EvOnto : un outil d'évolution de ressource termino-ontologique pour l'annotation sémantique

Anis Tissaoui, Nathalie Aussenac-Gilles, Philippe Laublet, Nathalie Jane Hernandez

EvOnto : un outil d'évolution d'une ressource termino-ontologique pour l'annotation sémantique

Anis Tissaoui* — Nathalie Aussenac-Gilles* — Philippe Laublet — Nathalie Hernandez***

* *Institut de Recherche en Informatique de Toulouse*
Université Paul Sabatier, 118 Route de Narbonne, F-31062, Toulouse cedex 09
{Tissaoui, Aussenac, Hernande}@irit.fr

** *Laboratoire Sens, Texte, Informatique et Histoire*
Université de Paris-Sorbonne, 1 Rue Victor Cousin, 75005 Paris
Philippe.Laublet@paris-sorbonne.fr

RÉSUMÉ. *Dans un environnement dynamique, les ressources termino-ontologiques et les annotations sémantiques qu'elles permettent de construire doivent être modifiées régulièrement et en cohérence pour s'adapter à l'évolution du domaine sur lequel elles portent et des collections documentaires annotées. En support à un environnement d'annotation automatique de documents (TextViz), nous avons développé EvOnto pour faciliter l'évolution d'une ressource termino-ontologique en tenant compte des annotations sémantiques définies avec celle-ci. EvOnto permet de formuler une demande de changement, d'évaluer l'impact de ce changement sur la ressource termino-ontologique et sur les annotations sémantiques, et finalement de décider de la mise en œuvre de ce changement. Cet article présente les principes d'EvOnto et une étude de cas qui illustre son apport à l'évolution d'ontologie.*

ABSTRACT. *In dynamic environments, Ontological and Terminological Resources and semantic annotations built from them must be changed to adapt to domain evolutions and to new needs for annotated documents. Therefore, we developed the EvOnto tool that extends the TextViz tool for automatic document annotation with a termino-ontological resource. EvOnto is intended for the ontologist, it provides an interactive guide-line allowing him to formulate a change request, to evaluate its impact on the termino-ontological resource and on the semantic annotations, and finally to decide how the change will be implemented. Our paper presents the main principles of EvOnto and it reports a case-study that illustrates its support to ontology evolution.*

MOTS-CLÉS : *ressources termino-ontologique, annotation sémantique, évolution.*

KEYWORDS: *ontological and terminological resource, semantic annotation, evolution.*

1. Introduction

La perspective du Web sémantique est d'enrichir le contenu du Web actuel avec des descriptions formelles de sorte que les agents logiciels puissent être en mesure de traiter les informations fournies par l'homme sur les pages Web (Tim Berners Lee *et al.*, 2001). Les annotations sémantiques décrivent le contenu de ces pages à partir de concepts et/ou de relations représentés dans une ontologie. Dans ce cadre, les ontologies jouent un rôle clé : elles fournissent le vocabulaire et la connaissance du domaine qui seront utilisées pour analyser le contenu des pages Web. L'annotation sémantique basée sur l'ontologie exige que l'ontologie soit riche : elle doit représenter non seulement les concepts et relations potentiellement mentionnés dans les documents mais aussi les termes (et leurs variantes) employés pour dénoter ces entités dans les textes. En ajoutant une composante lexicale aux ontologies, les ressources termino-ontologiques (RTO) s'avèrent donc bien adaptées.

Pour annoter de nouvelles ressources, prendre en compte les évolutions du domaine (Maedche, 2002) ou de nouveaux usages (Flouris *et al.*, 2006), une RTO doit être fréquemment modifiée. Ce processus, complexe et coûteux, soulève des questions pratiques : comment identifier la nécessité d'un changement ? Comment traiter les conséquences de ce changement sur les entités de la RTO ou sur les artefacts dépendants de cette RTO, comme les annotations posées à l'aide de la version non modifiée ? Depuis 2002, l'évolution des ontologies a fait l'objet de recherches qui ont permis d'identifier les différents types de changements possibles, les étapes d'un processus de changement, les aides possibles pour assister la gestion des versions d'une ontologie ou encore la gestion des répercussions logiques d'une modification locale sur l'ontologie toute entière (Stojanovic, 2004) (Klein, 2004).

Notre recherche s'inscrit dans cette lignée, mais se concentre sur deux caractéristiques spécifiques : l'évolution de RTO, ontologies à composantes lexicales dans lesquelles les termes ont un statut à part entière à côté des composants habituels ; l'utilisation de la RTO pour un objectif précis : l'annotation sémantique de documents.

Une hypothèse forte de notre travail est que la prise en compte des impacts d'une modification de la RTO sur la RTO elle-même et sur les annotations, et ce dès la proposition de cette modification, permettra de réduire les effets négatifs de cet impact. Dans le cas d'annotations sémantiques, les effets négatifs peuvent être par exemple de devoir recommencer inutilement l'annotation de certains documents, ou de dégrader la qualité des annotations existantes. Le fait de prendre en compte les termes et les annotations sémantiques utilisant la RTO renouvelle les questions classiques soulevées par l'évolution d'ontologie : Comment appliquer un changement tout en assurant la cohérence au sein de RTO mais aussi la cohérence de la RTO par rapport aux applications qui l'utilisent ? Comment analyser les effets d'un changement (au sein de la RTO mais aussi sur ses utilisations) et les gérer au

mieux ? Et si plusieurs solutions sont possibles, quelles informations ou critères présenter à l'ontologue[1] pour l'aider à choisir ?

Dans cet article, nous présentons notre proposition pour gérer l'évolution conjointe d'une RTO et d'annotations sémantiques. La section 2 décrit le contexte de ce travail le projet DYNAMO, et son méta-modèle de RTO et d'annotation. Nous dressons dans la section 3 un état de l'art sur la gestion de l'évolution d'ontologie et les outils d'aide à l'évolution. La section 4 présente notre approche d'évolution, qui gère les conséquences de l'évolution de la RTO sur les annotations, et inversement, les conséquences des évolutions du corpus et des besoins en annotation sur la RTO. Pour valider cette approche, nous avons développé le logiciel EvOnto, présenté en section 5. Enfin, en section 6, nous présentons une expérimentation d'évolution d'une RTO avec EvOnto.

2. Contexte scientifique

2.1 *Le projet Dynamo*

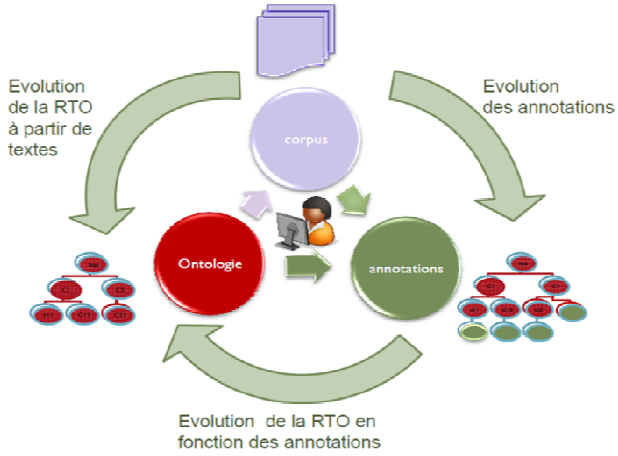

Figure 1. *Le cycle des évolutions*

Ce travail est réalisé dans le cadre du projet ANR DYNAMO[2] (DYNAMic Ontology for information retrieval). L'objectif principal de ce projet est de concevoir une approche méthodologique et un ensemble d'outils logiciels qui gèrent la construction et la maintenance de ressources ontologiques à partir de documents et

[1] L'ontologue est la personne qui construit l'ontologie
[2] http://www.irit.fr/DYNAMO

l'utilisation de ces ressources pour une indexation sémantique facilitant la recherche d'information. L'originalité de DYNAMO réside dans la spécification conjointe du fonctionnement des modules de maintenance d'ontologie et de recherche d'information, de manière à prendre en compte, d'une part, les répercutions d'une évolution du corpus sur les ressources ontologiques et, d'autre part, la dynamique de l'annotation (éventuellement sa remise en cause) en fonction des évolutions constatées dans l'ontologie (Figure 1). Dans ce projet, les corpus documentaires fournis par les partenaires sont issus de trois domaines particuliers : la recherche en archéologie des techniques, le diagnostic de pannes automobiles et la gestion d'incidents logiciels.

