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

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

* *Institut de Recherche en Informatique de Toulouse - IRIT- UMR 5505*
Université Paul Sabatier, 118 Route de Narbonne F-31062, Toulouse cedex 9
{Tissaoui, Aussenac, Hernande}@irit.fr

** *Laboratoire Sens, Texte, Informatique et Histoire - STIH – EA 4509*
Maison de la recherche - Université de Paris-Sorbonne, 28 rue Serpente, 75006 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

Définies comme des représentations consensuelles et stables, les ontologies sont utilisées aujourd'hui comme des modèles de connaissances d'un domaine pour des applications spécifiques, dans des contextes en évolution régulière. L'ontologie doit alors être modifiée pour être adaptée à la tâche ou aux données auxquelles elle est associée, tout en restant manipulables par les systèmes informatiques et interprétables par les êtres humains (Maedche, 2002). Appliquer des changements à une ontologie vise à la modifier pour la rendre plus adéquate au domaine qu'elle modélise et à ses objectifs d'usage. La modification d'ontologie présente plusieurs difficultés aussi bien sur un plan pratique que théorique. Pratiquement, il n'est pas toujours évident d'appréhender ce qui est attendu d'un besoin de changement, ni comment un changement peut être réalisé, ni encore comment ce changement va modifier le comportement des systèmes utilisant l'ontologie. De plus, les procédés d'évolution peuvent être longs et demandent une constante vérification de l'ontologie au regard du domaine à modéliser. Théoriquement, l'évolution remet en question la stabilité de l'ontologie et du domaine, et suppose de maîtriser la sémantique de la représentation des connaissances choisie pour cette ontologie

Depuis 2002 environ, l'évolution des ontologies a fait l'objet de recherches qui ont permis d'identifier les différents types de changement possibles, les étapes d'un processus de changement, les aides possibles pour mieux gérer les versions d'ontologies ou encore la gestion des répercussions logiques d'une modification locale sur l'ontologie toute entière (Stojanovic, 2004) (Klein, 2004). Notre recherche se situe dans cet axe en y apportant deux originalités. Tout d'abord, nous nous intéressons à l'évolution d'ontologies à composantes lexicales, ou ressources termino-ontologiques (RTO), 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 le cas qui nous intéresse. Une hypothèse forte de notre travail est que la prise en compte des impacts d'une modification de l'ontologie sur l'ontologie 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 l'ontologie renouvelle les questions classiques soulevées par la gestion de l'évolution : comment appliquer un changement tout en assurant la cohérence au sein de la ressource termino-ontologique mais aussi la cohérence de la RTO avec ses utilisations ? Comment analyser les effets d'un changement (au sein de la RTO mais aussi sur ses utilisations) et les gérer au mieux (par exemple en les minimisant) ? Et si plusieurs solutions sont possibles, quelles informations ou critères présenter à l'ontographe pour l'aider à choisir ?

Ces questions nous ont guidés pour vérifier notre hypothèse en définissant un logiciel et un processus de gestion de changement pour une ressource termino-ontologique servant à l'annotation sémantique de documents textuels : EvOnto (***Ev***olution d'***Onto***logie). EvOnto est destiné à un ontographe et le guide interactivement pour formuler une demande de changement, évaluer son impact (effets supplémentaires) sur la qualité de la ressource termino-ontologique et aussi sur les annotations sémantiques, et décider ensuite de leur mise en œuvre. Des informations sur l'utilisation de l'ontologie sont fournies à l'ontographe pour qu'il prenne l'initiative d'une évolution de la RTO, en connaisse les conséquences, et les adapte pour minimiser les effets négatifs, les impacts non souhaitables ou les coûts correspondants sur l'ontologie elle-même et son utilisation dans des annotations. EvOnto a été défini dans le cadre du projet ANR DYNAMO. Il se base sur l'outil d'annotation automatique de documents TextViz (Reymonet *et al.*, 2009), défini comme un plug-in de l'éditeur d'ontologies Protégé.

Nous présentons dans la section 2 le contexte de ce travail, à savoir le projet DYNAMO, le modèle de RTO utilisé et le modèle d'annotation basé sur ce modèle. Nous dressons dans la section 3 un état de l'art sur les logiciels de gestion de l'évolution d'ontologie. Dans la section 4, nous présentons les principes du système EvOnto et détaillons son fonctionnement sur une étude de cas. Les perspectives de cet article portent sur les améliorations prévues d'EvOnto.

