l'ontologie (étape 7 et 9).

Itération 2. Identification des changements complexes	
Résultat attendu : identification des changements complexes *Fusionner, Séparer, Déplacer_Entité*	
Identification sans trace d'évolution	**Identification avec trace d'évolution intégrée**
Analyse et définition des solutions conceptuelles	
- analyser les heuristiques de techniques de fusions - identifier les heuristiques pertinentes pour l'identification des changements ontologiques - adapter ces heuristiques pour définir des règles d'identification des changements complexes	- idem itération 1, mais pour des changements complexes
Implémentation dans OntoAnalyseur → OntoAnalyseur V 0.2 (inclut V 0.1)	
- idem itération 1, mais pour des changements complexes	- idem itération 1, mais pour des changements complexes
Test fonctionnel et validation (phase commune)	
- idem itération 1	

Itération 3. Découverte des relations sémantiques entre entités ontologiques de V_N et V_{N+1}
Résultat attendu : découverte des relations d'**Identité, Équivalence, Inclusion, Généralisation, Spécialisation**
Analyse et définition des solutions conceptuelles
- analyser les effets de changements élémentaires et complexes (identifiés par OntoAnalyseur) sur les entités de V_N et de V_{N+1} - définir les règles de découverte des relations sémantiques en fonction des changements ontologiques identifiés
Implémentation dans OntoAnalyseur → OntoAnalyseur V 0.3 (inclut V 0.2)
- implémenter les règles de découverte des relations sémantiques
Test fonctionnel et validation
- test fonctionnel de l'implémentation - validation de la conception : (1) considérer une version OWL-DL de V_N et la modifier pour obtenir V_{N+1} (2) décrire ensuite les relations sémantiques entre les entités ontologiques de V_N et celles de V_{N+1}, (3) les comparer avec les relations découvertes par OntoAnalyseur et (4) recommencer avec une autre version jusqu'à obtenir une validation satisfaisante. **Participants** : utilisateurs futurs du système (nombre à définir) et concepteur du système. **Instruments fournis** : liste des changements à effectuer pour passer de V_N à V_{N+1} et réponses prédéfinies à choisir par les participants (exemple de réponse possible : à la suite du changement, l'entité de V_{N+1} est une spécialisation de celle de V_N).

Itération 4. Analyse de la compatibilité entre V_N et V_{N+1}
Résultat attendu : classification des changements selon les critères **Totalement Compatible, Rétrocompatible, Incompatible**
Analyse et définition des solutions conceptuelles
- classifier les changements élémentaires et complexes (identifiés par OntoAnalyseur) en fonction de leur effet sur la compatibilité de V_{N+1} : totalement compatibles, rétrocompatibles, incompatibles
Implémentation dans OntoAnalyseur → OntoAnalyseur V 0.4 (inclut V 0.3)
- implémenter la classification des changements - implémenter les procédures de calcul, définies dans le chapitre III § 1.2.3, en fonction de trois dimensions de l'analyse
Test fonctionnel et validation
- test fonctionnel de l'implémentation - validation de la conception (effectuée par le concepteur du système) : (1) considérer une version OWL-DL de V_N et la modifier pour obtenir V_{N+1}, (2) classifier ensuite les changements exécutés selon leur compatibilité, (3) comparer la classification avec celle fournie par OntoAnalyseur et (4) recommencer avec une autre version jusqu'à obtenir une validation satisfaisante. **Participants et Instruments** : il n'y pas besoins ni des utilisateurs ni des instruments de validation spécifiques pour l'itération 4.

1.2 Description des itérations pour UKIsModificateur

Itération 5. Modification des UKIs
Résultat attendu : modification des UKIs en fonction des changements exécutés pour passer de V_N à V_{N+1}
Analyse et définition des solutions conceptuelles **(en fonction des résultats fournis par OntoAnalyseur V 0.4)**
- identifier la ou les dimensions de compatibilité nécessaires pour la modification des UKIs dans notre contexte particulier - raffiner la procédure de modification des UKIs pour les changements totalement compatibles, rétrocompatibles ainsi que pour les changements incompatibles qui n'affectent pas les UKIs (cf. chapitre III § 2.2.1) - définir la procédure de modification des UKIs pour les changements incompatibles qui affectent les UKIs
Implémentation dans UKIsModificateur → UKIsModificateur V 0.1
- implémenter un protocole de communication entre OntoAnalyseur et UKIsModificateur - implémenter un protocole de communication entre UKIsModificateur et les clients externes, incluant l'accès aux UKIs à modifier - implémenter les deux procédures de modifications des UKIs
Test fonctionnel et validation
- test fonctionnel de l'implémentation - validation de la conception pour la modification des UKIs affectés par des changements incompatibles : (1) référencer des objets à une version V_N, (2) évaluer le référencement de ces objets à une nouvelle version V_{N+1} selon les critères : degré de pertinence élevé, moyen ou faible. Observation : la modification du référencement est réalisée par UKIsModificateur. **Participants** : utilisateurs futurs du système (nombre à définir). **Instruments fournis** : outil de référencement, interface compréhensive pour la visualisation de la modification des UKIs.

