Table des matières

Sommaire

Liste des figures

Liste des tableaux

CHAPITRE I

Introduction Générale

Sommaire

« Face au monde qui bouge, il vaut mieux penser le changement que changer le pansement ! »
Francis Blanche (1921-1974)

1.1 Contexte de la thèse

1.1.1 Contexte scientifique

Le web sémantique est une évolution coordonnée du web (lancée entre autre par le W3C). L'objectif principal du Web sémantique est de fournir aux utilisateurs finaux des services plus intelligents basés sur l'utilisation par la machine de connaissances représentées en utilisant des ontologies et des bases de connaissances [Berners-Lee, *et al.*, 2001]. Les ontologies sont des tentatives pour modéliser plus précisément une partie du monde pour, entre autres, permettre des interactions entre les données disponibles dans différents formats. Depuis que Tim Berners-Lee a proposé cette idée en 1994, il y a eu de nombreux efforts impulsés par le Consortium World Wide Web (W3C). La plupart de ces tentatives visent à spécifier, développer et déployer des langages pour le partage de connaissances (RDF, OWL, ...).

Le nombre d'ontologies qui sont développées et utilisées pour différentes applications ne cesse d'augmenter. Mais un des problèmes majeurs avec les ontologies est leur changement ou leur évolution. Les ontologies peuvent changer pour des raisons diverses, par exemple lorsque les besoins des utilisateurs changent, nécessitant une conceptualisation différente. Mais aussi un défaut de conception peut avoir été remarqué dans la conceptualisation d'origine, en appliquant des corrections de modélisation, ou bien encore il peut s'avérer nécessaire d'élargir la représentation du domaine.

Les changements dans une ontologie peuvent causer des problèmes sérieux pour les données instanciées, les applications et les services qui sont dépendants de l'ontologie, ainsi que pour toutes les ontologies qui importent cette ontologie modifiée [Maedche *et al.*, 2003]. Par conséquent, cela nécessite l'invention de méthodes pour gérer les changements afin de réduire au minimum les effets négatifs sur l'ontologie elle-même et sur les applications déployées depuis le début de leur cycle de vie.

Ces dernières années, la gestion du changement au sein des ontologies est étudié de telle sorte que ces mises à jour puissent être enregistrées et utilisées pour fournir une meilleure maintenance et une meilleure accessibilité. La plupart des travaux à ce jour a porté sur les moyens offerts à la personne qui maintient l'ontologie, pour l'aider à gérer le changement au sein d'une ontologie avec différents aspects, tels que la caractérisation de changement [Maedche *et al.*, 2003], l'évolution d'ontologie [Noy *et al.*, 2004], [Luong, 2007], [Rogozan, 2008], le versioning d'ontologie [Huang *et al.*, 2005], la maintenance et la cohérence [Noy *et al.*, 2002] [Djedidi, 2009], [Klein *et al.*, 2002], [Haase *et al.*, 2005]. La caractérisation commune de la plupart de ces travaux est que leurs efforts sur la gestion des changements d'ontologie sont mis sur l'identification et la manipulation du changement de l'ontologie elle-même. Mais l'impact que le changement peut avoir aussi sur les artefacts dépendants de l'ontologie (annotations sémantiques basées sur l'ontologie, d'autres applications, …) peut être très important. Par conséquent, réexaminer la question de la gestion du changement dans une ontologie à partir du moment où le changement se produit est nécessaire en s'intéressant tout particulièrement à certaines conséquences de ces changements.

Aujourd'hui, l'évolution et la maintenance des ontologies sont mal maîtrisées et soulèvent des problèmes peu abordés. Beaucoup de difficultés sont identifiées : difficulté de trouver et de caractériser les connaissances à modifier pour que l'ontologie réponde au mieux à un besoin particulier, difficulté d'évaluer l'impact et la pertinence d'une modification sur la structure conceptuelle de l'ontologie et, finalement, difficulté à traiter les effets de la modification sur les artéfacts dépendants de cette ontologie.La maintenance régulière de l'ontologie et des artéfacts associés devient une nécessité afin de conserver leur validité et leur pertinence. Cette dynamique des connaissances touche alors, en cascade, toutes les utilisations de ces ontologies, en particulier les annotations sémantiques déjà produites à l'aide de celles-ci, qui doivent être mises à jour en conséquence.

1.1.2 Contexte applicatif

Toutes les recherches conduites au cours de notre thèse ont été réalisées dans le cadre du projet DYNAMO[1] (*DYNAMic Ontology for information retrieval*). L'objectif principal du projet 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.

Ce projet réunit d'une part, des acteurs publics du monde académique : 3 équipes de l'IRIT (Institut de Recherche en Informatique de Toulouse) à savoir IC3[2] (Ingénierie des Connaissances, de la Cognition et de la Coopération) , SIG-RI-EVI[3] (Systèmes d'Informations Généralisés : Exploration et Visualisation d'Information) et SMAC[4] (Systèmes Multi-Agents Coopératifs) ; l'équipe LaLIC (Langues, Logiques, Informatique, Cognition) de l'Université Paris Sorbonne aujourd'hui composante du laboratoire STIH[5], le laboratoire P&T[6] (Préhistoire et technologie) de l'Université Paris Ouest Nanterre La Défense et, d'autre part, des acteurs privés avec la section Recherche de la société ACTIA[7], spécialisé dans le diagnostic en électronique automobile et la société ARTAL Technologies[8], pour la gestion de projets informatiques.

Dans le projet Dynamo, nous appelons *ressource termino-ontologique* un modèle conceptuel comportant une composante conceptuelle, prenant la forme d'une ontologie, et une composante lexicale, ou terminologie, à savoir un ensemble de termes dénotant les concepts.

[1]Ce projet a bénéficié d'un financement ANR 07 TLOG 004 01 de l'Agence Nationale de la Recherche (2007-2012) http://www.irit.fr/DYNAMO
[2]http://www.irit.fr/-Equipe-MELODI Depuis septembre 2011, l'équipe IC3 est devenue MELODI.
[3]http://www.irit.fr/-Equipe-SIG
[4]http://www.irit.fr/-Equipe-SMAC
[5]http://www.stih.paris-sorbonne.fr/
[6]http://www.mae.u-paris10.fr/index.html
[7]http://www.actia.fr
[8]http://www.artal.fr

L'annotation sémantique d'un document textuel revient à associer des instances d'éléments de l'ontologie à des mots ou groupes de mots précisément localisés dans des textes. Pour une collection de documents donnée, le projet Dynamo fait l'hypothèse que l'on va concevoir une ressource termino-ontologique (ou RTO) adaptée pour réaliser l'annotation sémantique des documents de la collection. Un environnement informatique, la plate-forme TextViz, a été développé dans le cadre ce projet. TextViz permet de construire et gérer la RTO, de gérer la collection de documents, de générer des annotations sémantiques de la collection par la RTO, puis de faire des recherche d'informations au sein de la collection.

Ce projet s'intéresse à des ontologies et annotations définies dans un environnement dynamique, dans lequel le domaine et les connaissances associées, les collections de documents et les annotations sémantiques construites avec l'ontologie doivent être modifiés régulièrement et en cohérence pour s'adapter à l'évolution du domaine sur lequel elles portent et des collections documentaires. Les domaines considérés dans le projet correspondent aux besoins des trois partenaires Actia, Artal et P&T (cf. 3.2.3).

1.2 Problématique posée

La problématique originale de la thèse concerne la spécification de modules gérant, de manière interactive, l'évolution conjointe d'une ressource termino-ontologique et des annotations sémantiques d'un corpus. D'une part, l'évolution de ce corpus a des répercussions sur la ressource termino-ontologique et sur les annotations sémantiques produites à l'aide de cette ressource termino-ontologique. D'autre part, l'évaluation de la qualité des annotations nécessite éventuellement l'évolution de la ressource termino-ontologique.

Le fait de prendre en compte les termes associés aux concepts et les annotations sémantiques utilisant l'ontologie renouvelle les questions classiques soulevées par la gestion de l'évolution :

1. Comment mieux représenter les opérations de changement de la ressource termino-ontologique afin d'aider l'ontologue à choisir une opération de changement ?

2. Comment analyser les effets d'un changement (au sein de la ressource termino-ontologique mais aussi sur ses utilisations) et les gérer au mieux (par exemple en les minimisant) ?

3. Comment détecter les incohérences générées par les changements dans la ressource termino-ontologique ?

4. Comment résoudre les problèmes de cohérence détectés à cause de la modification de ressource termino-ontologique pour assurer la consistance globale du système (c'est-à-dire de la ressource termino-ontologique et de ses utilisations) ?

5. Si plusieurs solutions sont possibles, quelles informations doivent être présentées à l'ontologue pour l'aider à choisir une solution ?

1.3 Contributions apportées

Pour répondre à toutes ces questions, nous avons défini des principes d'évolution et nous avons dressé une typologie de changements avec, pour chacun, la nature de ses conséquences, sur la ressource termino-ontologique et sur les annotations sémantiques. Nous avons également implémenté ces principes dans différents modules, qui forment le logiciel EvOnto, extension de la plate-forme TextViz, et nous avons réalisé des évaluations des propositions et de ces modules du logiciel.

Notre contribution comporte donc trois facettes :

1. *Une méthodologie* qui propose une vue unifiée sur l'évolution, son déroulement, ses exigences et ses besoins, les principes mis en jeu et les règles à respecter concernant une RTO et des annotations sémantiques dans un environnement dynamique. L'évolution est gérée dans deux directions : le fait d'annoter un texte peut conduire à faire évoluer la RTO, et inversement, les évolutions de la RTO

conduisent à revoir l'annotation de certains textes. La détection des besoins de changement à partir des annotations s'appuie sur des critères de qualité des annotations. Nous avons défini et implémenté un module d'expression et d'application de ces critères.

2. *Une typologie des changements* : elle concerne des ressources termino-ontologiques représentées à l'aide du langage standard OWL selon le méta-modèle défini dans le projet Dynamo. Nous avons recensé les changements applicables à une RTO, et leur avons donné une signification bien précise dans le cadre du méta-modèle. Les changements peuvent être élémentaires (ajout ou effacement d'un concept, d'un terme ou d'une relation), mais aussi plus complexes (déplacement ou fusion de concepts). Cette typologie élargit des travaux antérieurs en gérant le lexique de la RTO et tous les changements portant sur les termes et les liens de dénotation entre termes et concepts.

3. *Des stratégies d'évolution adaptables :* Le lien entre RTO et annotations intervient à deux reprises : au moment de choisir quoi faire des éléments reliés à un objet modifié, et au moment de revoir les annotations une fois la RTO modifiée. Pour chaque type de changement, sont proposées 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. Une fois les conséquences choisies ou confirmées par l'utilisateur, elles sont rendues effectives dans la RTO, puis répercutées sur les annotations.

Pour mettre en œuvre ces propositions au sein du projet Dynamo, nous avons développé le logiciel « EvOnto » en complément du logiciel TextViz. Les deux sont intégrés dans l'éditeur

d'ontologies Protégé sous forme de plugins. Pour faire évoluer une RTO, EvOnto aide l'ontologue à adapter la RTO au corpus pour produire les annotations les plus précises possible, et à adapter les annotations aux évolutions de la ressource. Il permet de guider interactivement l'ontologue pour formuler une demande de changement, évaluer son impact (effets supplémentaires) sur la qualité de la RTO et aussi sur les annotations sémantiques, et décider ensuite de leur mise en œuvre.

1.4 Organisation du manuscrit

Le manuscrit de la thèse est structuré en 8 chapitres dont 6 chapitres sont organisés en trois parties, un chapitre pour l'introduction et le dernier concerne la conclusion et les perspectives.

La première partie contient un seul chapitre d'état de l'art (chapitre 2), et met l'accent, à partir d'une vue globale des recherches existantes, sur les différentes méthodologies de gestion de l'évolution des ontologies et des annotations sémantiques. Ensuite, nous exposons et comparons des outils d'évolution d'ontologies ainsi que des outils d'évolution des annotations sémantiques.

La partie 2 (chapitres 3, 4, 5 et 6) détaille nos propositions théoriques et pratiques et se divise en quatre chapitres. Le chapitre 3 décrit le contexte de notre travail, à savoir le projet DYNAMO. Le quatrième chapitre expose notre méthodologie et propose une vue unifiée sur l'évolution, son déroulement, ses exigences et ses besoins, les principes mis en jeu et les règles à respecter par une RTO et les annotations sémantiques dans un environnement dynamique. Le chapitre 5 présente le processus d'évolution de RTO que nous proposons. Il décrit la conceptualisation et la formalisation de la typologie de changement que nous avons définie ainsi que les stratégies d'évolution de la RTO et ses conséquences. Le chapitre 6 porte sur la propagation du changement vers les annotations sémantiques. Nous y proposons

le processus d'évolution des annotations sémantiques et aussi le module d'évaluation de la qualité des annotations.

Dans la partie 3, on se focalise sur le prototype « EvOnto » et la mise en œuvre et l'évaluation de ce système. Cette partie critique également les résultats issus de l'analyse de cette évaluation.

Finalement, le chapitre de conclusion établit un bilan général des travaux de cette thèse ainsi que quelques pistes intéressantes en guise de perspectives de recherche pour une suite des travaux.

Première partie

ÉTAT DE L'ART – ÉVOLUTION D'ONTOLOGIES ET D'ANNOTATIONS SÉMANTIQUES

Tout au long de ce chapitre, nous mettons l'accent sur la problématique d'évolution d'ontologie et offrons une vision globale des différentes approches de l'état de l'art. Nous nous focalisons sur la gestion des effets des changements ontologiques sur les annotations sémantiques et les solutions proposées pour mettre à jour les annotations affectées par rapport à l'ontologie modifiée. Nous terminons par un panorama des logiciels de gestion d'ontologies assurant à la fois leur évolution et l'évolution d'annotations sémantiques.

CHAPITRE II

Évolutions

Sommaire

2.1 Introduction

Les ontologies sont des structures souvent grandes et complexes. Lorsqu'elles sont utilisées dans un contexte dynamique, c'est-à-dire chaque fois qu'il y a un changement dans le domaine ou que l'application utilisant l'ontologie doit répondre à de nouveaux besoins, l'ontologie doit pouvoir faire l'objet d'une évolution qui peut regrouper plusieurs changements. Ces changements effectués dans l'ontologie peuvent nécessiter d'autres changements en cascade dans l'ontologie et affecter ce qui est appelé dans la littérature des « artefacts dépendants », c'est-à-dire des ontologies ou des modèles de connaissances adaptés de cette ontologie ou encore des applications faisant appel à l'ontologie. Dans cette thèse, nous nous intéresserons particulièrement aux applications qui produisent ou utilisent des annotations sémantiques basées sur l'ontologie.

Tout au long de ce chapitre, nous mettons l'accent sur la problématique d'évolution d'ontologie et offrons une vision globale des différentes approches de l'état de l'art. Tout d'abord, nous discutons les problèmes posés par l'évolution d'ontologie tels que l'identification des causes de changement d'une ontologie, la représentation de ces changements, leurs effets et leur propagation (section 2.2.3.3, 2.2.3.4 et 2.2.3.5). Nous présentons ensuite un état de l'art sur les approches méthodologiques d'évolution des ontologies (section 2.2.4). Puis, nous nous focalisons sur la gestion des effets des changements ontologiques sur les annotations sémantiques. Nous mentionnons les solutions proposées pour mettre à jour les annotations inconsistantes par rapport à l'ontologie modifiée (section 2.3). Cette analyse présente un grand intérêt pour notre thèse dans la mesure où nous nous intéressons aux ontologies pour l'annotation de collections documentaires régulièrement mises à jour. Enfin, nous concluons par un panorama des logiciels de gestion d'ontologies assurant à la fois leur évolution et l'évolution d'annotations sémantiques (section 2.4 et 2.5).

2.2 Évolution d'ontologie : gestion du changement

2.2.1 Définition

Une des définitions les plus connues et sans doute la plus utilisée de la notion d'ontologie est celle de [Gruber, 1993] : "une ontologie est une *spécification formelle explicite* d'une *conceptualisation partagée d'un domaine*". Cette définition insiste sur l'aspect consensuel des connaissances de l'ontologie, sans indiquer si l'ontologie est figée ou non. Or, ni les données, ni l'ontologie elle-même ne sont stables dans le temps. Pour tenir compte de cette instabilité, Fensel considère une ontologie comme un réseau dynamique de significations, dans lequel un consensus est atteint à l'intérieur d'un processus continu de changements d'informations et de significations [Fensel, 2001]. Dans un environnement dynamique, les ontologies doivent être modifiées régulièrement tout en maintenant leur cohérence, pour s'adapter à l'évolution du domaine sur lequel elles portent. L'évolution d'ontologie fait l'objet de plusieurs définitions dans la littérature.

***Définition 1* :** [Klein, 2004] définit l'évolution d'une ontologie comme un processus offrant la capacité de gérer les changements apportés à une ontologie en créant et en maintenant différentes versions de cette ontologie. Cette capacité consiste à identifier et à différencier les versions, à les modifier, à spécifier des relations qui rendent explicites les différences entre ces versions.

"*We will combine ontology evolution and versioning into a single concept defined as the ability to manage ontology changes and their effects by creating and maintaining different variants of the ontology. This ability consists of methods to distinguish and recognize versions, specifications of relationships between versions, update and change procedures for*

ontologies, and access mechanisms that combine different versions of an ontology and the corresponding data.[9] "

***Définition 2* :** [Stojanovic, 2004] définit l'évolution d'une ontologie comme une adaptation opportune de cette ontologie aux changements se produisant dans son environnement d'utilisation et la propagation consistante de ces changements aux artefacts dépendant de l'ontologie *:*

"Ontology Evolution is the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artefacts.".

2.2.2 Origines des changements d'une ontologie

Formellement, si nous retenons qu'une ontologie est une spécification explicite d'une conceptualisation d'un domaine, alors, une modification de l'un des trois éléments de cette définition peut causer des changements dans une ontologie : (1) *des changements dans le domaine*, (2) *des changements de conceptualisation* ou (3) *des changements de spécification explicite* [Klein, 2004].

Pour mieux comprendre les changements dans le domaine décrit par une ontologie, tout d'abord, prenons l'exemple simple suivant où deux départements universitaires avec des structures administratives différentes fusionnent. L'ontologie décrivant le domaine correspondant à une des deux structures administratives doit être mise à jour pour refléter ce changement. Dans ce cas, l'ontologie, comme n'importe quelle structure contenant des informations d'un domaine particulier, a besoin d'être changée tout simplement parce que le domaine d'intérêt a changé [Stojanovic *et al.*, 2003].

[9] Nous associons évolution et gestion des versions d'une ontologie au sein d'un concept unique défini comme la capacité à gérer les changements et leurs effets en créant et maintenant différentes variantes de l'ontologie. Cette capacité s'appuie sur des méthodes pour distinguer et reconnaître différentes versions, des spécifications des relations entre versions, des procédures de mise à jour et de changement des ontologies. Elle s'appuie enfin sur des mécanismes d'accès qui combinent différentes versions d'une ontologie et des données (instances) correspondantes.

Mais, même si nous supposons un domaine statique, ce qui est une hypothèse plutôt peu réaliste pour la plupart des applications, il peut être nécessaire de modifier l'ontologie si la perspective sous laquelle le domaine est considéré change [Noy et Klein, 2004]. Or, un changement dans la perspective d'utilisation induit un changement dans la conceptualisation de l'ontologie. Différentes tâches peuvent impliquer différents points de vue sur le domaine et, par conséquent, la conceptualisation résultante sera différente. Il se peut aussi que l'utilisation de l'ontologie permette de découvrir un défaut de conception dans la conceptualisation initiale du domaine [Plessers et De Troyer 2005], ou qu'elle conduise à vouloir incorporer des notions supplémentaires, correspondant aux changements des besoins des utilisateurs [Haase et Stojanovic 2005]. L'ontologie devra alors refléter cette nouvelle conceptualisation.

Enfin, les changements de spécification explicite se produisent lorsqu'une ontologie est traduite d'un langage de représentation des connaissances à un autre. La préservation de la sémantique de l'ontologie durant cette traduction n'est pas une tâche triviale parce que les langages de représentation des connaissances différent, non seulement au niveau syntaxique mais aussi au niveau sémantique [Corcho et Gomez-Pérez 2000]. Pour cela, des ajustements importants sont probablement nécessaires.

2.2.3 Une méthode de référence pour la gestion des changements

Dans cette thèse, nous nous intéressons principalement au problème d'évolution d'ontologie suite à des changements du domaine ou à une nouvelle conceptualisation (les deux premiers cas précédents). Plus précisément, nous nous focalisons sur les cas où l'ontologie est liée à un corpus de documents textuels qui évolue, et que cette ontologie sert à annoter. Nous souhaitons prendre en compte tous les effets potentiels des changements effectués sur les entités ontologiques et vérifier la cohérence de l'ontologie après l'application

de ces changements. Un autre point très important est de respecter la validité des applications utilisant l'ontologie après sa mise à jour et ce à travers la propagation des changements vers les données de ces applications. Dans notre cas, ces données sont les annotations sémantiques des documents basées sur l'ontologie.

Nous avons également l'intention de garder une trace de chaque changement dans l'ontologie et des scénarios considérés pour la faire évoluer. Dans cette perspective, plusieurs approches ont été proposées. Dans la suite, nous présentons tout d'abord des méthodologies existantes qui présentent un processus - plus ou moins global - d'évolution d'ontologie [Stojanovic *et al.*, 2002a] [Stojanovic *et al.*, 2002b] [Klein, 2004]. Ensuite, nous nous concentrons sur quelques approches qui traitent de problématiques particulières de gestion de changement telles que le processus d'évolution, la représentation d'un changement, la gestion de ses effets et de sa propagation vers les applications dépendantes.

2.2.3.1 Modèle d'une ontologie

Définition 3 **:** Le modèle d'une ontologie est le méta-modèle qui définit le langage de représentation des connaissances dans cette ontologie [Guizzardi, 2008]. Par exemple OWL définit un modèle.

Définition 4 **:** Un élément d'un modèle d'ontologie est une des primitives (structures) de représentation des connaissances proposées par ce modèle. En OWL, on a par exemple les notions de classe, de propriété, d'instance etc.

Etant données ces 2 premières définitions, [Stojanovic, 2004] définit ainsi une ontologie consistante par rapport à son modèle :

Définition 5 : Une ontologie est consistante par rapport à son modèle si et seulement si elle préserve les contraintes (syntaxiques et sémantiques) définies pour le modèle d'ontologie de base, c'est-à-dire par le langage utilisé pour sa représentation :

"A single ontology OI is defined to be consistent with the respect to its model if and only if it preserves the constraints defined for underlying ontology model.".

On parlera parfois aussi de cohérence pour dire que l'ontologie répond à certains critères qui peuvent varier suivant les situations ou les usages.

Une autre manière de considérer la notion de modèle et de cohérence d'une ontologie est de faire référence à la théorie des modèles en logique, au sens de Tarski. Dans ce cas, on s'intéresse à une interprétation des formules logiques, c'est-à-dire de l'ensemble des prédicats et axiomes ou assertions définis dans l'ontologie. Valider la cohérence de l'ontologie revient alors à être capable de prouver la valeur de vérité des formules de l'ontologie. Nous avons donc une deuxième définition de consistance d'une ontologie

Définition 6 **:** Une ontologie est consistante (en logique de description) si et seulement si elle a au moins un modèle, c'est-à-dire une interprétation qui satisfait toutes ses formules.

Par la suite, nous utiliserons « modèle » et « consistance d'un modèle » au sens de la définition 5.

2.2.3.2 Évolution d'ontologie selon Stojanovic

Pour assurer une bonne évolution, il faut qu'il y ait un cadre méthodologique qui l'organise. L. Stojanovic fut la première qui ait abordé l'évolution de l'ontologie, et plus particulièrement qui l'ait associée à une méthodologie. Son travail est considéré comme une référence sur laquelle se base la majorité des travaux qui l'ont suivi.

En adaptant aux ontologies les définitions relatives à l'évolution des schémas de bases de données (un état de l'art assez complet des recherches existantes sur l'évolution et la gestion

des versions de schéma est présenté dans [Roddick, 1995]), L. Stojanovic distingue la gestion de l'ontologie, sa modification, son évolution de la gestion de ses versions comme des variantes mettant en jeu différents niveaux de complexité [Stojanovic, 2004] :

- *Modification de l'ontologie :* consiste à apporter des changements à l'ontologie considérée, sans tenir compte de son état de cohérence.
- *Évolution de l'ontologie :* facilite la modification d'une ontologie en préservant son état de cohérence.
- *Gestion des versions de l'ontologie :* permet la manipulation des modifications de l'ontologie en créant et en gérant ses différentes versions.
- *Gestion de l'ontologie :* Il s'agit d'un ensemble de méthodes et techniques qui sont élaborées pour utiliser efficacement de multiples variantes des ontologies, éventuellement à partir de sources différentes en vue de résoudre des tâches différentes. Par conséquent, un système de gestion d'ontologie est considéré comme un cadre pour créer, modifier, gérer des versions, faire des requêtes, sauvegarder des ontologies, etc.

Dans notre travail, nous nous sommes concentrés sur le deuxième et le quatrième niveau (*évolution de l'ontologie et gestion de l'ontologie*). Il est alors important d'explorer les conséquences des changements apportés à des ontologies et la stratégie à employer pour garantir leur consistance. Il est aussi nécessaire de propager tout changement appliqué localement à l'ontologie, aux applications basées sur celles-ci pour préserver leur pertinence et leur performance opérationnelle, et enfin de valider les changements globalement.

Sur le plan méthodologique et pour assister un système d'évolution d'ontologie, Stojanovic (2004) définit diverses notions nécessaires faire évoluer l'ontologie, assurer sa consistance et ce après avoir identifié les différents types de changements.

L'article [Stojanovic *et al.*, 2002] énonce tout d'abord les conditions à satisfaire par un système d'évolution d'ontologie puis propose un processus d'évolution qui satisfait ces conditions. Ce processus, constitué de six phases, analyse systématiquement les causes et les conséquences des changements, et assure la consistance de l'ontologie ainsi que celle de ses parties dépendantes après la réalisation de ces changements.

Les six phases du processus d'évolution sont les suivantes (figure II-1) : (1) saisie du changement, (2) représentation du changement, (3) sémantique du changement, (4) propagation du changement, (5) implémentation du changement et (6) validation du changement.

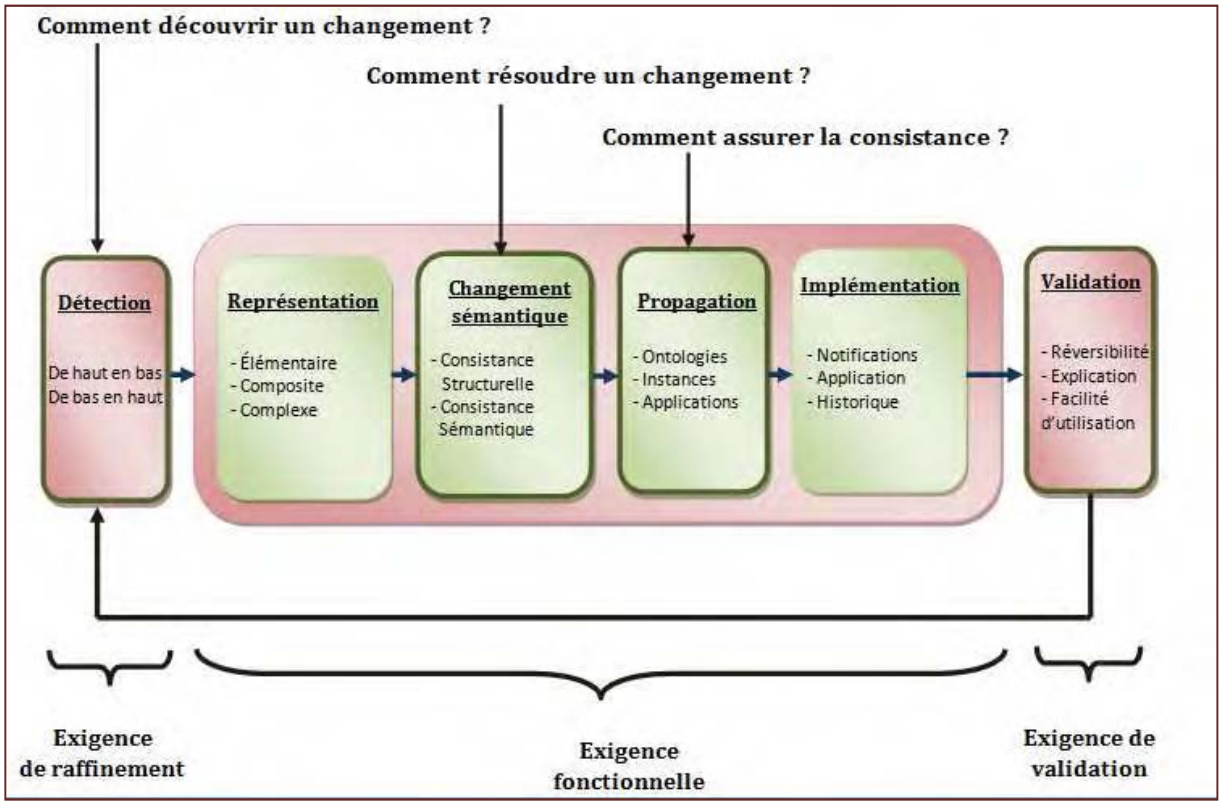


Figure II - 1 : Processus d'évolution de l'ontologie en six phases [Stojanovic, 2004]

A- Saisie (ou recueil) du changement

Le processus d'évolution commence par la détection des besoins explicites de changements dans l'ontologie, selon une approche descendante (top-down) ou à partir du résultat de

techniques de découverte de changements selon une approche ascendante (bottom-up). Trois modalités sont possibles pour découvrir des changements :

- Au niveau structurel (structure-driven), les changements peuvent être déduits de problèmes ou d'erreurs identifiés grâce à la structure de l'ontologie elle-même, en exploitant par exemple un ensemble d'heuristiques testant des propriétés syntaxiques ou sémantiques des éléments de l'ontologie ;
- Au niveau de données (data-driven), les changements sont générés par des modifications sur l'ensemble de données associées à l'ontologie, telles que des documents textuels, des instances de concepts ou une base de données, représentant la connaissance modélisée par l'ontologie.
- Au niveau des usages (usage-driven), les changements sont découverts par le suivi et l'analyse des traces des utilisateurs de l'ontologie, et qui font ressortir des erreurs, des manques, des connaissances inutiles, etc.

B- Représentation du changement

Avant de mettre en œuvre les changements, ils doivent être identifiés et représentés dans un format adéquat. Les changements peuvent être représentés à différents niveaux de granularité, en distinguant par exemple des changements élémentaires, des changements composites et des changements complexes tels que ceux définis par Stojanovic pour le modèle d'ontologie KAON [Stojanovic, 2004]

Un *changement élémentaire* est un changement primitif qui modifie, ajoute ou supprime, une seule entité de l'ontologie correspondant à une structure particulière du modèle du langage utilisé pour cette ontologie. Une liste de changements élémentaires adaptés au modèle d'ontologie de KAON est présentée dans le tableau II-1.