2.2 Le modèle de la ressource termino-ontologique

Dans DYNAMO, nous nous intéressons à l'utilisation et l'évolution de Ressources Termino-Ontologique (RTO), à savoir un modèle conceptuel comportant une composante conceptuelle, prenant la forme d'une ontologie, et une composante lexicale ou terminologie (Aussenac-Gilles *et al.*, 2006), (Maedche, 2002) et (Cimiano, 2006). La RTO contient alors non seulement une représentation des concepts du domaine et de leurs relations, mais aussi une représentation explicite des termes associés (termes désignant les concepts). Par exemple, dans une RTO représentant le domaine de l'automobile, nous pourrons avoir le concept automobile. Ce concept est désigné par plusieurs termes comme voiture, auto, automobile. Ainsi, la manifestation linguistique de chaque concept dans les documents est sauvegardée avec l'ontologie non seulement dans les labels, mais aussi sous forme de structures spécifiques. Dans DYNAMO, ces termes permettent d'annoter ou indexer des documents textuels dans le cadre d'une annotation sémantique basée sur la reconnaissance dans les textes des termes qui leur sont associés dans l'ontologie.

Le modèle de RTO retenu (Figure 2) est le méta-modèle OWL proposé dans (Reymonet *et al.*, 2009). Dans ce méta-modèle, les concepts et les termes sont deux meta-classes sous-classes de owl:Class. La relation « dénote » entre la classe Term et la classe Concept permet de relier un ou plusieurs termes avec un ou plusieurs concepts. Chaque instance d'une sous-classe de Term est une occurrence de ce terme à un endroit précis d'un document particulier. Elle est reliée à une instance de concept par la relation « désigne ». Les instances présentes dans un document ont comme valeur de la propriété hasTermOccurrence une instance de la classe document qui réifie la notion de document. Le projet Dynamo a donné lieu à un premier prototype TextViz (Reymonet et al, 2009) qui a été développé comme un plug-in de Protégé.

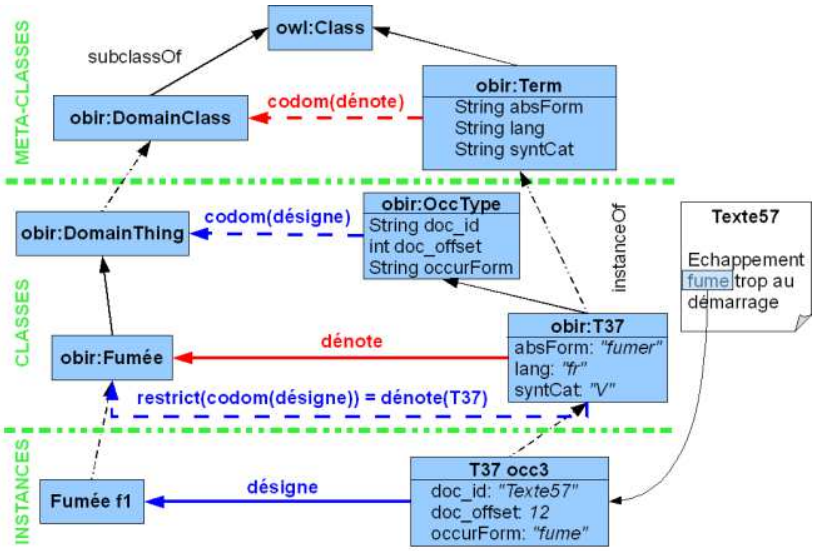

Figure 2. ***Le méta-modèle de RTO***

3. Évolution d'ontologies : travaux connexes

Les résultats en matière d'évolution d'ontologies ont permis d'établir des aides pour faciliter l'identification des besoins de changement (Cimiano *et al.*, 2005) et la détection des changements (Klein, 2004), (Plessers *et al.*, 2005), l'implémentation des changements (Stojanovic, 2004), (Flouris, 2006), la maintenance de la cohérence (Stojanovic, 2004), (Djedidi, 2009) et la gestion de versions d'ontologie (Klein, 2004). D'autres recherches ont défini un processus d'évolution plus ou moins global comprenant aussi bien l'analyse et la résolution des impacts de changement que la propagation des changements aux artefacts dépendant de l'ontologie modifiée, c'est-à-dire les objets, les ontologies et les applications qu'elle référence (Stojanovic, 2004) (Klein, 2004) et (Luong, 2007). (Flouris et al. 2006) distinguent les logiciels facilitant l'évolution de ceux assistant la gestion et la comparaison de versions. Enfin, le processus d'évolution est organisé différemment suivant que l'on cherche à faire évoluer l'ontologie selon les besoins de l'utilisateur sans utiliser d'autres sources de connaissances du domaine, ou au contraire que l'on veuille alimenter ou enrichir l'ontologie à l'aide de nouvelles connaissances tirées d'autres sources, comme des ontologies existantes ou des corpus textuels. On remarquera que les travaux existants sur l'évolution d'ontologies ne traitent pas l'évolution dans le temps de la partie terminologique associée à une ontologie.

3.1 Le processus d'évolution d'ontologies

Un des premiers résultats relatifs à l'évolution d'ontologie est le processus global défini par Stojanovic pour la méthode KAON (Stojanovic *et al.*, 2003) (Stojanovic, 2004). Cette méthode repose sur une taxonomie des changements que nous adoptons dans nos travaux. Cette méthode propose aussi des stratégies de résolution des inconsistances au sein de l'ontologie modifiée, stratégies qui sont présentées à l'utilisateur pour qu'il décide de leur application en se basant sur son expertise du domaine. Une des limites de cette approche est de ne pas conserver l'historique des modifications et de ne garder que la version la plus récente de l'ontologie. De plus, elle ne rend pas compte des changements du domaine venant de sources externes (documents, bases de données, …).

Klein adopte les mêmes principes tout en gérant et conservant les différentes versions de l'ontologie en évolution (Klein, 2004). Il propose une taxonomie de changements pour le langage OWL qui distingue les changements de base des changements complexes. Klein n'étudie pas la propagation de ces changements vers les artefacts dépendants. En revanche, Luong prend en compte à la fois l'ontologie et les annotations sémantiques produites avec cette ontologie (Luong, 2007). L'auteur traite le cas où l'évolution de l'ontologie engendre l'évolution des annotations. Par contre, l'inverse n'est pas pris en compte. De même, R. Djedidi a proposé une approche qui s'appuie sur différents types de patrons pour optimiser et automatiser la gestion des changements tout en assurant la cohérence et la qualité de l'ontologie après évolution (Djedidi, 2009).

Une autre approche (Rogozan *et al.*, 2005) (Rogozan, 2009) intègre deux méthodologies présentées par Klein (Klein, 2004) et Stojanovic dans une méthodologie unifiée appliquée à un contexte particulier dans lequel l'ontologie est utilisée comme référentiel sémantique pour les éléments pédagogiques d'un système d'apprentissage sur le Web sémantique. Les auteurs ont identifié des changements après l'évolution d'ontologie et analysent leur effet sur l'annotation de ressources à base d'ontologie. Cependant, cette approche a besoin d'un ensemble de modifications d'ontologie formalisée et elle ne traite que de la modification des adresses URI d'éléments d'annotation. Les auteurs (Rogozan et al., 2005) ont également décrit un cadre «OntoAnalyzer» pour le maintien de l'intégrité de l'annotation sémantique de ressources au-delà de l'évolution d'ontologie, mais cette approche modifie simplement les liens de référence des objets pour leur permettre de bien se référer à la nouvelle version de l'ontologie.