Itération 6. Mise à jour des UKIs
Résultat attendu : mise à jour des UKIs en fonction des changements exécutés pour passer de V_N à V_{N+1} selon les modes **Modification automatique**, **Semi-Automatique**, **Manuelle avec Conseils**
Analyse et définition des solutions conceptuelles
- analyser la pertinence des modifications d'UKIs affectés par des changements incompatibles en fonction des résultats de la validation de l'UKIsModificateur V 0.1. - en fonction de ces résultats, classifier les modifications d'UKIs à réaliser automatiquement ou semi automatiquement, sachant que la modification de tout UKI non affecté par des changements est automatique (cf. chapitre III § 2.2.1) - pour la mise à jour manuelle des UKIs, définir des conseils au niveau de l'interface de visualisation de V_N et de V_{N+1}
Implémentation dans UKIsModificateur → UKIsModificateur V 0.2 (inclut UKIsModificateur V 0.1)
- implémenter les modes de mise à jour automatique et semi-automatique - concevoir et implémenter l'interface pour la mise à jour manuelle avec conseils
Test fonctionnel et validation
- test fonctionnel de l'implémentation - validation de la mise à jour des UKIs : (1) référencer des objets à une version V_N, (2) pour une version V_{N+1}, évaluer la mise à jour des UKIs en fonction des critères de : satisfiabilité, pertinence, facilité de compréhension (pour la mise à jour automatique et semi-automatique), facilité d'utilisation (pour la mise à jour manuelle), autres. **Participants** : utilisateurs futurs du système (nombre à définir). **Instruments fournis** : outil de référencement, interface compréhensive pour la mise à jour des UKIs, questionnaires, autres.

1.3 Validation systémique des solutions conceptuelles et de leur implémentation

Afin de valider d'une manière systémique les solutions conceptuelles proposées, nous mettrons en place, à la fin des itérations, une expérimentation d'OntoAnalyseur V0.4 et d'UKIsModificateur V0.2, avec des utilisateurs, dans un contexte réel. Pour cela, nous envisageons d'utiliser le Laboratoire-Observatoire de Recherche en Ingénierie du Téléapprentissage (LORIT) du Centre de recherche LICEF, étant donné qu'il possède une infrastructure adéquate aux expérimentations demandant une technologie de pointe. En ce qui concerne la définition du protocole d'expérimentation et du code déontologique, les deux étant nécessaires pour toute expérimentation engageant des sujets humains, nous allons nous inspirer des travaux de Moscovici et Buschini (2003) ainsi que de l'expérience que nous avons acquise lors de la mis en place d'une expérimentation au LORIT (Basque, Rogozan, Pudelko, et Bures, 2003).

2. État d'avancement des travaux

Avant de conclure, nous présentons l'état d'avancement de nos travaux de recherche en fonction de la démarche de recherche-développement que nous envisageons d'appliquer pour atteindre nos objectifs.

Tableau 5. État d'avancement des activités de recherche-développement

<table>
<tr><th colspan="2">Activités de recherche-développement</th><th>État</th></tr>
<tr><td colspan="2">Identification de la problématique : Évolution de l'ontologie comme condition sine qua non d'un système d'apprentissage de type ERPAD
- résultats diffusés sous la forme d'article de recherche : (Rogozan, 2003b; Rogozan et al., 2002)
- résultats diffusés sous la forme de communication : (Rogozan, 2003b; Rogozan et al., 2001)</td><td>Réalisé</td></tr>
<tr><td colspan="2">Analyse des approches méthodologiques et des outils pour l'évolution de l'ontologie
- résultats diffusés sous la forme de communication : (Rogozan, 2003a)</td><td>Réalisé</td></tr>
<tr><td colspan="2">Définition de l'ébauche de la méthodologie d'évolution (intrant à l'étape de raffinement)
- résultats diffusés sous la forme d'article de recherche : (Rogozan et al., à paraître en 2004)
- résultats diffusés sous la forme de communication : (Psyché, Rogozan, Bourdeau, et Paquette, 2004)
- résultats diffusés sous la forme de rapport technique : (Rogozan, 2004)</td><td>Réalisé</td></tr>
<tr><td colspan="2">Analyse OWL (intrant aux itérations 1, 2)</td><td>Réalisé</td></tr>
<tr><td>Conception d'une taxonomie des changements élémentaires et complexes en OWL-DL (intrant aux itérations 1, 2, 3, 4)</td><td rowspan="3">Résultats diffusés :
- article de recherche TICE (Rogozan et al., à paraître en 2004)