Tableau II - 1 : Taxonomie des changements élémentaires, selon [Stojanovic, 2004]

Élément d'une ontologie KAON	Add	Remove
Concept	AddConcept	RemoveConcept
Concept Hierarchy	AddSubConcept	RemoveSubConcept
Property	AddProperty	RemoveProperty
Property Hierarchy	AddSubProperty	RemoveSubProperty
Property Domain	AddPropertyDomain	RemovePropertyDomain
Property Range	AddPropertyRange	RemovePropertyRange
Property Symmetric	AddPropertySymmetric	RemovePropertySymmet.
Property Transitive	AddPropertyTransitive	RemovePropertyTransit.
Property Inverse	AddPropertyInverse	RemovePropertyInverse
Max Cardinality	AddMaxCardinality	RemoveMaxCardinality
Min Cardinality	AddMinCardinality	RemoveMinCardinality

Un *changement composite* met en œuvre une suite de changements élémentaires appliqués ensemble. Une liste de changements composites adaptés au modèle d'ontologie de KAON est présentée dans le tableau II-2.

Un *changement complexe* est un changement qui peut être décomposé en une combinaison d'au moins deux changements élémentaires ou composites. A titre d'exemple, une liste de 12 changements complexes est présentée dans [Stojanovic *et al.*, 2002c].

Tableau II - 2 : Taxonomie des changements composites pour l'entité concept, selon [Stojanovic, 2004]

Changement composite	Syntaxe et sémantique
Déplacer le concept vers le haut	**DéplacerConceptHaut (c)** Attacher le concept c à tous les parents de ses parents.
Déplacer le concept vers le bas	**DéplacerConceptBas (c)** Attacher le concept c à tous les enfants de ses parents sauf lui-même.
Grouper concepts	**GrouperConcepts (c1,c2, newC)** Créer un superconcept commun newC pour le concept c1 et c2 et transférer les propriétés communes.
Diviser le concept	**Diviser Concepts (c,newC1, newC2)** Diviser un concept c en deux concepts newC1 et new C2 et distribuer les propriétés et les instances entre eux.
Fusionner concepts	**FusionnerConcepts (c1,c2, newC)** Remplacer les concepts c1 et c2 par un concept newC et agréger toutes les propriétés et les instances.

Copier concept	**CopierConcepts (c, newC)** Dupliquer un concept c avec toutes ses propriétés et les instances directes par la création d'un nouveau concept newC et l'attacher à tous les parents de c.
Généraliser concept	**AjouterGeneralisationConcept (c, newC)** Ajouter un nouveau concept newC entre un concept donné c et tous ses parents.
Héritage de concept	**AjouterHéritageConcept (c1, c2, newC)** Ajouter un nouveau concept newC et attacher le à son parent le concept c1 et à son enfant le concept c2.
Spécialisation de concept	**AjouterSpécialisationConcept (c, newC)** Ajouter un nouveau concept newC entre un concept donné c et tous ses enfants.

Stojanovic a défini une autre caractérisation des changements selon leurs effets [Stojanovic, 2004] : les *changements additifs* ajoutent de nouvelles entités à l'ontologie sans modifier les entités existantes, alors que les *changements soustractifs* suppriment certaines entités ou parties d'entités.

C- *Sémantique du changement*

L'ontologie doit évoluer d'un état consistant vers un autre état consistant. Cette phase permet d'identifier et de réaliser des changements induits systématiquement par le changement demandé explicitement, et ce en assurant la consistance de toute l'ontologie (définition 3). De manière plus simple, la phase « sémantique du changement » consiste à compléter le changement de base (demandé par l'ontologue) par un ensemble de changements supplémentaires, définis en fonction du type d'entité concernée et de l'opération effectuée. Ces changements supplémentaires rendent mieux explicite la sémantique du changement et de ses conséquences au sein de l'ontologie. Ils garantissent le passage de l'ontologie d'un état consistant à un autre état consistant.

Toujours dans le cas d'une ontologie décrire dans KAON, le modèle de consistance de l'ontologie est défini comme l'union de 16 invariants (dont deux exemples sont montrés dans l'exemple 1), de 2 contraintes dites « faibles » (exemple 2) et de 4 contraintes définies par

l'utilisateur (exemple 3) [Stojanovic, 2004]. Les invariants sont des règles de consistance que doit respecter toute ontologie dans KAON. Chaque changement d'une ontologie doit maintenir l'exactitude des invariants mais une ontologie ne doit pas nécessairement satisfaire toutes les contraintes souples et les contraintes définies par l'utilisateur.

Exemple 1

IC_1 : identité unique invariante : toutes les entités (concepts, propriétés et instances) ont chacune une identité unique.

$$C \cap P = \phi \ \wedge \ C \cap I = \phi \ \wedge \ P \cap I = \phi$$

Cette contrainte de cohérence impose une séparation stricte des concepts, propriétés et instances. Exemple : la disjonction des concepts, des propriétés et des instances est exigée. Cela signifie par exemple, qu'un concept ne peut pas être en même temps une instance.

IC*2* : hiérarchie de concepts invariante : la hiérarchie de concept est un graphe acyclique direct :

$$\neg \exists c \in C \ (c, c) \in Hc^{*}$$

Exemple 2

- SC_1:*Min-cardinality Soft Constraint*:

$$\forall c \in C \ \forall p \in P \ mincard(c,p) \text{ is defined} \Rightarrow$$

$$\forall i \in I \ |instprop(p,i)| \geq mincard(c,p)$$

- SC_2: *Max-cardinality Soft Constraint*:

$$\forall c \in C \ \forall p \in P \ maxcard(c,p) \text{ is defined} \Rightarrow$$

$$\forall i \in I \ |instprop(p,i)| \leq maxcard(c,p)$$

Exemple 3

UC_1 : le domaine et le co-domaine de la propriété sont des contraintes définies par l'utilisateur : une propriété avec un concept domaine/co-domaine est considérée comme consistante si :

$$\forall p \in P \; \exists c \in C \; c \in domain(p)$$

$$\forall p \in P \; \exists c \in C \cup L \; c \in range(p)$$

En s'inspirant des travaux d'évolution de schéma de bases de données, Stojanovic définit deux approches pour assurer la consistance de l'ontologie : l'approche procédurale et l'approche déclarative [Stojanovic, 2004].

- ***Approche procédurale*** : cette approche est basée sur les contraintes du modèle de consistance et sur un ensemble prédéfini de règles qui doivent être suivies pour maintenir les contraintes satisfaites après chaque changement.

Un changement peut être traité de plusieurs manières, ce qui signifie que différents ensembles de changements supplémentaires peuvent être générés à partir d'un même changement initial. [Stojanovic, 2004] présente un ensemble de notions pour aider un ontologue à modifier une ontologie selon ses préférences :

- *Un point de résolution* représente une action qui pourrait se présenter pendant le traitement d'un changement,
- *Une stratégie d'évolution élémentaire* est une manière de résoudre un point de résolution,
- *La méthode de résolution* est un ensemble de stratégies d'évolution élémentaires possibles pour traiter un point de résolution particulier.

- *Une stratégie d'évolution* est un ensemble de couples (point de résolution, méthode de résolution définie pour ce point de résolution).

Pour mettre en œuvre un changement particulier, le processus d'évolution doit déterminer des réponses à plusieurs points de résolution. Chaque réponse possible correspondant à chaque point de résolution est une stratégie d'évolution élémentaire. Les points de résolution et leurs stratégies d'évolution élémentaires couvrent toutes les solutions possibles qu'un ontologue peut adopter pour appliquer des changements sur l'ontologie.

Approche déclarative : cette approche est basée sur un ensemble d'axiomes qui formalisent la dynamique de l'évolution. L'approche déclarative permet à un ontologue de modifier une ontologie selon les besoins, puis de calculer tous les changements supplémentaires. Cette approche est organisée en deux phases :

Formalisation de la demande de changement : l'ontologue définit sa demande de changement de façon déclarative sous la forme d'une série de changements avec l'un des deux statuts suivants : les *changements à exécuter* (ex. suppression du concept A) et les *changements à ne pas exécuter* (les sous-concepts de A peuvent être aussi bien préservés que supprimés).

Résolution du changement : seul le premier ensemble de changements (à exécuter) est appliqué afin de détecter les incohérences qu'ils causent dans l'ontologie.

D- Propagation du changement

La phase de propagation du ou des changements définis à l'étape précédente consiste, après avoir effectué une mise à jour de l'ontologie, à assurer la consistance des artefacts

dépendant de l'ontologie : ses instances, les ontologies qui l'importent ainsi que les programmes d'application qui l'utilisent.

Pour résoudre les problèmes qui peuvent survenir au sein de ces différents artefacts après une modification de l'ontologie, Stojanovic envisage les solutions suivantes :

- pour les ontologies qui l'importent, une procédure récursive est mise en œuvre : il s'agit d'appliquer des changements sur ces ontologies afin de préserver leur consistance conceptuelle et structurale.
- les instances de l'ontologie doivent être changées de manière à ce que l'ontologie et les instances restent conformes l'une à l'autre.

E- Implémentation du changement

L'implémentation du changement est l'application physique du changement à l'ontologie. Le rôle de cette phase est (i) d'informer l'ontologue de toutes les conséquences d'une demande de changement, (ii) d'appliquer tous les changements nécessaires et dérivés et (iii) de garder une trace des changements effectués.

F- Validation du changement

Dans cette phase, l'utilisateur évalue et teste les changements exécutés. Deux approches sont possibles pour vérifier la consistance de l'ontologie [Stojanovic, 2004] :

- la vérification a *posteriori*, où les changements sont d'abord exécutés, puis on vérifie si l'ontologie mise à jour satisfait les contraintes de consistance.
- la vérification a *priori*, qui définit un ensemble de pré-conditions relatives à chaque changement. Pour chaque changement, la consistance sera maintenue si l'ontologie est consistante avant une mise à jour et les pré-conditions sont satisfaites.

Selon [Rogozan, 2008], l'étape de propagation du changement, proposée par Stojanovic, est assez unidirectionnelle vu qu'elle vise seulement la modification des ontologies dépendantes afin de préserver leur consistance avec l'ontologie de base. Les auteurs ne proposent aucune étape d'analyse des effets des changements de l'ontologie sur les artefacts dépendants. Ces changements peuvent provoquer la suppression des liens de référencement sémantique entre des ressources annotées et l'ontologie, ou la détérioration des fonctionnalités de systèmes qui utilisent une ontologie modifiée. [Djedidi et al., 2007] considèrent qu'une étape d'évaluation de la qualité doit être intégrée dans le processus avant la mise en opération de l'ontologie modifiée.

Nous nous concentrons par la suite sur d'autres approches qui traitent des problématiques particulières de gestion des changements telles que le processus d'évolution, la représentation d'un changement, de ses effets et de sa propagation vers les applications dépendantes.

2.2.4 Autres approches d'évolution d'ontologie

2.2.4.1 Evolution et gestion de versions

Les ontologies sont utilisées par diverses communautés d'utilisateurs, applications ou d'autres applications qui en dépendent. Un changement dans une ontologie pourrait causer des problèmes dans ces applications. Dans ce cas, la gestion de versions peut permettre à certains utilisateurs de conserver ou de revenir à la version antérieure jusqu'à ce que l'ensemble de l'application soit adaptée et les problèmes résolus. La gestion de versions consiste à conserver des liens entre toutes les versions de l'ontologie, à donner les moyens d'accéder à chaque version selon les besoins et cela de différentes manières (via des données, services, applications ou une autre ontologie).

Certains auteurs affirment que l'on ne peut pas distinguer la gestion de l'évolution de la gestion des versions, qu'ils considèrent comme étant un seul et même processus [Klein, 2004], [Noy et Klein, 2004]. Les ontologues maintiennent alors non seulement les différentes versions d'une ontologie, mais aussi des informations associées par exemple, comment les versions diffèrent et si elles sont compatibles entre elles ou non. Dans [Klein et Noy, 2003] et [Klein, 2004], un Framework gérant l'évolution de l'ontologie permet aussi d'en gérer des versions. Les étapes principales du processus associé sont : (1) l'identification de deux versions d'une même ontologie, (2) la spécification des changements qui ont eu lieu entre ces deux versions, et (3) la génération de nouvelles connaissances sur le lien entre les deux versions. Les auteurs utilisent le terme gestion de versions pour décrire leur approche et proposent deux éléments fondamentaux :

- Un modèle d'analyse de la relation entre deux versions d'une ontologie. Ce modèle permet (i) d'expliciter ce qui a été changé dans la définition des entités ontologiques, (ii) 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), (iii) de décrire les changements par un ensemble de métadonnées spécifiant la date, l'auteur et le but de chaque changement, (iv) de décrire le contexte dans lequel les changements sont valides.

- une technique d'identification des versions d'une ontologie reposant sur deux principes de base : (1) les changements dans la définition d'une classe ou propriété produisent une nouvelle version, ayant un nouvel URI (*Uniform Ressource Identifier*) et (2) la forme de l'URI indique si la version est compatible avec la version antérieure.

Rogozan considère que cette méthodologie facilite la gestion des versions d'une ontologie après son évolution et non l'évolution d'ontologie proprement dite. En effet, la méthode fournit un modèle d'analyse de la relation entre les versions de l'ontologie, mais ne se préoccupent pas de la gestion de l'accès aux artefacts dépendants.

D'autres méthodes ont été proposées pour la gestion des versions d'une ontologie et pour des raisons de simplicité, nous n'allons pas les détailler. Nous renvoyons le lecteur intéressé aux références bibliographiques suivantes : [Zhang *et al*., 2003] [Noy et Musen, 2004] [Plessers et Troyer, 2005] et [Liang, 2006].

En revanche, nous allons décrire brièvement des travaux se basant sur les résultats de Stojanovic et réalisés par [Rogozan, 2008], [Luong, 2007] et [Djedidi, 2009].

2.2.4.2 Evolution d'une ontologie dans un contexte d'annotation [Rogozan et Paquette, 2005] et [Rogozan, 2008]

Dans sa thèse, Rogozan s'inspire des deux travaux de Klein [Klein, 2004] et de Stojanovic [Stojanovic, 2004] pour créer une méthode unifiée applicable au contexte particulier dans lequel l'ontologie est utilisée comme référentiel sémantique pour les éléments pédagogiques d'un système de téléapprentissage.

Une référence sémantique est précisément l'identifiant d'une connaissance formelle, exprimée sous la forme d'une classe dans une ontologie. Chaque référence sémantique est spécifiée formellement par ce qu'on appelle un UKI (*Uniform Knowledge Identifier*). Un UKI est défini comme un URI avec la particularité que les trois premiers composants doivent nécessairement identifier une ontologie (ou une version d'ontologie) tandis que le quatrième doit identifier une classe à l'intérieur de cette ontologie (figure II-2).

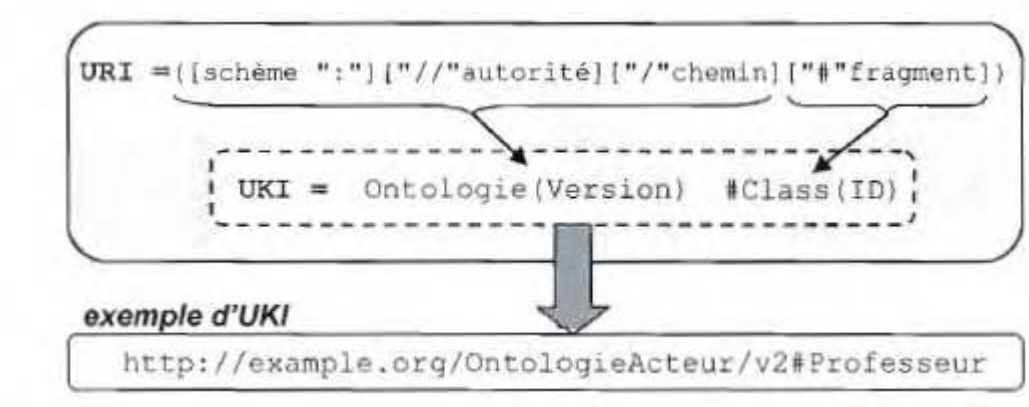


Figure II - 2 : Un UKI est un URI de l'ontologie/version d'ontologie (ex. http://example.org/OntologieActeur/v2) avec un identifiant de fragment qui est le nom d'une classe (ex. Professeur)

Rogozan a défini un processus composé de huit étapes (figure II.3). Les six premières étapes correspondent à celles de la méthode proposée par Stojanovic. La septième étape analyse les effets des changements, ce qui permet de recenser ceux susceptibles de causer des problèmes de dysfonctionnement de systèmes, des inconsistances des ontologies dépendantes ou encore la perte d'accès aux ressources référencées sémantiquement. Pour résoudre le problème de pertes d'accès aux ressources, Rogozan (2008) s'inspire des stratégies de résolution de Stojanovic pour créer des stratégies de modification du référencement sémantique affecté par l'évolution des ontologies. Ces stratégies sont fondées sur l'analyse des types de changement et de leur impact sur l'accès aux ressources référencées, mais aussi sur l'interprétation de celles-ci au moyen de la nouvelle version de l'ontologie.

La dernière étape du processus (étape huit) consiste en la mise en opération de la version V_{N+1}, avec la prise en compte de la propagation des changements vers les applications dépendantes. *Cette méthode ignore la vérification de la qualité de l'ontologie après évolution et avant la mise en opération de la nouvelle version.*

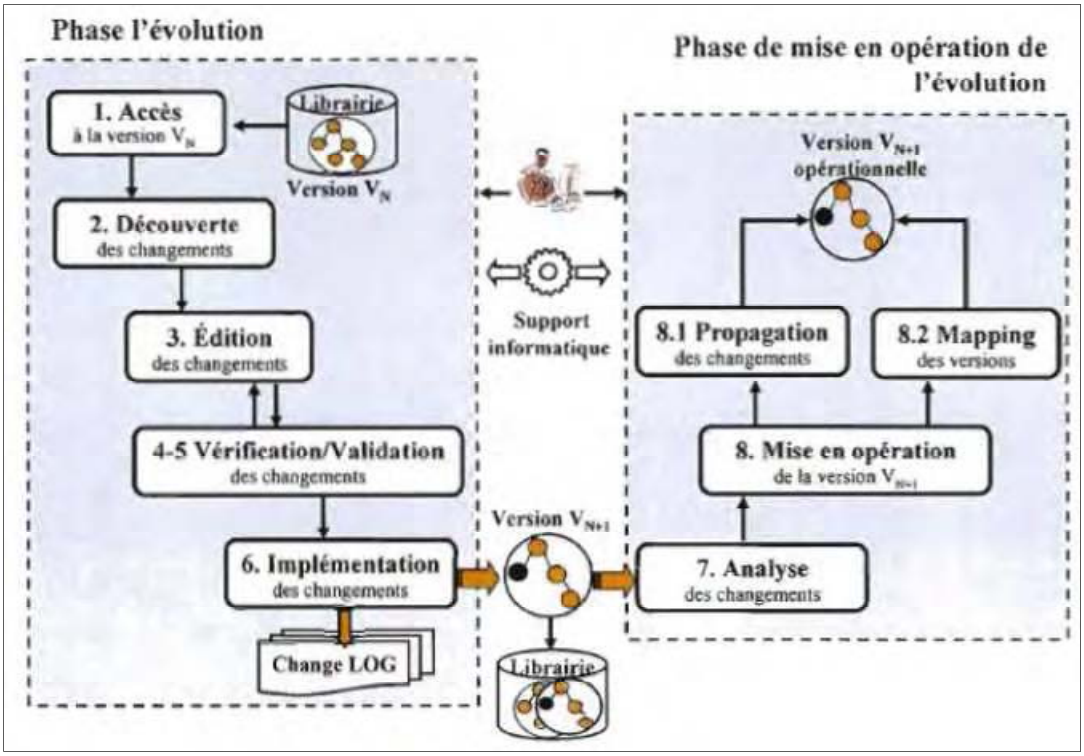


Figure II - 3 : Processus d'évolution d'ontologie, selon [Rogozan, 2008]

2.2.4.3 Evolution d'ontologie adaptée de la méthode KAON [Djedidi et Aufaure, 2008a], [Djedidi, 2009] et [Djedidi et Aufaure, 2009]

Une autre approche inspirée de la méthode KAON présente l'originalité de mettre l'accent sur des points non traités dans les approches précédentes et en particulier l'évaluation de l'ontologie qui évolue. L'idée est de classer les manières proposées pour gérer les conséquences d'un changement – quand plusieurs sont possibles – pour ne choisir que celle qui préserve – si ce n'est améliore – la qualité de l'ontologie. En guidant ainsi le choix des conséquences à appliquer, cette approche évite de solliciter l'intervention de l'ontologe directement, comme le font la plupart des approches existantes. L'évaluation de la qualité intervient en évaluant deux types de cohérence :

- la cohérence conceptuelle de l'ontologie : on vérifie la complexité de sa structure, la cohésion de ses composants, la richesse de sa conceptualisation et son niveau d'abstraction ;

- la cohérence de la modélisation de domaine : on vérifie sa complétude par rapport à des sources représentatives du domaine.

De plus, l'évaluation tient compte de contraintes des utilisateurs, liées au domaine de modélisation et/ou au contexte d'application de l'ontologie, à travers les pondérations des critères.

Cette méthode s'appuie sur une modélisation à l'aide de trois types de patrons et suit un processus en six étapes (figure II.4) : *identification du changement*, *représentation du changement*, *implémentation du changement*, *validation du changement*, *évaluation de la qualité de l'ontologie et annotation*. Les deux premières étapes, identiques à celles de KAON sont suivies d'une implémentation qui consiste à appliquer provisoirement le changement de base et ses changements dérivés tout en préservant la trace des modifications effectuées et les métadonnées associées. L'application finale des changements doit être précédée par la vérification de la consistance de l'ontologie et par la préservation de leur qualité. *Cependant, cette méthode ne traite pas la propagation des changements vers les applications dépendantes*.

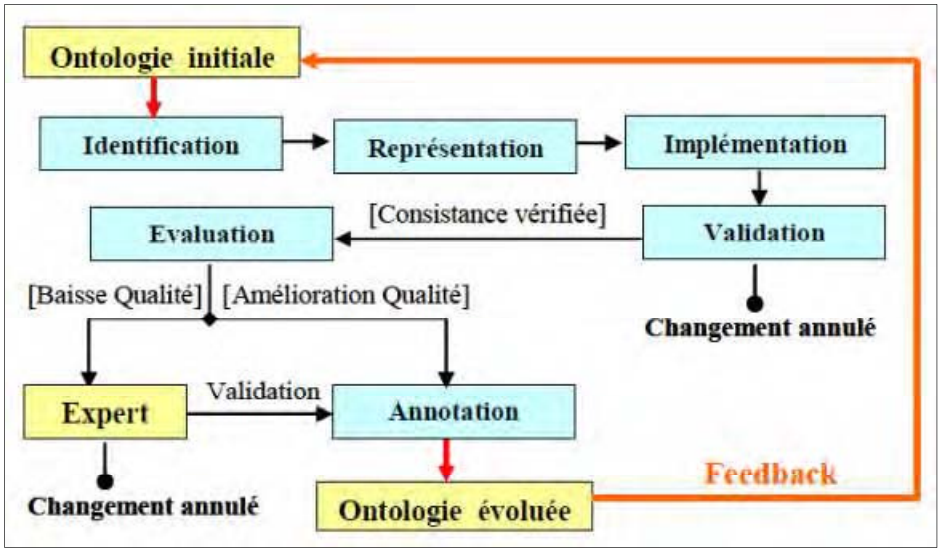


Figure II - 4 : Processus d'évolution d'ontologie, selon [Djedidi, 2009]

2.2.4.4 Tracer les évolutions [Luong, 2007], [Luong et Dieng-Kuntz, 2007]

Luong s'inspire aussi des travaux de Stojanovic. Il présente une approche de gestion d'évolution d'un web sémantique d'entreprise dont les composants principaux sont les ontologies et les annotations. Le processus d'évolution est organisé en trois étapes (figure II-5) : la représentation des changements, la « résolution » des changements et la propagation des changements aux annotations.

L'approche se focalise principalement sur la résolution des changements de l'ontologie qui provoquent des inconsistances lors de leur propagation vers les annotations sémantiques. Pour éviter ces inconsistances dans deux contextes, que l'on dispose de traces des changements dans l'ontologie ou pas, deux techniques sont mises en œuvre et implémentées dans le logiciel CoSWEM (*Corporate Semantic Web Evolution Management*). Nous les détaillerons dans la partie 2.3 qui traite de la gestion de l'impact des évolutions de l'ontologie sur les annotations. Nous présenterons CoSWEM en partie 2.5

Luong s'inspire des stratégies de résolution proposées dans [Stojanovic, 2004] pour « résoudre les changements » au sein de l'ontologie (section 2.2.5.3), auxquelles il ajoute d'autres stratégies pour résoudre l'incohérence d'annotations sémantiques par rapport à l'ontologie modifiée. Ces stratégies de résolution des changements des annotations font appel à 3 opérations : remplacer un des éléments d'annotation, le laisser inchangé ou le supprimer. CoSWEM peut être utilisé en mode automatique pour l'approche procédurale (il applique alors des stratégies choisies par défaut) ou semi-automatique dans le cas où l'on n'a pas de trace des évolutions de l'ontologie (le système fait des propositions de changement dans les annotations, que l'utilisateur valide ou corrige).

Dans le système ECCO, les changements sont censés être appliqués en amont, c'est-à-dire qu'on applique le changement de base en entrée et non les changements additionnels associés. C'est pourquoi seulement des propositions de résolution (stratégies de résolutions) ont été définies pour les implémenter dans ECCO [Luong, 2007]. La tâche de résolution des annotations sémantiques inconsistantes est résolue manuellement dans le cas où le système connaît la stratégie de résolution correspondante appliquée pour l'ontologie.

Figure II - 5 : Processus d'évolution d'ontologie selon [Luong, 2007]

2.2.4.5 Evolution d'ontologie à partir de données multimédia [Castano, 2006], [Castano *et al.*, 2007]

L'originalité de la méthode BOEMIE (*Bootstrapping Ontology Evolution with Multimedia Information Extraction*) est de s'intéresser à l'évolution d'ontologie à partir de données multimédia. Ainsi, l'extraction des connaissances exploite des contenus multimédia pour peupler et enrichir des ontologies. De plus, le déploiement de ces ontologies sert à améliorer la robustesse du système d'extraction d'information multimédia [Castano, 2006] [Castano *et al.*, 2007] [Petasis, 2007]. BOEMIE propose une approche d'évolution à l'aide de patrons. Les patrons d'évolution spécifient les inputs du processus d'évolution, c'est-à-dire les informations extraites des sources multimédia et présentées sous forme de ABox (instances de concepts et de relations) ; ils déterminent l'opération d'évolution à exécuter : peuplement (ajout de nouvelles instances) ou enrichissement (extension par de nouveaux concepts, relations, propriétés).

Quatre patrons d'évolution sont distingués [Castano *et al.*, 2007]. Les patrons de peuplement sont utilisés lorsque l'interprétation d'une ressource multimédia (input) peut être

expliquée par un seul (P1) ou de multiples (P2) concept(s) déjà présents dans l'ontologie. Les patrons d'enrichissement sont utilisés lorsqu'il n'y a pas de concepts de l'ontologie expliquant les ressources par des métadonnées (P3) ou sans (P4).

Le processus d'évolution de BOEMIE comporte quatre étapes principales (figure II-6) : *sélection des patrons*, *peuplement* (ajout de nouvelles instances), *enrichissement* (ajout de nouveaux concept, nouvelles relations, ...), *et coordination* (production d'un journal des changements effectués dans l'ontologie depuis la version précédente) [Castano *et al.*, 2007] et [Petasis *et al.*, 2007].

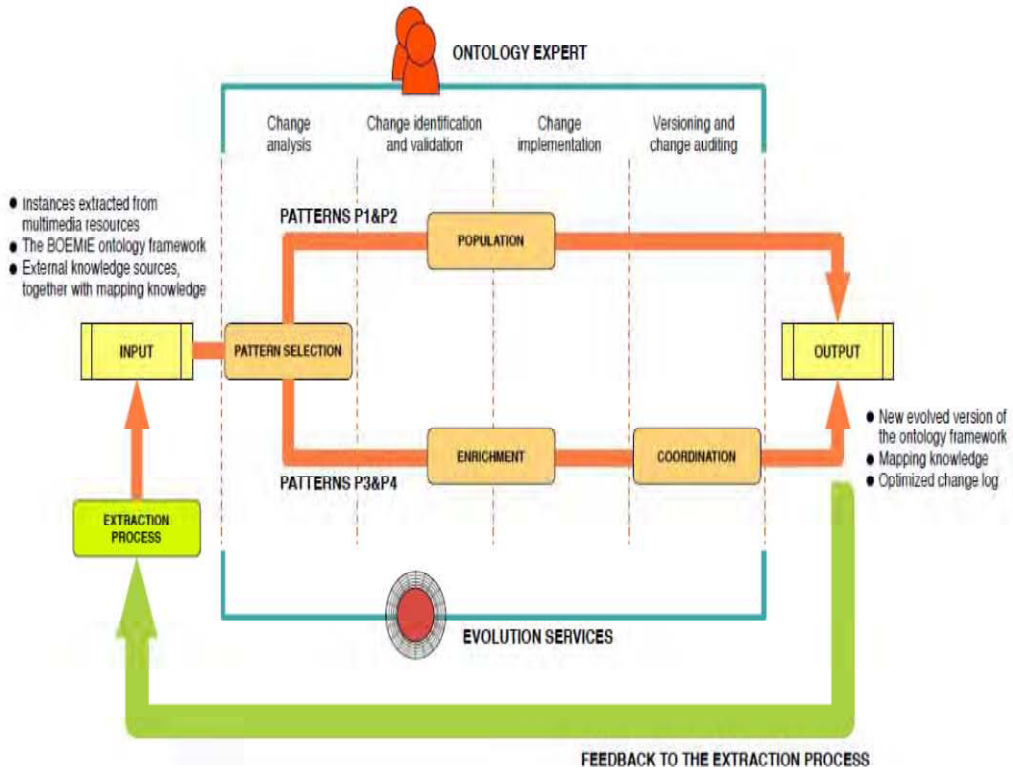


Figure II - 6 : Méthodologie d'évolution BOEMIE selon [Castano *et al.*, 2007]

2.2.4.6 Approches logiques de l'évolution d'ontologie

L'évolution d'ontologie peut aussi s'appuyer sur le modèle logique de données des ontologies qui évoluent.