Dans ces quatre travaux,, le processus défini est comparable : étant donné une demande de changement indiquée par l'ontologue et un état courant de l'ontologie, il s'agit de traduire ce changement sous une forme suffisamment formelle pour en anticiper les conséquences sur l'ontologie, vérifier que sa qualité ne sera pas dégradée, et aider l'ontologue à choisir un ensemble de conséquences sur les composants de l'ontologie, parmi toutes celles possibles. De manière complémentaire, d'autres travaux visent d'abord à automatiser l'identification de

connaissances qui pourraient enrichir une ontologie (toutes les modifications automatiques sont des ajouts), avant d'assurer la gestion de son évolution. Ainsi, pour alimenter une ontologie, la méthode EVOLVA de F. Zablith exploite les termes présents dans des textes du domaine et réutilise des ressources comme des ontologies ou des bases de données trouvés sur le web (Zablith, 2009). Les termes extraits de corpus textuels mènent à la création de nouveaux concepts, s'ils apparaissent dans les URIs d'un concept d'une ressource en ligne (comme WordNet). Ces concepts sont placés dans l'ontologie lorsqu'EVOLVA est capable de les mettre en relation avec un des concepts déjà présents dans l'ontologie. Pour cela, EVOLVA cherche à extraire des relations de ressources en ligne, en particulier d'ontologies, à l'aide du logiciel Scarlet[3]. EVOLVA contient cinq composants dont le dernier, «Gestion de l'évolution», permet à l'ontologue de contrôler l'évolution après enrichissement.

3.2 Outils d'aide à l'évolution

Ces différentes méthode s'appuient sur des logiciels qui guident leur mise en œuvre et font des suggestions à l'ontologue. Un des premiers éditeurs d'ontologie à intégrer un outil de gestion automatisée des évolutions d'une ontologie a été la plateforme KAON4 proposée à l'Université de Karlsruhe. Ce système guide la formulation de besoins de changement en suggérant des améliorations de l'ontologie. Il offre une suite d'interfaces pour éditer des changements, vérifier la consistance de l'ontologie et créer des stratégies d'évolution personnalisées (Maedche *et al.*, 2003). L'ontologie est représentée formellement à l'aide d'un langage propre à KAON et plus récemment en OWL. KAON enregistre dans un journal tous les changements apportés à l'ontologie, et utilise le concept ChangeLog pour spécifier les types des changements : AddEntity, DeleteEntity ou ModifyEntity. L'outil ne permet pas malheureusement de gérer des changements complexes tels que fusionner ou diviser des entités ontologiques.

ECCO5 offre un environnement collaboratif et contextuel de construction d'ontologies sur lequel se base le système de gestion d'évolution CoSWEM6 (Corporate Semantic Web Evolution Management) (Luong *et al.*, 2007). CoSWEM gère l'enrichissement de vocabulaires structurés à partir de textes, ces vocabulaires étant utilisés pour produire des annotations sémantiques. Il garde l'historique du processus d'élaboration et de modification d'ontologie sous forme de métadonnées stockées au format RDF. Ces traces sont récupérées pour propager les changements de l'ontologie aux annotations qui la référencent selon des règles qui détectent et corrigent les annotations devenues obsolètes après l'évolution, en accord avec les

[3] http://scarlet.open.ac.uk/
[4] http://kaon.semanticweb.org
[5] http://www-sop.inria.fr/edelweiss/projects/ewok/publications/ecco.html
[6] http://www-sop.inria.fr/edelweiss/projects/ewok/publications/coswem.html

stratégies d'évolution définies dans (Luong, 2007). Plus récemment, au sein de la boîte à outils NEON ToolKit, EVOLVA7 met e œuvre le processus d'enrichissement que nous avons présenté dans la partie précédente (Zablith *et al.*, 2009).

Bien qu'EVOLVA et CoSWEM partent de termes pour enrichir une ontologie, ces termes deviennent des labels de concepts mais ne constituent pas des structures à part entière. Pour répondre au besoin de gérer des termes et des annotations associées à une RTO conforme au modèle de Dynamo, nous avons développé un nouvel logiciel qui maintient l'uniformité et la cohérence entre les différentes entités de la RTO d'une part, et les annotations sémantiques d'une autre part.

4. Approche pour l'évolution conjointe d'une RTO et d'annotations

4.1 Le processus d'évolution d'EvOnto

Le projet Dynamo prévoit plusieurs scénarios d'évolution, qui correspondent à des parcours différents sur la figure 3, de manière très proche aux scénarios proposés par (Djedidi, 2009). L'originalité d'EvOnto est d'assurer un support à ces différents scénarios, tout en gérant une forte imbrication entre la ressource termino-ontologique et les annotations sémantiques qu'elle permet de produire sur la collection de documents. D'une part, toute évolution de la RTO implique de détecter les annotations devenues incohérentes à cause des changements de la RTO et aussi de les corriger en vue de garantir la consistance de la base d'annotations. D'autre part, les évolutions de la base de documents et donc des annotations produites ou à produire peuvent indiquer la nécessité d'évolutions de la RTO, qui en retour impliquent ce maintien de cohérence sur l'ensemble des annotations. L'étude de la qualité des annotations comment moyen de justifier l'évolution de la RTO est un des originalités d'EvOnto. La qualité des annotations est évaluée par la vérification de certaines contraintes définies avant l'application du processus d'annotation. Si ces contraintes ne sont pas vérifiées alors nous avons besoin de faire évoluer la RTO.

Or les termes sont au cœur de cette imbrication, dans la mesure où le processus d'annotation de TextViz s'appuie sur les termes de la RTO pour annoter les documents par un graphe d'instances de concepts en relation. L'évolution de la RTO premier bloc sur figure 3) se déroule selon les étapes classiques de l'état de l'art. En revanche, chacune de ses étapes est innovante en traitant de manière couplée l'ontologie et sa composante terminologique. De plus, l'étape de propagation est particulièrement fouillée, et correspond au processus d'évolution des annotations.

EvOnto s'appuie sur trois caractéristiques originales : il accorde une attention particulière aux termes qui contribuent pour définir les annotations ; il définit des critères de qualité pour évaluer le résultat de l'annotation automatique de textes et

[7] http://evolva.kmi.open.ac.uk/

détecter des manques dans l'ontologie ; il suppose que la qualité d'une ontologie est mesurée par son utilisation pour l'annotation.

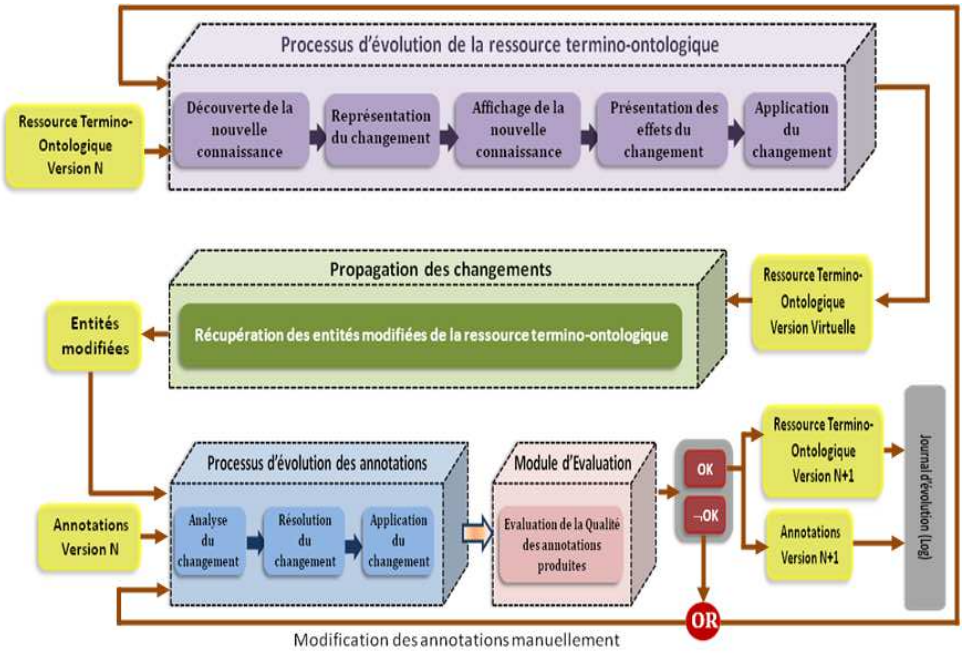

Figure 3. *L'architecture générale de l'approche EvOnto*

La méthode définit un processus cyclique que nous détaillons dans les parties suivantes : après l'annotation de nouveaux documents, EvOnto propose à l'ontologue de vérifier la qualité de ces annotations et, à partir des erreurs ou des manques identifiés, de formuler des besoins d'évolution de la RTO. Pour cela, différentes opérations de changement sont proposées et, pour chacune d'elles, EvOnto permet d'en régler les conséquences au sein de la RTO mais aussi sur les annotations. Enfin, le changement est rendu effectif dans la RTO puis au sein des annotations.