2. DYNAMO : ontologies dynamiques pour l'annotation sémantique

2.1. Contexte

L'objectif principal du projet DYNAMO[1] (DYNAMic Ontology for information retrieval) est de concevoir une méthode et un ensemble d'outils logiciels qui gèrent la construction et surtout l'évolution de ressources termino-ontologiques à partir de documents textuels ainsi que l'utilisation de ces ressources pour une indexation sémantique facilitant la recherche d'information dans ces documents. 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 de documents 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. Plusieurs approches complémentaires sont expérimentées dans le projet : EvOnto, présenté dans cet article, et DYNAMO-SMA, une approche à base de systèmes multi-agents adaptatifs (Sellami et al., 2009). Dans ce projet, les corpus documentaires fournis par les partenaires sont issus de trois domaines particuliers :

[1] Ce projet bénéficie d'un financement ANR 07 TLOG 004 01 de l'Agence Nationale de la Recherche. http://www.irit.fr/DYNAMO

la recherche en archéologie des techniques, le diagnostic de pannes automobiles et le diagnostic de défauts logiciels.

2.2. Le modèle d'ontologie

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. La RTO contient alors non seulement une représentation des concepts du domaine et de leurs relations, mais aussi une représentation séparée des termes associés (termes désignant les concepts). 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 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 de terme 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 propriété hasTermOccurrence une instance de la classe `document` qui réifie la notion de document.

2.3. Le prototype TextViz et le modèle d'annotation sémantique

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. Il bénéficie ainsi des interfaces de Protégé pour la gestion de la RTO, interfaces adaptées de manière à gérer les termes en plus des concepts et relations. TextViz permet de visualiser les documents textuels du corpus ainsi que leurs annotations. Ces annotations sémantiques sont créées automatiquement par un algorithme qui identifie en corpus des fragments de texte (un mot ou une séquence de mots), proches des termes de la RTO. Ces fragments de textes sont alors annotés par des instances des concepts dénotés par ces termes (des entités anonymes) et des occurrences de termes. Lorsque ces concepts sont reliés dans l'ontologie et que des indices linguistiques de surface permettent d'identifier les instances en relation, des relations entre instances sont également définies. L'ensemble de ces instances et des relations sémantiques identifiées entre elles dans le document forme un ou plusieurs graphes qui constituent l'annotation sémantique. Les annotations proposées par le système peuvent être vérifiées et modifiées par un annotateur. TextViz propose également une fonction d'interrogation des documents annotés. Pour cela, la requête est

également annotée sous forme de graphe, puis le graphe requête et les annotations de documents sont rapprochés en utilisant une distance sémantique.

3. Evolution d'ontologie : état de l'art

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 dépendants 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. Notre objectif étant de fournir une aide à l'évolution de l'ontologie, nous ne mentionnons ici que des logiciels du premier type.

Un des premiers éditeurs d'ontologie à intégrer un outil de gestion automatisée des évolutions d'une ontologie a été la plateforme **KAON**[2] 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'outils 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.

ECCO[3] offre un environnement collaboratif et contextuel de construction d'ontologies sur lequel se base le système de gestion d'évolution **CoSWEM**[4] (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'une 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 des stratégies d'évolution définies dans (Luong, 2007).

[2] http://kaon.semanticweb.org
[3] http://www-sop.inria.fr/edelweiss/projects/ewok/publications/ecco.html
[4] http://www-sop.inria.fr/edelweiss/projects/ewok/publications/coswem.html

Plus récemment, au sein de la boîte à outils NEON ToolKit, **EVOLVA**[5] facilite l'évolution d'ontologies en exploitant des textes et en réutilisant des ressources comme des ontologies ou des bases de données (Zablith *et al.,* 2009). Les nouveaux concepts intégrés par EVOLVA découlent de l'extraction de termes à partir d'un corpus textuel. Ils sont placés dans l'ontologie à l'aide de nouvelles relations en utilisant des ressources en ligne (comme WordNet) et des ontologies retrouvées à l'aide du logiciel Scarlet. EVOLVA contient cinq composants dont le dernier, «Gestion de l'évolution», permet à l'ontologue de contrôler l'évolution.