La première de ces approches consiste à adapter les principes de changement de croyance (Belief Changes) à l'évolution d'ontologie [Flouris et Plexousakis, 2005][Flouris, 2005]. Les changements de croyance se réfèrent à l'adaptation automatique d'une base de connaissances à de nouvelles connaissances et ce, sans intervention humaine [Flouris *et al.,* 2006a]. Les changements sont exprimés sous forme d'un ensemble de propositions ou d'axiomes conformes au modèle des logiques de description dans lequel toute l'ontologie, et ses axiomes en particulier, sont exprimés. Quatre opérations de changement sont possibles : *Révision* et *contraction* pour les changements liés à la conceptualisation (c'est la perception du domaine qui change et non le domaine lui même) et *mise à jour* et *suppression* pour les changements du domaine (nouvelles réalités).

Une deuxième approche de ce type s'appuie sur un langage de définition des changements CDL (*Change Definition Language*) permettant aux utilisateurs de représenter de manière formelle et déclarative les changements qu'ils veulent apporter à l'ontologie (différences entre l'ancienne version et la version courante de l'ontologie) [Plessers et De Troyer, 2005], [Plessers *et al.,* 2007]. Les changements sont exprimés sous forme de requêtes temporelles appliquées à l'historique des versions (un changement de l'ontologie s'exprime en termes de pré-conditions et post-conditions). L'historique des versions permet de sauvegarder, pour chaque concept défini dans l'ontologie, toutes ses versions. La syntaxe et la sémantique de langage CDL sont détaillées dans [Plessers *et al.,* 2007].

2.2.5 Représentation des changements

Pour un langage de représentation d'ontologie donné, une taxonomie de changements spécifiant des classes de changements et leurs propriétés est indispensable. Plusieurs classifications ont été proposées dans la littérature. Dans la section qui suit, nous présentons

un état de l'art des classifications des changements en complément des classifications de Stojanovic proposées en section 2.2.3.

2.2.5.1 Représentation des changements selon Klein [Klein et Noy, 2003] et [Klein, 2004]

Dans le cadre qu'ils définissent pour l'évolution d'ontologie, Klein et Noy distinguent plusieurs types de changements pour les ontologies au format OWL. Ils les organisent dans une ontologie des changements. Nous reproduisons dans le tableau II-3 la taxonomie des changements élémentaires (dits atomiques).

Tableau II - 3 : Taxonomie des changements atomiques selon [Klein, 2004]

Élément d'une ontologie OWL	Opération de changement
Opérations appliquées aux classes	
Class	Add, remove
Cardinality upper/lowerbound	Add, remove
Cardinality upper/lowerbound	Modify
Superclass relation	Add, remove
Class disjointness	Add, remove
Class equivalence	Add, remove
Opérations appliquées aux propriétés	
Property	Add, remove
Domain Range	Add, remove Add, remove
Superproperty relation Property equivalence	Add, remove Add, remove
Property inverse Property symmetry Property functionality Property transitivity Property inverse-functionality	Add, remove Set, unset Set, unset Set, unset Set, unset

La classification des changements de Klein s'appuie sur deux dimensions : atomique *vs* composite, simple *vs* riche (tableau II-4).

Tableau II - 4 : Classification de changements OWL, selon [Klein, 2004]

changement	simple	riche
Atomique	élémentaire	complexe
Composite	complexe	complexe

Les *opérations atomiques* sont des opérations qui ne peuvent pas être subdivisées dans des opérations plus petites, alors que les *opérations composites* sont des opérations composées de plusieurs changements atomiques.

Les *changements simples* sont ceux qui peuvent être identifiés en analysant uniquement la structure de l'ontologie alors que les *changements riches* font appel au raisonnement. Selon Rogozan, l'identification de changements riches requiert une interrogation de la théorie logique de l'ontologie, étant donné que ces changements englobent une information à propos de leurs conséquences sur le modèle logique de l'ontologie. Par exemple, étendre le co-domaine d'une propriété à la superclasse de la classe qui le spécifiait est un exemple de changement riche.

Pour Klein, un changement n'est élémentaire que s'il est simple et atomique. Dans les autres cas, il est complexe (au sens de Stojanovic). Si l'on compare cette taxonomie à celle de Stojanovic, on voit que Klein introduit en plus : (a) des opérations de changement de type *modifier* ainsi que *mettre à* (Set) et *annuler* (Unset) pour les caractéristiques de propriétés (telles que la transitivité ou la symétrie) ; (b) des opérations d'ajout et d'effacement d'une relation hiérarchique entre deux classes ou d'une relation entre deux propriétés ; (c) des opérations d'ajout et d'effacement d'une équivalence ou d'une disjonction entre deux classes. Cette taxonomie est le point de départ des travaux de [Klein *et al.*, 2002], [Klein et Noy, 2003], [Noy et Klein, 2003], [Stuckenschmidt et Klein, 2003].

2.2.5.2 Représentation des changements dans l'ontologie CHAO [Noy *et al.*, 2006]

Un prolongement des travaux de [Klein, 2004] a donné lieu à une ontologie des changements et des annotations (*Change and Annotation Ontology CHAO*). Cette ontologie étend la taxonomie précédente pour représenter des changements entre deux versions d'une ontologie et les annotations (au sens de commentaires) qu'un utilisateur associe à ces changements. L'ontologie contient deux classes principales (figure II-7) : la classe «*Change*» pour représenter les changements de l'ontologie (les auteurs réutilisent les travaux précédents) et la classe « *Annotation* » pour stocker les annotations liées aux changements (ce qui constitue l'originalité de cette ontologie par rapport aux travaux précédents).

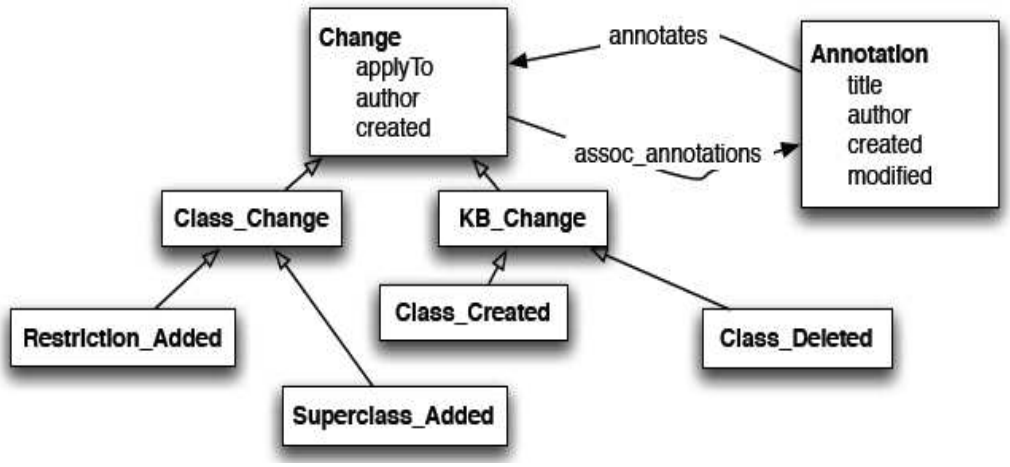


Figure II - 7 : Les classes *Change* et *Annotation* de l'ontologie de changement et d'annotation CHAO

2.2.5.3 Modélisation formelle des changements [Luong, 2007] [Luong et Dieng-Kuntz, 2007]

Dans le cadre de ses recherches (cf.3.2.4.4), Luong élabore une ontologie d'évolution s'inspirant toujours des travaux de [Stojanovic, 2004] mais avec le but de modéliser d'une manière formelle les types de changements. Il considère que l'ensemble des changements élémentaires proposés par Stojanovic ne contient que des opérations additives et soustractives

des concepts et des propriétés qui ne sont pas suffisantes pour exprimer les relations entre les hiérarchies des concepts ou des propriétés. Par exemple, le changement *CreateHierarchyConceptLink (c1, c2)* décrit une opération de création d'un lien entre deux concepts c1 et c2, ou un autre changement *RemovePropertyDomainLink (c, p)* enlève le concept c du domaine d'une propriété p dans l'ontologie.

Luong a défini un certain nombre de notions qui permettent de décrire formellement la syntaxe, les paramètres et la sémantique d'un changement, de même que les conditions à satisfaire avant et après son application. Par exemple, le changement *CreateHierarchyConceptLink (c_1, c_2)* qui relie deux concepts de l'ensemble des concepts C de l'ontologie par un lien hiérarchique est décrit dans le tableau II-5 :

Tableau II - 5 : Description d'un changement selon [Luong, 2007]

Syntaxe	***Sémantique***
CreateHierarchyConceptLink (c_1, c_2)	Créer un lien de hiérarchie entre les concepts c_1 et c_2, c_2 devient le père de c_1
Pré-condition	***Post-condition***
$c_1, c_2 \in C, (c_1, c_2) \notin H_C$	$c_1, c_2 \in C, (c_1, c_2) \in H_C$

Les changements modélisés par Luong correspondent à des changements élémentaires et composites (Tableau II-6) : un changement élémentaire affecte une seule entité de l'ontologie, alors qu'un changement composite est celui qui crée, retire ou modifie un et seulement un niveau du voisinage (voisinage direct) d'un élément de l'ontologie.

Tableau II - 6 : Liste des changements élémentaires et composites selon [Luong, 2007]

Changements élémentaires	
Sur le concept	**Sur la propriété**
InsertConcept	InsertProperty
DeleteConcept	DeleteProperty
RenameConcept	RenameProperty
CreateHierarchyConceptLink	CreateHierarchyPropertyLink
RemoveHierarchyConceptLink	RemoveHierarchyPropertyLink
	CreatePropertyDomainLink
	RemovePropertyDomainLink
	CreatePropertyRangeLink
	RemovePropertyRangeLink
Changements composites	
Sur le concept	**Sur la propriété**
CreateCommonConcept	CreateCommonProperty
MergeConcept	MergeProperty
SplitConcept	SplitProperty
InsertConceptGeneralisation	InsertPropertyGeneralisation
InsertConceptSpecialisation	InsertPropertySpecialisation
MoveConcept	MoveProperty

Les changements sont aussi classés selon leur impact sur les annotations sémantiques produites à l'aide de l'ontologie. Luong a divisé l'ensemble des changements de l'ontologie en deux groupes principaux : (a) *correction obligatoire* et (b) *correction facultative*. Une *correction obligatoire* entraîne nécessairement une ou des inconsistances dans les annotations concernées par ce changement. Une *correction facultative* n'entraîne pas d'inconsistances dans les annotations concernées par ce changement.

La figure II-8 représente un type de changement entraînant une correction obligatoire. Supposons qu'il existe un triplet de l'annotation sémantique qui utilise les concepts *Actor*, *Role_in_the_CoP* et la propriété *plays_role* de l'ontologie de base. La suppression du concept *Actor* va causer une inconsistance entre l'annotation et l'ontologie.

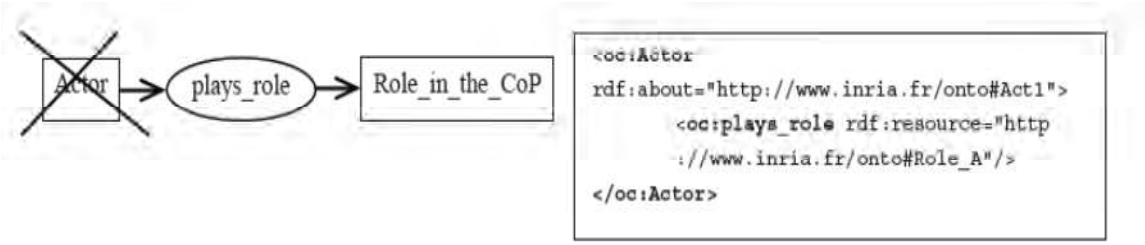


Figure II - 8 : Exemple de changement entraînant une correction obligatoire

Luong propose donc une *ontologie d'évolution* qui a pour but de définir formellement la classification des changements ontologiques ainsi que leurs relations avec les entités de l'ontologie du domaine. Cette ontologie formalise également des informations concernant le processus d'évolution de l'ontologie [Luong, 2007]. La figure II-9 présente un large extrait de l'ontologie d'évolution utilisée dans CoSWEM. Cette ontologie modélise des traces de changements à l'aide du concept *Trace* et de ses sous-concepts *TraceOnto* et *TraceAnnot* qui vont servir à représenter des traces d'évolution, respectivement de l'ontologie et de l'annotation sémantique. Pour chaque trace, le système enregistre des informations comme l'auteur qui a réalisé des changements (*hasAuthor*), l'identifiant et la date de la trace effectuée (*hasNumber*, *hasDate*). Chaque trace conserve aussi les changements effectués entre deux versions consécutives de l'ontologie (*hasVersionBefore*, *hasVersionAfter* entre deux concepts *Ontologie*), etc. La deuxième information importante à modéliser correspond aux changements effectués. Chaque trace contient (via la propriété *hasTrace*) plusieurs changements différents (ex. le concept *Change*) [Luong, 2007].

Figure II - 9 : Les concepts (colonne de gauche) et les propriétés (colonne de droite) de l'ontologie d'évolution

2.2.5.4 Représentation des changements selon Rogozan [Rogozan, 2008] [Rogozan et Paquette, 2005]

Rogozan présente une ontologie des changements applicables à des ontologies OWL-DL Cette ontologie élargit la conceptualisation de [Klein, 2004], par l'ajout d'un certain nombre de caractéristiques comme une typologie des changements complexes et une description des effets de changements sur le référencement sémantique. Cette ontologie contient trois classes principales (figure II-10) : (1) la classe *ChangeOperation* spécifie les changements applicables aux ontologies OWL-DL ; elle se compose d'un ensemble de changements élémentaires, spécifié par la sous-hiérarchie *ElementaryChange* (figure II-11), ainsi que d'un

ensemble de changements complexes, spécifié par la sous-hiérarchie *ComplexChange* (figure II-12); (2) la classe *ChangeObject* présente les objets sur lesquels agissent les changements ; (3) la classe *ChangeEffect* décrit les types de conséquences qu'on peut rencontrer après avoir effectué un changement.

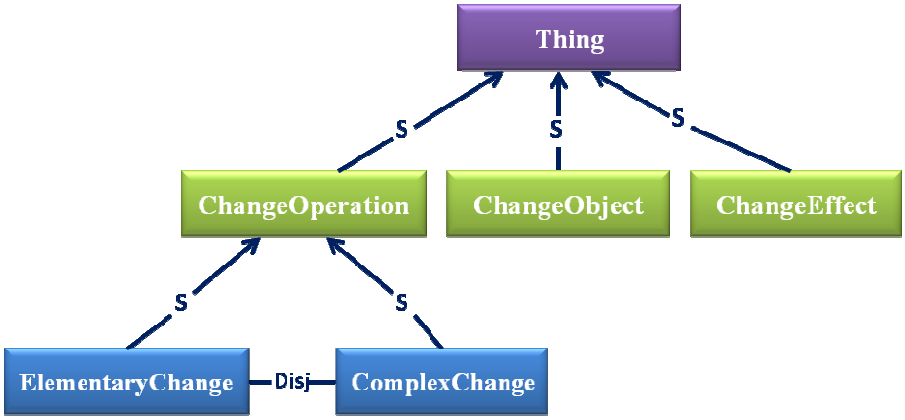


Figure II - 10 : Classes racines de l'ontologie des changements

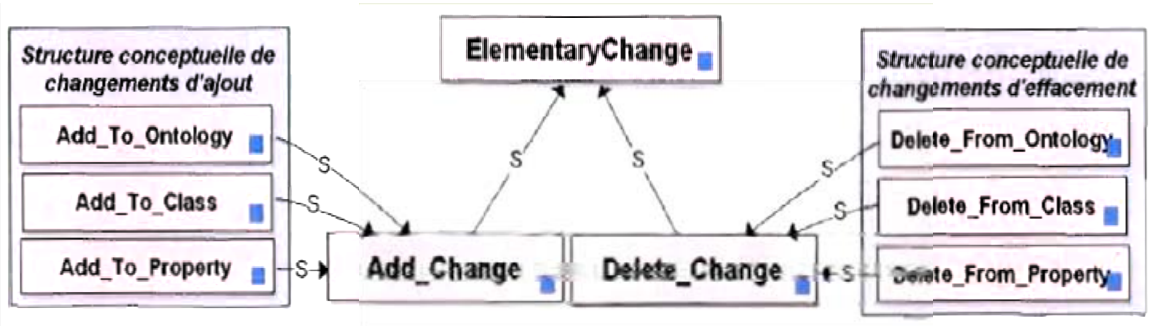


Figure II - 11 : Hiérarchie des changements élémentaires selon [Rogozan, 2008]

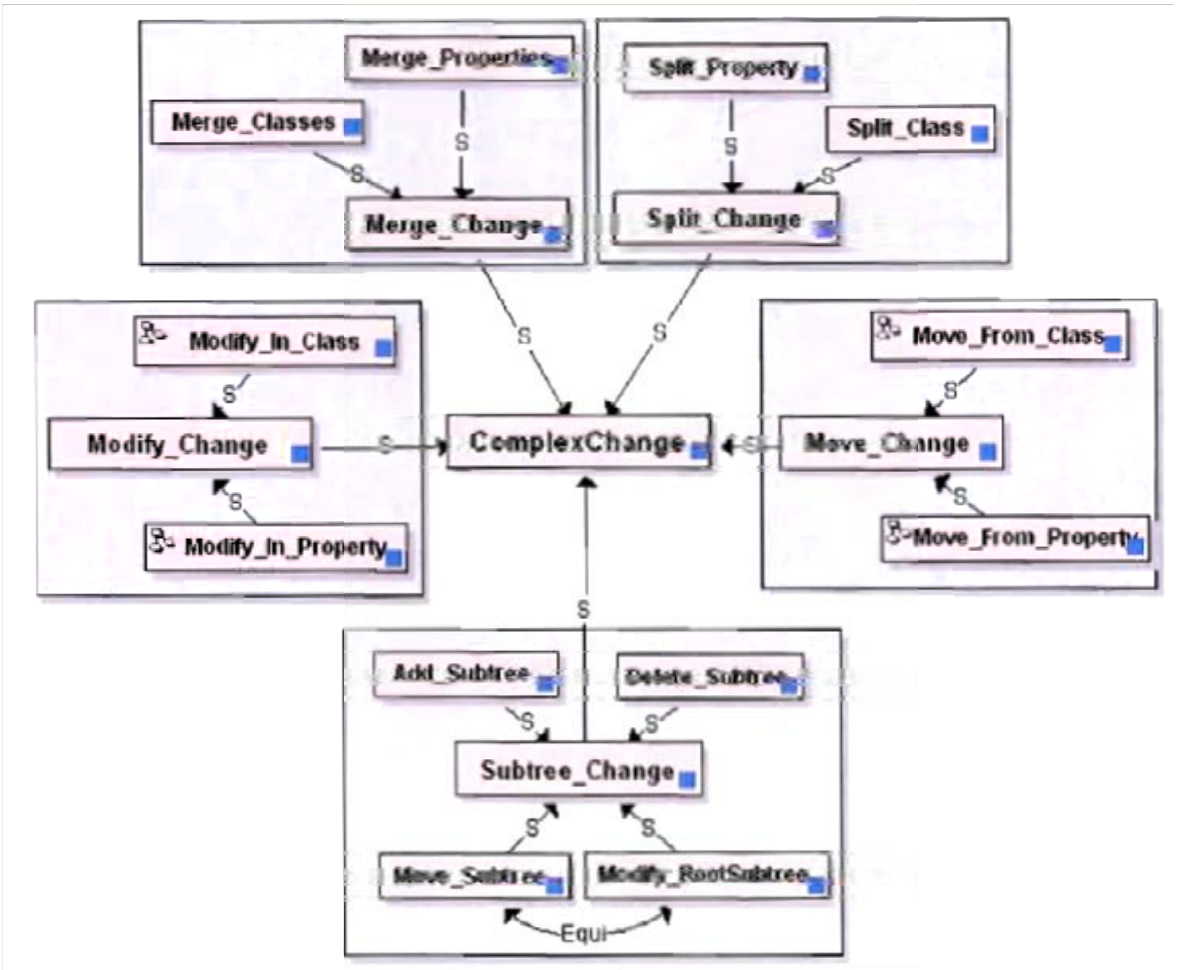


Figure II - 12 : Hiérarchie des changements complexes selon [Rogozan, 2008]

A cette phase de représentation des changements, on peut s'interroger sur les conséquences de ces changements, c'est-à-dire sur le fait de savoir si la définition des changements ne devrait pas contenir une description de leurs conséquences sur l'ontologie elle-même ou sur les annotations sémantiques de collections documentaires. Comme le souligne bien Rogozan, ni Klein ni Stojanovic ne représentent les conséquences d'un changement dans leur ontologie des changements. Klein parle cependant de changement riche, (comme l'élargissement du domaine d'une propriété), mais ne décrit pas ses effets sur l'ontologie. De même, pour les changements complexes proposés par Stojanovic comme la généralisation ou la spécialisation d'un concept. C'est pour cela que la section suivante aborde les effets des changements.

2.2.6 Effets des changements

Dans les situations où les ontologies sont utilisées par diverses communautés d'utilisateurs, ou par plusieurs applications, le changement d'un élément dans une ontologie peut avoir un impact sur d'autres éléments au sein de la même ontologie ou d'autres éléments liés aux applications. Par conséquent, les effets de changements doivent être analysés non seulement par rapport à l'ontologie elle-même, mais aussi en tenant compte des raisons de changement et de l'utilisation de l'ontologie.

Selon une approche de gestion d'ontologies distribuées [Klein, 2004], les effets des changements dépendent de ce que nous avons besoin de préserver :

- les données, pour maintenir un typage correct des instances entre les différentes versions d'une ontologie ;
- l'ontologie elle-même, pour maintenir les résultats de requêtes de sorte que le résultat d'une requête q_1 sur une version O_i de l'ontologie, s'il reste valide, soit inclus dans le résultat de la même requête q_1 sur une version O_{i+1} de ladite ontologie ;
- les conséquences, pour maintenir les inférences de sorte que les faits inférés d'un axiome a_1 sur une version O_i de l'ontologie soient aussi inférés du même axiome a_1 sur une version O_{i+1} de ladite ontologie si a_1 reste vrai ;
- la cohérence, pour assurer que la nouvelle version ne contienne pas d'incohérences logique.

Selon [Djedidi, 2009] [Noy et Klein, 2004] et par analogie avec les changements de schémas de bases de données, les effets de changements peuvent aussi être classés en tenant compte de leur impact sur les instances de l'ontologie. On parlera alors des types suivants :

- Changement préservant : lorsque les instances ne sont pas perdues (par exemple en ajoutant un concept ou une propriété) ;

- Changement par traduction : préserve les instances en traduisant les connaissances sous une autre forme (en groupant par exemple, des concepts en une union de leurs super-concepts, sous-concepts et propriétés, les instances peuvent être préservées) ;
- Changements non-préservants : les instances sont perdues (la suppression d'une propriété par exemple, peut causer la perte des instances ou de relations entre instances).

[Stuckenschmidt et Klein, 2003] présentent les situations où les classes deviennent plus spécialisées ou plus générales sous l'effet des changements qui ajoutent ou qui suppriment des entités d'une ontologie (figure II-13). Toutefois, ces auteurs traitent uniquement les effets des changements élémentaires, ce qui est insuffisant pour une interprétation pertinente de la modification conceptuelle introduite dans la nouvelle version de l'ontologie.

Operation	Effect
Attach a relation to concept *C*	*C*: Specialized
Complex. Change the superclass of concept *C* to a concept lower in the hierarchy	*C*: Specialized
Complex. Restnct the range of a relation *R* *(efleet on all C that have a restriction on R)*	R: Specialized, *C*: Specialized
Remove a superclass relation of a concept *C*	*C*. Generalized
Change the concept definition of *C* from primitive to defined	*C*: Generalized
Add a concept definition *A*	*C*: Unknown
1 *Complex* Add a (not further specified) sub-class *A* of *C*	*C*: No effect
Define a relation *R* as functional	/?: Specialized

Figure II - 13 : Une modification dans une ontologie et ses effets sur la classification des concepts dans la hiérarchie, selon [Stuckenschmidt et Klein, 2003]

2.3 Gestion de la dynamique des annotations sémantiques

Dans cette section, nous examinons les effets des changements ontologiques sur l'annotation à savoir la détection des annotations inconsistantes et la résolution des annotations inconsistantes. Peu nombreux sont les travaux actuels qui identifient les effets potentiels de l'évolution des ontologies sur les annotations sémantiques.

2.3.1 Dynamique des annotations dans la plateforme NEON [Maynard *et al.*, 2007]

Dans le cadre du projet NEON, une approche intégrée définit l'évolution des ontologies et des métadonnées associées. Elle aborde la problématique de propagation des effets des changements ontologiques aux métadonnées associées et vice-versa. Dans ce contexte, le terme métadonnées est défini comme l'annotation des données textuelles, et par conséquent, la production des méta-données utilisent des informations sur le contenu linguistique de l'ontologie. Ce type de métadonnées est souvent appelé *métadonnées sémantiques*. Les auteurs proposent de classer les changements de l'ontologie selon ce qu'ils provoquent sur l'ontologie : la mise à jour des concepts, des instances ou des propriétés. Les changements ont des effets sur les métadonnées (sur les instances) qui décrivent les données textuelles. Pour propager les modifications conformément à la nouvelle version de l'ontologie, ils proposent différentes mesures qui peuvent être prises : (i) ne rien faire, (ii) corriger par une action manuelle ou (iii) corriger par une action automatique. L'action (ii) correspond à une sorte de ré-annotation manuelle des textes avec la nouvelle version de l'ontologie, tandis que l'action (iii) nécessite un ensemble de procédures qui réalisent le reclassement des instances dans l'ontologie (les auteurs fournissent quelques exemples basés sur l'API de la plate-forme de Traitement Automatique des Langues GATE).

2.3.2 CoSWEM : évolution d'annotations suite à l'évolution d'ontologie [Luong, 2007] [Luong et Dieng-Kuntz, 2007]

L'approche de Luong (que nous avons présentée en 3.2.4.4) est la plus complète pour gérer à la fois l'évolution de l'ontologie et celle des annotations sémantiques produites avec cette ontologie. Rappelons que cette approche permet de comparer des annotations et une nouvelle version de l'ontologie qui a servi à l'annoter pour identifier les annotations qui ne sont plus consistantes avec l'ontologie (utilisant par exemple des entités qui n'existent plus dans la nouvelle version). Il propose à l'utilisateur des solutions pour rectifier les annotations devenues inconsistantes. Pour cela, il propose deux approches.

Selon la première approche, procédurale, le système récupère les historiques d'évolution fournis par l'éditeur d'ontologie ECCO[10] (*Éditeur Collaboratif et Contextuel d'Ontologie*) sous forme d'un journal d'évolution (figure II-14). Cette trace d'évolution est représentée sous forme d'annotations XML qui facilitent son exploitation (figure II-15). Dans un fichier de trace, des concepts et des propriétés de l'ontologie d'évolution décrivent les modifications apportées (*TraceOnto, DeleteConcept, hasChange, hasAdditionalChange, etc.*) aux entités (sur la figure II-14, concepts *Member_role, Facilitator, Coordinator, Role_in_the_CoP* et la propriété *coordinate)* de l'ontologie modifiée (ici l'ontologie O'CoP dont un extrait peut être vu dans [Luong, 2007] page 79).

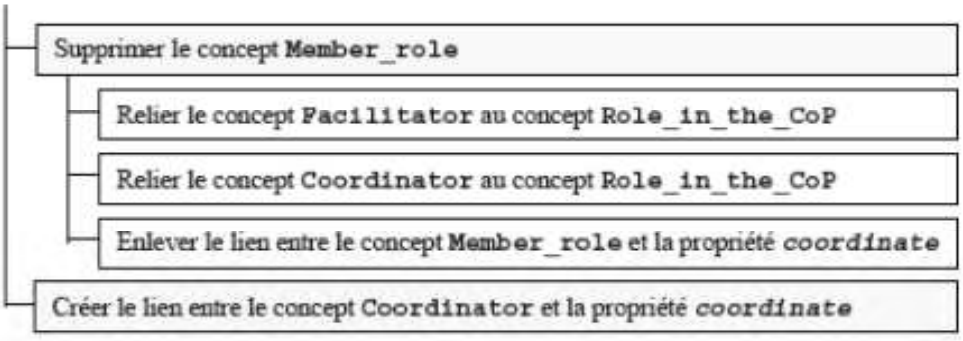


Figure II - 14 : Séquence de changements effectués sur l'ontologie selon [Luong, 2007]

[10] http://www-sop.inria.fr/edelweiss/projects/ewok/publications/ecco.html

```
<TraceOnto>
   <hasChange>
      <DeleteConcept >
         <hasDeleteConcept Member_role>
         <hasAnnotInconsistency>yes</hasAnnotInconsistency>
            <hasAdditionalChange>
               <CreateHierarchyConceptLink>
                <hasAnnotInconsistency>no</hasAnnotInconsistency>
                 <hasSubConceptCreate Facilitator >
                 <hasSuperConceptCreate Role_in_the_CoP >
               <CreateHierarchyConceptLink>
            </hasAdditionalChange>
            <hasAdditionalChange>
               <CreateHierarchyConceptLink>
                 <hasAnnotInconsistency>no</hasAnnotInconsistency>
                 <hasSubConceptCreate Coordinator >
                 <hasSuperConceptCreate Role_in_the_CoP  >
               <CreateHierarchyConceptLink>
            </hasAdditionalChange>
            <hasAdditionalChange>
               <RemoveConceptDomainLink>
                  <hasAnnotInconsistency>yes</hasAnnotInconsistency>
                  <hasConceptDomainCreate Member_role >
                  <hasPropertyDomainCreate coordinate >
                <RemoveConceptDomainLink>
            </hasAdditionalChange>
      </DeleteConcept>
   </hasChange>

   <hasChange>
      <CreateConceptDomainLink>
         <hasAnnotInconsistency>no</hasAnnotInconsistency>
         <hasConceptDomainCreate Coordinator >
         <hasPropertyDomainCreate coordinate >
      <CreateConceptDomainLink>
   </hasChange>
<TraceOnto>
```

Figure II - 15 : Fichier XML des traces générées par les changements de la figure II-14

A partir de ce journal, le système CoSWEM identifie puis corrige les annotations incohérentes. Pour cela, il s'appuie sur le moteur Corese pour trouver les annotations qui contiennent un des éléments modifiés dans l'ontologie, et localiser ainsi les annotations devenues inconsistantes avec l'ontologie modifiée. Il applique ensuite des procédures

d'adaptation des annotations qui corrigent systématiquement chaque type de changement selon une stratégie par défaut.

Le système commence par appliquer des règles de détection d'inconsistance, comme celle de l'exemple 4, pour détecter les annotations qui contiennent des entités (concepts, propriétés, domaine ou co-domaine de propriétés) supprimées de l'ontologie. Il applique ensuite des règles de correction d'inconsistance (exemple 5) pour proposer à l'utilisateur de rectifier les annotations devenues incohérentes par rapport à l'ontologie modifiée.