4.2 Déclencher un changement de la RTO

Alors que la plupart des travaux font l'hypothèse que les évolutions sont déclenchées à partir d'une analyse de l'ontologie elle-même (de son incohérence ou de ses lacunes), dans EvOnto, nous avons prévu trois motifs de son évolution : (i) une évolution des connaissances du domaine (apparition d'un nouveau terme ou d'un concept, modification du sens d'un terme, etc.)., (ii) le repérage d'erreurs ou d'améliorations possibles par l'ontologue (un terme n'est pas associé au bon

concept, la signature d'une propriété n'est pas correcte, deux concepts devraient être fusionnés, etc.), (iii) une analyse des nouvelles annotations suite à une évolution du corpus. Cette dernière situation est une des originalités d'EvOnto : faire évoluer la RTO à partir des annotations. Ce processus est basé sur 3 étapes.

Tout d'abord, lorsque des documents sont ajoutés ou retirés de la collection de textes, l'ontologue peut déclencher leur annotation automatique par TextViz. A cette étape, il se peut que l'ontologie ne soit plus en adéquation avec le corpus. D'une part, le retrait de documents peut rendre des termes et/ou des concepts inutiles. D'autre part, l'ajout de nouveaux documents peut mettre en évidence des termes des documents, pertinents pour leur annotation mais absents de la RTO avec le sens considéré. Dans ce cas, le système n'est pas capable d'annoter correctement le document : les annotations générées peuvent être incomplètes ou fausse

L'étape suivante d'EvOnto consiste alors à vérifier les annotations produites automatiquement, ce qui est une originalité de notre approche. Une vérification manuelle, basée sur la lecture des annotations par l'ontologue, peut conduire à des oublis. Pour repérer rapidement et systématiquement tous les documents mal annotés, EvOnto propose un module d'évaluation de la qualité des annotations.

Ce module demande à l'utilisateur de définir des critères de qualité des annotations propres au corpus et au domaine d'application. Un critère regroupe un ensemble de concepts et / ou de relations qui doivent être trouvés dans chaque annotation. Une annotation ne sera valable que si elle contient au moins une instance de chacun de ces concepts, éventuellement en relation. En général, ces concepts sont des classes de haut niveau dans l'ontologie. Un autre critère est le nombre de mots du texte auxquels on a pu associer un concept. Dans certains domaines, une bonne annotation est celle qui permet de couvrir le plus possible de termes dans une partie considérée du document (résumé, partie « symptôme » dans des fiches de diagnostic, etc.). Pour chaque document, un score évalue le pourcentage d'éléments reconnus, indiquant dans quelle mesure les critères sont vérifiés.

Finalement, l'ontologue peut vérifier une à une les annotations des documents dont le score est faible. Bien que ce processus soit manuel, il est efficace parce qu'il guide l'identification de concepts ou de termes manquants ou à modifier. Ceci revient à suggérer des corriger les annotations et surtout à faire évoluer l'ontologie.

4.3 Représentation du changement

Par le biais de l'éditeur d'ontologies, l'ontologue apporte les modifications souhaitées. Chaque modification peut avoir de nombreuses conséquences sur les éléments dépendants de celui modifié, que ce soit localement dans la RTO, ou au niveau des annotations produites avec cette RTO. La représentation du changement a pour but d'en présenter les conséquences avant d'en assurer la mise en œuvre.

Pour exprimer l'évolution, nous avons recensé l'ensemble des modifications que l'on peut apporter aux RTOs et nous leur avons donné une signification bien précise (Tissaoui, 2009), (Tissaoui *et al.,* 2011). Les changements peuvent être élémentaires (ajouter ou effacer un terme, un concept ou une relation), ou complexes (déplacement ou fusion de concepts). Cette typologie élargit celle de (Stojanovic, 2004) par la prise en compte du lexique de la RTO, qui peut évoluer lorsque les changements portent sur les termes et les liens de dénotations.

Pour chaque type de changement, nous avons identifié différentes stratégies d'évolution, c'est-à-dire les manières de gérer les conséquences de ce changement sur les entités de la RTO liées à l'entité modifiée initialement et sur les usages de la RTO. Ces stratégies permettent d'anticiper les différentes conséquences possibles d'un changement avant de le rendre effectif : par exemple, les fils d'un concept détruit peuvent être associés à son père, à Top ou à un autre concept choisi par l'utilisateur. Les stratégies sont présentées à l'ontologue avec leurs conséquences. Une fois une stratégie sélectionnée et adaptée, le changement est répercuté sur la RTO et les annotations.

Les stratégies d'évolution sont adaptables. Elles reposent sur un ensemble de conséquences prédéfinies pour chaque type de changement, mais l'ontologue peut ajuster les conséquences entité par entité s'il le juge nécessaire. Les informations présentées comprennent les éléments de RTO reliés et les documents annotés par l'élément modifié, mais aussi les documents annotés par les éléments de la RTO impactés par cette modification (Figure 3). Les stratégies sont adaptables pour mieux répondre à la diversité des raisons à l'origine d'un changement. En effet, pour une même modification, l'utilisateur adopte telle ou telle conséquence selon les documents ou les concepts. Or le système ne propose que des choix systématiques. Gérer cette diversité de manière interactive est donc une nécessité. Une interface homme-machine propose à l'utilisateur une séquence d'informations modifiables qui le guident pour ajuster les conséquences ou revoir la modification demandée.

4.4 Impact sur les annotations

Lorsque l'ontologue a choisi de faire évoluer la RTO, ce changement est répercuté sur les annotations. Pour gérer l'évolution conjointe des annotations, nous proposons un processus en deux temps : détecter les annotations qui impliquent au moins une entité de la RTO qui a évolué suite au changement puis modifier ces annotations devenues incohérentes.

Pour détecter les annotations à modifier, deux méthodes sont utilisées. La première est de rechercher dans les graphes d'annotation les nœuds qui correspondent à des entités de la RTO qui ont été modifiées. Par exemple, un triplet RDF où apparaît un concept qui a été supprimé doit clairement être remis en cause.

Pour la recherche des annotations concernées, nous utilisons des requêtes SPARQL sur l'ensemble des graphes correspondant aux annotations des documents.

Une fois les annotations incohérentes déterminées, elles doivent être corrigées. Nous proposons deux méthodes ici encore. Selon la première, l'ontologue relance le module d'annotation automatique seulement pour les documents concernés. Le temps de calcul reste raisonnable si la base de documents ne comprend que quelques centaines de documents courts. Pour les applications qui nécessitent de compléter manuellement les annotations, cette solution n'est pas pleinement satisfaisante car elle oblige l'ontologue à refaire les annotations manuelles. Afin de réduire ce coût, la deuxième méthode consiste à adapter les annotations localement dans le fichier d'annotation, par des requêtes changeant le type des instances par exemple, lorsque leur type est concerné par le changement de la RTO.

5. Outil EvOnto : deux scenarios d'évolutions

L'outil EvOnto met en œuvre cette approche. Il se présente comme un complément à TextViz (Reymonet *et al.*, 2009), intégré dans Protégé sous forme d'un plugin. A la phase actuelle d'avancement du projet, nous disposons d'un prototype dont les résultats nous permettent de montrer la faisabilité et l'importance de notre démarche.