Résultats soumis pour diffusion :
- article de recherche workshop EKAW (Rogozan et Paquette, soumis en 2004)</td><td>En cours</td></tr>
<tr><td>Définition des solutions générales pour l'identification des changements et l'analyse de la compatibilité entre V_N et V_{N+1} (intrants aux itérations 1, 2, 3 et 4)</td><td>Réalisé</td></tr>
<tr><td>Définition des solutions générales pour la modification et la mise à jour des UKIs (intrants aux itérations 5 et 6)</td><td>Réalisé</td></tr>
<tr><td colspan="2">Itération 1</td><td>En cours</td></tr>
<tr><td colspan="2">Itérations 2, 3, 4, 5, 6</td><td>À faire</td></tr>
</table>

Conclusion

En conclusion, nous faisons un rappel de notre problématique de recherche, des contributions envisagées et de leur originalité.

La gestion de l'évolution de l'ontologie dans un environnement dynamique, multi-acteurs et distribué sur le Web constitue un des principaux défis que les chercheurs en ingénierie des connaissances cherchent à relever à l'heure actuelle, un défi apparu du fait de l'utilisation des ontologies pour construire l'environnement virtuel de demain, le Web sémantique. En vérité, il est irréaliste de vouloir construire le Web sémantique, un environnement dynamique et distribué dont l'architecture est fondée sur le référencement sémantique des objets du Web, sans considérer l'évolution d'une ontologie qui sert de référentiel sémantique. Cette conclusion s'est rapidement propagée dans la communauté des chercheurs, ceux-ci s'engageant dans de nouvelles recherches ayant comme objet d'étude la gestion du processus d'évolution. Toutefois ces recherches, loin d'être achevées, apportent parfois plus de questions que de réponses, par exemple la question de la définition d'un ensemble complet et cohérent d'éléments méthodologiques pour assurer l'évolution de l'ontologie d'une manière consistante, ou encore celle de savoir comment effectuer une analyse pertinente des effets des changements sur le référencement sémantique des objets du Web.

Les contributions de notre projet de recherche s'avèrent alors des éléments importants dans la conception d'un Web sémantique capable de gérer l'évolution de l'ontologie. Notre première contribution est d'ordre méthodologique. Nous proposons l'ébauche d'une méthodologie pour la gestion de l'évolution en unifiant dans un cadre cohérent les approches existantes dans la littérature et en proposant de nouvelles étapes portant sur l'analyse de la nouvelle version de l'ontologie et sur son opérationnalisation. Les méthodes pour supporter ces deux dernières étapes sont actuellement presque inexistantes. Notre deuxième contribution cherche alors à répondre à ces besoins en proposant deux méthodes : (1) une méthode pour l'identification des changements ontologiques et l'analyse des effets des changements sur la relation, sémantique et de compatibilité, entre les versions d'une ontologie, et (2) une méthode pour la modification et pour la mise à jour des UKIs des objets référencés par des entités d'une ontologie évolutive. Ces méthodes sont supportées informatiquement par deux systèmes informatiques, OntoAnalyser et UKIsModificateur.

Quant à l'originalité de nos travaux, elle réside principalement dans :

- l'utilisation des règles d'identification des similarités entre des ontologies différentes, règles utilisées par les techniques de fusion d'ontologies, pour la définition des règles d'identification des différences entre les versions d'une même ontologie ainsi que des changements complexes qui les ont provoquées.