Exemple 4

Règle de détection pour les concepts

Soient :
C_O et P_O les ensembles des concepts et des propriétés de l'ontologie *O*,
C_A et P_A les ensembles des concepts et des propriétés figurant dans l'annotation *A*.
D_O et D_A les ensembles de tous les types de données définis respectivement dans l'ontologie *O* et dans l'annotation *A*.
Tous les triplets dans l'annotation ont la forme (*r* p_t *v .)* avec la ressource *r* du type c_t (*r type* c_t .).
Si un type de concept est utilisé dans l'annotation mais n'est pas défini dans l'ontologie de base, alors l'annotation mène à un état inconsistant.

```
    R-1 :∀ annotation A, Si (c_t ∈ C_A) et (c_t ∉ C_O) Alors
marquer(inconsistant)

    R-2 :∀ annotation A, Si (c_t ∈ C_A) et (c_t ∈ C_O) Alors
marquer (OK)
```

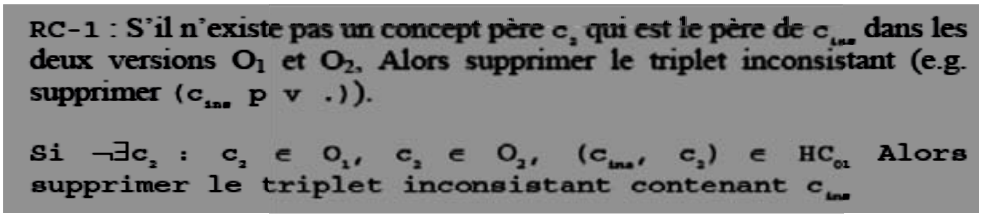
Exemple 5 :

Règle de correction d'inconsistance : RC

Soient :
O_1 : ancienne version de l'ontologie (avant la modification)
O_2 : nouvelle version de l'ontologie
CO_1, CO_2, C_A pour indiquer respectivement les ensembles de concepts de l'ontologie O_1, O_2et de l'annotation *A*.
PO_1, PO_2, P_A représentent respectivement les ensembles de propriétés de l'ontologie O_1, O_2 et l'annotation A.
HC_{O1}, HC_{O2} représentent les hiérarchies de concepts de O_1 et O_2, par exemple $(c, c_2) \in HC_{O1}$ signifie que une le concept *c* a pour père $c_2$dans l'ontologie O_1.

Si un triplet est inconsistant, il contient alors au moins un élément inconsistant, par exemple (c_{ins} *p v* .) ou (*c* p_{ins} *v* .) ou (c_{ins} p_{ins} *v* .), etc.

Un élément est inconsistant (ex. c_{ins}, p_{ins}) s'il existe dans l'ancienne version de l'ontologie O_1 et de l'annotation *A* mais n'existe pas dans la nouvelle version d'ontologie O_2.



Pratiquement, ces analyses sont restées à l'état de propositions et n'ont pas été implémentées dans CoSWEM.

Les travaux de Luong ne traitent la résolution des inconsistances que de manière partielle, et leurs résultats restent des propositions encore théoriques. Notre recherche cherche à aller plus loin en implémentant un système d'aide qui traite l'évolution des annotations sémantiques après les modifications d'une ressource termino-ontologique (ontologie avec une composante terminologique). Ce système doit détecter des annotations inconsistantes et les corriger. Il doit également gérer les effets sur les ressources termino-ontologiques des problèmes rencontrés dans l'annotation sémantique.

2.3.3 SAM : modifier les annotations à partir d'un journal d'évolution [Rogozan et Paquette, 2005] [Rogozan, 2008]

La méthode – *Semantic Annotation Modifier* (SAM) – assure un référencement sémantique évolutif des ressources décrites à l'aide des concepts (classes) d'une ontologie qui évolue dans le temps (cf. 3.2.4.2). SAM permet aussi d'analyser les effets des changements de l'ontologie sur le référencement sémantique de ressources, assure le maintien de l'accès à ces ressources et leur interprétation.

La première fonctionnalité de SAM est l'analyse des effets des changements. Pour cela, les utilisateurs fournissent à SAM le référencement sémantique des ressources sous la forme d'un fichier contenant des UKI associés aux ressources, et cela, dans un fichier joint à leur demande d'analyse des changements, comme illustré dans la figure II-16. Ensuite, SAM extrait les changements apportés à la version Vn, en utilisant le journal d'évolution. Finalement, il met en correspondance les UKI avec les changements ainsi extraits. Le but ici est d'identifier quels sont les UKI affectés et quels sont les changements qui les affectent.

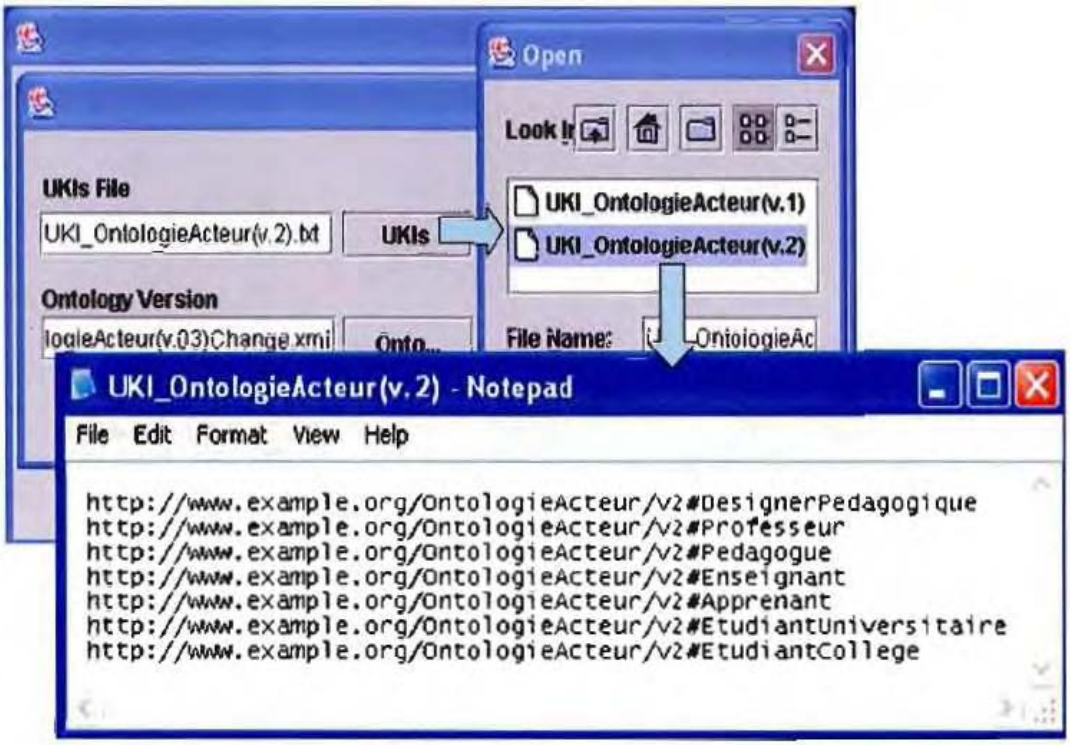


Figure II - 16 : Sélection et format d'un fichier des UKI

Après l'analyse des effets des changements, le système SAM modifie le référencement sémantique pour préserver l'accès aux ressources ainsi que leur interprétation après l'évolution de l'ontologie de référence. Cette modification se déroule en trois étapes : *l'organisation des UKI à modifier, la modification des UKI, la génération d'un fichier avec les UKI modifiés*. Pendant la première étape - *organisation des UKI à modifier* – le système répartit les UKI à modifier en trois catégories à savoir, les UKI non affectés par les

changements, les UKI affectés par des changements n'ayant aucun effet et, finalement, les UKI affectés par des changements ayant des effets. Lors de *la modification des UKI*, le système fait appel à l'utilisateur pour valider les modifications qu'il propose ou pour choisir parmi plusieurs modifications possibles. A l'issue de la dernière étape du processus, SAM génère le fichier final avec les UKI modifiés et le fournit aux utilisateurs.

Après cette présentation des approches existantes en évolution d'ontologie et en analyse des effets de changements sur les annotations sémantiques, nous décrivons dans la section suivante, les principaux logiciels offrant des fonctionnalités d'évolution d'ontologie et des annotations sémantiques.

2.4 Logiciels d'aide à l'évolution d'ontologie

Il existe peu d'outils qui assistent un processus complet d'évolution d'ontologie. La gestion des évolutions est réalisée partiellement au sein d'éditeurs d'ontologies, ou par des logiciels qui offrent des fonctionnalités plus spécialisées pour cette tâche. Nous fournissons dans la section 2.4.1 un aperçu des logiciels d'édition d'ontologie qui se présentent rarement comme des plateformes de gestion d'évolution, les modifications de l'ontologie étant prises en compte immédiatement et de façon élémentaire. Dans la section 2.4.2., nous donnons un aperçu des outils qui traitent spécifiquement du processus d'évolution d'ontologie.

2.4.1 Gestion des évolutions dans les éditeurs d'ontologie

Protégé et ses plug-in [Noy et Musen, 2000], [Noy et Musen, 2003], [Noy et Musen, 2002] et [Noy *et al.*, 2004]

La plateforme Protégé[11] est un outil open-source développé par l'équipe SMI (*Stanford Medical Informatics*) de l'Université de Stanford. C'est un éditeur de création de base de connaissances très convivial, réalisé en Java. Actuellement, cet éditeur est largement utilisé et

[11]http://protege.stanford.edu

figure parmi les plus répandus. Une grande communauté l'utilise pour créer et gérer des ontologies aux formats RDF(S) et OWL, dans différents domaines.

Protégé intègre de nombreux plugins permettant des manipulations sur les bases de connaissances créées. Une communauté d'utilisateurs développe régulièrement ces plug-ins pour répondre à des besoins spécifiques. Certains de ceux-ci sont dédiés tout particulièrement à l'évolution d'ontologie. On peut citer *PROMPT* [Noy et Musen, 2003] et *PROMPTdiff* [Noy et Musen, 2002] .

PROMPT[12] est une suite de plug-in de Protégé en vue de gérer plusieurs ontologies à la fois. Cet outil offre les fonctionnalités principales suivantes :

(1) comparer deux versions d'une ontologie ;

(2) fusionner deux ontologies ;

(3) extraire une partie d'une ontologie.

PROMPTDiff utilise des heuristiques pour comparer différentes versions d'une ontologie au niveau structurel et créer une "*structuraldiff*" des différences entre ces versions.

Une suite de PROMPT a été proposée par [Noy et Musen, 2003] : PROMPT *suite*. Il contient des outils tels que *AnchorPROMPT*, *iPROMPT*, *PROMPTFactor* afin de réaliser des tâches telles que la fusion d'ontologies, leur alignement, la gestion de ses versions.

2.4.2 Logiciels dédiés à l'évolution d'ontologie

2.4.2.1 Gestion des évolutions dans KAON [Stojanovic *et al.*, 2002a], [Maedche et Staab, 2003] et [Gabel *et al.*, 2004]

[12]http://protege.stanford.edu/plugins/prompt/prompt.html

Un des premiers éditeurs d'ontologie à assurer en partie la gestion automatisée des évolutions d'une ontologie est la plateforme KAON[13] (***KA****rlsruhe* ***ON****tology and Semantic Web infrastructure*) proposée à l'Université de Karlsruhe. Le processus d'évolution défini par Stojanovic (cf. 3.2.3.2) a été prévu pour s'intégrer dans KAON. Il offre plusieurs fonctions pour saisir des changements, vérifier la consistance de l'ontologie et créer des stratégies d'évolution personnalisées. 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 et a ignoré une étape très importante du processus d'évolution qui est la propagation des changements.

2.4.2.2 EVOLVA [Zablith, 2011], [Zablith *et al.*, 2010], [Zablith *et al.*, 2010], [Zablith *et al.*, 2009]} et [Zablith, 2009]

Plus récemment, au sein de la boîte à outils NEON ToolKit, l'outil Evolva[14] facilite l'évolution d'ontologies en exploitant des textes et en réutilisant des ressources comme des ontologies ou des bases de données. Evolva ne traite qu'un seul type d'évolution : l'ajout de concepts et de relations. Les nouveaux concepts intégrés à une ontologie 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 sur le web. Pour identifier des relations dans des ontologies en ligne, Evolva utilise le logiciel Scarlet [Sabou *et al.*, 2008], qui découvre des relations sémantiques à l'aide du moteur de recherche d'ontologies Watson[15] [d'Aquin *et al.*, 2008]. Il s'appuie sur des

[13]http://kaon.semanticweb.org
[14] http://evolva.kmi.open.ac.uk/
[15] http://watson.kmi.open.ac.uk/Overview.html

heuristiques d'extraction de connaissances qui permettent de sélectionner les ontologies pertinentes qui peuvent être réutilisées par rapport à l'ontologie en cours et à chaque nouveau concept à ajouter. D'autres heuristiques filtrent les propositions de Scarlet pour mettre en relation les connaissances extraites et les concepts de l'ontologie à enrichir.

Le système Evolva contient cinq composants : (a) Découverte d'Information, (b) Validation des données, (c) Changements dans l'ontologie, (d) Validation de l'évolution, et (e) Gestion de l'évolution.

1. *Découverte d'Information* : ce module commence par découvrir de nouvelles informations à partir de sources de données : textes, bases de données et ontologies. Il compare l'ontologie à enrichir au contenu de ces sources pour détecter de nouveaux termes en traitant chacune des sources de manière spécifique. (i) Les textes sont analysés à l'aide d'outils d'extraction de termes ; (ii) Les ontologies sont traduites de manière à être utilisables dans un langage compatible avec celui de l'ontologie de base ; enfin (iii) le contenu des bases de données est traduit sous un format ontologique.

2. *Validation des données :* en se basant sur un ensemble de règles heuristiques (ex. la longueur des termes extraits des textes), ce module permet de valider les termes découverts. A ce niveau, la liste des termes extraits est affichée pour l'utilisateur. La validation montre les termes qui correspondent aux concepts de l'ontologie déjà existants. En outre, le bruit est éliminé sur la base de la longueur des termes extraits. Un contrôleur teste la longueur de la chaîne de caractères pour exclure par exemple les termes formés de seulement une ou deux lettres. Les paramètres pour contrôler la validation automatique comprennent le changement de la longueur minimale des termes acceptables ainsi que l'utilisation d'un seuil de similitude. En plus de la

validation automatique, l'utilisateur est en mesure de sélectionner manuellement quels termes doivent être inclus à l'étape suivante.

3. *Changements dans l'ontologie :* ce module est chargé d'intégrer les termes, les données mises sous forme ontologique ou les éléments tirés d'ontologies dans l'ontologie à enrichir. Après la validation des fragments à réutiliser, il les intègre à l'ontologie. Pour savoir où mettre ces concepts dans l'ontologie, Evolva établit de nouvelles relations entre les termes extraits et les concepts de l'ontologie à enrichir en explorant diverses sources de connaissances générales, essentiellement d'autres ontologies. C'est à cette étape que Scarlet[16] est utilisé pour trouver une ou des relations pertinentes entre les concepts à ajouter et ceux de l'ontologie. Par exemple, pour relier deux concepts *Researcher* et *AcademicStaff*, Scarlet identifie les ontologies en ligne qui peuvent fournir des informations sur la façon dont ces deux concepts sont reliées. Puis il combine ces informations pour en déduire la relation à retenir.

   Pour cela, Evolva s'appuie sur deux stratégies de plus en plus sophistiquées de Scarlet. Celui-ci peut identifier si cette relation est définie (directement ou par transitivité) dans une ontologie en ligne unique ou non. Outre les relations de subsomption, *Scarlet* est également en mesure d'identifier des relations disjointes et des relations nommées.

4. *Validation de l'évolution :* l'application automatique des changements peut conduire à des incohérences et des inconsistances dans une ontologie. Par exemple, des données peuvent être dupliquées. Certaines inconsistances et incohérences dans l'ontologie modifiée sont détectées et gérées en utilisant un processus de raisonnement temporaire.

[16]C'est un moteur de découverte de relations sémantiques basé sur le moteur de recherche d'ontologies Watson

5. *Gestion de l'évolution :* Le but est de propager les changements vers les composants dépendants de l'ontologie (ex. d'autres ontologies ou des applications qui l'utilisent). Cette phase reste encore à l'état de proposition théorique et n'est pas implémentée.

2.4.2.3 ONTO-EVO^{A}L [Djedidi et Aufaure, 2008a], [Djedidi, 2009], [Djedidi et Aufaure, 2009] et [Djedidi et Aufaure, 2010]

L'outil ONTO-EVO^{A}L (*Ontology evolution evaluation*) s'appuie sur une modélisation à l'aide de patrons de gestion de changement CMP (Change Management Patterns). Ces patrons (cf. 3.2.4.3) spécifient des classes de changements, des classes d'incohérences et des classes d'alternatives de résolution (*patrons de changements*, *patrons des incohérences* et *patrons des alternatives*).

Ces patrons permettent de définir et de classer respectivement les *types des changements* gérés par le processus en se basant sur le modèle OWL DL, les *types des incohérences logiques* traitées par le processus en se référant aux contraintes logiques de OWL-DL et *les types des alternatives de résolution d'incohérence* pouvant être générées.

Un *Patron de Changement* peut correspondre à un Patron de Changement Elémentaire (exemple 6) ou à un *Patron de Changement* Complexe tel que spécifié dans la classification des changements OWL [Klein, 2004]. Un Patron de Changement Complexe est composé d'un ensemble de *patrons de Changement* Basiques et/ou Complexes. Prenons l'exemple d'un patron de changement basique correspondant à l'ajout d'une relation de sous-classe entre deux concepts donnés. Ce patron est décrit comme suit (tableau II-7).

Tableau II - 7 : Exemple de patron de changement basique selon [Djedidi *et al.*, 2009]

Type	Entités concernées	Arguments	Contraintes	Axiome OWL DL
P_Chgt_Bas_ Ajouter_ Sous_Classe	Classe. Classe	Sub_classID Super_classID	¬(Sub_classID disjointWith Super_classID)	SubClassOf (Sub_classID, Super_classID)

Pour illustrer une instanciation possible de ce patron (exemple 6), prenons l'exemple d'une ontologie OWL *O* définie par les axiomes suivants [Djedidi *et al.*, 2009] :

Exemple 6

{Animal ⊑ Faune-Flore, Plante ⊑ Faune-Flore, Herbivore ⊑ Animal, Carnivore ⊑ Animal, PlanteCarnivore ⊑ Plante, Plante ⊑ ¬Animal}.

Soit le changement *Ch1* définissant la classe *PlanteCarnivore* comme *sous-classe* de la classe *Animal*. L'instanciation du patron *Ch1* permet de spécifier la signature suivante (tableau II-8) :

Tableau II - 8 : Exemple d'instanciation d'un patron de changement élémentaire selon [Djedidi *et al.*, 2009]

Type	Entités concernées	Arguments	Contraintes	Axiomes OWL DL
P_Chgt_Bas_Ajouter_Sous_Classe	Classe, Classe	Animal, PlanteCarnivore	¬(PlanteCarnivore disjointWith Animal)	SubClassOf (PlanteCarnivore, Animal)

Le *patron d'incohérence* (exemple 7) permet de catégoriser les types d'incohérences logiques (qui peuvent être causées aussi bien par un changement complexe qu'un changement élémentaire) et de les expliciter afin de faciliter l'interprétation des résultats du raisonneur, d'orienter la localisation des axiomes responsables des incohérences et par là préparer leur correction [Djedidi, 2009].

Exemple 7

Reprenons l'exemple de changement élémentaire *Ch1* (exemple 6), l'instanciation du patron d'incohérence de disjonction est détectée et décrite comme suit (tableau II-9) :

Tableau II - 9 : Exemple d'instanciation d'un patron d'incohérence de disjonction, selon [Djedidi *et al.*, 2009]

Type	Entités Impliquées	Entités Concernées	Axiomes OWL DL concernés
P_Incons_Disj	Animal, Plante, PlanteCarnivore,	Animal, Plante	Plant ⊑ ¬Animal, PlanteCarnivore ⊑ Plant

Un *patron d'alternative* (exemple 8) représente un changement additionnel à appliquer pour lever une incohérence logique. Il est décrit comme un changement (basique ou complexe) et hérite des propriétés d'un patron de changement.

Exemple 8

Reprenant l'exemple de changement basique Ch1 (exemple 6), deux résolutions sont possibles pour résoudre l'incohérence de disjonction causée (exemple7). Une alternative de résolution (patron et son instanciation) est décrite comme suit (tableau II-10 et figure II-17) :

Tableau II - 10 : Exemple de patron d'alternative résolvant une disjonction, selon [Djedidi *et al.*, 2009]

P_Alt_Disj_Chgt_Bas_Ajout_Sous_Classe (*alt*)				
Entités Concernées	**Arguments**	**Pré-conditions**	**Contraintes**	**Axiomes OWL DL**
Classe, Classe	Sub_classID, Super_classID, Id1_cls_disj, Id2_cls_disj	SuperClass (Id1_cls_disj) ∩ SuperClass (Id2_cls_disj) ⊑ Super_classID	¬(Sub_classID disjointWith Super_classID)	SubClassOf (Sub_classID, Super_classID)

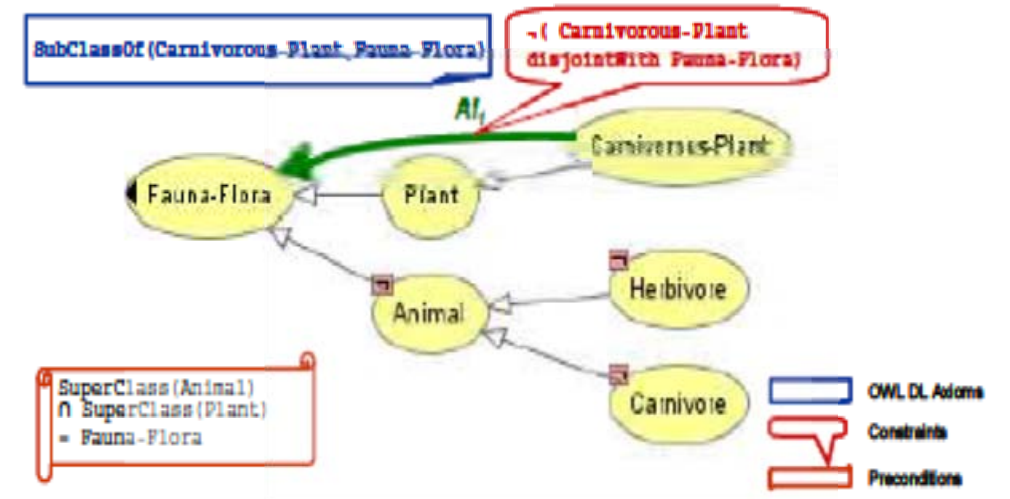


Figure II - 17 : Instanciation du patron de l'alternative selon [Djedidi, 2009]

L'outil permet également une activité d'évaluation. Une fois la consistance de l'ontologie vérifiée, l'outil passe à la phase d'évaluation dans laquelle il mesure l'impact du changement sur la qualité de l'ontologie.

Le modèle de qualité d'ontologie permet de guider la gestion des incohérences en évaluant l'impact des résolutions proposées sur le contenu et l'usage de l'ontologie à travers un ensemble de métriques quantitatives et ce, afin de choisir une résolution qui préserve la qualité de l'ontologie.

Les caractéristiques de qualité considérées correspondent aux aspects de contenu et usage de l'ontologie :

- Le contenu de l'ontologie inclut les dimensions structurelle et sémantique de l'ontologie. Son évaluation se base sur les critères de complexité, de cohésion, de conceptualisation et d'abstraction ;
- L'usage de l'ontologie tient compte de l'utilisabilité de l'ontologie. Il est évalué par les critères de complétude, et de compréhension.

2.5 Logiciels gérant l'évolution d'annotations sémantiques

2.5.1 TextViz [Reymonet *et al.*, 2007], [Reymonet *et al.*, 2009] et [Reymonet *et al.*, 2010]

Le projet Dynamo a donné lieu à un premier prototype TextViz 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. L'architecture du système TextViz est constituée de trois composantes essentiels, les ressources documentaires (corpus des textes), la ressource termino-ontologique et les annotations sémantiques. TextViz permet d'annoter les documents textuels du corpus ainsi que de visualiser ces annotations. Il permet de mettre à jour la ressource termino-ontologique et les annotations des documents basées sur cette RTO. Les textes sont supposés avoir tous la même structure et être conformes à la même DTD XML. Ils sont aussi supposés être du même

domaine, celui traité par l'ontologie. En cela, ils forment une collection homogène. Le processus d'annotation complet sera détaillé en section 3.5.

2.5.2 Éditeur ECCO et outil CoSWEM [Luong *et al.*, 2007], [Luong et Dieng-Kuntz, 2007] et [Luong, 2007]

Associé à la méthode présentée en section 2.2.4.4, l'éditeur d'ontologie ECCO[17] offre un environnement collaboratif et contextuel de construction d'ontologies sur lequel se base le système de gestion d'évolution CoSWEM (*Corporate Semantic Web Evolution Management*). ECCO couvre les différentes étapes du cycle de développement d'une ontologie en partant de la récupération de termes extraits par des outils de traitements de texte (NLP) pour créer un vocabulaire qui sera ensuite enrichi, hiérarchisé, puis formalisé en une ontologie OWL Lite. CoSWEM gère l'enrichissement de vocabulaires structurés (ontologie OWL) 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.

CoSWEM fournit des fonctions d'aide telles que la visualisation des changements effectués, la comparaison des versions de l'ontologie (même dans le cas où on ne sait pas quels sont les changements ontologiques effectués), la détection des annotations sémantiques inconsistantes à cause de modifications de l'ontologie, la correction de ces annotations inconsistantes, etc.

L'architecture de CoSWEM comprend trois composants principaux (figure II-18) : *composant utilisateur*, *composant intermédiaire* et *composant d'évolution*.

[17] http://www-sop.inria.fr/edelweiss/projects/ewok/publications/ecco.html

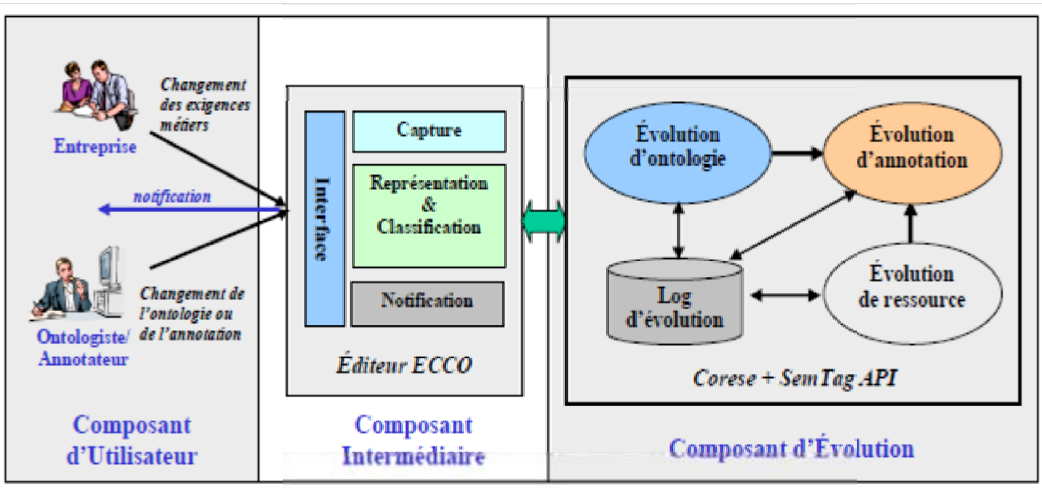


Figure II - 18 : Architecture du système CoSWEM, selon [Luong, 2007]

1. *Le Composant Utilisateur* permet de gérer les rôles des agents humains qui interagissent avec CoSWEM, tels que *l'Utilisateur* qui manipule les connaissances de son domaine métier ; *l'Ontologue* qui crée les ontologies, qui comprend le domaine métier et modélise ce domaine sous forme d'une représentation formelle et consensuelle ; l'*Annotateur* qui utilise les ontologies ainsi créées pour annoter les documents.

2. *Le Composant Intermédiaire* contient trois modules : le module interface s'occupe des aspects d'interaction avec les utilisateurs ; le module représentation des changements les représente de manière formelle sous forme de changements élémentaires et composites ; enfin, le module notification communique aux utilisateurs le résultat de la résolution des changements ainsi que les éléments qui ont été mis à jour.

3. *Le Composant d'Evolution*, composant principal du système, permet de résoudre des problèmes d'adéquation de l'ontologie et des annotations sémantiques. Ce composant est divisé en trois modules : - *Module Évolution d'ontologie* **:** il permet de travailler avec différentes versions de l'ontologie. Dans ce module, Luong se concentre sur la différence au niveau conceptuel entre des versions de l'ontologie ainsi que sur la

propagation des changements de l'ontologie vers les annotations sémantiques concernées. Dans ce module, les changements effectués sont conservés pour être réutilisés ultérieurement dans le module d'évolution des annotations. - *Module Évolution des annotations.* Après avoir reçu les informations sur les changements ontologiques propagés, ce module détectera les annotations sémantiques devenues obsolètes par rapport à la nouvelle version de l'ontologie à cause des changements effectués. Ces annotations sémantiques obsolètes sont ensuite corrigées selon le choix de solution de l'utilisateur pour assurer la consistance de toute la base d'annotations. - *Journal d'évolution :* un journal d'évolution permet de garder l'historique des changements, des éléments affectés, l'ordre d'exécution de ces changements, etc. Ces informations, une fois sauvegardées, sont transformées en annotations sémantiques, cela permet de les manipuler facilement pour découvrir les informations utiles concernant le processus d'évolution.

Développés dans le cadre du projet E-WOK_HUB (*Environmental Web Ontology Knowledge Hub*), ECCO et CoSWEM sont deux logiciels dédiés respectivement à l'édition et à l'évolution d'ontologie et d'annotations sémantiques. Mais malheureusement, les différentes stratégies de résolution déjà définies ne sont pas toutes intégrées dans l'éditeur d'ontologie ECCO.