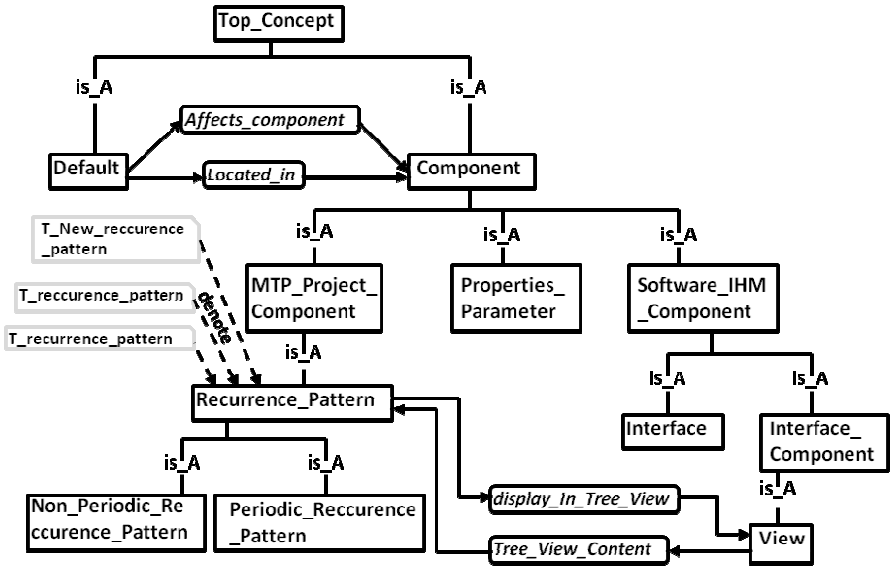

Figure 4. *Extrait d'une RTO du domaine des incidents de logiciel*

Nous illustrons le processus d'évolution d'une ressource termino-ontologique avec EvOnto à l'aide d'un scénario basé sur une RTO du domaine des incidents logiciels, dont un extrait est présenté en Figure 4. Cette RTO contient des concepts de haut niveau (*Trigger_Event, Default, Component*) et des relations sémantiques, comme *Located_in*, *Affects_Component*, *Causes* et *Concerns_Default*. *T_Default*, T*_Problem* et *T_Bug* sont des termes qui dénotent le concept *Default*. En figure 5, nous donnons un extrait de la représentation interne correspondant à un graphe d'annotation d'un document particulier, graphe qui s'appuie sur cette partie de la RTO.

```
1.  <Term_fr rdf:ID=" T_New_Reccurence_Pattern ">
2.  <dénote>
3.    <Concept rdf:ID= "#Recurrence_Pattern">
4.      <rdfs:subClassOf  rdf:resource="# MTP_Project_Component "/>
5.        <Recurrence_Pattern rdf:ID ="Onto_Individual_131">
6.        <Recurrence_Pattern rdf:ID ="Onto_Individual_132">
7.    </Concept>
8.   < T_New_Reccurence_Pattern rdf:ID="Onto_Individual_149">
9.      <document_id rdf:datatype="&xsd;string">Fiche-152.xml</document_id>
10.     <désigne rdf:resource="#Onto_Individual_131"/>
11.     <field_id rdf:datatype="&xsd;string">Resum&#233;</field_id>
12.     <spotted_words rdf:datatype="&xsd;string">[New Reccurence Pattern]</spotted_words>
13.     <term_offset rdf:datatype="&xsd;string">[0]</term_offset>
14.   </T_New_Reccurence_Pattern>
15.   <T_Reccurence_Pattern rdf:ID="Onto_Individual_148">
16.     <document_id rdf:datatype="&xsd;string">Fiche-151.xml</document_id>
17.     <désigne rdf:resource="#Onto_Individual_132"/>
18.     <field_id rdf:datatype="&xsd;string">Resum&#233;</field_id>
19.     <spotted_words rdf:datatype="&xsd;string">[Reccurence pattern]</spotted_words>
20.     <term_offset rdf:datatype="&xsd;string">[0]</term_offset>
21.   </ T_Reccurence_Pattern>
22. </dénote>
23. <owl:ObjectProperty rdf:ID="display_In_Tree_View">
24.   <rdfs:range>
25.    <Concept rdf:ID="Tree_View">
26.     <rdfs:subClassOf rdf:resource="#View"/>
27.    </Concept>
28.   </rdfs:range>
29.   …
30. </Term_fr>
```

Figure 5. *Extrait d'une représentation interne du graphe d'instance*

5.1 Aide à l'évaluation d'annotations

Chaque fois que TextViz produit une nouvelle annotation, une vérification de la qualité des annotations est lancée, selon des critères définis par l'utilisateur. Prenons l'exemple de la RTO de la figure1, les annotations doivent vérifier certains critères comme une couverture minimale du noyau de la RTO. Par exemple, on peut choisir comme critère que l'annotation d'une fiche doive contenir au moins :

- Une instance indirecte du concept « Default » ;
- Une instance indirecte du concept « Component » ;
- Une relation « Affects_Component » entre ces deux instances.

La figure 6 présente l'interface de définition de ces critères. Dans le rectangle A, l'ontologue peut choisir les concepts dont des instances directes et/ou indirectes doivent être identifiées dans chaque document. Dans le rectangle B, il indique au système les relations entre instances qui doivent être identifiées.

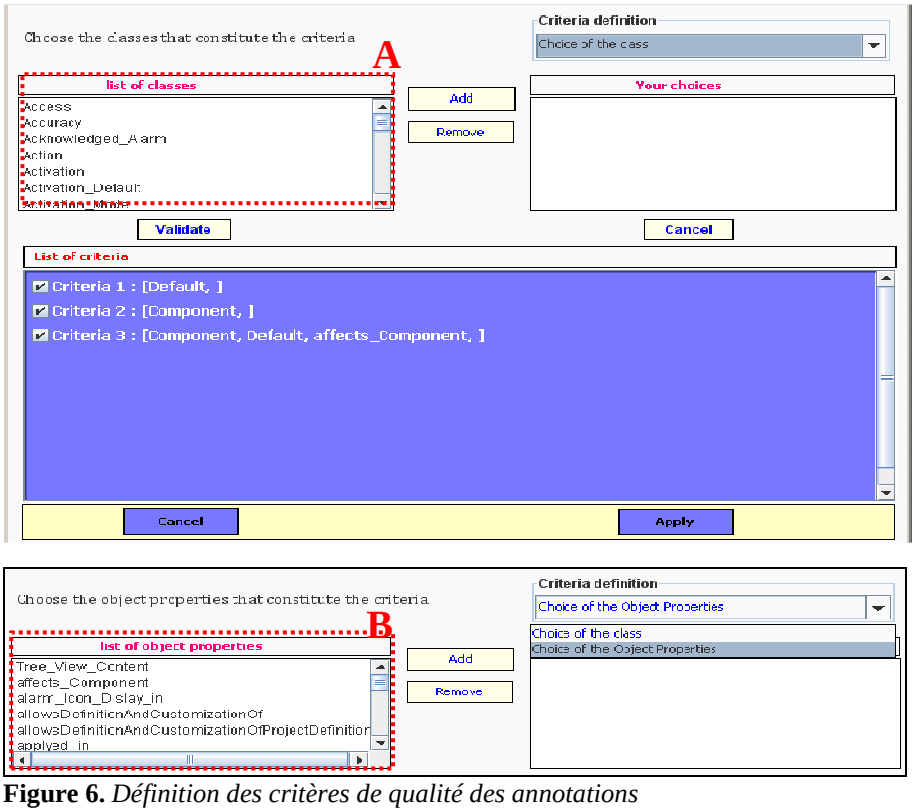

Figure 6. *Définition des critères de qualité des annotations*

Dans la version initiale de TextViz, ces critères étaient fixes pour une application donnée et insérés dans le code. En revanche, dans EvOnto, nous avons construit

cette interface qui permet de définir ces critères de manière interactive. Lorsque tous les critères sont vérifiés pour un texte donné, celui-ci est considéré comme bien annoté. Lorsqu'un ou plusieurs critères ne sont pas satisfaits, TextViz dispose d'information pour modifier automatiquement les annotations. Par exemple, si une contrainte spécifie que toute instance d'un concept donné doit être associée à une instance d'un second concept via une relation sémantique spécifique (i.e. la relation « affects_Component» avec une cardinalité minimum entre les concepts « Default » et « Component ») et si deux instances adéquates existent, alors le système pourrait créer la relation entre celles-ci.