- la définition d'un formalisme de représentation des changements ontologiques (*OntologyChange*) intégrée dans la nouvelle version de l'ontologie, ce formalisme permettant l'explicitation formelle de l'intention des développeurs ayant effectué les changements.
- la prise en compte de la dimension sémantique dans la comparaison entre les versions de l'ontologie en permettant la découverte des relations d'identité, d'équivalence, d'inclusion, de généralisation et de spécification entre leurs entités ontologiques.
- la définition d'une technique d'analyse de la compatibilité entre les versions de l'ontologie selon trois dimensions : la préservation des instances, la préservation de la structure conceptuelle et la préservation de l'axiomatisation de l'ontologie.
- la définition d'une méthode de référencement sémantique évolutif qui supporte la modification et la mise à jour des UKIs (*Uniform Knowledge Identifier*) en fonction de l'analyse des changements effectués pour passer d'une version de l'ontologie à une autre.

L'application des méthodes et outils que nous proposons s'étend à toutes les applications du Web sémantique qui demandent des informations sur l'évolution de l'ontologie ou qui utilisent des ontologies évolutives en guise de référentiels sémantiques de divers objets. Par exemple, les architectures ontologiques pour l'interopérabilité sémantique (Missikoff et Taglino, 2004; Wache et *al.*, 2001) ou celles pour le référencement sémantique des pages Web (Handschuh et *al.*, 2001) ou encore les agents de recherche d'informations basés sur l'ontologie (Hendler, 2001).

Dans notre contexte particulier, ces méthodes et outils sont essentiels pour assurer l'agrégation sémantique d'un système d'apprentissage réalisée par le référencement de ses éléments pédagogiques à un espace ontologique commun. De plus, leur application dans un système d'apprentissage de type ERPAD (Environnement de Réalisation de Projets d'Apprentissage à Distance) permettra aux apprenants de faire évoluer l'ontologie du domaine d'apprentissage et celle de tâche, tout en préservant l'utilisation de ces ontologies comme référentiels communs pour la construction émergente de l'espace de collaboration (EDC) et de l'espace de documentation (EDD). Les apprenants deviendront alors les concepteurs de leur propre environnement de travail à distance lors d'une vraie démarche de réalisation des projets d'apprentissage.

Références

Anderson, T., et Whitelock, D. (2004). The Educational Semantic Web: Visioning and racticing the Future of Education. *Journal of Interactive Media in Education, 1*(Special Issue on the Educational Semantic Web).

Arpirez, J., Corcho, O., Fernandez-Lopez, M., et Gomez-Perez, A. (2001). *WebODE : a Workbench for Ontological Engineering.* Paper presented at the First international Conference on Knowledge Capture (K-CAP'01), Victoria, Canada.

Bachimont, B. (2000). Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances. In Z. M. Charlet J., Kassel G., Bourgault D., (Ed.), *Ingénierie des connaissances. Évolution Récentes et nouveaux défis* (pp. 305-323). Paris: Eyrolles.

Basque, J., Rogozan, D., Pudelko, B., et Bures, E. (2003). *Rapport d'utilisation du LORIT lors d'une expérimentation sur la co-construction de connaissances en situation de téléapprentissage* (No. rapport interne LICEF).

Bechhofer, S., et Goble, C. (2001). *Towards Annotation Using DAML+OIL.* Paper presented at the KCAP'01 Workshop on Semantic Markup and Annotation, Victoria, Canada.

Bechhofer, S., Horrocks, I., Goble, C., et Stevens, R. (2001). OilEd: a Reason-able Ontology Editor for the Semantic Web. In *Joint German/Austrian Conference on Artificial Intelligence (KI'01)* (Vol. 2174). Vienne: Lecture Notes in Artificial Intelligence.

Bellamy, R. K. E., Grant, W., Cooper, E., Borovoy, R., et Adams, S. (1994). *Media Fusion: A Tool that Supports Learning through Experience, Reflection, and Collaboration* (No. 20): Apple Classrooms of Tomorrow (ACOT)™.

Berners-Lee, T. (2000). *Semantic Web - XML 2000 Conference*, from http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html

Berners-Lee, T., Hendler, J., et Lasilla, O. (2001). The Semantic Web. *Scientific American, 284 (5)*.

Borst, W. N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse.* University of Tweenty, Enschede, Centre for Telematica and Information Technology.

Calvanese, D., Giacomo, G., et Lenzerini, M. (2001). *A Framework for Ontology Integration.* Paper presented at the 2001 Int. Semantic Web Working Symposium (SWWS 2001).

Charlet, J., Bachimont, B., et Troncy, R. (2003). Ontologies pour le Web sémantique. In J. Charlet, P. Laublet et C. Reynaud (Eds.), *Web sémantique*: Action_spécifique_32_CNRS/STIC.

Charlet, J., Zacklad, M., Kassel, G., et Bourigault, D. (Eds.). (2000). *Ingénierie des connaissances: Évolution récentes et nouveaux défis*: Éditions Eyrolles.