4. EvOnto : outil pour soutenir l'évolution des RTOs

4.1. Principes

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 outil 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. Cet outil, EvOnto, se présente comme un complément à TextViz (Reymonet et al., 2009), intégré dans Protégé sous forme d'un plugin.

Selon EvOnto, le processus d'évolution utilise des documents textuels et des connaissances du domaine pour faire évoluer une RTO et mettre à jour l'annotation de ces documents à l'aide de cette RTO. La nouveauté de notre problématique réside dans la spécification conjointe des modules d'évolution d'une RTO et d'évolution des annotations sémantiques d'un corpus. D'une part, l'évolution de ce corpus a des répercussions sur la RTO et sur les annotations sémantiques produites à l'aide de cette RTO. D'autre part, l'évaluation de la qualité des annotations provoque éventuellement l'évolution de la RTO.

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) pour définir une typologie de changements applicables à une RTO définie selon le modèle de (Reymonet *et al.,* 2009). Les changements peuvent être élémentaires, comme l'ajout ou l'effacement d'un concept, d'un terme ou d'une relation, mais aussi plus complexes, comme le déplacement ou la fusion de concepts. Cette typologie élargit la conceptualisation de (Stojanovic, 2004), par l'ajout d'un certain nombre de caractéristiques dont la plus notable est la présence d'un lexique de RTO qui peut évoluer à cause des changements portant sur les termes et les liens de dénotations.

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

Pour chaque type de changement, nous avons identifié différentes stratégies d'évolution, c'est-à-dire des manières de gérer les conséquences de ce changement sur les autres composants de la RTO liés à celui modifié. Ces stratégies permettent de présenter à l'ontologue les différentes conséquences possibles d'un changement avant de le rendre effectif dans la RTO, puis de le répercuter sur les utilisations de l'ontologie. Parmi ces informations, l'originalité d'EvOnto est de mentionner les documents annotés par l'élément modifié (nous parlerons de *conséquences directes sur les annotations*) et par les éléments de la RTO modifiés en conséquence de cette modification (nous parlerons de *conséquences indirectes sur les annotations*). De plus, les stratégies d'évolution sont adaptables. Elles proposent d'abord un ensemble de conséquences qui traitent en bloc toutes les entités d'un même type, que l'utilisateur peut ensuite ajuster entité par entité si nécessaire.

4.2. Scénario d'utilisation d'EvOnto

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 logiciel, dont un extrait est présenté en figure 1. 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.

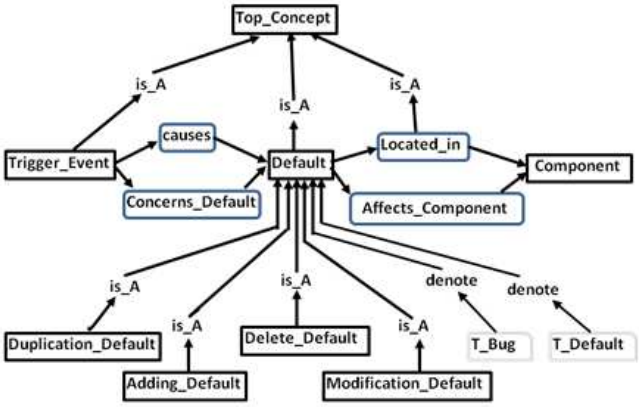

Figure 1. *Extrait d'une RTO du domaine des incidents de logiciel*

Le processus commence par la sélection de l'entité à modifier, qui se fait comme dans Protégé. Le 'popup_menu' que nous avons ajouté au panel de la hiérarchie des concepts présente séparément les changements élémentaires (Create, Rename et Delete) des changements composés (Move, Merge et Split). Prenons l'exemple de l'opération « Split Class » qui permet de supprimer un concept sélectionné pour le remplacer par deux nouveaux concepts. Supposons que la RTO de la figure 1 soit

modifiée en divisant le concept [*Default*] en deux sous concepts [*Edition_Default*] et [*Classification_Default*]. La figure 2 montre la fenêtre affichée par EvOnto après la sélection de «split class». Cette interface présente les conséquences de cette modification selon la stratégie prévue par défaut. Ces conséquences visent à maintenir la cohérence de la RTO et entre la RTO et les annotations de documents. L'ontologue peut décider de changer de stratégie, d'ajuster plus finement chacune des conséquences, ou de renoncer à cette modification. La fenêtre d'information est composée de six panels :