2.6 Discussion

Nous résumons les fonctionnalités principales de chaque approche présentée ainsi que les outils proposés dans un tableau comparatif (tableau II-11). La plupart des outils traitant de la gestion des changements d'ontologie se concentrent sur le suivi des changements et sur la gestion des différentes versions d'ontologies (par exemple, OntoView [Klein *et al.*, 2002],

PROMPTdiff [Noy *et al.*, 2003], [Ognyanov et Kiryakov, 2002] et OntoAnalyzer [Rogozan, 2008]). D'autres travaux mettent en œuvre un ensemble de fonctionnalités pour assister de nombreuses tâches du processus d'évolution d'ontologies comme KAON [Stojanovic, 2004]. Plus rares sont les travaux qui assistent la propagation des changements vers les artefacts dépendants (à savoir les ontologies, les référencements sémantiques, les applications) [Rogozan, 2008] [Luong, 2007]. Par ailleurs, il n'y a que ces deux méthodes qui assurent une propagation des évolutions vers les annotations sémantiques de textes. Un autre aspect qui, à notre connaissance, n'a jamais été abordé dans le processus de gestion d'évolution d'ontologie est la prise en compte de l'aspect lexical ou terminologique, à savoir des termes dénotant les concepts de l'ontologie. Le fait de prendre en compte les termes et les annotations sémantiques utilisant l'aspect lexical renouvelle plusieurs questions.

Tableau II - 11 : Comparaison des approches et outils d'évolution d'ontologie et d'annotations

Approches		[Stojanovic, 2004]	[Klein, 2004]	[Plessers *et al.*, 2007]	[Castano, 2006] [Castano *et al.*, 2007]	[Luong, 2007]	[Rogozan, 2008]	[Djedidi, 2009]
Evolution d'ontologie : fonctionnalités	**Prototype**	Suite d'outils KAON	OntoView PROMPTdiff	Deux Plug-ins de Protégé : Générateur d'historique de versions + Détecteur de changements	Ontology Evolution Toolkit, un composant du prototype BOEMIE	CoSWEM + ECCO	Onto-Analyseur	Onto-Evo[a]l
	Processus d'évolution	Oui (Processus Global)	Pas de processus	Partiel (certaines phases)	Partiel	Partiel	Partiel	Oui (Processus Global)
	Langage d'ontologie	KAON	OWL	OWL DL	OWL	RDF(S)	OWL	OWL
	Spécification des changements	Construction d'une ontologie d'évolution	Construction d'une ontologie d'opérations de changement et de transformations entre versions	Langage de définition de changement CDL	Utilisation de ABox	Construction d'une ontologie d'évolution	Construction d'une ontologie des changements	Utilisation des patrons de changements
	Détection du changement	- - -	Relations conceptuelles et d'évolution entre versions	Utilisation du modèle logique de données pour récupérer les traces de changement	- - -	récupération de traces de changement	- - -	Utilisation d'un raisonneur

Evolution d'ontologie : fonctionnalités	**Vérification de consistance**	localisation de la sous-ontologie incohérente	compatibilité entre différentes versions	Algorithme de localisation utilisant deux types d'arbres de localisation (tracking trees)	Oui : Pour le peuplement et l'enrichissement	modèle de cohérence des annotations	compatibilité entre deux versions	Oui : avec le raisonneur Pellet
	Résolution d'inconsistance	stratégies de résolution par type de changements	Dériver des changements additionnels	Proposition de règles de correction	- - -	stratégies de résolution par type de changements	stratégies de résolution pour chaque type de changement	Proposition de patrons d'alternatives (résolutions)
	Propagation des changements dans l'ontologie	Diffusion vers les artefacts avec différentes synchronisation	propagation partielle aux données	- - -	Mappings des connaissances entre l'ontologie et les sources de connaissances externes	Annotations sémantiques	Oui	Oui
	Log d'évolution	Oui	Oui	Historique des versions de chaque entité de l'ontologie	Oui	Phase non vérifié	Oui	Oui
	Gestion des Versions	- - -	Comparaison des versions	- - -	- - -	Comparaison des versions	Comparaison des versions	Comparaison des versions
	Propagation du changement aux annotations sémantiques	- - -	- - -	- - -	- - -	Oui	Oui	- - -
	Prototype	CREAM	- - -	- - -	- - -	CoSWEM + ECCO	Semantic Annotation Modifier	- - -
	Processus d'évolution	Oui	- - -	- - -	- - -	Oui	Oui	- - -

Evolution d'annotations : fonctionnalités	**Analyse des effets de changements ontologiques**	- - -	- - -	- - -	- - -	Oui : vérification des contraintes de consistance de l'ontologie pour trouver des annotations inconsistantes	Oui : analyse fournie par l'outil SAM	- - -
	Détection des annotations inconsistantes	- - -	- - -	- - -	- - -	Oui : règles de détection sur les aspects du concept, de la propriété….	Partiel	- - -
	Rectification des annotations inconsistantes	- - -	- - -	- - -	- - -	Stratégies de résolution des annotations	Partiel : SAM : suggestion des modifications du référencement sémantique	- - -

2.7 Conclusion

Dans ce chapitre, nous avons présenté une analyse aussi exhaustive que possible de l'état de l'art des sujets traités dans cette thèse, et nous avons discuté leurs limites. Les différentes approches d'évolution d'ontologie proposent des fonctionnalités similaires : la représentation des changements, leurs effets et leur propagation vers les artéfacts dépendants. A ces approches sont associés des logiciels qui assistent l'évolution d'ontologie et, pour certains, qui gèrent aussi l'évolution des annotations sémantiques. Enfin, une étude comparative des approches d'évolutions d'ontologie a été présentée.

Dans le chapitre suivant, nous présentons le projet sur lequel la suite de notre travail va reposer, à savoir le projet DYNAMO, en détaillant le caractère novateur du projet ainsi que le méta-modèle de la ressource termino-ontologique utilisée.

Deuxième partie

EvOnto – UNE APPROCHE POUR L'ÉVOLUTION DES RTO ET DES ANNOTATIONS SÉMANTIQUES

Tout au long des quatre chapitres qui composent cette partie, nous présentons notre contribution qui vise à fournir une méthodologie pour gérer l'évolution conjointe de ressources termino-ontologiques et des annotations sémantiques posées sur des documents textuels. Le premier chapitre détaille le contexte de notre travail, à savoir le projet DYNAMO. Le deuxième présente notre méthodologie d'évolution EvOnto et son principe de fonctionnement. Le processus d'évolution de la ressource termino-ontologique avec les différents scénarios seront présentés dans le troisième chapitre. Le dernier chapitre est consacré à la présentation du processus d'évolution des annotations sémantiques et la vérification de la qualité de ces annotations.

CHAPITRE III

Le projet DYNAMO : ontologies dynamiques pour l'annotation sémantique

Sommaire

3.1 Introduction

Ce chapitre a pour objectif de présenter le projet dans lequel notre travail a été réalisé à savoir le projet DYNAMO[18]. La première partie de ce chapitre présente le contexte du projet et son caractère novateur. La deuxième partie présente le méta-modèle de la RTO utilisé dans Dynamo. La troisième partie détaille le processus d'annotation sémantique, avant d'en finir avec l'éditeur d'ontologie et d'annotation TextViz.

3.2 Le projet Dynamo

3.2.1 Contexte

L'objectif principal du projet DYNAMO 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 annotation sémantique facilitant la recherche des documents souhaités par un utilisateur.

L'architecture du système Dynamo, présentée dans la figure III-1, repose sur trois ressources essentielles : la collection de documents, la RTO et les annotations sémantiques.

- *La collection de documents* est constituée d'un ensemble de documents textuels (éventuellement structurés en parties). C'est au sein de cette collection que les documents pertinents seront recherchés.

- *La ressource termino-ontologique* (RTO) représente pour chaque domaine d'application les termes, concepts et relations du domaine en suivant le méta-modèle présenté dans la section 3.3 qui permet de les manipuler en utilisant toute

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

l'expressivité d'OWL. Elle contient donc une composante conceptuelle, prenant la forme d'une ontologie, et une composante lexicale, terminologie du domaine.

- *Les annotations sémantiques* des documents sont créées automatiquement par l'outil d'annotation de Dynamo, mais elles peuvent être modifiées ou refusées par un spécialiste du domaine et de la tâche concernés.

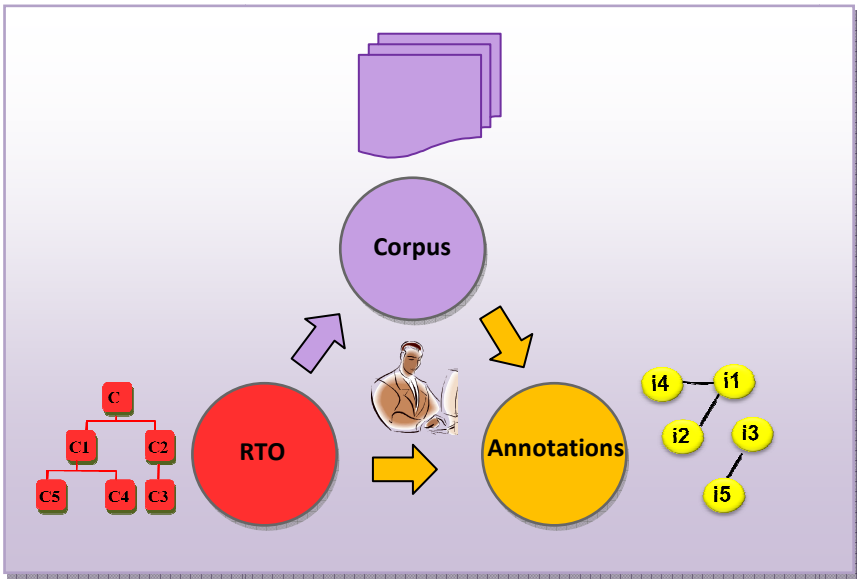


Figure III - 1 : Ressources utilisées dans le cadre du Dynamo

Le projet DYNAMO se focalise donc sur la maintenance cohérente des trois briques du processus de recherche d'information lié à une activité.

Dans ce projet, les corpus documentaires fournis par les partenaires sont issus de trois domaines particuliers : la recherche en archéologie des techniques – corpus ARKEOTEK du laboratoire Préhistoire et Technologie –, le diagnostic de pannes automobiles – ACTIA – et le diagnostic de défauts logiciels – ARTAL Technologies–

3.2.2 Caractère novateur

Le caractère innovant du projet DYNAMO réside tout d'abord dans l'utilisation de ressources termino-ontologiques. L'utilisation d'ontologies à composantes terminologiques vise en effet à proposer un meilleur support pour l'annotation des documents et la spécification des requêtes utilisateurs : l'ajout d'une composante lexicale ou terminologique au modèle conceptuel d'une ontologie permet de mieux articuler textes et ontologies via l'expression terminologique des concepts dans les textes.

3.2.3 Les applications dans Dynamo

Partenaire ARTAL Technologies

Le domaine d'application d'ARTAL concerne le diagnostic de défauts logiciels des projets informatiques. Les fiches d'ARTAL permettent de décrire un dysfonctionnement ou une non conformité d'un logiciel livré, d'exprimer une demande d'ajout de nouvelles fonctionnalités, etc. Une fiche d'anomalie de logiciel est un fichier XML contenant deux champs : (i) *Résumé* qui résume l'anomalie signalée par l'utilisateur du logiciel ou l'ingénieur et (ii) *Description* pour décrire en détail l'anomalie.

Le corpus d'Artal contient uniquement des documents en anglais. Il est composé de 800 fiches d'anomalies. Seul le champ résumé qui présente brièvement la description d'une anomalie ou d'une demande d'évolution est conservé.

Le noyau de l'ontologie d'Artal est basé sur quatre concepts : *Default*, *Function*, *Component* et *Trigger_Event*.

- *Default* : un défaut désigne une panne ou une anomalie ;
- *Function :* une fonction est concernée par un événement ;

- *Component* : un composant désigne une partie identifiée d'une interface ou d'une information ;
- *Trigger_Event* : un événement est une action qui peut déclencher un défaut.

Ces concepts sont reliés par les relations suivantes :

- Chaque défaut affecte un composant ;
- Chaque défaut est localisé dans un composant (pas forcément celui affecté) ;
- Chaque défaut est causé par un événement ;
- Chaque événement concerne un composant (pas forcément celui affecté ou celui dans lequel le défaut a été localisé) ;
- Chaque événement concerne une fonction ;

Partenaire ACTIA

Le domaine d'application d'ACTIA concerne le diagnostic d'incidents fournis par des constructeurs automobiles ou des réseaux indépendants.

Le corpus d'ACTIA contient des documents en français et anglais. Il est composé de 4500 fiches d'anomalies. Chaque fiche est un fichier XML formé de 5 champs principaux (i) *applicability* décrit le type de voiture concerné, (ii) *service* qui présente le service ayant détecté l'anomalie, (iii) *symptoms* qui décrit les problèmes rencontrés par le garagiste, (iv) *diagnosis* est le diagnostic du problème, qui correspond aux causes du comportement anormal constaté, et (v) *remedy* qui décrit une solution au problème.

Le noyau de l'ontologie d'ACTIA est basé sur cinq concepts (figure III-2) : symptôme, problème, prestation, composant et contexte.

- un symptôme est défini par un problème et une prestation du véhicule.

- un problème affecte une ou plusieurs prestations et peut survenir dans certains contextes.
- une prestation correspond à une fonctionnalité rendue à l'usager par son véhicule.
- un composant est une pièce ou un ensemble de pièces mécaniques, électriques et/ou électroniques ; il participe à la réalisation d'une prestation.
- Les problèmes constatés par l'utilisateur et/ou le garagiste et les contextes dans lesquels ils sont constatés.

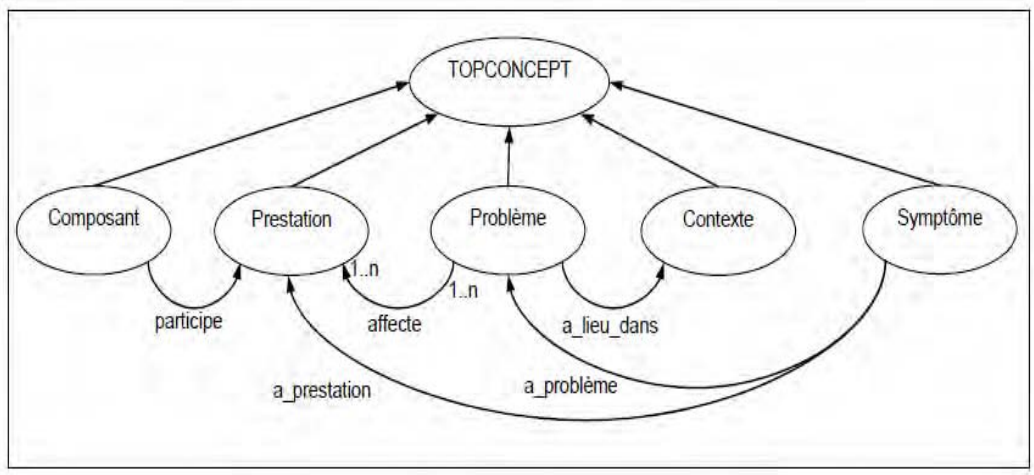


Figure III - 2 : Noyau de la RTO du partenaire ACTIA

Partenaire ARKEOTEK

Le domaine d'application d'ARKEOTEK concerne des articles scientifiques en archéologie.

Le corpus d'ARKEOTEK est un ensemble de documents composés d'un ensemble de règles tirées d'articles scientifiques en archéologie. Il est composé de 300 règles. Une règle est composée de deux parties : prémisse et conclusion. Elle est décrite dans un fichier XML formé de 3 champs :

- si contient les prémisses d'une règle ;

- alors contient une conclusion ou une interprétation d'une règle ;
- proposition décrit les observations ou les analyses réalisées par l'archéologue.

Le noyau de l'ontologie d'ARKEOTEK est basé sur deux concepts de base à savoir, objets et interprétations.

3.3 Le Méta-modèle de RTO dans Dynamo

Dans DYNAMO, nous nous intéressons à l'utilisation et l'évolution de Ressources Termino-Ontologiques. Dans le cadre du projet, les ressources termino-ontologiques suivent un méta-modèle comportant non seulement une représentation des concepts du domaine et de leurs relations, mais aussi une représentation dédiée aux termes associés (termes désignant les concepts). Ainsi, la manifestation de chaque concept dans les documents est représentée dans l'ontologie sous forme de lexicalisations spécifiques. Ce méta-modèle est construit à partir de deux méta-classes.

3.3.1 Les méta-classes du méta-modèle

Afin de représenter les deux composantes d'une RTO et pour être conforme avec le standard le plus populaire pour représenter les ontologies, à savoir OWL[19], [Reymonet *et al.*, 2007a] ont défini un premier méta-modèle en OWL-DL, capable de représenter des termes. Cependant, certains choix opérationnels avaient rendu le modèle peu satisfaisant du point de vue théorique [Reymonet *et al.*, 2007b]. C'est pourquoi les auteurs ont apporté un certain nombre de modifications en se basant sur les primitives de méta-modélisation disponibles en OWL-Full. Le plus haut niveau d'abstraction du modèle définit deux sous-classes de la classe *owl:Class* : la classe *Concept* et la classe *Term* (figure III-3). Le méta-modèle permet ainsi de faire la distinction entre un objet de type concept, représenté par une sous-classe de

[19] http://www.w3.org/TR/owl-features/

DomainThing instance de Concept, et un objet de type terme, représenté par une sous-classe de TermOccurrence qui est par ailleurs instance de la méta-classe *Term*. Nous décrivons dans les sections suivantes chacun de ces objets ainsi que les liens qui existent entre eux.

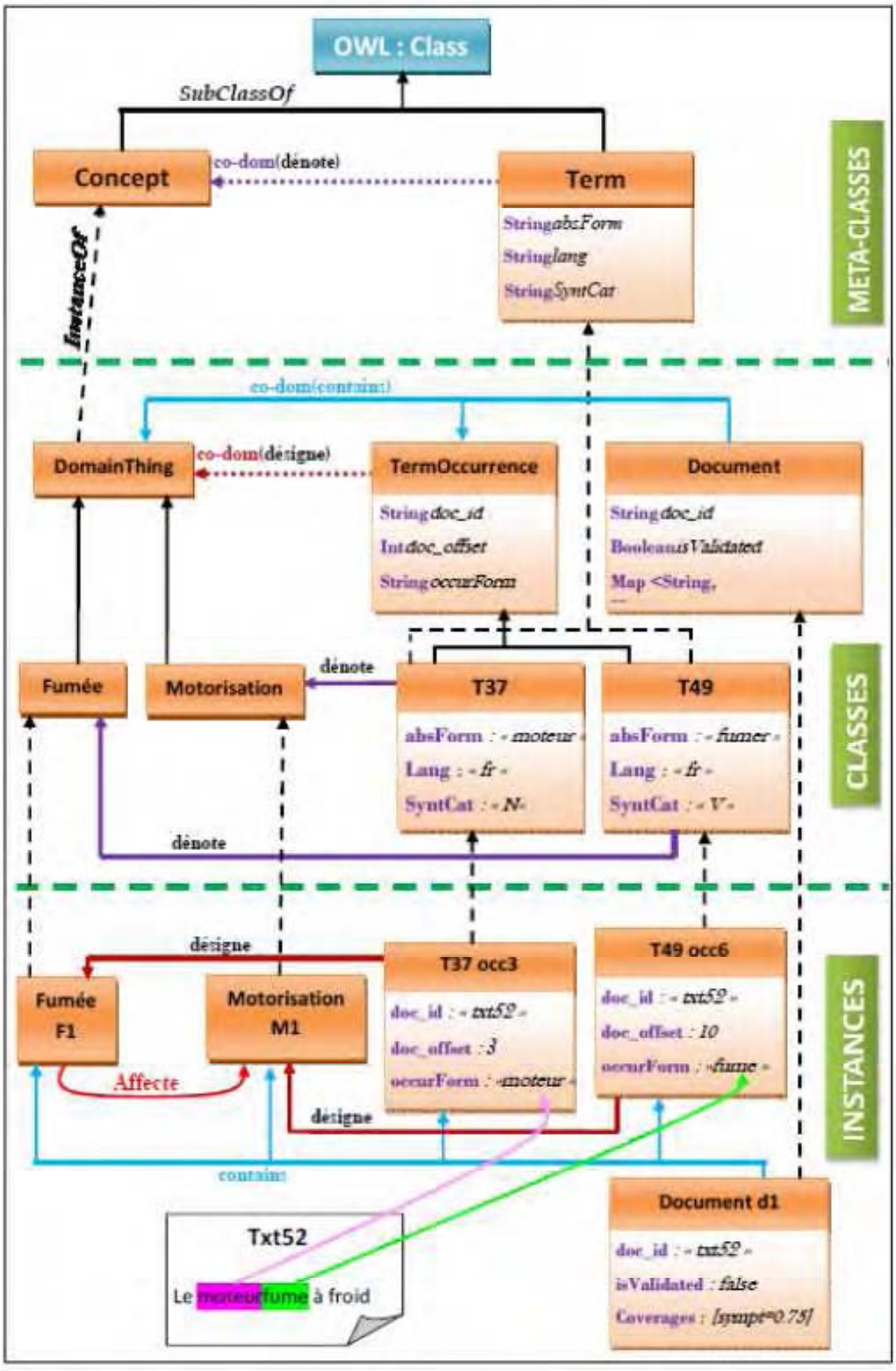


Figure III - 3 : Modèle de la RTO [Reymonet *et al.*, 2009]

3.3.2 La représentation d'un concept

En OWL, les éléments de base d'une ontologie sont matérialisés de la façon suivante :

- les concepts de l'ontologie sous forme de sous-classes de *owl:Thing*, instances de *owl:Class*,
- les attributs de concepts sous forme d'instances de *owl:DatatypeProperty*,
- les relations entre concepts sous forme d'instances de *owl:ObjectProperty*

Dans la partie ontologique de la RTO de Dynamo, les concepts sont, de manière classique pour une ontologie, organisés dans une hiérarchie selon des relations *subClassOf* (classe/sous-classe), et décrits par des relations sémantiques autres que *subClassOf* (dites relations transverses). Dans le méta-modèle, chaque concept est représenté par une sous-classe de la classe *DomainThing.*

3.3.3 La représentation d'un terme

Dans une RTO de DYNAMO, les termes qui sont associés à un concept sont utilisés pour produire les annotations sémantiques décrivant le contenu des textes. L'annotation est posée lors de la reconnaissance dans les textes des termes qui dénotent des concepts de la composante ontologique de la RTO. De façon à pouvoir revenir aux textes pour lesquels une annotation a été produite, le méta-modèle permet de représenter à la fois les termes et leurs occurrences dans les textes.

Selon le modèle de RTO de [Reymonet, 2008] un objet de type Terme est une sous-classe de la classe *TermOccurrence et* instance de la méta-classe *Term*. Ce choix de modélisation s'avère pratique car il permet d'assimiler une occurrence de terme à une instance de la classe représentant le terme. Dans l'exemple de la figure III-3, les termes *T37* et *T49* ont comme propriétés les éléments suivants : la forme sous laquelle le terme est repéré et leur catégorie syntaxique et morphologique sous forme de *owl:DatatypeProperty*.

- La catégorie syntaxique (SyntCat)
  - T37 : SyntCat =V
  - T49 : SyntCat =N
- La forme sous laquelle le terme est repéré (absForm)
  - T37 : absForm =moteur
  - T49 : absForm =fume

Pour permettre une approche multilingue des ressources termino-ontologiques dans laquelle plusieurs lexiques de langues différentes seraient associés à une même ontologie, les termes ont également un attribut indiquant leur langue d'origine [Reymonet, 2008]. Dans la figure III-3, la langue des termes *T37* et *T49* est le français «Fr».

3.3.4 Représentation d'une occurrence de terme

Chaque instance de terme est une occurrence de ce terme dans un texte (*TermOccurrence*). La classe *TermOccurrence* joue pour les termes le même rôle que pour la classe *DomainThing* pour les concepts. Dans l'exemple de la figure III-3, les termes *T37* et *T49* ont pour occurrences respectivement *occ3* et *occ6*. Les propriétés d'une occurrence (*termOccurrence*) présentent sa localisation dans le corpus (un identifiant de texte et un identifiant de position). Dans la figure III-3 les occurrences *T37*occ3 et *T49occ6* ont pour attributs :

- Un identifiant de texte (doc_id)
  - T37occ3 : doc_id = txt52
  - T49occ6 : doc_id = txt52
- Un identifiant de position relative dans le document (doc_offset)
  - T37occ3 : doc_offset = 3
  - T49occ6 : doc_offset = 10

3.3.5 La modélisation des liens terme-concept

Selon le méta-modèle de la RTO, les concepts et les termes sont des instances de deux méta-classes adaptées de *owl:Class*. Pour les relier, Reymonet [Reymonet, 2008] a défini la propriété *dénote* ayant pour domaine la classe *Term* T et pour co-domaine la classe *Concept* C. La relation *dénote* permet de relier un ou plusieurs termes avec un ou plusieurs concepts. De plus, chaque instance de terme (en fait de TermOccurrence) est reliée à une instance d'un concept par la relation désigne. Dans l'exemple de la figure III-3, l'occurrence du terme fumer dans le document *txt52* est représentée comme l'instance*T49occ6* et désigne l'instance du concept Fumée à savoir l'instance Fumée *f1*. Une autre relation *contains* ajoutée au modèle de la RTO permet de lier un document aux occurrences de termes et aux instances des concepts.

3.4 Le processus d'annotation sémantique dans Dynamo

Comme nous l'avons vu, le méta-modèle permet de représenter les liens entre la RTO et les documents considérés, et constitue ainsi un méta-modèle adapté pour représenter des annotations sémantiques faites à l'aide d'une RTO. Dans cette section, nous détaillons ce processus d'annotation sémantique qui exploite les textes et la RTO pour produire des annotations, tel qu'il est défini dans [Reymonet, 2008].

3.4.1 Représentation des annotations

Les annotations dans Dynamo sont dites sémantiques dans le sens où elles s'appuient sur les ressources termino-ontologiques du domaine correspondant à chacune des applications. Elles sont composées d'éléments représentés dans la RTO et dont on trouve une trace linguistique dans le document ou la partie de document à annoter. Dans Dynamo, on définit un graphe d'annotation comme un graphe dont les nœuds sont les instances anonymes des concepts retrouvés dans le texte annoté, et les relations sont des relations entre ces instances

conformes à la RTO. Le processus d'annotation génère aussi des instances anonymes de la classe TermOccurrence, qui permettent de localiser précisément les annotations. Ces instances sont reliées aux instances de *DomainThing* par la relation *designe.*

L'idée principale du processus consiste dans un premier temps à retrouver la trace de certains termes de la RTO dans chaque document et à construire les occurrences correspondantes, puis à associer ces occurrences à de nouvelles instances de concepts ; enfin, les instances de concepts doivent être reliées selon les relations sémantiques existant dans la RTO [Thomas *et al.*, 2010]. Nous faisons ci-après une description plus fine de chacune des étapes du processus d'annotations.

3.4.2 Pose des annotations

Les annotations sémantiques visent à identifier en corpus des fragments de texte (un mot ou une séquence de mots susceptibles d'être retenus comme terme) qui dénotent des concepts et à les annoter en conséquence avec des instances, de ces concepts créées dynamiquement à ce moment-là, (on les désigne dans ce cas sous le terme d'entités anonymes). L'ensemble de ces instances et des relations sémantiques identifiées dans le document forme un ou plusieurs graphes qui constituent l'annotation sémantique.

De façon plus détaillée, le processus d'annotation sémantique des documents se déroule en deux étapes :

Première étape - *projection de la RTO sur les documents* –Cette phase consiste à utiliser l'ensemble des termes constitutifs de la ressource afin de les projeter sur les documents.

Deuxième étape : *identification des concepts.* A l'issue de cette phase de projection de la terminologie sur les documents, le système peut alors déterminer quels concepts sont mis en

jeu dans chaque document à partir des relations définies dans la RTO entre ces concepts et les termes identifiés dans ce document lors de la projection.

Une fois ce processus terminé, nous obtenons donc un ensemble d'instances de termes (leurs occurrences) ainsi que les instances de concepts qui leur correspondent. Le document se retrouve donc annoté avec ces occurrences des termes retrouvées ainsi que par les instances des concepts que dénotent ces termes. Reste à déterminer les liens entre ces instances de manière à créer les graphes d'annotation. Cette partie sera détaillée dans la section suivante.

3.4.3 Génération des graphes d'annotations

La dernière étape du processus automatique d'annotation sémantique consiste, à construire un ou plusieurs graphes d'annotation (i.e. à retrouver les relations sémantiques existant entre deux instances de concepts) à partir de l'ensemble des instances de concepts associées à un document. Dans la figure III-3, la relation *affecte* entre *Fumée F1*, instance de concept *Fumée*, et *Motorisation M1*, instance de concept *Motorisation* est un exemple de relation.

Pour construire les graphes d'annotations, le système doit créer des relations entre les instances de concepts identifiées. Contrairement à l'identification des instances de concepts, qui s'appuie sur leur lexicalisation, aucun indice linguistique n'est ici utilisé. En effet, les textes ne comportent pas souvent dans nos applications de formulation explicite des relations. Pour trouver ces relations entre instances de concepts, différentes heuristiques ont été définies [Thomas *et al.,* 2010]. Certaines sont basées sur la proximité dans le document des occurrences de termes désignant les deux instances de concepts que l'on cherche à mettre en relation. D'autres heuristiques sont basées sur la cardinalité des relations, comme la suivante : « *pour chacune des instances de concept découverte dans le document, pour chacune des relations ayant pour domaine ce concept et ayant une cardinalité minimale de 1, nous créons*

une instance du concept co-domaine. Les restrictions sur les relations prennent ici toute leur importance pour créer des instances aussi spécifiques que possible. »

3.5 TextViz : éditeur d'ontologie et d'annotation

Le projet Dynamo a donné lieu à un prototype TextViz [Reymonet et al, 2009] qui a été développé comme un plug-in de Protégé. Il bénéficie ainsi des interfaces de Protégé qui ont été enrichies pour répondre aux besoins du projet. Ainsi, TextViz permet la gestion de RTO (les interfaces de Protégé ont été adaptées de manière à gérer les termes en plus des concepts et relations). TextViz permet également de visualiser les documents textuels du corpus ainsi que leurs annotations.