Dans le cas le plus général et le plus fréquent, le système fournit à l'ontologue le texte mal annoté ainsi que les critères non satisfaits. Les résultats sont affichés dans un tableau avec les concepts attendus (une ligne pour chaque triplet qui contient le concept attendu). Les annotations ne remplissant pas les critères correspondent aux textes qui contiennent les triplets présents sur des lignes dont une ou plusieurs cellules sont vides. L'ontologue peut choisir alors d'enrichir ou de modifier la RTO. Il peut ajouter à la main de nouvelles instances de concepts comme annotations, il peut aussi modifier l'ontologie pour répondre aux besoins de l'annotation (ajouter de nouveaux termes à des concepts existants, définir de nouveaux concepts, …) en utilisant l'interface d'évolution de la RTO. Ensuite, les documents à l'origine de cette évolution doivent être ré-annotés et les annotations à nouveau évaluées.

5.2 Evolutions de la RTO

Cette section présente un exemple qui, nous l'espérons, permet de mieux faire comprendre ce qui motive notre approche. Cet exemple sera utilisé dans la suite du texte afin d'illustrer notre proposition.

L'opération de changement DeleteConcept supprime le concept choisi. Supposons que nous voulions supprimer le concept *Recurrence_Pattern* de la Figure 4. EvOnto ouvre une nouvelle fenêtre qui affiche toutes les conséquences de ce changement sur la ressource termino-ontologique et les annotations (Figure 7) selon la stratégie par défaut pour DeleteConcept, qui consiste à attacher les sous-concepts et les termes associés au concept père du concept *Recurrence_Pattern*. L'ontologue peut choisir une autre stratégie et vérifier ses conséquences, ou il peut décider d'abandonner la modification.

Quel que soit le changement sélectionné, concept et stratégie, les informations sont réparties en 4 zones (figure 7) : nom du changement actuel et la version modifiée du concept (CONCEPT A MODIFIER) ; informations sur la stratégie choisie (ou celle par défaut) ; situation de ce concept dans la hiérarchie du concept (concept Browser) et ses termes associés (les classes commençant par T_ dans le navigateur de terme) ; toutes les conséquences du changement (Informations Lexicales et Conceptuelles), avec la partie supérieure dédiée aux conséquences sur la

ressource termino-ontologique, et la partie inférieure qui répertorie les textes dont les annotations sont touchées.

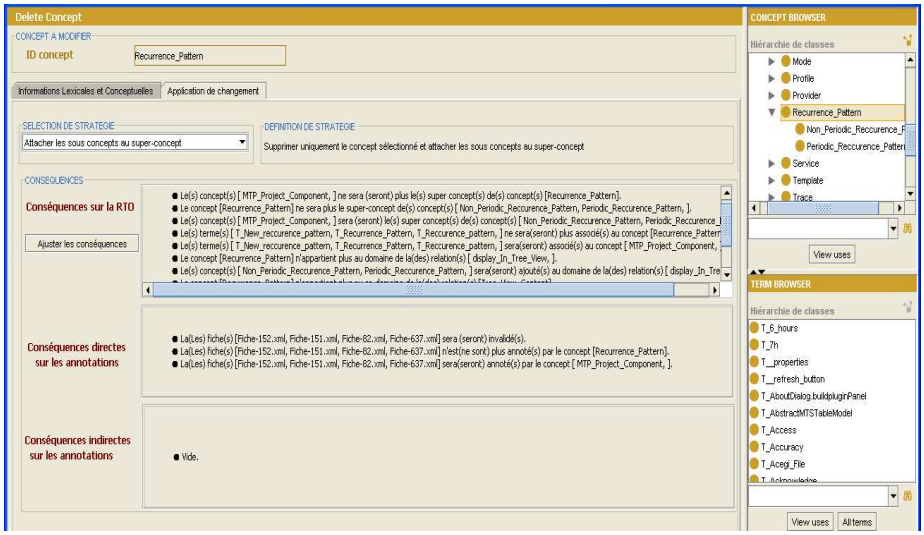

Figure 7. *Informations fournies lors de la sélection du changement DeleteConcept*

Dans l'exemple, trois stratégies d'évolution de la RTO (SRTO) sont possibles pour gérer les termes et les concepts fils du concept détruit par DeleteConcept :

- SRTO1 : Les attacher au concept père de celui détruit (stratégie par défaut) ;
- SRTO2 : Les attacher à DomainThing ;
- STRO3 : Les supprimer.

Les conséquences sur l'ensemble des éléments de la RTO de la stratégie par défaut pour DeleteConcept sont les suivantes :

– Le concept [*Recurrence_Pattern*] ne sera plus le père des concepts [*Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern*],

– Le concept [*MTP_Project_Component*] sera le père des concepts [*Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern*],

– Les termes [*T_New_reccurence_pattern, T_reccurence_pattern, T_recurrence_pattern*] ne seront plus associés au concept [*Recurrence_Pattern*], mais au concept [*MTP_Project_Component*].

– Le concept [*Recurrence_Pattern*] sera supprimé du domaine et du co-domaine des relations [*display_In_Tree_View, Tree_View_Content*],

– Les concepts [*Non_Periodic_Reccurence_Pattern, Periodic_ Reccurence_Pattern*] seront ajoutés au domaine et au co-domaine des relations [*display_In_Tree_View, Tree_View_Content*],

Ensuite, soit l'ontologue valide cette stratégie et tous les changements précédents, soit il choisit une autre stratégie, soit encore, quelle que soit la stratégie choisie, il modifie seulement une partie des conséquences (opération « Ajuster les conséquences »). Dans le cas particulier de DeleteConcept, il semble naturel d'adapter l'option par défaut qui suggère de distribuer la même liste de termes à deux concepts distincts. Cette option conduirait à des termes polysémiques qui étiquettent deux concepts et annotent les mêmes fragments de texte avec deux concepts différents. Or cette situation doit être évitée dans TextViz.

La fenêtre qui permet l'ajustement des conséquences est propre à chaque type de changement, suivant qu'il implique un ou plusieurs composants de l'ontologie. La figure 8 montre l'interface d'adaptation des conséquences du changement DeleteConcept après plusieurs modifications manuelles. Pour chaque type de données à modifier (sous-concepts, propriétés, etc.), la partie gauche de la fenêtre présente les entités liées au concept supprimé ; l'option Select permet de sélectionner une de ces entités, qui est alors déplacée vers le cadre de droite. Les changements possibles seraient de distribuer les termes et concepts fils du concept supprimé vers plusieurs autres concepts de la RTO.