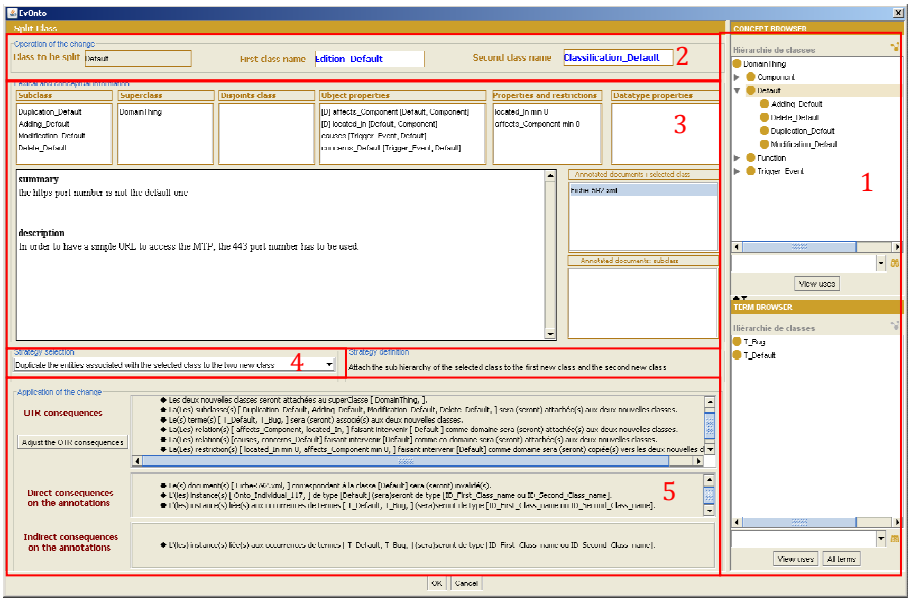

Figure 2. *Affichage des conséquences de l'opération Split Class*

(1) Un panel présente la composante conceptuelle, l'ontologie dans le « Concept Browser » et la composante lexicale, la terminologie dans le «Term Browser» (classes commençant par T_).

(2) Le panel nommé « Change Operation » affiche le nom de l'opération en cours, le concept sélectionné («*Default*» dans notre exemple) et, dans le cas de « split class », deux zones de textes pour saisir les deux nouveaux concepts qui remplaceront celui sélectionné.

(3) Le panel nommé « Lexical and Conceptual Information » affiche toutes les entités liées au concept sélectionné : ses concepts fils [*Duplication_Default*, *Delete_Default*, *Adding_Default*, *Modification_Default*], son concept père [*Top_Concept*], les concepts disjoints, les relations ou restrictions de relations dont il est domaine ou co-domaine [*Located_in*, *Affects_Component*,…], les datatypes

properties, les documents annotés par les instances du concept sélectionné (ici, *fiche_592.xml*) ou de ses fils et par des occurrences des termes associés.

(4) « Strategy Selection » : Ce panel propose la stratégie définie par défaut pour ce changement et permet de la modifier en sélectionnant une stratégie alternative. Nous avons adapté les stratégies déjà définies par Luong (2007) et Stojanovic (2004) aux évolutions d'une RTO comportant des termes. Dans le cas de « Split Class », trois stratégies sont possibles :

- associer aux deux nouveaux concepts les entités associées au concept sélectionné (affichée par défaut),
- associer ces entités au premier nouveau concept,
- associer ces entités au deuxième nouveau concept.

(5) « Change Validation » : Pour chaque stratégie d'évolution sélectionnée, le système avertit l'utilisateur et présente les conséquences engendrées par ce changement sur la RTO (partie supérieure) et les effets du changement sur les annotations sémantiques, directes et indirectes (partie inférieure). Ces informations lui permettent de choisir la stratégie qui lui semble la mieux adaptée. Par exemple, si l'utilisateur sélectionne la première stratégie (associer les entités aux deux concepts créés, cas présenté en figure 1), le système affiche les conséquences suivantes :

- Les concepts [*Duplication_Default]*, [*Delete_Default*], [*Adding_Default]*, [*Modification_Default*] seront fils des concepts [*Edition_Default*] et [*Classification_Default*].

- Les termes [*T_Bug*, *T_Default*] seront associés aux deux nouveaux concepts.