Ding, Y., et Fensel, D. (2001). *Ontology Library Systems: The key for successful Ontology Reuse.* Paper presented at the First Semantic Web Working Symposium (SWWS'1), Stanford, USA.

Ding, Y., Fensel, D., Klein, M., Omelayenko, B., et Schulten, E. (2004). The role of Ontologies in eCommerce. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 593-616): Springer Verlag.

Donzelini, G., Ponta, D., Bailey, C., Xu D. (1999). *Learning Electronic Systems Design with a Project Based Course on the Network.* Paper presented at the ENABLE 99 - Enabling network-based learning International Conference, Espoo, Finland, 1999.

Ermine, J.-L., Chaillot, M., Bigeon, P., Charreton, B., et Malavieille, D. (1996). MKSM, Méthode pour la gestion des connaissances. *Ingénierie des systèmes d'information (AFCET-Hermès), 4*.

Euzenat, J. (1995). Building consensual knowledge bases: context and architecture. In N. Mars (Ed.), *Towards very large knowledge bases* (pp. 143-155). Amsterdam: IOS press.

Euzenat, J., et all. (2001). *Research challenges and perspectives of the Semantic Web*: EU-NSF strategic workshop.

Fayolle, J., Jacquet, G., et Fouquet, R. (2001). *L'apport des projets dans une démarche d'enseignement à distance en 3ème année d'école d'ingénieur.* Paper presented at the Colloque sur la Pédagogie par Projet dans l'enseignement supérieur: enjeux et perspectives, Brest, France.

Fernandez, M., Gomez-Perez, A., et Juristo, N. (1997). *Methontology: From Ontological Art Toward Ontological Engineering.* Paper presented at the Spring Symposium Series on Ontological Engineering. AAAI97, Stanford, USA.

Finkle, S., et Torp, L. (1994). *Introductory documents*. Illinois: IMSA Center for Problem-Based Learning.

Fung, P. (1996). Issues in Project -Based Distance Learning in Computer Science. *Journal of Distance Education/Revue de l'enseignement à distance, 11*(2).

Gomez-Perez, A. (1999). *Ontological Ingineering: A state of the art*. Madrid: Facultad de Informatica, Universidad Politecnica de Madrid.

Gruber, T. (1995). *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* Paper presented at the International Workshop on Formal Ontology, Padova, Italy.

Gruninger, M., et Lee, J. (2002). Ontology Applications and Design. *Communication of the ACM, 45*(2), 39-41.

Guarino, N., et Giaretta, P. (1995). Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Mars N. J. I. (Ed.), *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing* (pp. 25-32). Amsterdam: IOS Press.

Guarino, N., et Welty, C. (2002). Evaluating ontological decision with Ontoclean. *Communication of the ACM, 45*(2), 61-65.

Handschuh, S., Staab, S., et Maedche, A. (2001). *CREAM — Creating Relational Metadata with a Component-Based, Ontology-Driven Framework.* Paper presented at the First International Conference on Knowledge Capture (K-CAP 2001), Victoria, Canada.

Heflin, J. (2001). *Towards the Semantic Web: Knowledge and representation in a dynamic, distributed environment.* Faculty of the Graduate School of the University of Maryland.

Heflin, J., et Hendler, J. (2000). *Dynamic Ontology on the Web.* Paper presented at the AAAI, 17th National Conference on artificial Intelligence.

Heflin, J., Hendler, J., et Luke, S. (1999). Coping with Changing Ontologies in a Distributed Environment. *Ontology Management. Papers from the AAAI Workshop*, 74-79.

Hendler, J. (2001). Agents and the Semantic Web. *IEEE Intelligent systems, March/April 2001*, 30-37.

Horrocks, I., Sattler, U., et Tobies, S. (1999). *Practical reasoning for expressive description logics.* Paper presented at the 6th International Conference on Logic for Programming and Automated Reasoning.

IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology.* (No. Std. 610.12-1990). New York (USA): IEEE Computer Society.

IEEE. (1995). *IEEE Guide for Software Quality Assurance Planning* (No. Std. 730.1-1995). New York (USA). IEEE Computer Society.

Klein, M. (2002a). *Supporting evolving ontologies on the internet.* Paper presented at the Proceedings of the EDBT 2002 PhD Workshop, Prague, Czech Republic.

Klein, M. (2002b). *Versioning of distributed ontologies* (No. Deliverable D20 V1.1, EU/IST Project WonderWeb).