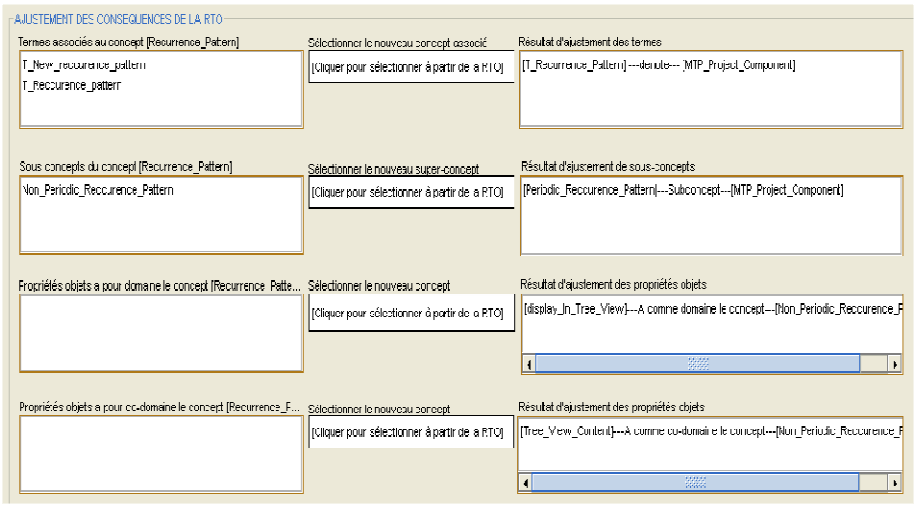

Figure 8. *Interface pour adapter les conséquences sur d'autres composants de la RTO du changement DeleteConcept*

La confirmation de ces choix provoque la mise à jour des conséquences directes et indirectes sur les annotations sur le panel 5 de la figure 2. En effet, les annotations étant liées à la présence d'occurrences de termes dans un document, le seul fait d'associer les termes à un concept précis décide des termes, des concepts, voire des relations, qui annotent ce document. Une fois les différentes conséquences validées, les changements sont effectués dans la RTO ainsi que dans les annotations.

5.3 Propagation des annotations

Pour gérer l'évolution des annotations, nous proposons un processus en trois temps (figure 9) :

- Analyser les types des entités modifiées, entités lexicales (les termes) ou entités conceptuelles comme les concepts et les relations - analyse du changement -.
- Rechercher dans la base d'annotations celles concernées par les modifications de la RTO – résolution du changement –
- Modifier les annotations devenues incohérentes à cause de ces changements – application du changement -.

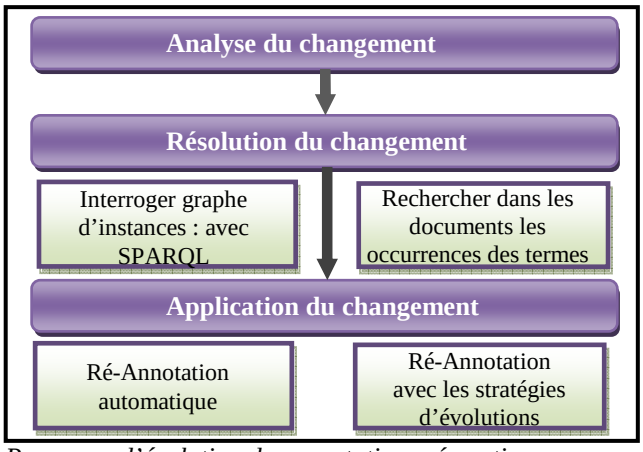

Figure 9. *Processus d'évolution des annotations sémantiques*

Pour la deuxième étape, deux méthodes sont utilisables.

- La première est de rechercher dans les graphes d'annotation les parties qui subissent un impact des modifications de la RTO. Ainsi, un triplet RDF où apparaît un concept qui a été supprimé doit être remis en cause. Pour la recherche des graphes concernés, nous utilisons des requêtes SPARQL sur l'ensemble des graphes correspondant à tous les documents déjà annotés.

- La deuxième méthode est de relancer l'annotation automatique des documents concernés, ce qui revient à rechercher dans les documents les occurrences des termes liés aux éléments de la RTO modifiés.

Si la première méthode, recherche des graphes impactés, a été utilisée, il est possible, au moins dans certains cas, d'opérer localement sur ces graphes. A cette fin, nous appliquons des stratégies d'évolution d'annotations (SA). Chacune d'entre elles est associée à une des stratégies d'évolution de la RTO (SRTO). Une SA a pour but de corriger les inconsistances que peut produire le changement dans les annotations sémantiques. Elle définit pour chaque opération de changement des règles pour corriger les unités d'annotations.

Le tableau 1 présente les conséquences de l'application de SRTO2 (stratégie 2 pour la RTO) et de la SA2 associée, dans le cas de l'exemple de DeleteConcept(RecurrencePattern). Rappelons que ces stratégies ne seront appliquées telles quelles que si l'utilisateur les accepte sans en modifier les conséquences.

Stratégies d'évolution pour la RTO : SRTO2
– Le concept [DomainThing] sera le père des concepts [Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern], – Les termes [T_New_reccurence_pattern, T_reccurence_pattern, T_recurrence_pattern] ne seront plus associés au concept [Recurrence_Pattern], mais associés au concept [DomainThing]. – Le concept [Recurrence_Pattern] sera supprimé du domaine et du co-domaine des relations [display_In_Tree_View, Tree_View_Content], – Les concepts [Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern] seront ajoutés au domaine et au co-domaine des relations [display_In_Tree_View, Tree_View_Content],
Stratégies d'évolution pour l'annotation : SA2
– Le concept Recurrence_Pattern appartient au domaine (respectivement co-domaine) des relations [display_In_Tree_View, Tree_View_Content], alors remplacer toutes les instances du type Recurrence_Pattern par une instance de type [DomainThing] y compris dans les triplets contenant la relation display_In_Tree_View et Tree_View_Content

Tableau 1. *Application des stratégies SRTO2 et SA2 sur l'ontologie et les annotations.*

6. Évaluation des résultats

Pour la première expérimentation d'Evonto, nous avons fait des tests avec deux corpus et deux ressources termino-ontologiques fournis par les sociétés ARTAL technologies et ACTIA. La RTO d'ARTAL possède environ 569 concepts et de 787 termes. Le corpus associé contient uniquement des documents en anglais. Il est composé de 769 fiches annotées sémantiquement à l'aide de plusieurs triplets. La

RTO d'ACTIA possède environ 340 concepts et 564 termes. Le corpus d'ACTIA est composé de 702 fiches annotées sémantiquement par plusieurs triplets.

La première expérimentation illustre un processus d'évolution des annotations suite à l'évolution de la RTO en utilisant EvOnto et en exploitant les informations de l'exemple précédent (DeleteConcept : *Recurrence_Pattern*). Le Tableau 2 montre le résultat de la détection des triplets inconsistants, le nombre de triplets inconsistants (3ème colonne) contenant l'élément affecté équivalent (2ème colonne). La quatrième colonne de ce tableau représente le nombre de triplets inconsistants détectés par EvOnto. Chaque ligne correspond à une des conséquences du changement demandé. Le nombre de triplets inconsistants est donné après la lecture des fiches par l'ontologue. Le système a détecté 24 triplets inconsistants pour l'élément *Recurrence_Pattern*, 6 triplets inconsistants pour les éléments *Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern*, 9 triplets inconsistants pour les éléments *T_New_reccurence_pattern, T_reccurence_pattern, T_recurrence_pattern* et 4 triplets inconsistants pour les éléments *display_In_Tree_View, Tree_View_Content*. Le résultat de la détection des triplets inconsistants est exact.

Changement effectué	Eléments affectés	Nombre des triplets inconsistants	Nombre des triplets inconsistants détectés
DeleteConcept : changement de base	Recurrence_Pattern	24	24
CreateHierarchy-ConceptLink: changement supplémentaire	Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern	6	6
MoveTerm : changement supplémentaire	T_New_reccurence_pattern, T_reccurence_pattern, T_recurrence_pattern	9	9
MoveRelation: changement supplémentaire	display_In_Tree_View, Tree_View_Content	4	4

Tableau 2. *Résultat de la détection des triplets inconsistants*

Une fois les triplets détectés, EvOnto applique les règles de rectification, dont les résultats sont synthétisés dans le tableau 3. C'est l'onologue qui juge de la pertinence des modifications. Parmi les 9 triplets inconsistants détectés associés aux termes *T_New_reccurence_pattern, T_reccurence_pattern, T_recurrence_pattern*, 8 ont été bien rectifiés dans le fichier d'annotation. 24 triplets inconsistants détectés associés au concept *Recurrence_Pattern* ont été bien rectifiés, 4 triplets inconsistants détectés associés aux relations *display_In_Tree_View, Tree_View_Content* ont été bien

rectifiés et 6 triplets associés aux concepts T *Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern* ont été bien rectifiés.