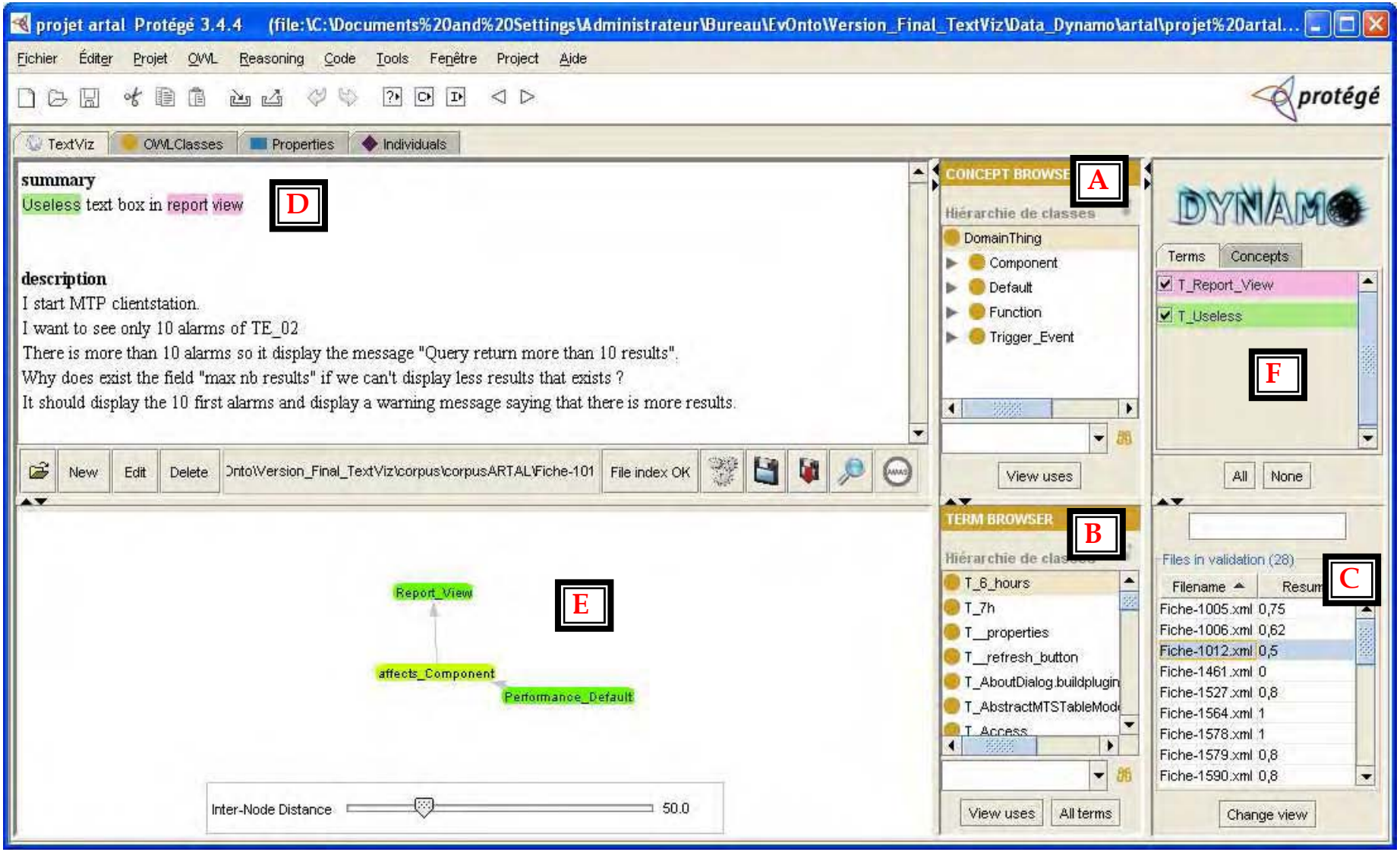


Figure III - 4 : Copie de l'écran principal de TextViz

L'interface TextViz contient six panneaux (figure III-4) offrant une vue globale sur la RTO ainsi que les annotations des documents.

A **CONCEPT BROWSER** : affiche la hiérarchie des concepts d'une RTO. Si l'ontologue sélectionne un concept dans le panneau CONCEPT BROWSER, les termes associés à ce concept sont affichés dans le panneau TERM BROWSER.

B **TERM BROWSER** : affiche la liste des termes d'une RTO. Si l'ontologue sélectionne un terme dans le panneau TERM BROWSER, le concept dénoté est alors présenté dans le panneau CONCEPT BROWSER.

C **Files in validation**: affiche la liste des documents annotés mais qui ne sont pas encore validés par l'ontologue. Pour chaque document, un score évalue le pourcentage d'éléments reconnus (une bonne annotation est celle qui permet de couvrir le plus possible de termes dans une partie considérée du document). Pour afficher le contenu textuel d'un document, l'ontologue doit cliquer sur le nom d'un document dans le panneau files in validation. Le contenu textuel du document sélectionné sera affiché dans le panneau D.

D Le panneau D contient le contenu textuel d'un document sélectionné dans le panneau C (files in validation).

E Le panneau E affiche le graphe d'annotation composé des instances de concepts et des relations identifiées dans la fiche sélectionnée.

F Le panneau F affiche la liste des termes d'un document sélectionné dans le panneau C et la liste des concepts associés.

Tel quel, TextViz permet de traiter quelques opérations simples d'édition telles que :

- l'ajout de terme, qui consiste à créer un nouveau terme dans la RTO et à poser le lien de dénotation entre ce terme et un concept qu'il dénote (voir figure III-5).
- la suppression d'un terme, qui entraîne nécessairement la suppression de ses occurrences ainsi que le (les) lien(s) de dénotation le reliant à un (des) concept(s).
- l'ajout d'un lien de dénotation d'un terme vers un concept (tous deux préexistants).
- la modification d'un lien de dénotation, qui consiste à changer le concept dénoté par un terme (le lien est par définition lié au terme, celui-ci ne peut donc être modifié) ; cette opération peut être assimilée à une suppression du lien suivie d'un ajout.
- la suppression d'un lien de dénotation suite à laquelle le terme et le concept originellement reliés continuent à exister indépendamment de la disparition de la relation.
- l'ajout de concept, qui peut comprendre une phase d'association à un ou plusieurs termes existants.
- la suppression de concept, envisagée ici comme n'influant pas sur l'existence d'autres concepts mais qui entraîne la disparition de tous les liens de dénotation pointant vers le concept.

L'application d'une opération de changement sur la RTO a des conséquences sur la base d'annotations. Pour traiter ses conséquences, la seule solution proposé dans l'outil TextViz est de ré-annoter l'ensemble des documents.

TextViz s'intègre dans un objectif global de recherche d'information. Notons que la recherche d'information utilise, entre autres, le même processus d'annotation pour traiter les requêtes et les représenter de manière comparable aux annotations des documents.

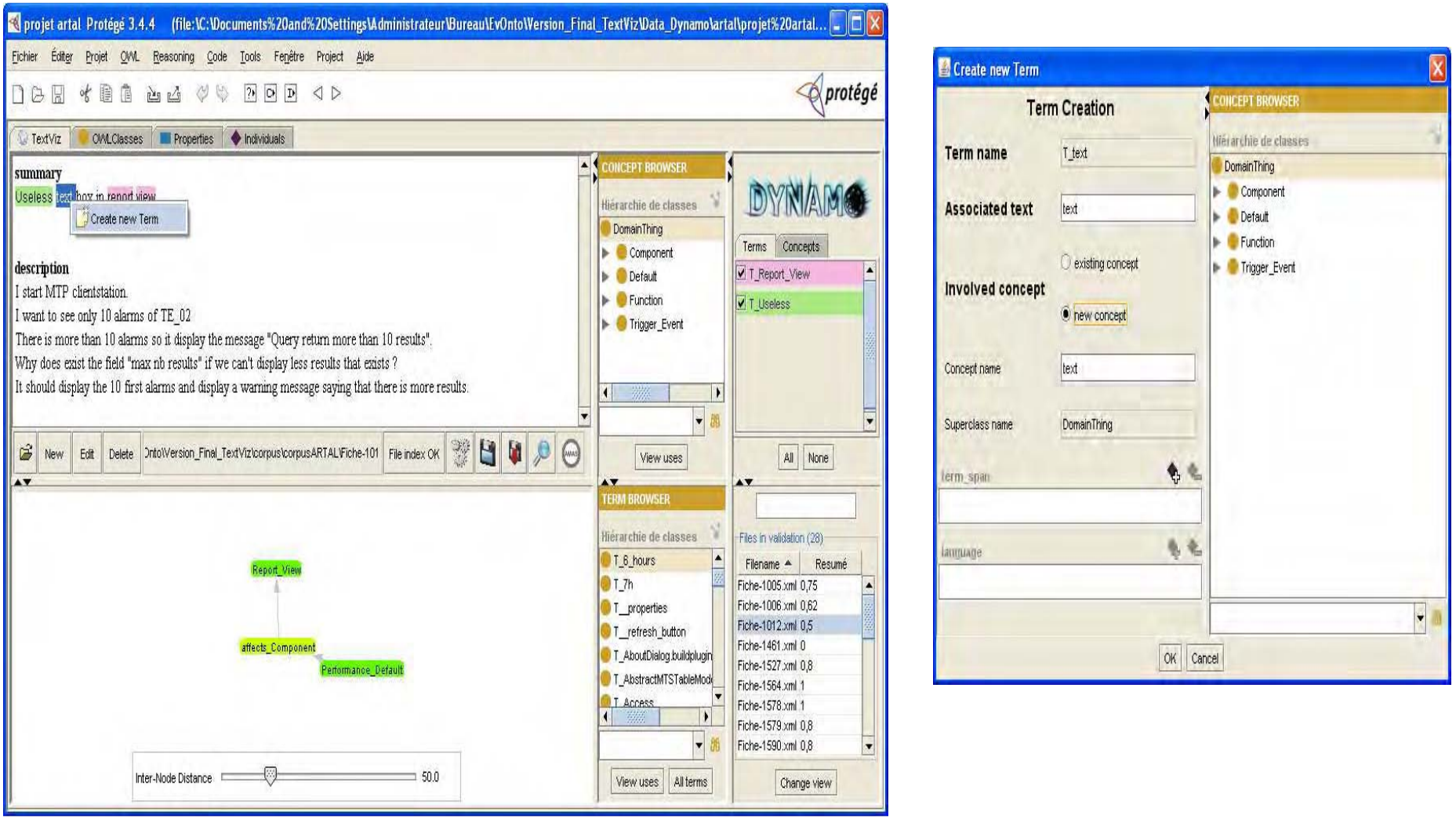


Figure III - 5 : Ajout d'un nouveau terme et du nouveau concept qu'il dénote dans la RTO avec TextViz

L'outil TextViz a été spécifiquement conçu pour permettre l'édition et la construction d'ontologie et d'annotations sémantiques des documents. TextViz permet l'enregistrement des opérations accomplies par les utilisateurs lors de la modification d'une RTO. Cependant, ces opérations ne sont (a) ni classifiées en fonction d'un modèle de changements (ex. les changements élémentaires et complexes) ; (b) ni documentées complètement, en précisant par exemple les paramètres d'entrée et/ou de sortie de toute opération accomplie lors du processus de modification de la RTO ; (c) ni enregistrées dans des versions antérieures (journalisation des changements) pour garder une trace de modification et en vue d'une récupération future.

La forte limitation est que l'outil TextViz ne traite pas les changements complexes (un changement complexe est un changement qui affecte deux ou plusieurs entités de la RTO). Bien qu'il soit vrai que les changements élémentaires permettraient l'édition de tout type de changement complexe, il est aussi vrai qu'il est beaucoup plus avantageux d'utiliser directement des changements complexes [Klein et Noy, 2003]. En effet, les changements complexes facilitent : (a) la validation globale et collective de différentes opérations de l'évolution d'une RTO; (b) la visualisation d'une seule opération de changement complexe (ex. SplitClasses), contrairement à une somme de changements élémentaires (ex. un seul DeleteClass et plusieurs AddClass), ce qui permet aux utilisateurs de comprendre plus aisément la modification d'une RTO; (c) l'analyse plus claire des conséquences sur la conceptualisation de la RTO ainsi que celle des effets sur l'annotation sémantique de ressources. Par exemple, si deux classes C1 et C2ont été fusionnées à l'aide d'un changement MergeClasses et qu'un certain nombre de sous-classes des classes C1 et C2 ont été déplacées à la classe C3, résultante de la fusion, on peut déduire que les ressources annotées par les classes C1 et C2 peuvent maintenant être annotées par la classe C3, puisque celle-ci contient les instances des classes C1 et C2.

Dans le but d'analyser les effets des changements sur les annotations sémantiques des ressources et préserver l'intégrité de ces annotations après l'évolution d'une RTO, l''utilisation des changements complexes est d'autant plus importante. Il ne suffit pas d'utiliser uniquement des opérations simples, mais il faudrait enrichir l'ensemble de ces opérations par l'utilisation des changements complexes.

Nous situons justement nos travaux de thèse dans ce contexte particulier, les limites du prototype proposé nous ont permis de dégager des questions fondamentales sur lesquelles se basera la suite de la thèse. Par exemple, doit-on s'intéresser seulement aux changements simples, comme l'ajout ou l'effacement des composants d'une RTO, ou également aux changements plus complexes, comme la fusion ou la division, qui possèdent une sémantique plus riche pour exprimer l'évolution ? Les utilisateurs d'une ontologie doivent-ils ou non être informés de changements conséquences de l'évolution ? Si oui, par quel moyen les identifier et comment les présenter pour en faciliter la compréhension des utilisateurs ? Quels choix doivent être offerts à l'utilisateur ? Ou encore, comment s'assurer qu'après l'évolution d'une RTO, les ressources annotées restent accessibles et cohérentes avec la RTO ?

3.6 Conclusion

Ce chapitre nous a permis de mettre en avant le contexte de notre travail à savoir le projet DYNAMO. Partant du constat que les différents modèles existants ne permettaient pas de représenter correctement la partie terminologique associée à une ontologie, les travaux de [Reymonet, 2008] proposent un méta-modèle de RTO en OWL-FULL qui réifie les termes et leur accorde un statut similaire à celui des concepts. Ce méta-modèle permet notamment d'associer aux termes un nombre quelconque d'informations utiles, comme par exemple la liste de leurs occurrences en corpus. Ce méta-modèle intègre également la notion de

document et permet ainsi la représentation d'annotations sémantiques de documents à l'aide d'une RTO.

L'analyse du logiciel TextViz a mis en évidence l'absence de fonctionnalités importantes pour gérer l'évolution des RTOs. Dans cette thèse, nous nous donnons comme but de trouver des solutions pour répondre à ce manque.

Dans les chapitres suivants, nous présentons notre méthode d'évolution des ressources termino-ontologiques qui assiste la propagation des changements de la RTO vers les annotations sémantiques et vice-versa.

CHAPITRE IV

La méthode EvOnto : contexte et présentation générale

Sommaire

0.1 Introduction

Les travaux existants (cf. chapitre II) n'abordent que peu l'évolution de la partie lexicale des ontologies et l'impact de l'évolution des ontologies sur les annotations sémantiques de documents. De plus, les collections de documents ne sont jamais figées dans le temps et peuvent à tout moment être modifiées, par l'ajout de nouveaux documents ou la modification (la suppression) de documents déjà présents. Par conséquent, il reste encore des questions ouvertes à la réflexion.

Dans nos travaux, nous nous sommes intéressés particulièrement aux problèmes liés à la gestion des changements d'une RTO telle que nous l'avons définie au chapitre III. Rappelons que nous appelons *ressource termino-ontologique* (RTO) un modèle conceptuel comportant une composante conceptuelle, prenant la forme d'une ontologie, et une composante lexicale, ou terminologie, à savoir des termes dénotant les concepts. Nous étudions ces changements à la fois dans le contexte local de cette ressource et au sein d'applications particulières que sont les *annotations sémantiques* d'un corpus.

Nous présentons dans ce chapitre la problématique d'évolution de RTO et d'annotations sémantiques dans le cadre du projet Dynamo et nous introduisons notre méthode EvOnto qui offre une vue unifiée sur l'évolution dans le cas où la RTO sert à produire des annotations sémantiques d'un corpus qui évolue dans le temps. Nous présentons les exigences d'EvOnto et ses besoins ainsi que les principes mis en jeu.

0.2 Problématiques d'évolution dans Dynamo

Les documents et les textes jouent un rôle très important dans la création d'ontologie et en retour, l'ontologie est utilisée pour annoter ces mêmes textes.

Le fait de disposer d'un modèle de type « ontologie de domaine » avec une composante lexicale (une liste des termes qui dénotent des concepts de l'ontologie) doit permettre de retrouver dans les textes les formes lexicales des concepts du domaine.

Concrètement, le projet DYNAMO se focalise sur la dynamique des trois types de ressources à savoir, la RTO, une collection de documents et les annotations sémantiques associées à ces documents à partir de la RTO (Figure IV-1). Deux raisons principales peuvent provoquer ces évolutions :

MODIFICATION DE LA RESSOURCE TERMINO-ONTOLOGIQUE

L'ontologue peut souhaiter modifier la structure de la RTO pour plusieurs raisons. Il peut par exemple se rendre compte d'erreurs de représentation dans la RTO (un terme n'a pas été associé au bon concept, la signature d'une propriété n'est pas correcte, deux concepts devraient être fusionnés etc.). Il peut également vouloir faire évoluer la RTO lorsque le domaine représenté dans la RTO évolue (apparition d'un nouveau terme, modification du sens d'un terme, etc.). Par le biais de l'éditeur d'ontologies du système, l'ontologue apportera la modification souhaitée. Cette modification peut avoir de nombreuses conséquences, que ce soit localement au niveau des éléments dépendants de l'élément modifié dans la RTO, ou au niveau des éléments dépendant de la RTO et aussi sur la base d'annotations.

ÉVOLUTION DE LA COLLECTION DES DOCUMENTS

Un autre scénario d'évolution correspond à l'ajout de nouveaux documents dans le système. Les termes, pertinents pour l'annotation, contenus dans ces nouveaux documents peuvent ne pas être représentés dans la RTO ou leur sens peut avoir évolué. L'ontologie n'étant pas en adéquation avec le texte, l'outil d'annotation ne pourra pas être efficace. La RTO devra donc évoluer. Plus généralement, ceci devra être le cas lorsque le système n'est

pas capable d'annoter correctement un document. Une mauvaise annotation peut être détectée par l'annotateur qui pourra demander une évolution de la RTO. Elle peut également être détectée automatiquement par le système.

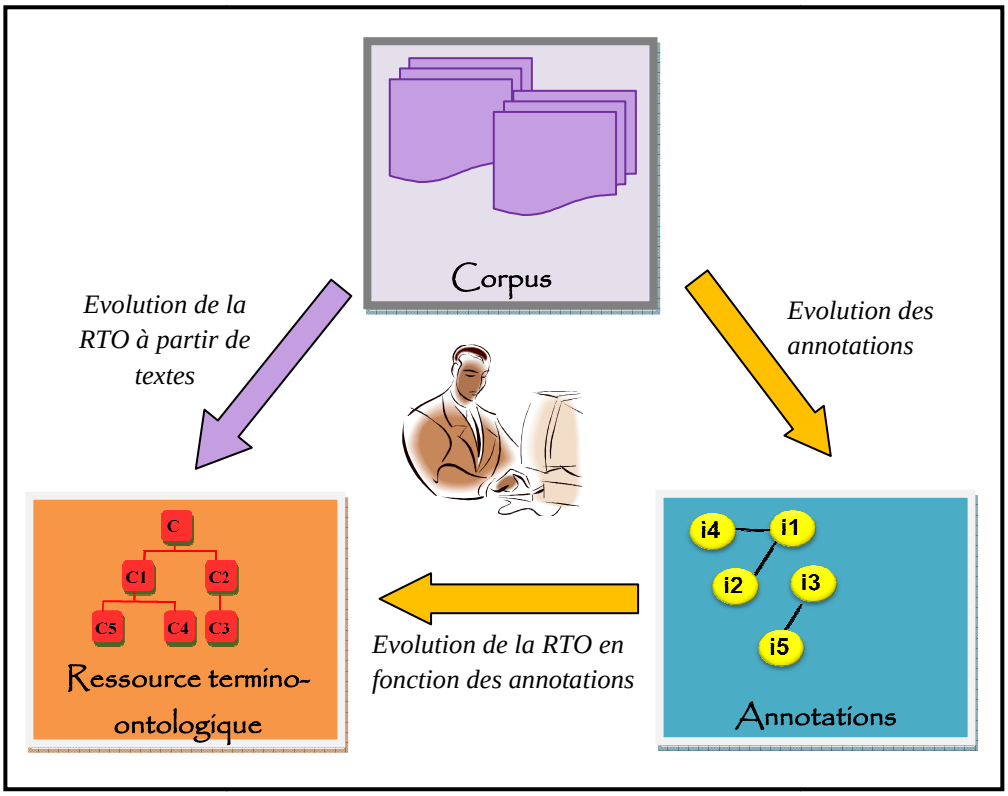


Figure IV - 1 : Gestion d'évolutions de différents types de connaissances [Laublet et al., 2010, livrable Lot 2 Dynamo]

Pour répondre à ces deux types de scénarios, plusieurs questions se posent :

1. Quels sont les critères permettant de déterminer que dans sa version actuelle la RTO ne permet pas la génération automatique d'une annotation de qualité ?
2. Comment représenter au mieux les opérations de changement de la RTO afin d'aider l'ontologue à choisir son opération de changement ?
3. 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) ?

4. Comment peut détecter l'absence d'incohérences générées à cause des changements de RTO ?
5. Comment résoudre l'absence d'incohérences détectées à cause de la modification de RTO pour assurer la consistance globale du système (au sein de la RTO mais aussi sur ses utilisations) ?
6. Si plusieurs solutions sont possibles pour traiter les effets, quelles informations doivent être présentées à l'ontologue pour l'aider à choisir une solution ?

0.2.1 Besoin en évolution

L'évolution des ontologies n'est pas un processus trivial, en raison de la variété des sources et les conséquences des changements. Il est difficile à réaliser sans l'aide d'un ingénieur d'ontologie car il est compliqué d'anticiper tous les effets secondaires d'un changement au sein de l'ontologie et plus encore de propager le changement aux artefacts dépendants de celle-ci. Par conséquent, nous avons identifié un ensemble d'exigences pour la gestion d'évolution d'une RTO qui permet aux utilisateurs d'être en mesure de modifier facilement une RTO. Ces exigences peuvent être représentées comme suit :

- La première exigence est essentielle pour toute approche d'évolution d'une RTO. Après l'application d'un changement à une RTO cohérente, la ressource doit rester dans un état cohérent.
- La deuxième exigence est de présenter à l'utilisateur toutes les informations nécessaires pour contrôler les changements et prendre des décisions appropriées. Le système doit offrir un contrôle du processus d'évolution grâce à un aperçu (affichage des informations) sur le changement de base (celui que l'ontologue veut l'appliquer) et les changements additionnels (ceux qui sont causés directement ou indirectement par un changement de base).

- Il est aussi nécessaire que le système puisse propager le changement appliqué localement à la RTO (car une modification dans une partie de la ressource peut générer des incohérences subtiles dans les autres parties de la même ressource), aux artefacts dépendants (les objets, applications et ontologies dépendants) et de valider globalement les changements.
- La définition de critères doit aider pour analyser la qualité d'une annotation et inciter l'ontologue à mettre à jour si besoin l'ontologie
- La traçabilité des changements doit être gardée pour pouvoir les justifier, les expliquer, voire les annuler et gérer les différentes versions d'une ontologie.

0.2.2 Exemples de scénario d'évolution dans Dynamo

Comme décrit dans la section 4.1, le projet Dynamo définit deux scénarios typiques menant à une évolution dans la RTO. Nous donnons dans cette section un exemple pour chacun d'entre-eux. Notons bien que la propagation vers les annotations sémantiques se fait dans les deux cas et sera détaillée dans le chapitre VI.

0.2.2.1 Evolution de la RTO suite à l'ajout de nouveaux documents

Tous les documents de la collection sont annotés en se basant sur la RTO. L'évolution du corpus (ajout, modification de documents), comme l'évolution de la RTO, amène à annoter ou ré-annoter les documents dans le cas où les annotations sont non valides.

La démarche prévue dans Dynamo (figure IV-2) est d'annoter automatiquement à partir du module dédié ces nouveaux documents. Cependant la RTO peut ne plus être en adéquation avec le corpus, ce qui mènerait à la création automatique d'annotations incomplètes ou fausses. Une annotation est incomplète si elle ne respecte pas les critères d'adéquation entre RTO et documents, choisis au préalable par l'ontologue (par exemple un nombre minimal

d'annotations, une couverture minimale du texte et un ensemble de concepts à retrouver dans chaque document (voir section 6.5). Pour chaque document, un score est défini et permet d'évaluer dans quelle mesure les critères sont vérifiés par les annotations qui ont été produites à partir de son analyse.

Finalement, l'ontologue peut parcourir la liste des documents en commençant par ceux dont le score de vérification des critères est faible. Bien que ce processus soit manuel, il est efficace parce qu'il permet de guider l'ontologue rapidement vers des fiches mal annotées (celles qui ont un score faible). Or une mauvaise annotation correspond en général à l'identification de concepts ou de termes manquants dans la RTO. Il faut itérer sur l'utilisation du module d'évolution de la RTO jusqu'à obtenir une annotation satisfaisante pour l'ontologue. Celui-ci peut décider alors d'effectuer des changements, soit, dans la RTO, en général l'ajout de nouveaux termes pour les concepts existants ou l'ajout de nouveaux concepts qui sont des sous-classes des concepts attendus dans les critères d'annotation, soit, par la modification manuelle des annotations.

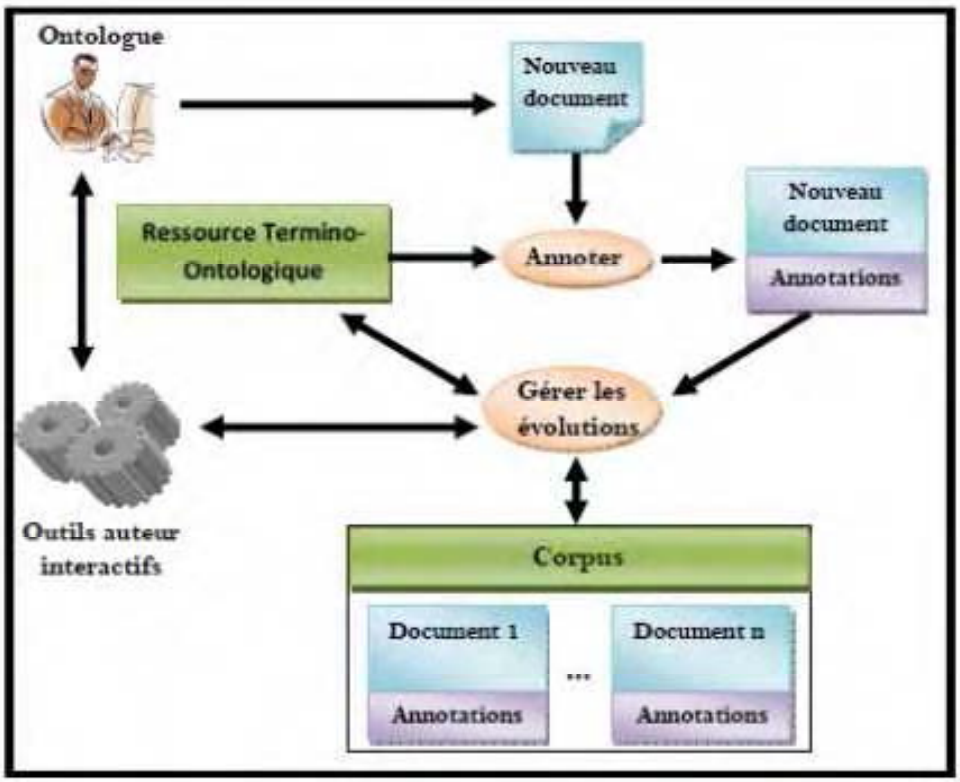


Figure IV - 2 : Cycle d'évolution de la RTO et de l'annotation suite à l'ajout de document dans le corpus

Pour illustrer l'évolution de la RTO et des annotations sémantiques correspondantes dans le cas du premier scénario, nous examinons un extrait d'une RTO du domaine des incidents logiciels.

Le noyau de l'ontologie d'ARTAL (figure IV-3) est basé sur quatre concepts : *défaut, composant, événement et fonction*. Notons bien que la RTO d'ARTAL est déjà détaillée dans le chapitre III.

- Un défaut désigne une panne ou une anomalie (figure IV-4) ;
- Un composant désigne une partie identifiée d'une interface ou d'une information (figure IV-6);
- Un événement est une action qui peut déclencher un défaut (figure IV-5).
- Une fonction est concernée par un événement

Ces concepts sont reliés par les relations suivantes :

- Chaque défaut affecte un composant ;
- Chaque défaut est localisé dans un composant (pas forcément celui affecté) ;
- Chaque défaut est causé par un événement ;
- Chaque événement concerne un composant (pas forcément celui affecté ou celui dans lequel le défaut a été localisé).

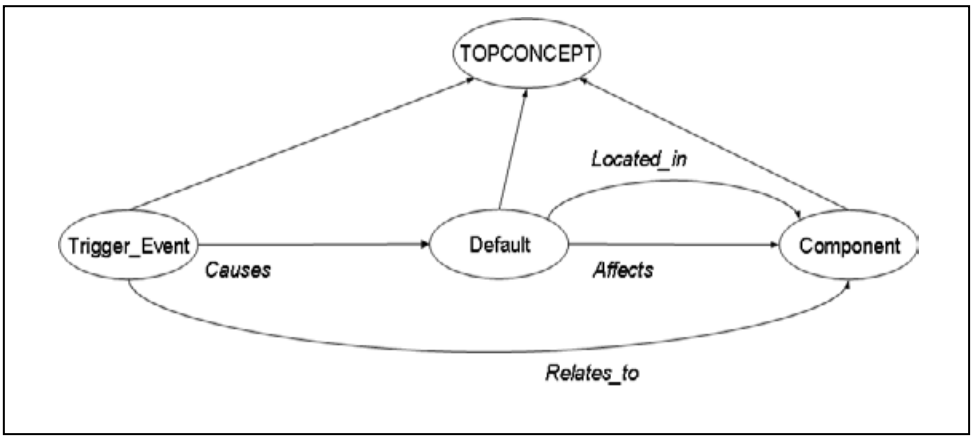


Figure IV - 3 : Noyau de l'ontologie d'ARTAL

Les figures suivantes montrent des sous-classes des concepts principaux.