Klein, M., Ding, Y., Fensel, D., et Omelayenko, B. (2002). Ontology management - Storing, aligning and maintaining ontologies. In J. Davids, D. Fensel et F. vanHarmele (Eds.), *Towards the Semantic Web: Ontology-Driven Knowledge Management* (pp. 47-69): Wiley.

Klein, M., et Fensel, D. (2001). *Ontology versioning for the Semantic Web.* Paper presented at the International Semantic Web Working Symposium (SWWS), USA.

Klein, M., Fensel, D., Kiryakov, A., et Ognyanov, D. (2002). *Ontology versioning and change detection on the web.* Paper presented at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Sigüenza, Spain.

Klein, M., et Noy, N. (2003). *A component-based framework for the ontology evolution.* Paper presented at the Workshop on Ontologies and Distributed Systems, IJCAI-2003, Acapulco, Mexico.

Koper, R. (2004). Use of the Semantic Web to Solve Some Basic Problems in Education: Increase Flexible, Distributed Lifelong Learning, Decrease Teachers' Workload. *Journal of Interactive Media in Education, 6*(Special Issue on the Educational Semantic Web).

Larman, C. (2002). *UML et les Design Patterns*. Paris: CampusPress.

Laublet, P., Reynaud, C., et Charlet, J. (2002). *Sur quelques aspects du Web sémantique.* Paper presented at the Deuxièmes assises nationales du GdR I3, Nancy.

Lebrun, M. (2002). *Des technologies pour enseigner et apprendre* (2 ed.). Paris: De Boeck.

Leontiev, A. (1984). *Activité, Conscience, Personnalité* (2 ed.). Moscou: Éditions du Progrès.

Maedche, A., Motik, B., et Stojanovic, L. (2003). Managing Multiple and Distributed Ontologies in the Semantic Web. *VLDB Journal - Special Issue on Semantic Web, 12*, 286-302.

Maedche, A., Motik, B., Stojanovic, L., Studer, R., et Volz, R. (2003). Ontologies for Enterprise Knowledge Management. *Intelligent Infromation Processing, March/April 2003*.

McGuinness, D. (2000). *Conceptual Modeling for Distributed Ontology Environments.* Paper presented at the ICCS 2000, Darmstadt, Germany.

Missikoff, M., et Taglino, F. (2004). An Ontology-based Platform for Semantic Interoperability. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 617-633). Berlin: Springer-Verlag.

Morgan, A. (1994). Developping learner autonomy: project-based learning in open and distance learning. In F. Lockood (Ed.), *Materials Production in Open and Distance Learning* (pp. 112-120). London: Paul Chapman.

Moscovici, S., et Buschini, F. (Eds.). (2003). *Les méthodes des sciences humaines* (1 ed.). Paris: Presses Universitaires de France.

Motta, E., Vargas-Vera, M., Domingue, J., Lanzoni, M., et Ciravegna, F. (2002). *MnM: OntologyDriven Semi-Automatic and Automatic Support for Semantic Mark-up.* Paper presented at the Semantic Authoring, Annotation et Knowledge Markup Workshop at ECAI 2002, Lyon, France.

Nejdl, W., et Banagl, M. (1994). Asking About Possibilites - Revision and Update Semantics for Subjunctive Queries. In G. Lakemeyer et B. Nebel (Eds.), *Foundations of Knowldege Representation and Reasoning*. Berlin: Springer-Verlag.

Noy, N. (2004). Tools for Mapping and Merging Ontologies. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies*: Springer Verlag.

Noy, N., Fergerson, R., et Musen, M. (2000). The knowledge model of Protégé-2000: Combaining interoperability and flexibility. In R. Dieng et O. Corby (Eds.), *12 International Conference on Knowledge Engineering and Knowledge Management (EKAW'00)* (Vol. 1937): Lectures Notes in Computer Science.

Noy, N., et Klein, M. (2003a). Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems, 5*.

Noy, N., et Klein, M. (2003b). *Tracking Complex Changes During Ontology Evolution.* Paper presented at the Second International Semantic Web Conference (ISWC-2003) Poster Session, Florida.

Noy, N., et Musen, M. (2000). *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment.* Paper presented at the Seventeenth National Conference on Artificial Intelligence (AAAI-2000).

Noy, N., et Musen, M. (2002). *PROMPTDIFF: A fixed-point algorithm for comparing ontology versions.* Paper presented at the 18th National Conference on Artificial Intelligence (AAAI-2002), Edmonton, Canada.