- Les propriétés [*Located_in)] et* [*Affects_Component*] auront pour domaine les deux concepts [*Edition_Default*] et [*Classification_Default*].

- Les propriétés [*Causes*] et [*Concerns_Default*] recevront désormais le premier nouveau concept comme co-domaine.

- Le document [*fiche_952.xml*] sera annoté désormais par la même instance du terme [*T_default*] mais l'instance de concept associée sera celle du premier concept [*Edition_Default*]. En effet, TextViz n'autorise qu'une seule instance de concept pour une occurrence de terme donnée.

Avant de valider pour appliquer pratiquement le changement et ses conséquences, l'utilisateur peut ajuster les conséquences en sélectionnant le bouton « Adjust Consequences on the OTR » qui ouvre un nouveau panel. Ceci est tout à fait justifié dans le cas de cette stratégie car il n'est pas très cohérent, dans une perspective d'annotation sémantique, de laisser la même liste de termes exactement à deux concepts différents. L'interface d'ajustement est spécifique à chaque type de changement afin de tirer le meilleur parti possible de ce que signifie ce changement. La figure 3 présente le panel d'ajustement des conséquences de l'opération « Split Class » avant toute intervention de l'utilisateur.

(6) « Adjust Consequences on the OTR » : Le système donne le droit à l'utilisateur d'ajuster les conséquences entité par entité, c'est-à-dire que l'utilisateur peut annuler ou modifier une des conséquences du changement prévues par la stratégie. L'utilisateur peut décider d'associer à d'autres concepts chacun des concepts fils du concept sélectionné, les termes qui le dénotaient, ses propriétés (object properties, datatype properties) et les restrictions. Dans le cas de l'opération « Split Class », l'utilisateur doit répartir chacune des entités entre les deux concepts remplaçant le concept supprimé. En effet, il ne semblait pas faire sens d'associer ces éléments à d'autres concepts. En revanche, ils peuvent être supprimés de chacune des listes ou des deux et, dans ce dernier cas, ils ne sont associés à aucun concept.

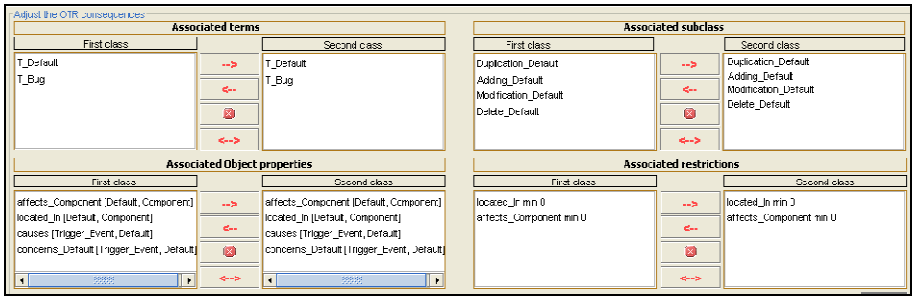

Figure 3. *Interface pour adapter les conséquences de l'opération « Split Class »*

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. Ce 2^{e} processus consiste à modifier le moins possible les annotations en remplaçant par de nouveaux triplets les triplets d'annotation touchés par le changement.

4.3. Perspectives d'évaluation

EvOnto aide l'ontologue à faire évoluer une RTO et les annotations de documents faites grâce à cette RTO. L'outil permet également de prendre des décisions que nous espérons plus pertinentes grâce à l'affichage des conséquences de chaque changement. Or il est extrêmement compliqué d'évaluer correctement un outil d'aide à l'utilisateur. Bien que nous en soyons à la première phase de la mise en œuvre et donc qu'une évaluation complète en situation n'ait pas été réalisée, nous avons fait quelques expériences avec deux de nos applications, basées l'une sur la RTO du domaine des incidents logiciel et l'autre sur une RTO du domaine du diagnostic automobile. Nous avons repris l'opération « Split Class » et avons

comparé le temps nécessaire pour effectuer ce type d'opération avec EvOnto et dans Protégé directement. Nous n'avons pris en compte que les conséquences sur la RTO. En effet, en utilisant Protégé, les conséquences sur les annotations ne peuvent être traitées qu'en lançant une nouvelle annotation automatique à l'issue de l'évolution de la RTO. Le temps d'exécution d'un changement de base et des changements conséquences de celui-ci sur la RTO est estimé à environ 1 minute avec EvOnto alors qu'avec Protégé, ce temps est estimé à 2 minutes. L'analyse du journal des changements de deux outils montre que le journal d'évolution d'EvOnto prend en compte tous les changements (changements de base et conséquences) alors qu'avec Protégé, seulement trois des changements ont été enregistrés : création de deux nouveaux concepts et suppression du concept sélectionné.