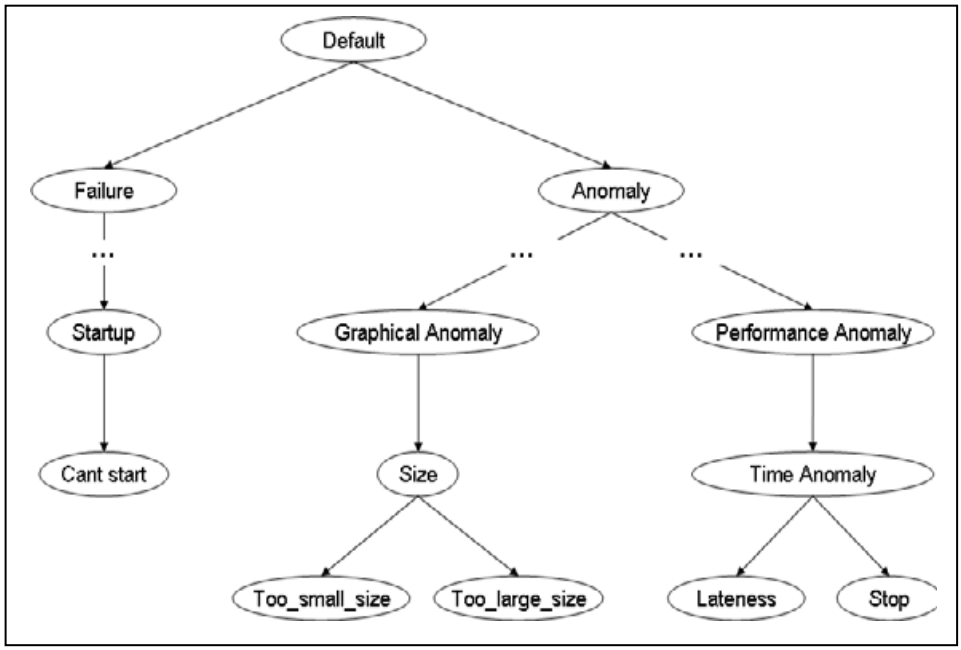


Figure IV - 4 : Extrait de la taxonomie des défauts

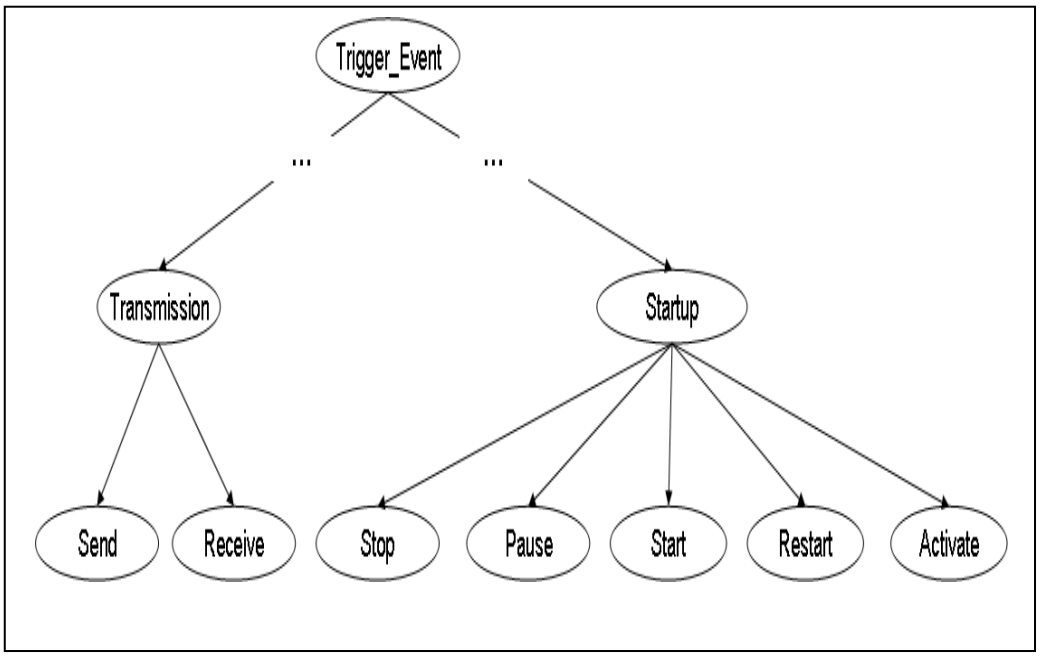


Figure IV - 5 : Extrait de la taxonomie des événements

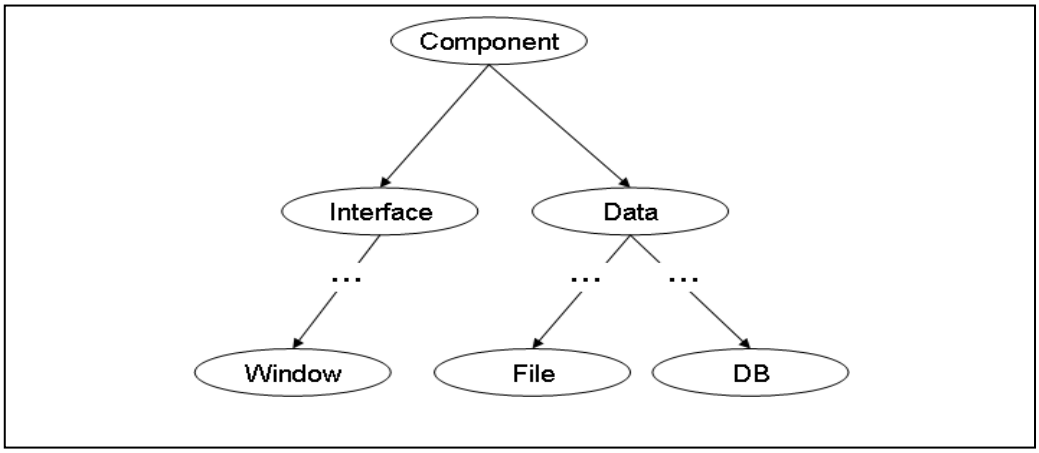


Figure IV - 6 : Extrait de la taxonomie des composants

Considérons maintenant la fiche suivante «600» (dans le cas d'ARTAL, on ne traite que les résumés). Dans la projection de la liste de termes de la RTO sur la fiche 600, l'outil TextViz détecte les termes *ClientStation*, *TE* et *After* qui dénotent respectivement les concepts *ClientStation*, *TE*, et *Posterior_Temporal_events*. L'outil n'a pas détecté d'autres termes pour dénoter le concept Default ou ses sous concepts indirects. Une fois le processus d'annotation

TexTViz appliqué de manière automatique, nous obtenons l'annotation présentée dans le tableau IV-1

Fiche 600 (Résumé)
"ClientStation is stucked after a while using trace mode on 10 TEs"

Tableau IV - 1 : Annotation de la fiche automatiquement

Annotation de la fiche 600	
Concepts	**Concepts indirects**
Default	
Component	**ClientStation**
	TE (TraceableElement)
Trigger_Event	**Posterior_Temporal_Events**
Domaine/ Relation/Co-domaine	**i1 →iR→ i2,** ***i1 et i2 =instancesanonymes*** ***iR = instances de relations***
Default →Affects→ Component	
Default →Located_in→ Component	
Trigger_Event→Causes→ Default	**Posterior_Temporal_Events→*iR*:Causes→**
Trigger_Event→Relates_to→ Component	**Posterior_Temporal_Events→*iR*:Relates_to→TE**

Ici on constate que la relation Causes n'a pas de valeur. Comme nous l'avons expliqué dans le scénario, une mauvaise annotation peut être *détectée par l'ontologue* ou également être *détectée automatiquement par le système*.

Dans le premier cas – *mauvaise annotation détectée par l'ontologue* – l'ontologue pourra demander une évolution de la RTO. Dans notre exemple, l'ontologie contient le concept *Time_Anomaly* auquel deux autres concepts fils sont rattachés (*Lateness* et *Stop*). Pour ré-annoter la fiche 600, l'ontologue décide de créer un nouveau terme *stuck*. Ce terme dénotera un nouveau concept *Stucked* crée par l'ontologue. Sémantiquement, le concept créé peut être attaché à une anomalie liée au temps. Ce dernier n'exprime pas un retard ou un arrêt (les deux

concepts couverts par l'actuelle ontologie), il présente plutôt un blocage. Il doit donc être ajouté et rattaché au concept *Time_Anomaly* (sous-concept indirect de Default).Suite aux modifications apportées à la RTO par l'ontologue, les documents à l'origine de cette évolution doivent être ré-annotés. Nous obtenons la nouvelle annotation présentée dans le tableau IV-2.

Tableau IV - 2 : Ré-annotation de la fiche manuellement

Annotation de la fiche 600	
Concepts	**Concepts indirects**
Default	***Stucked***
Component	**ClientStation**
	TE (TraceableElement)
Trigger_Event	**Posterior_Temporal_Events**
Domaine/ Relation/Co-domaine	**i1 →iR→ i2,** *i1 et i2 =instancesanonymes* *iR = instances de relations*
Default →Affects→ Component	***Stucked*→** *iR* **:Affects→**
Default →Located_in→ Component	***Stucked*→** *iR*:**Located_in→**
Trigger_Event→Causes→ Default	**Posterior_Temporal_Events→***iR*:**Causes→*Stucked***
Trigger_Event→Relates_to→ Component	**Posterior_Temporal_Events→***iR*:**Relates_to→TE**

Dans le deuxième cas - *mauvaise annotation détectée par le système* - et comme nous l'avons expliqué, il y a des critères de qualité qui doivent être vérifiés par le système. Certaines annotations peuvent ne pas contenir toutes les instances de concepts noyaux de l'ontologie. Toutefois il faudrait vérifier une couverture minimale du noyau. L'annotation d'une fiche ARTAL doit contenir :

Une première couverture minimale :

- Une instance indirecte de *Default* ;
- Une instance indirecte de *Component* ;

- Une instance de la relation *Affects* entre les deux instances indirectes de *Default* et *Component*.
- Une instance de la relation *Located_in* entre les deux instances indirectes de *Default* et *Component*.

Ou bien

Deuxième couverture minimale :

- Une instance indirecte de *Default* ;
- Une instance indirecte de *Component* ;
- Une instance indirecte de *Trigger_Event* ;
- Une instance de la relation *Causes* entre les deux instances indirectes de *Trigger_Event* et *Default* ;
- Une instance de la relation *Relates_to* entre les deux instances indirectes de *Trigger_Event* et *Component* ;

Dans notre exemple, l'affectation (*Affects*) et la localisation (*Located_in*) de défaut ne sont pas mentionnées. La première couverture minimale n'est pas respectée. Mais l'annotation assure la deuxième couverture minimale.

0.2.2.2 Modification directe de la RTO par l'ontologue

Dans ce deuxième scénario, l'ontologue décide directement d'effectuer des changements dans la RTO, en général l'ajout, la suppression, la modification, la fusion, la division … des concepts et/ou des relations, l'ajout de nouveaux termes pour des concepts existants ou la suppression de termes…La liste des changements possibles est présentée dans les sections 5.4.4 et 5.4.5. Ce choix de l'ontologue peut provenir par exemple de modifications ou d'évolutions dans le domaine concerné par la RTO ou bien de choix de modélisation différents comme dans l'exemple ci-dessous.

La démarche prévue pour ce scénario (figure IV-7) est de modifier la RTO existante en utilisant EvOnto le module d'évolution de la RTO.

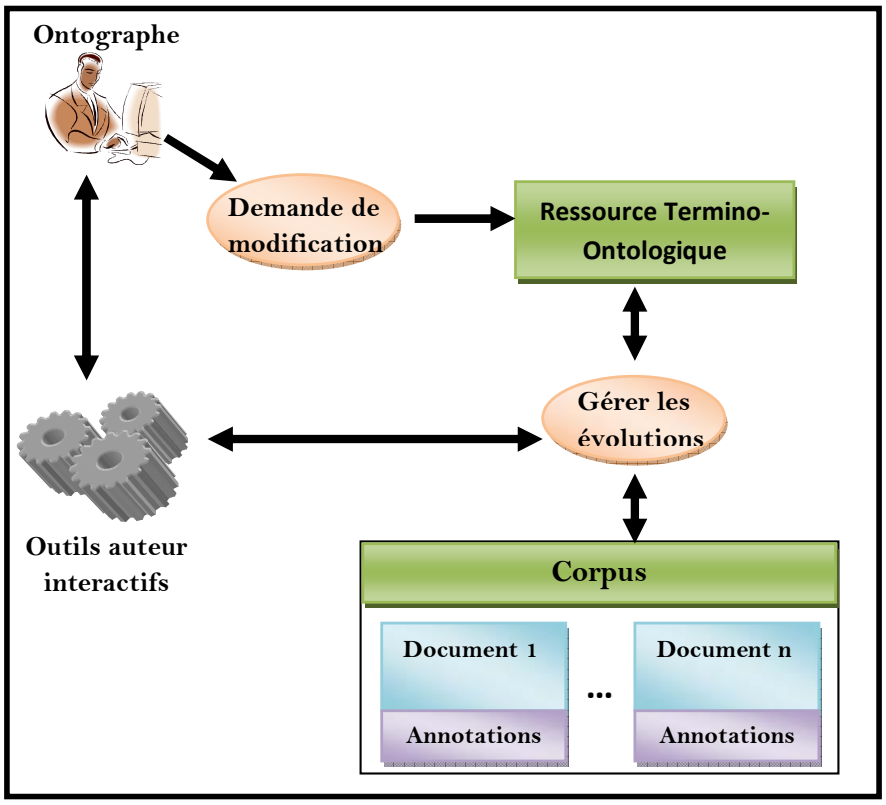


Figure IV - 7 : Cycle d'évolution de l'annotation dans le contexte d'évolution de la RTO

Reprenons l'exemple présenté dans la section précédente dans le cas de ce deuxième scénario. Après la création du terme *stuck* et du concept *Stucked*, faisons l'hypothèse que l'ontologue se rend compte qu'il a commis une erreur. Le terme stuck correspond à un synonyme du concept *freeze* qui existe déjà dans la RTO. Ces deux termes présentent la notion du blocage dans le temps. L'ontologue décide donc de supprimer le terme *stuck*[20] ainsi que le concept *Stucked*. Remarquons qu'une fois la RTO modifiée, elle ne permet plus d'annoter les ressources jusqu'ici annotées avec ce concept (comme par exemple la fiche 600). Les critères d'adéquation entre la RTO et les documents ne sont pas respectés, ni la première couverture minimale, ni la deuxième couverture minimale. Par conséquent, l'ontologue doit de nouveau faire évoluer la RTO (l'ajout ou la suppression d'une entité de la

[20] L'ontologue peut garder le terme *stuck* comme deuxième terme dénotant le concept *freeze*

RTO) et/ou l'annotation (par exemple ajout ou suppression des instances) jusqu'à ce que les critères soient vérifiés.

Après avoir vu ces deux scénarios, nous allons, dans la section suivante, présenter notre approche d'évolution EvOnto, une approche semi-automatique pour aider d'une manière interactive l'ontologue à faire évoluer la RTO.

0.3 EvOnto : approche semi-automatique pour aider d'une manière interactive l'utilisateur

Gérer l'application des changements tout en maintenant la cohérence au sein des différentes ressources est une tâche coûteuse en termes de temps. La complexité de la tâche peut aussi être une source d'erreurs.

Pour assister l'ontologue dans cette tâche, nous avons défini une méthode de gestion de changement EvOnto **(***EVOlutionONTOlogie*) avec un logiciel associé. Nous proposons un processus semi-automatisé permettant d'aider de manière interactive l'utilisateur et de conduire l'application des changements tout en maintenant la cohérence entre la RTO modifiée et les annotations sémantiques de collections documentaires

0.3.1 Aspects importants de l'évolution

Afin de décrire notre approche, nous définissons ici les notions sur lesquelles elle repose.

Définition 1 – Changement

Un changement se réfère à une modification de la ressource termino-ontologique lors du processus d'évolution de l'ontologie dans sa version V_N à l'ontologie dans sa version V_{N+1}.

Pour exprimer l'évolution, nous avons recensé l'ensemble des modifications que l'on peut apporter aux ressources termino-ontologiques et nous leur avons donné une signification bien

précise. Cette typologie (voir 6.4.3) élargit la conceptualisation de [Stojanovic, 2004] par l'ajout d'un certain nombre de caractéristiques dont la plus notable est la prise en compte du lexique de la RTO qui peut évoluer lorsque les changements portent sur les termes et les liens de dénotations. La typologie sera discutée en détail dans le chapitre suivant. Les changements peuvent être élémentaires, comme l'ajout ou l'effacement d'un terme, d'un concept, ou d'une relation, mais aussi plus complexes, comme le déplacement ou la fusion de concepts.

Définition 2 – Changement élémentaire

Un changement élémentaire est celui qui peut être identifié en analysant uniquement la structure conceptuelle (concept, relation, attribut) ou lexicale (terme) de la RTO. Ce type de changement affecte une seule entité de la RTO.

Définition 3 – Changement complexe

Un changement complexe est un changement qui affecte deux ou plusieurs entités de la RTO. Chaque changement complexe est la conséquence de l'association de deux ou plusieurs changements élémentaires. Théoriquement, l'ensemble de changements complexes est infini, parce que le nombre de combinaisons possibles est aussi infini.

Un objectif de notre travail est que la prise en compte des impacts d'une modification de RTO sur la ressource elle-même et sur les annotations, et ce dès la proposition de cette modification, permettra de réduire les effets négatifs. 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 de la RTO ou des annotations existantes.

Pour vérifier notre hypothèse, nous utilisons la notion de stratégie d'évolution (empruntée à Stojanovic [Stojanovic, 2004]) et l'élargissons en intégrant dans la stratégie les conséquences d'un changement sur les annotations. Ces stratégies permettent aussi de résoudre les changements de la RTO d'une manière semi automatique avec l'intervention de l'ontologue en vue d'assurer le *rôle* et la *consistance* de la RTO après sa modification.

Définition 4 – Cohérence de la ressource termino-ontologique

La ressource termino-ontologique doit évoluer d'un état cohérent vers un autre état cohérent. L'état cohérent d'une ressource termino-ontologique est assuré si le changement de base et les changements additionnels, qui sont causés par un changement de base, sont validés (appliqués) ensemble.

Définition 5 – Lien de référence

Un lien de référence est le chemin qui relie un objet (une ressource annotée) à une entité de la ressource termino-ontologique utilisée comme référence sémantique

Définition 6 – Conformité entre la ressource termino-ontologique et les annotations sémantiques

La conformité entre la ressource termino-ontologique et l'annotation sémantique est assurée si les liens de références existent et prennent en compte la nouvelle modification après une évolution de la ressource termino-ontologique ou d'annotations.

Définition 7 – Stratégie d'évolution

Un ensemble de manières possibles dont les utilisateurs décident pour résoudre les inconsistances issues de l'application d'un changement, par l'application d'un nombre des changements additionnels.

Par exemple, si deux concepts sont fusionnés, leurs sous-concepts et propriétés peuvent être effacés ou déplacés au concept résultant ou à un autre. Dans le tableau IV-2, nous présentons des exemples de stratégies d'évolution pour une fusion de concepts. Dans l'exemple, trois stratégies sont possibles pour gérer les concepts fils des concepts A et B. La stratégie d'évolution est la manière de gérer les conséquences de la fusion des concepts A et B sur les entités de la RTO liées à l'entité modifiée initialement tels que les propriétés et les termes qui dénotent A et B. Les stratégies d'évolutions seront détaillées dans le chapitre suivant.

Tableau IV - 3 : Exemples de stratégies d'évolution pour une fusion de concepts

Stratégies d'évolution pour un changement qui fusionne les concepts A et B pour obtenir le concept C			
Concepts : Quoi faire avec les sous concepts de A et B	*Propriétés : Quoi faire avec les propriétés de A et B*	*Termes : Quoi faire avec les termes qui dénotent A et B*	*Autre entité*
Stratégie d'évolution SE1			
Suppression	*Effacement domaine ou co-domaine*	*Suppression des termes*	…
Stratégie d'évolutionSE2			
Attacher au concept C	*Attacher au concept C*	*Attacher à un ou plusieurs sous-concepts selon l'indication de l'ontologue*	…
Stratégie d'évolutionSE3			
Attacher à un autre concept de la RTO	*Attacher à un autre concept de la RTO*	*Attacher à un autre concept de la RTO*	…

0.3.2 Vue unifiée de l'approche EvOnto : méthodologie

EvOnto (**Ev**olution d'**Onto**logie) est un modèle de gestion de changement (figure IV-9) pour une RTO servant à l'annotation sémantique de documents textuels. Il est destiné à un ontologue et le guide interactivement pour formuler une demande de changement, évaluer son impact (effets supplémentaires) sur la qualité de la RTO 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'ontologue 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 la ressource elle-même et son utilisation dans des annotations.

Dans l'état de l'art, nous avons présenté plusieurs processus d'évolution d'ontologie. Nous définissons une méthode et un outil qui se situent dans la continuité de ces travaux, et qui les étendent pour traiter des problèmes spécifiques : la gestion des termes associés aux concepts et la gestion conjointe des annotations. Pour mieux gérer ces deux facettes, l'approche générale (figure IV-9) EvOnto est structurée selon deux processus :

PROCESSUS D'EVOLUTION DE LA RESSOURCE TERMINO-ONTOLOGIQUE

Le processus d'évolution de la RTO représente la phase d'évolution consacrée aux changements et à la manière de les appliquer tout en préservant la cohérence de la RTO. En effet, le processus offre un guide interactif à l'ontologue pour formuler une demande de changement, évaluer son impact. De plus, les stratégies d'évolutions proposées sont adaptables. Elles proposent d'abord un ensemble de conséquences qui traitent en bloc toutes les entités d'un même type, que l'ontologue peut ensuite ajuster entité par entité si nécessaire. Le processus d'évolution comporte les 4 phases représentées sur la figure IV-9.

1 **Expression du changement :** *La phase d'expression d'un changement est lancée suite à la demande d'un changement à appliquer sur une ressource termino-ontologique initiale supposée cohérente. Elle vise à expliciter le changement pour préparer les phases d'analyse et de résolution des conséquences avant d'en assurer la mise en œuvre.*

2 **Présentation des stratégies et simulation de leurs conséquences :** *La phase de Présentation des stratégies et simulation de leurs conséquences a pour but de présenter pour chaque stratégie proposée les conséquences avant d'en assurer la mise en œuvre et donner le droit à l'ontologue d'ajuster ses conséquences.*

3 **Choix de la stratégie et ajustement des conséquences :** *Cette phase donne le droit à l'ontologue de choisir une stratégie d'évolution qui est présenté par le système. Or le système ne propose que des choix systématiques. Pour chaque stratégie choisie les conséquences peuvent être modifiées. Ces stratégies sont adaptables pour mieux répondre à la diversité des raisons à l'origine d'un changement.*

4 **Application du changement à la RTO :** *Dans cette phase la séquence des changements est validée conjointement avec l'ontologue afin d'obtenir une nouvelle version de la ressource termino-ontologique.*

PROCESSUS D'EVOLUTION DES ANNOTATIONS

Pour préserver la cohérence de la nouvelle version de la RTO avec les annotations sémantiques des documents basées sur cette ressource, celles-ci devront être adaptées à la nouvelle version de la RTO. Pour cela nous avons proposé un processus d'évolution d'annotations qui comporte trois phases.

1 **Détection des annotations incohérentes :** *La phase de détection des annotations incohérentes permet de capturer une liste d'annotations sémantiques (graphes d'annotations) qui sont incohérentes par rapport à la ressource termino-ontologique. Pour cette phase, deux méthodes sont proposées :*

2 **Rectification des annotations incohérentes :** *La phase de rectification des annotations incohérentes permet de corriger les incohérences des triplets détectées en fonction des choix effectués par l'ontologue. Deux méthodes sont proposées pour corriger les incohérences.*

3 **Mise à jour de la base d'annotations :** *Une fois rectifiées les annotations de toutes les fiches concernées par la modification, la phase de mise à jour de la base d'annotations permet à l'ontologue ou au spécialiste du domaine de valider l'annotation fiche par fiche.*

MODULE D'EVALUATION DE LA QUALITE DES ANNOTATIONS

Ce module demande à l'utilisateur de définir des critères de qualité des annotations propres au corpus et au domaine d'application. Il consiste à vérifier automatiquement que les annotations produites par le système respectent, pour chaque document, les critères choisis au préalable par l'ontologue. EvOnto permet alors de 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 sera détaillé dans le chapitre VI.

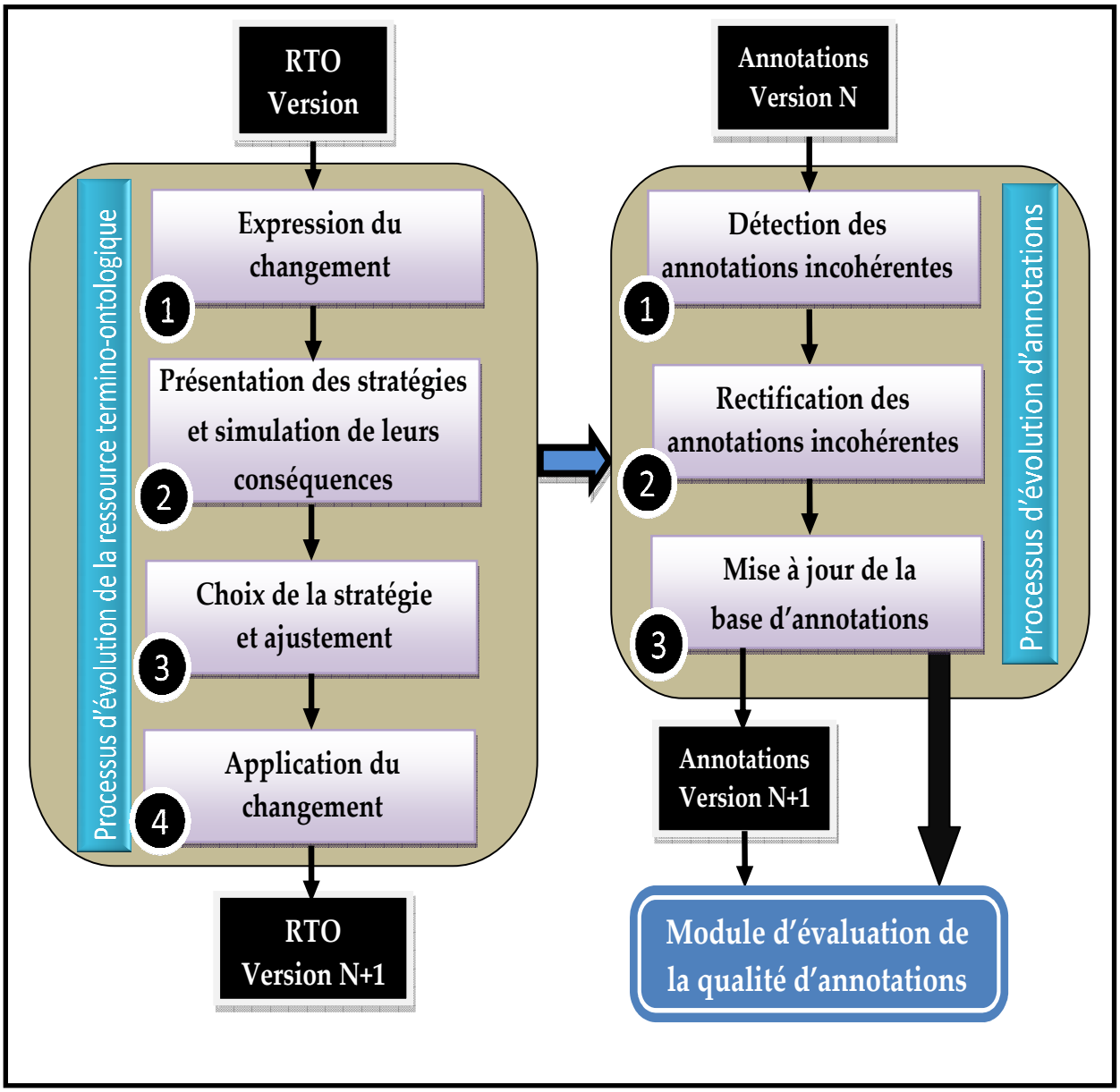


Figure IV - 8 : Architecture générale de l'approche EvOnto

0.4 Conclusion

Dans cette section, nous avons présenté une nouvelle approche pour faire face aux changements de la RTO et des annotations sémantiques des documents. L'approche est basée sur un processus d'évolution en deux niveaux. Un processus en quatre phases, qui analyse systématiquement les conséquences des changements et assure la cohérence de la RTO et un niveau qui permet la propagation des changements vers les annotations sémantiques.

En prenant en compte la dynamique des différents types de connaissances, deux scénarios menant à une évolution de la RTO et des annotations sémantiques sont définis. La description détaillée du fonctionnement et de la modélisation de l'approche fait l'objet des chapitres V et VI.

Par ailleurs, une autre approche a été proposée pour gérer l'évolution dans DYNAMO en utilisant le système multi-agent [Sellami *et al.*, 2010] [Sellami *et al.*, 2011].Le but d'utiliser le SMA est d'aider et assister l'ontologue à la fois dans la phase de construction et dans la phase de maintenance de la RTO (réduire le besoin d'intervention manuelle). Techniquement, l'idée est d'intégrer le module SMA dans l'outil Protégé. Ce SMA utilise l'état courant de la RTO et des résultats d'analyses de textes pour en dégager la nouvelle version d'une RTO existante. Le résultat produit par le SMA est ensuite proposé à l'ontologue pour validation via une interface graphique. Lorsque celui-ci 'intervient sur l'ontologie, ses actions sont transmises au SMA. Ce dernier actualise ses connaissances, réorganise les agents et fait une nouvelle proposition. Les évolutions se traduisent par le déplacement, l'ajout ou l'élimination d'un concept ou d'une relation, l'ajout d'un terme ou l'élimination d'un terme ou d'un lien de dénotation. Il s'agit d'une construction interactive d'une RTO.

CHAPITRE V

Processus d'évolution de la ressource termino-ontologique

Sommaire

5.1 Introduction

L'approche d'évolution EvOnto (*EVOlutionONTOlogie*) que nous avons définie (chapitre IV) est un processus d'évolution d'une ressource termino-ontologique mettant au centre l'ontologue responsable de l'évolution. Ce processus guide l'ontologue interactivement pour formuler une demande de changement et évaluer son impact sur la qualité de la RTO ainsi que la qualité des annotations sémantiques avant de décider de sa mise en œuvre. Des informations sur l'utilisation de l'ontologie sont ainsi fournies à l'ontologue pour qu'il prenne l'initiative d'une évolution de la RTO, en connaissant les conséquences, et les adapte pour minimiser les effets négatifs, les impacts non souhaitables ou les coûts correspondants (évolution en cascade sur la RTO et/ou sur les annotations). La première partie de notre méthodologie à savoir le processus d'évolution de la RTO et ses étapes sont discutés ci-après. Le chapitre est organisé comme suit : dans la section 2, nous présentons le modèle de représentation de la RTO utilisé dans notre travail ainsi qu'un exemple en section 3. Les sections suivantes (sections 5.4, 5.5, 5.6 et 5.7) présentent respectivement les aspects importants de l'évolution de la RTO à travers les quatre étapes du processus d'évolution de la RTO (premier bloc sur la figure IV-9 du chapitre IV) à savoir : (i) expression du changement, (ii) présentation des stratégies et simulation de leurs conséquences, (iii) choix de la stratégie et ajustement et (iv) application du changement, tout en présentant pour chaque étape les travaux réalisés et leur rôle dans le processus d'évolution.