Noy, N., et Musen, M. (2003a). Ontology Versioning as an Element of an Ontology-Management Framework. *IEEE Intelligent Systems, in press*.

Noy, N., et Musen, M. (2003b). The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. *International Journal of Human-Computer Studies, 59*(6), 983-1024.

Oberle, D., Volz, R., Motik, B., et Staab, S. (2004). An extensible ontology software environment. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 299-320): Springer Verlag.

Oberle, D., Wenke, D., Volz, R., et Staab, S. (2003). *Ontobroker and OntoEdit Adaptation* (No. Deliverable 9): IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web.

Oliver, D. E., Shahar, Y., Musen, M., et Shortliffe, E. H. (1999). Representation of change in controlled medical terminologies. *AI in Medicine, 15*(1), 53–76.

OntoWeb. (2002a). *EC project IST-OntoWeb*, from http://www.ontoweb.org

OntoWeb. (2002b). *A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies* (No. IST-2000-29243 - Deliverable 1.4).

OntoWeb. (2002c). *A survey on ontology tools* (No. IST-2000-29243-Deliverable 1.3).

Paquette, G. (2002a). *L'ingénierie pédagogique : pour construire l'apprentissage en réseau*. Québec: Presses de l'Université du Québec.

Paquette, G. (2002b). *Modélisation des connaissances et des compétences*. Québec: Presses de l'Université du Québec.

Paquette, G., et Rosca, I. (2002). Organic Aggregation of Knowledge Objects in Educational Systems. *Canadian Journal of Learning and Technology, vol. 28*(No. 3), 11-26.

Penuel, W., et Means, B. (2000). *Designing a performance assessment to measure students' communication skills in multi-media-supported, project-based learning.* Paper presented at the Annual Meeting of the American Educational Research Association, New Orleans.

Pinto, S., et Martins, J. (2002). *Evolving Ontologies in Distributed and Dynamic Settings.* Paper presented at the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2002), San Francisco.

Pinto, S., Staab, S., et Tempich, C. (2004). *DILIGENT: Towards a fine-grained methodology for Distributed Loosely-controllled and evolvInG Engineering of oNTologies. I.* Paper presented at the 16th European Conference on Artificial Intelligence ECAI-2004, Valencia.

Psyché, V., Rogozan, D., Bourdeau, J., et Paquette, G. (2004). Un processus d'ingénierie ontologique pour la Base de connaissances pour le téléapprentissage (BCTA). Montréal: Colloque DIVA à l'ACFAS 2004.

Rogozan, D. (2003a). Ontology Evolution: Ontology Editor Functions to Support Single-Ontology Evolution. Montréal: Groupe de travail LORNET (Thème 2).

Rogozan, D. (2003b). Solution techno-pédagogique aux interactions d'apprentissage en ligne entre groupes de projet. Rimouski: 3ème colloque du CIRTA. Le télé-apprentissage dans la société du savoir.

Rogozan, D. (2004). *Conception d'une méthodologie pour assurer l'évolution de l'ontologie dans un environnement dynamique et distribué* (No. à apparaître). Montréal: Centre de recherche LICEF.

Rogozan, D., Abdulrab, H., et Hotte, R. (2001). *Environnement ontologique d'aide à la réalisation de Projets d'Etudes Collectifs.* Paper presented at the Ingénierie des Connaissances (IC'2001), Grenoble, France.

Rogozan, D., Hotte, R., et Abdulrab, H. (2002). *Modélisation d'un espace dynamique dédié à la réalisation de projets d'apprentissage à distance.* Paper presented at the TICE Symposium on Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, France.

Rogozan, D., et Paquette, G. (2003). *Ontologie émergente pour le téléapprentissage par projet.* Paper presented at the Colloque DIVA (Développement, intégration et évaluation des technologies de formation et d'apprentissage), Montreal, Cananda.

Rogozan, D., et Paquette, G. (soumis en 2004). *Compatibility Analysis between Ontology Versions: Application to Change Propagation.* Paper presented at the EKAW'2004 Workshop on Knowledge Management and Semantic Web, Northamptonshire, UK.

Rogozan, D., Paquette, G., et Hotte, R. (2003). *Dynamic Approach for Constructing a Specific Collaboration Space among Distant Project Groups.* Paper presented at the E-Learn Conference, 2003, Phoenix, Arizona.