Enfin, outre la dimension quantitative, la contribution d'EvOnto est avant tout qualitative. Le fait de prévoir, dans les stratégies d'évolution, l'ensemble des conséquences liées à un changement, au sein de la RTO mais aussi sur les annotations associées, permet à l'utilisateur de ne pas oublier de les traiter. Dans Protégé, c'est à l'utilisateur d'aller vérifier les relations dont un concept modifié est domaine ou co-domaine, ou d'anticiper le déplacement des fils d'un concept avant de détruire leur père. De plus, la prise en compte du changement dans les annotations est complètement découplée de l'utilisation de Protégé. Dans TextViz, il est simple de relancer l'annotation automatique après chaque évolution de l'ontologie, mais cela remet en cause les annotations retouchées manuellement. Notre proposition consiste à réduire l'impact d'un changement et de ne retoucher que localement le fichier des annotations, là où les éléments modifiés sont utilisés.

5. Conclusion

Cet article a présenté l'implémentation au sein du logiciel EvOnto de principes d'évolution de RTO dans le cas où la RTO est utilisée pour annoter sémantiquement des documents. Notre recherche apporte plusieurs avancées aux travaux réalisés jusque là sur l'évolution d'ontologies. Tout d'abord, nous nous intéressons à l'évolution d'ontologies à composantes lexicales, ou RTO, dans lesquelles les termes ont un statut à part entière à côté des composants habituels d'une ontologie. Ensuite, 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. Nous poursuivons le développement d'EvOnto en intégrant d'autres opérations de changement, les stratégies associées et les interfaces d'ajustement des conséquences. Enfin, nous sommes en train de définir les stratégies de répercussion des changements sur les annotations qui perturbent le moins celles-ci. L'évaluation d'EvOnto soulève un ensemble de questions. Nous prévoyons un premier retour d'expérience à partir de l'utilisation du logiciel avec les données des différentes applications du projet Dynamo. Un protocole plus ciblé doit être défini pour évaluer l'apport quantitatif et qualitatif de son utilisation, et sa comparaison avec d'autres logiciels d'évolution d'ontologie.

6. Références

Cimiano, P., Völker, J. (2005). Text2Onto - a framework for ontology learning and data-driven change Discovery. In A. Montoyo, R. Munoz, & E. Metais (Eds.), *LNCS: Vol. 3513. Natural Language Processing and Information Systems*. Berlin: Springer, 227-238.

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

Flouris, G. (2006). On belief change and ontology evolution. Ph.D. Thesis, University of Crete, Department of Computer Science, Heraklion, Greece.

Flouris G., Plexousakis D., Antoniou G. (2006). *A Classification of Ontology Change*, CEUR-WS201, CEUR Workshop Proceedings, CEUR-WS.org.

Klein, M. (2004). Change management for distributed ontologies. Ph.D. Thesis, Dutch Graduate School for Information and Knowledge Systems. Germany.

Luong, P. H. (2007). Gestion de l'évolution d'un web sémantique d'entreprise. Thèse de doctorat, école doctorale STIC, école des mines de Paris.

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

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

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

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

Plessers, P, De Troyer O. (2005). 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.

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

Sellami Z., Gleizes M.P., Aussenac-Gilles N., Rougemaille S. (2009). Dynamic ontology co-construction based on adaptive multi-agent technology. *Intern. Conf. on Knowledge Engineering and Ontology Development KEOD 2009, Madeira (Portugal)*, 56-63.

Stojanovic L. (2004). Methods and Tools for Ontology Evolution, Ph.D. Thesis, Karlsruhe University. Germany.

Tissaoui A. (2009). Typologie de changements et leurs effets sur l'évolution de Ressources Termino-Ontologiques (Poster) *Journées Francophones d'Ingénierie des Connaissances IC2009*, Hammamet. http://ic2009.inria.fr/docs/posters/Tissaoui_Poster_IC2009.pdf

Zablith F., Sabou M., d'Aquin M. and Motta E. (2009). 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, 908-912.