Changement effectué	Eléments affectés	Nombre des triplets inconsistants détectés	Nombre des triplets inconsistants rectifiés
DeleteConcept : changement de base	Recurrence_Pattern	24	24
CreateHierarchy-ConceptLink: changement supplémentaire	Non_Periodic_Reccurence_Pattern, Periodic_Reccurence_Pattern	6	6
MoveTerm : changement supplémentaire	T_New_reccurence_pattern, T_reccurence_pattern, T_recurrence_pattern	9	8
MoveRelation: changement supplémentaire	display_In_Tree_View, Tree_View_Content	4	4

Tableau 3. *Résultat de la détection des triplets inconsistants rectifiés*

Le tableau 2 montre notre capacité à retrouver précisément les textes du corpus touchés par une évolution, grâce à des requêtes SPARQL dans la base d'annotations ainsi que dans la ressource termino-ontologique. Pour la deuxième évaluation (tableau 3), les résultats détectés sont aussi satisfaisants. Le tableau montre une proportion élevée de triplets détectés rectifiés. Nous pouvons dire que les résultats obtenus sont encourageants et que l'approche EvOnto semble pertinente.

7. Conclusion et perspectives

Le travail présenté dans cet article peut être comparé à certaines études similaires présentées dans (Luong, 2007) (Rogozan, 2009) et (Stojanovic et 2004). Notre contribution est d'associer l'évolution de la ressource termino-ontologique, les types de changements applicables (élémentaire ou complexe), même si nous nous sommes contentés d'un seul exemple d'opération dans ce papier, ainsi que l'évolution des annotations sémantiques des documents. Cette contribution préserve la cohérence de la RTO et conjointement celle des annotations sémantiques des collections documentaires annotées.

Notre recherche se situe dans cet axe en y apportant certaines originalités. Tout d'abord, nous nous intéressons à l'évolution d'ontologies à composantes lexicales, ou ressources termino-ontologiques, dans lesquelles les termes ont un statut à part entière à côté des composants habituels d'une ontologie. Ensuite, nous intégrons

dans le processus d'évolution le fait que l'ontologie soit utilisée pour un objectif précis, l'annotation sémantique de documents dans un contexte applicatif. Nous avons cherché particulièrement à éviter de devoir vérifier manuellement à nouveau l'annotation de certains documents, ce qui est coûteux en temps de travail humain. Enfin, nous nous intéressons à un scénario d'évolution où la RTO doit évoluer suite à la vérification des critères de qualité des annotations.

EvOnto accompagne le processus d'évolution en montrant à l'utilisateur les conséquences d'un changement sur les entités de la RTO reliées à celle modifiée, et sur les annotations des documents. Ces informations l'aident à prendre une décision.

Une expérience d'évolution de RTO réalisée avec EvOnto a permis d'obtenir des résultats très encourageants et a mis en avant la nécessité de faire évoluer le logiciel pour une meilleure qualité des résultats. Nous continuons à améliorer EvOnto en ajoutant des nouveaux changements et les stratégies correspondantes pour gérer les conséquences dans la RTO et sur les annotations.

8. Remerciements

Ce projet bénéficie du financement ANR 07-TLOG-004-01 de l'Agence Nationale de la Recherche. Les partenaires en sont : Préhistoire et Technologie[8] (), Actia[9], Artal[10], LaLIC et IRIT[11]. Nous remercions l'ANR ainsi que tous les partenaires du projet DYNAMO pour leur contribution à ce travail.

9. Bibliographie

Aussenac-Gilles N., Condamines A., Sèdes F. (eds), « Ressources termino-ontologiques », vol. Hors-série of *Revue I3 (Information – Interaction – Intelligence)*, Cépaduès-Editions, 2006.

Berners-Lee T., Hendler J., Lassila O., « *The Semantic Web* », The scientific American. May 2001.

Cimiano P., *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*, Springer, October, 2006.

Cimiano P., Völker J., « Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery », in A. Montoyo, R. Munoz, E. Metais (eds), *Proceedings of the 10th Int. Conference on Applications of Natural Language to Information Systems (NLDB)*, vol. 3513 of LNCS, Springer, Heidelberg, p. 227-238, June, 2005.

[8] http://www.mae.u-paris10.fr
[9] http://www.actia.com
[10] http://www.artal.fr
[11] http://www.irit.fr

Djedidi R., Approche d'évolution d'ontologie guidée par des patrons de gestion de changement. Thèse de doctorat, université Paris-Sud XI Orsay. 2009.

Flouris G., On belief change and ontology evolution, Thèse de doctorat, University of Crete, Department of Computer Science, Heraklion, Greece, 2006.

Flouris G., Plexousakis D., Antoniou G., « A classification of ontology change », *3rd Italian Semantic Web Workshop Scuola Normale Superiore*, Pisa, Italy, 18-20 December, 2006.

Klein, M., Change management for distributed ontologies. Thèse de doctorat. Thesis, Dutch Graduate School for Information and Knowledge Systems. Germany, 2004.

Luong P.H., Gestion de l'évolution d'un web sémantique d'entreprise. Thèse de doctorat, école de Mines de Paris, 2007.

Luong, P.H., Dieng-Kuntz, R., « A Rule-based Approach for Semantic Annotation Evolution ». *The Computational Intelligence Journal*, 23(3):320-338. Blackwell Publishing, Malden, MA 02148, USA, 2007.

Mäedche A., *Ontology learning for the Semantic Web*, vol. 665, Kluwer Academic Pub., 2002.

Mäedche A, Motik B, Stojanovic L., « Managing multiple and distributed ontologies in the Semantic Web ». *Very Large Data Bases* (*VLDB Journal)*, 12(4), 286–300, 2003.

Plessers, P, De Troyer O., « Ontology change detection using a version log ». In Y.Gil, E. Motta, V.R. Benjamins, M. Musen (Eds.), LNCS: Vol. 3729. *The Semantic Web – ISWC 2005. Berlin*, Germany: Springer-Verlag, 578-592, 2005.

Reymonet A., Thomas J., Aussenac-Gilles N., « Ontology Based Information retrieval: an application to automotive diagnosis ». *International Workshop on Principles of Diagnosis*. Stockholm M. Nyberg, E. Frisk, M. Krisander, J. Aslund (Eds.), Linköping University, Institut of Technology, pp 9-14, 2009.

Rogozan D., Gestion de l'évolution des ontologies : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions d'ontologie. Thèse de doctorat, université du Québec à Montréal, 2009.

Rogozan D., Paquette G., « Managing Ontology Changes on the Semantic Web ». *In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pp. 430-433, 2005.

Stojanovic L., Methods and Tools for Ontology Evolution, Thèse de doctorat, Karlsruhe University. Germany, 2004.

Stojanovic, L., Maedche, M., Stojanovic, N., Studer, R., « Ontology evolution as reconfiguration-design problem Solving ». *2nd international conference on Knowledge Capture K-CAP'03*, pp.162-171. New York, USA: ACM. ISBN: 1-58113-583-1, 2003.

Tissaoui A., « Typologie de changements et leurs effets sur l'évolution de Ressources Termino-Ontologiques ». (Poster) Dans *20e Journées Francophones d'Ingénierie des Connaissances*, Hammamet (Tunisie), 25-29 mai, 2009.

Tissaoui A., Aussenac-Gilles N., Laublet P., Hernandez N., « EvOnto: évolution de ressource termino-ontologique pour l'annotation sémantique ». Dans : *Journées Francophones sur*

les Ontologies (JFO 2011), Montréal (Canada), 22-23/06/2011, R. Bouaziz, P. Bourque, G. Falquet (Eds.), ACM SIGAPPFR, p. 5-17, 2011.

Zablith F., Sabou M., d'Aquin M., Motta E., « Ontology Evolution with Evolva ». In: *Proceedings of the 6th European Semantic Web Conference (ESWC)* LNCS 5554. Eds. L. Aroyo et al., Springer-Verlag, Berlin, Heidelberg, pp. 908-912, 2009.