Rogozan, D., Paquette, G., et Rosca, I. (à paraître en 2004). *Gestion de l'évolution de l'ontologie utilisée comme référentiel sémantique dans un environnement de téléapprentissage.* Paper presented at the TICE International Symposium on Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, Compiegne, France.

Shoham, Y., et Steve, C. (1994). Logics of Mental Attitudes in AI : a very preliminary survey. In G. Lakemeyer et B. Nebel (Eds.), *Foundations of Knowldege Representation and Reasoning*. Berlin: Springer-Verlag.

Staab, S., Maedche, A., et Handschuh, S. (2001). *An Annotation Framework for the Semantic Web.* Paper presented at the The First International Workshop on MultiMedia Annotation, Tokyo, Japan.

Stojanovic, L., Maedche, A., Motik, B., et Stojanovic, N. (2002). *User-driven Ontology Evolution Management.* Paper presented at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Sigüenza, Spain.

Stojanovic, L., et Motik, B. (2002). *Ontology Evolution within Ontology Editors.* Paper presented at the Knowledge Acquisition, Modeling and Management (EKAW), Siguenza, Spain.

Stojanovic, L., Staab, S., et Studer, R. (2001). *eLearning based on the Semantic Web.* Paper presented at the WebNet2001 - World Conference on the WWW and Internet, Orlando, Florida.

Stojanovic, L., Stojanovic, N., et Handschuh, S. (2002). *Evolution of the Metadata in the Ontology-based Knowledge Management Systems.* Paper presented at the Experience Management 2002, Berlin, Germany.

Stojanovic, N., et Stojanovic, L. (2002). *Usage-Oriented Evolution of Ontology-Based Knowledge Management Systems.* Paper presented at the Confederated International Conferences DOA, CoopIS and ODBASE 2002, Irvine, California, USA.

Stuckenschmidt, H., et Klein, M. (2003a). *Evolution management for interconnected ontologies.* Paper presented at the Workshop on Semantic Integration at ISWC 2003, Sanibel Island, Florida.

Stuckenschmidt, H., et Klein, M. (2003b). *Integrity and change in modular ontologies.* Paper presented at the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico.

Studer, R. (2003). *The Semantic Web: Methods, Applications and Future Trends.* Paper presented at the IFIP Conference on commerce, business and government, I3E 2003, Sao Paulo, Brazil.

Stutt, A., et Motta, E. (2004). Semantic Learning Webs. *Journal of Interactive Media in Education, 10*(Special Issue on the Educational Semantic Web).

Sunagawa, E., Kozaki, K., Kitamura, Y., et Mizoguchi, R. (2003). *An Environment for Distributed Ontology Development Based on Dependency Management.* Paper presented at the International Semantic Web Conference (ISWC), Florida, USA.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., et Wenke, D. (2002). OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In I. Horrocks et J. Hendler (Eds.), *First International Semantic Web Conference (ISWC'02)* (Vol. 2342). Italie: Lectures Notes in Computer Science.

Sure, Y., Staab, S., et Studer, R. (2004). On-To-Knowledge Methodology (OTKM). In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 117-132): Springer Verlag.

Thomas, J. W. (2000). *A Review of Research on Project-Based Learning*: The Autodesk Foundation.

Uschold, M., et Gruninger, M. (2002). Creating semantically integrated communities on the World Wide Web. Honolulu: Semantic Web Workhop.

Uschold, M., et King, M. (1995). *Towards a Methodology for Building Ontologies.* Paper presented at the Workshop on Basic Ontological Issues in Knowledge Sharing.

Van der Maren, J., M.,. (1996). *Méthodes de recherche pour l'éducation* (2 ed.). Montréal: Les Presses de l'Université de Montréal.

Van Heijst, G., Schreiber, A., et Wielinga, B. J. (1997). Using Explicit Ontologies in KBS Development. *International Journal of Human and Computer Studies, in press*.

Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., et al. (2001). *Ontology-based integration of information - a survey of existing approaches.* Paper presented at the IJCAI Workshop on Ontologies and Information Sharing.

Web_sémantique. (2001). *Action spécifique 32 CNRS / STIC*, from http://www.lalic.paris4.sorbonne.fr/stic/

WebOnt. (2001). *OWL Ontology Web Language*, from http://www.w3.org/2001/sw/WebOnt/

WebOnt. (2004a). *OWL Web Ontology Language Guide and Reference*, from http://www.w3.org/TR/2004/REC-owl-features-20040210/

WebOnt. (2004b). *OWL Web Ontology Language Use Cases and Requirements*, from http://www.w3.org/TR/2004/REC-owl-test-20040210/