5.2 Modèle de la RTO

La définition de formalismes pour représenter une RTO est l'objet de recherches récentes. Reymonet définit dans [Reymonet *et al.*, 2007] un modèle pour manipuler une RTO en OWL. Les termes y sont représentés comme des classes à part entière reliées aux concepts par une relation de dénotation. Cimiano mentionne la nécessité d'associer un lexique indépendant à

une ontologie et définit un modèle pour manipuler la RTO [Cimiano, 2006]. Les partenaires du projet Dynamo ont choisi de rester dépendant du langage utilisé (*OWL*) pour représenter l'ontologie et généraliser le modèle de [Reymonet *et al.*, 2007] afin de gérer de manière cohérente l'évolution des différentes composantes de la RTO. Un modèle de RTO a été défini qui nous servira plus tard à mieux expliquer la description d'un changement (la syntaxe, les paramètres, la sémantique d'un changement, …) dans les sections 5.5.2, 5.5.3 et 5.6.2.

Définition 1

Un modèle de RTO est une paire ***RTO-M : = (S, L)***, où

- **S** est l'ontologie qui contient les entités ontologiques,
- **L** est le lexique de l'ontologie qui contient les objets terminologiques.

Définition 2

La structure **(S)** d'une ressource termino-ontologique définie selon un modèle inspiré de [Cimiano, 2006] est un quadruplet : $\boldsymbol{S\ (RTO\text{-}M) : = (C, P, H^C, H^P)}$, où

Définition de la RTO :

- $\boldsymbol{C}$: ensemble des concepts ;
- $\boldsymbol{P}$: ensemble des propriétés ;
- $\boldsymbol{Top = DomainThing \in C}$ **:** concept de haut niveau de la hiérarchie
- $\boldsymbol{H^C}$: hiérarchie de concepts ; $\boldsymbol{(C_1, C_2) \in H^C}$ signifie que $\boldsymbol{C_1}$ est un sous-concept de $\boldsymbol{C_2}$ (relation orientée) ;
- $\boldsymbol{H^P}$: hiérarchie de propriétés ; Si $\boldsymbol{(P_1, P_2) \in H^P}$ signifie que $\boldsymbol{P_1}$ est une sous-propriété de $\boldsymbol{P_2}$.

Précisions :

On distingue deux types de propriétés

- $\boldsymbol{R \subseteq P}$: ensemble des relations ;

- ***A = P\R*** : ensemble des attributs ;
- ***P = A ∪ R***

Fonctions :

A chaque relation ***Rel*** de ***R***, on associe deux fonctions : domaine et co-domaine. ***Rel : relation rel : R → C x C***

- Domaine : fonction qui rend l'ensemble des concepts constituant le domaine d'une relation; ***R → C*** avec ***domaine(Rel) : =C***
- Co-domaine : fonction qui rend l'ensemble des concepts constituant le co-domaine d'une relation; ***R → C*** avec ***co-domaine(Rel) : = C***
- ***Rel(R) = (C_1, C_2)*** s'écrit aussi ***R (C_1, C_2)***

Définition 3

Le lexique **(L)** d'une ressource termino-ontologique définie selon le modèle du RTO est un tuple : ***L (RTO-M) : = (T^C, LD)***, où

T^C : ensemble des termes qui désignent les concepts.

LD : ensemble des liens de dénotation qui relient les termes et les concepts.

T^C → C, dénote (t) = c

LD = {(t, c) / dénote (t) = c}

5.3 Exemple illustratif

Comme nous l'avons présenté dans le chapitre IV, pour expliquer comment les changements de la RTO initiale peuvent affecter la cohérence de toutes les entités de la ressource, nous illustrons par un exemple réel les différentes étapes du processus d'évolution de la RTO. L'exemple montre un extrait de la RTO ARTAL (figure V-1) développée dans le cadre du projet Dynamo.

Rappelons que comme nous avons vu dans le chapitre III, les concepts *Trigger_Event, Default, Function* et *Component* sont les concepts de base (concepts de haut niveau). Sur la figure V-1, nous pouvons voir des relations transverses telles que la relation *Causes, Concerns_Function* ou *Affects_Component* et aussi des termes qui dénotent des concepts de la ressource telle que *T_unzoom* et *T_command*. Chaque terme est relié à un concept par la relation dénote.

Supposons que cet extrait de RTO soit modifié en appliquant les changements suivants :

- *DeleteConcept (Software_Programming_Component) :* supprimer le concept *Software_Programming_Component.*
- *SplitConcept (Interface, User_Interface, Screen)* : diviser le concept *Interface* en sous-concepts *User_Interface* et *Screen.*
- *CreateSubConcept (ConfigFile, File)* : création du concept *ConfigFile* comme concept fils du concept *File*.
- *MoveAssociatedTerm (T_Properties, File, ConfigFile)* : le terme *T_Properties* ne dénote plus le concept *File*. Il dénotera le concept *ConfigFile*.
- *GroupConcept (Unzoom_Function, Zoom_Function, Display_Function)* : créer un super-concept commun *Display_Function* pour les concepts *Unzoom_Function* et *Zoom_Function.*

Après avoir appliqué ces changements, nous obtenons une nouvelle version de la RTO (figure V-2) dans laquelle certains éléments ont été modifiés par rapport à la version initiale de la RTO. La question à laquelle nous nous intéressons est comment peut-on garantir une 'bonne évolution' (c.-à-d. la cohérence des de toutes les entités de la ressource par rapport à l'entité modifié) ?

Dans la section suivante, nous présentons notre méthode d'évolution des ressources termino-ontologiques et en particulier le premier bloc – processus d'évolution de la RTO - sur la figure IV-9 du chapitre IV, qui identifie les étapes essentielles à l'évolution et qui décrit le déroulement prévu pour chacune de ces étapes.

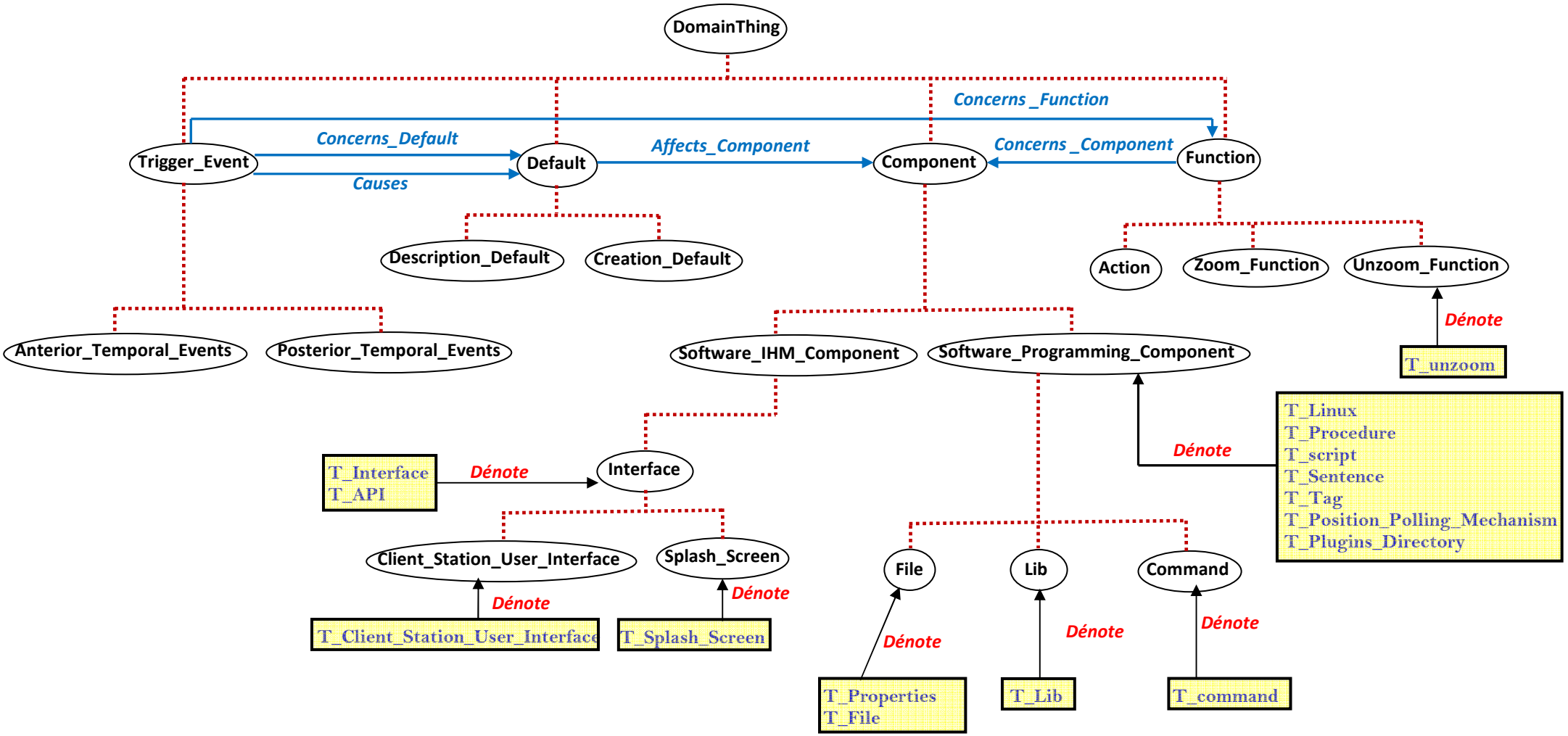


Figure V - 1 : Extrait de la ressource termino-ontologique pour la maintenance de logiciels avant modification

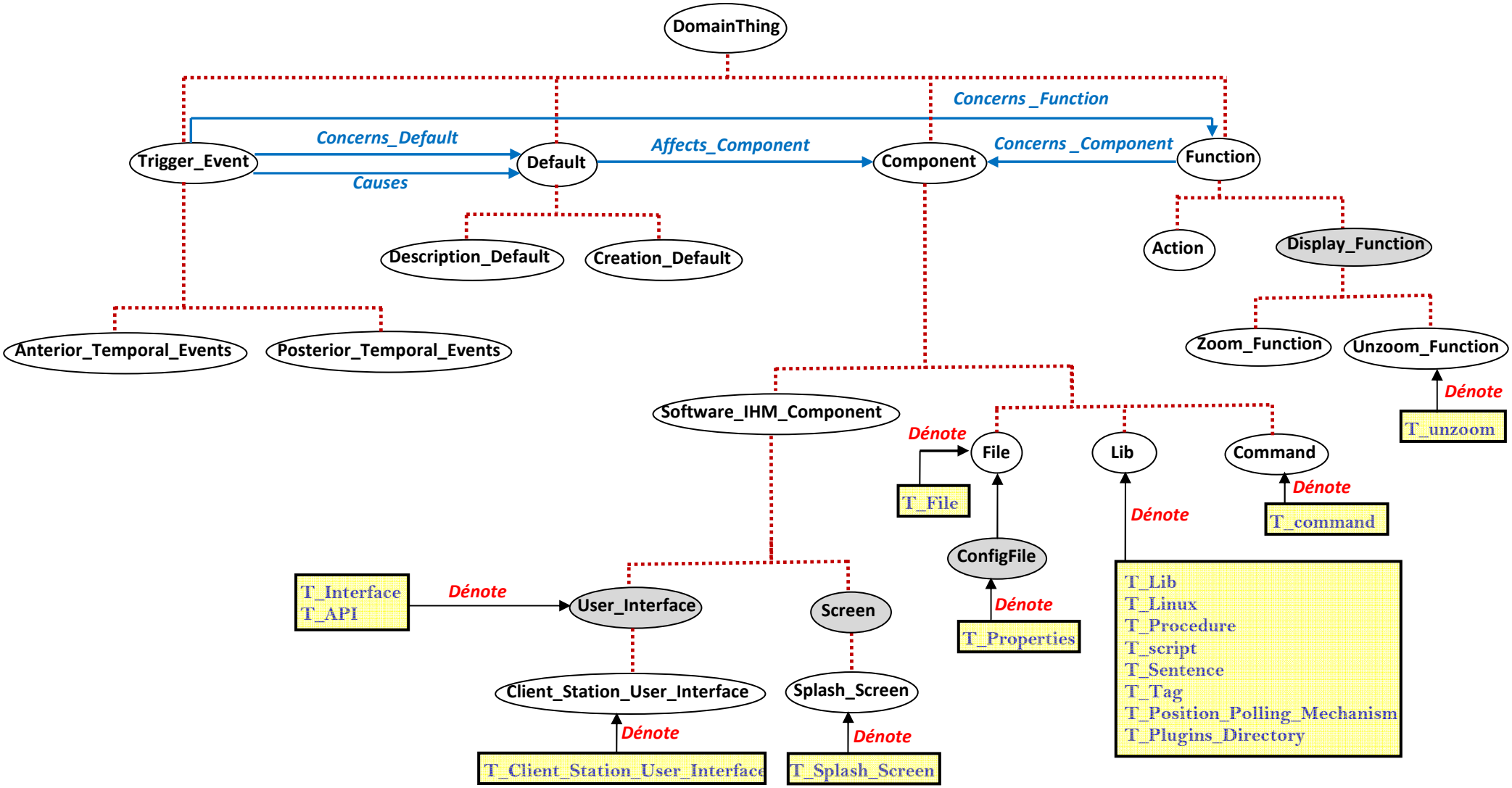


Figure V - 2 : Extrait de la ressource termino-ontologique pour la maintenance de logiciels après modification

5.4 Expression d'un changement

La phase d'expression d'un changement est initiée par l'ontologue lorsqu'il se rend compte d'un besoin en changement sur une ressource termino-ontologique initiale supposée cohérente. Cette phase vise à expliciter le changement de manière formelle et compréhensible pour préparer les phases d'analyse et de résolution des conséquences avant d'en assurer la mise en œuvre.

Un élément important à bien détailler avant de décrire le déroulement prévu pour la phase d'expression du changement est le type de besoin en changement dans le projet Dynamo. Ce dernier prévoit plusieurs scénarios d'évolution, qui correspondent à des parcours différents, de manière très proche des 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 RTO et les annotations sémantiques qu'elle permet de produire sur la collection de documents.

D'une part ce parti pris implique que toutes les évolutions de la RTO imposent 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 mettre en évidence la nécessité d'évolutions de la RTO, qui en retour (l'évolution des annotations) impliquent le maintien de cohérence sur l'ensemble des annotations.

Un besoin en changement correspond aux changements à apporter à la version V_N de la ressource en fonction de : (a) modifications apparues dans la RTO et (b) vérifications des annotations basés sur cette ressource.

5.4.1 Dans le contexte de modification de la RTO

Dans le contexte de la RTO elle-même, l'évolution peut être motivée par deux raisons principales. La première raison correspond au cas où l'ontologue se rend compte d'erreurs de représentation dans la RTO (un terme n'a pas été associé au bon concept, la signature d'une propriété n'est pas correcte, deux concepts devraient être fusionnés etc.). Il peut aussi vouloir préciser la modélisation : un exemple de ce type de changement est présenté dans la section précédente (figure V-2), nous avons fait une opération de division du concept *Interface* en sous-concepts *User_Interface* et *Screen*.

L'autre raison correspond au cas où le domaine représenté dans la RTO évolue (apparition d'un nouveau terme, modification du sens d'un terme, etc.). Par le biais de l'éditeur d'ontologies du système, l'ontologue apportera la modification souhaitée. Un exemple de changement est présenté dans la section précédente (figure V-2), nous avons fait une opération de déplacement de lien de dénotation du terme *T_Properties* pour dénoter un nouveau concept *ConfigFile*. Pour information, Le concept *ConfigFile* exprime un fichier de configuration et le terme T_Properties exprime un type de fichier dans lequel on peut stocker des informations de paramétrage pour pouvoir les modifier sans avoir à recompiler.

La figure V-3 présente un diagramme de séquence qui illustre un besoin de changement sur la RTO.

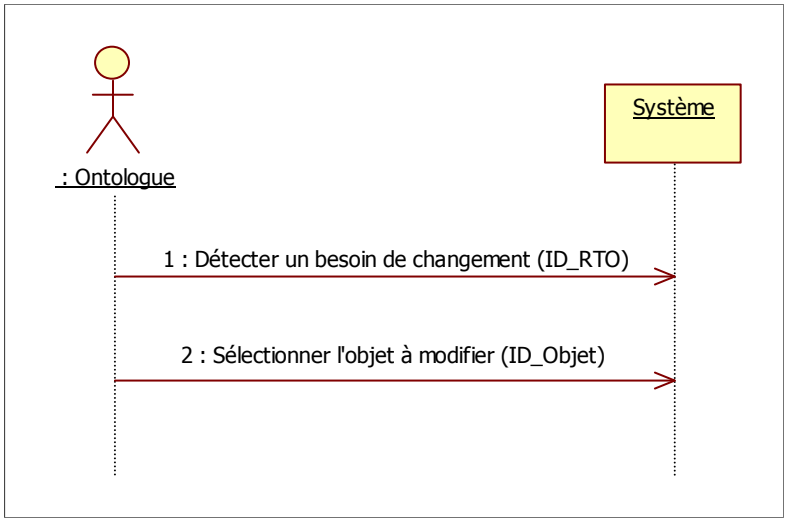


Figure V - 3 : Diagramme de séquence qui illustre un besoin de changement sur la RTO

5.4.2 Dans le contexte de vérification des annotations

Un autre scénario d'évolution correspond à l'ajout de nouveaux documents dans le système. Les termes, pertinents pour l'annotation, contenus dans ces nouveaux documents peuvent ne pas être représentés dans la RTO ou leur sens peut avoir évolué. La RTO n'étant pas en adéquation avec le texte, l'outil d'annotation ne pourra pas être efficace. La RTO devra donc évoluer. Plus généralement, ceci devra être le cas lorsque le système n'est pas capable d'annoter correctement un document. Une mauvaise annotation peut être détectée par l'ontologue qui pourra demander une évolution de la RTO. Un diagramme de séquence illustrant un besoin en changement détecté par l'ontologue suite à l'analyse d'une annotation est présenté dans la figure V-4.

Une mauvaise annotation peut également être détectée automatiquement par le système. Le système utilise alors des critères de qualité des annotations qui peuvent être propres au corpus et au domaine d'application. Si ces critères ne sont pas vérifiés, le système peut identifier un besoin en changement de la RTO, le signaler à l'ontologue, voire lui faire des suggestions d'évolution. Un diagramme de séquence illustrant un besoin de changement détecté

automatiquement par le système suite à l'analyse des annotations est présenté dans la figure V-5. Le processus est décrit plus en détail dans le chapitre VI.

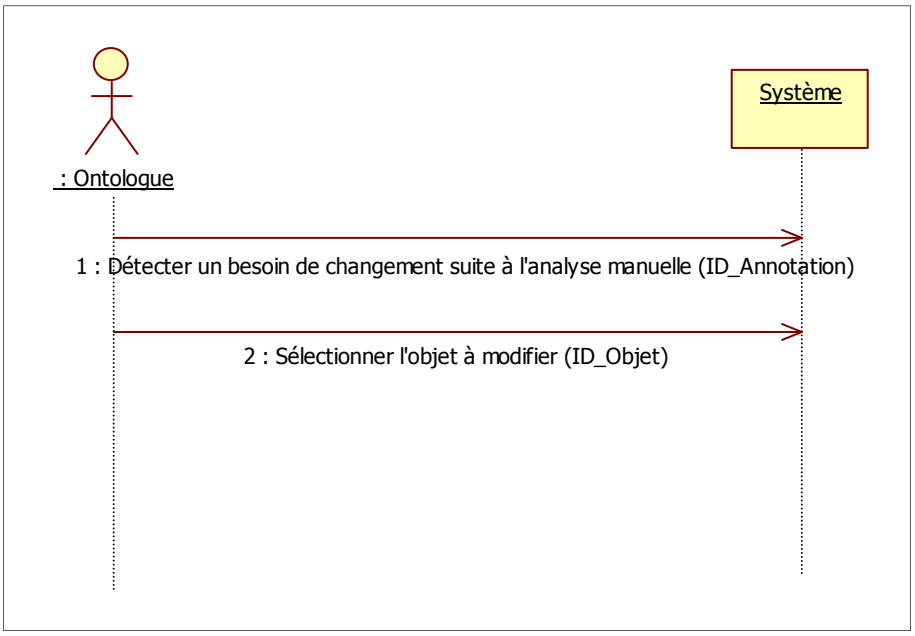


Figure V - 4 : Diagramme de séquence illustrant un besoin en changement détecté par l'ontologue à partir de l'analyse des annotations

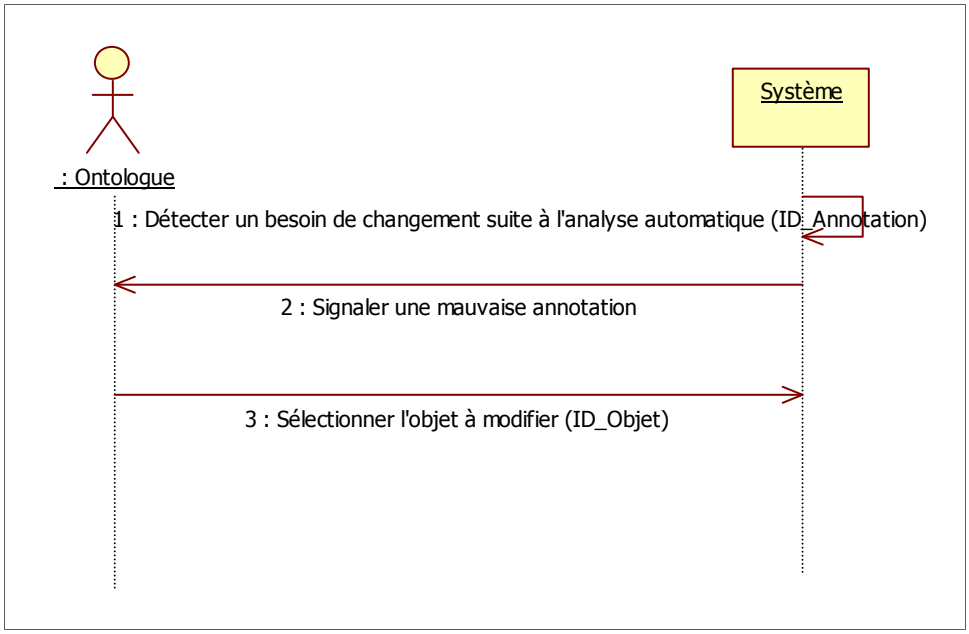


Figure V - 5 : Diagramme de séquence illustrant un besoin en changement détecté automatiquement par le système à partir de l'analyse des annotations

5.4.3 Typologies des changements

Pour exprimer l'évolution, nous avons besoin de recenser l'ensemble des modifications que l'on peut apporter aux ressources termino-ontologiques. Cette typologie est essentiellement un ensemble des changements typés, chaque changement ayant une signification bien précise.

Les changements peuvent être élémentaires, comme l'ajout ou l'effacement d'une entité de la RTO et comme nous l'avons présenté dans le chapitre IV, un changement élémentaire est celui qui peut être identifié en analysant uniquement la structure conceptuelle (concept, relation, attribut) ou lexicale (terme) de la RTO.

Les changements peuvent être aussi complexes, comme le déplacement ou la fusion d'une ou plusieurs entités de la RTO et comme nous l'avons présenté dans le chapitre IV, chaque changement complexe est la conséquence de l'association de deux ou plusieurs changements élémentaires. Théoriquement, l'ensemble de changements complexes est infini, parce que le nombre de combinaisons possibles est aussi infini.

Selon [Rogozan, 2008], les changements complexes possèdent une sémantique plus importante que les élémentaires. Par exemple, la sémantique d'un changement complexe de division *SplitConcept* est clairement plus riche que l'ensemble de changements élémentaires formé d'un seul *DeleteConcept* et de plusieurs *AddConcept*. De ce fait, les changements complexes peuvent mieux capter l'intention exacte des acteurs ayant fait évoluer la RTO et peuvent, par la suite, offrir une vue pertinente sur la vraie signification de l'évolution.

En partant du modèle de RTO [Reymonet *et al.*, 2007], nous proposons une typologie des changements applicables [Tissaoui, 2009] [Tissaoui *et al.*, 2011] [Tissaoui *et al.*, 2013] à une RTO définie selon ce modèle. Cette typologie élargit la conceptualisation de [Stojanovic, 2004] et [Luong, 2007], par l'ajout d'un certain nombre de caractéristiques dont la plus notable est la prise en compte du lexique de la RTO qui peut évoluer à cause des changements

portant sur les termes et les liens de dénotations. Nous avons aussi introduit un certain nombre de caractéristiques afin de mieux spécifier le type et l'effet de chaque changement sur la RTO (discutées en détail dans section 5.6).

Avant de mettre en œuvre un changement, nous proposons de vérifier les pré-conditions associées au changement et définies dans la typologie, afin de présenter à l'ontologue l'éventueles incohérences provoquées par la mise en œuvre du changement.

Nous proposons de définir deux types de changement : les changements élémentaires et les changements complexes.

- Tout d'abord, lorsque l'ontologue choisit d'effectuer un changement complexe le système pourra lui présenter directement les conséquences de ce changement au lieu de laisser imaginer l'enchainement des conséquences des changements élémentaires composant le changement complexe. Prenons l'exemple de *MergeConcept* qui permet de fusionner deux concepts dans la hiérarchie des concepts. L'opération *MergeConcept* est une séquence de changements élémentaires à savoir plusieurs *DeleteConcept* et un seul *AddConcept*. Prenons l'exemple ci-dessus (section 5.3), l'ontologue décide de fusionner les concepts *Unzoom_Function* et *Zoom_Function*, en un seul concept *Display_Function*. Dans un premier temps, l'ontologue doit supprimer les concepts *Unzoom_Function* et *Zoom_Function ;* dans un deuxième temps, il crée le concept *Display_Function* comme concept fils du concept *Function*.

Par conséquent, avec EvOnto et en utilisant notre typologie de changements, l'ontologue peut visualiser une seule opération, à savoir *MergeConcept* au lieu de visualiser deux ou plusieurs opérations telles que plusieurs *DeleteConcept* et un seul *AddConcept.*

En effet, un changement complexe facilite la validation collective de l'évolution de la RTO. Prenons le même exemple de *MergeConcept* (*Unzoom_Function*, *Zoom_Function*,

Display_Function). L'ontologue peut valider directement un seul changement complexe à savoir *MergeConcept* au lieu de valider plusieurs changements, comme dans notre exemple, la somme de plusieurs changements élémentaires à savoir *DeleteConcept (Unzoom_Function)*, *DeleteConcept (Zoom_Function)* et un seul *AddConcept (Display_Function)*.

Ensuite, un changement complexe permet à l'ontologue d'analyser au mieux les conséquences sur la conceptualisation de la RTO et d'étudier également les effets sur les annotations sémantiques des documents reposant sur cette ressource. Prenons l'exemple de la fusion de deux concepts C1 et C2, et les remplaçant par un nouveau concept NewC. Sachant que les sous concepts des concepts C1 et C2 ont été rattachés au concept NewC et sachant aussi que certains documents du corpus sont annotés par le concept C1 et d'autres par le concept C2, on peut déduire que les documents annotés par les concepts C1 et C2 peuvent maintenant être annotés par le concept NewC puisque celle-ci contient les instances de la Classe C1 et C2. Cette solution dépend de la stratégie choisie par l'ontologue s'il y en a plusieurs.

Enfin, du côté application, la distinction entre un changement élémentaire et un changement complexe facilite l'implémentation de notre prototype EvOnto dans la mesure où nous utilisons l'encapsulation de méthodes afin de faire des appels de fonctions. Par exemple, en définissant un changement complexe, nous pouvons faire appel à des méthodes ou des fonctions déjà définies dans un changement élémentaires ou un autre changement complexe.

Nous indiquons ci-dessous les différents types de changements et leur représentation dans la typologie. Cette typologie est utilisée dans l'implémentation du prototype.

5.4.4 Changements élémentaires

L'ensemble des changements élémentaires proposés par [Stojanovic, 2004] [Luong, 2007] ne contient que des opérations additives et soustractives des concepts et des propriétés qui ne permettent pas d'exprimer les relations entre la composante conceptuelle de la RTO (concept, relation, attribut) et lexicale (terme et lien de dénotation).Par conséquent, nous établissons une classification des changements élémentaires permettant de représenter tous les types de changements de RTO possibles sur le concept, la propriété, le terme et le lien de dénotation (Tableau V-1).

Tableau V - 1 : Liste des changements élémentaires

Changements élémentaires	
Opérations appliquées aux concepts	
Syntaxe de l'opération	**Sémantique de l'opération**
RenameConcept (OldC, NewC)	Remplacer le nom du concept modifié OldC par le nouveau nom NewC.
DeleteConcept (c)	Supprimer le concept c.
CreateConcept (NewC)	Créer le nouveau concept NewC.
CreateSubConcept (NewC, OldC)	Créer le nouveau concept NewC comme fils du concept OldC.
Create Hierarchy Concept Link (OldC1 , OldC2)	Créer un lien de subsomption entre les concepts OldC1 et OldC2,
Delete Hierarchy Concept Link (c1,c2)	Supprimer le lien de subsomption entre les concepts OldC1 et OldC2.
Opérations appliquées aux propriétés	
Syntaxe de l'opération	**Sémantique de l'opération**
RenameProperty (Oldp, Newp)	Remplacer l'identifiant de la propriété Oldp par le nouveau identifiant Newp.
DeleteProperty (p)	Supprimer la propriété p.
CreateProperty (Newp)	Créer la nouvelle propriété Newp.
CreateSubProperty (Newp, p)	Insérer la nouvelle propriété Newp comme fils de la propriété p.
Opérations appliquées aux termes	
Syntaxe de l'opération	**Sémantique de l'opération**
DeleteAssociatedTerm (OldT, OldC)	Supprimer le terme OldT associé au concept OldC.
CreateAssociatedTerm (NewT, OldC)	Créer le nouveau terme NewT et l'associer au concept sélectionné OldC (ajouter un nouveau terme à partir de la sélection d'un concept de la hiérarchie de concepts).
CreateTerm (NewT, (OldC or NewC))	Créer le nouveau terme NewT et l'associer à un concept existant OldC ou un nouveau concept NewC (ajouter un nouveau terme souvent à partir de la sélection du terme dans une fiche).
Opérations appliquées aux liens de dénotations	
Syntaxe de l'opération	**Sémantique de l'opération**
DeleteDenotationLink (ld, OldT, OldC)	Supprimer le lien de dénotation ld entre le terme OldT et le concept OldC.
CreateDenotationLink (ld, OldT, OldC)	Créer le nouveau lien de dénotation entre le terme OldT et le concept OldC.

En se basant sur le modèle de la RTO décrit dans la section 5.2, nous décrivons pour chaque type de changement la syntaxe, les paramètres, la sémantique de ce changement ainsi que les conditions qui doivent être satisfaites avant et après la mise en œuvre du changement. Nous présentons dans le tableau V-2 deux exemples d'une description du changement, *DeleteConcept (c)* qui permet de supprimer le concept c et *DeleteAssociatedTerm(t)* qui permet de supprimer le terme t associé au concept c.

Tableau V - 2 : Description d'un changement élémentaire

Type de changement : *DeleteConcept*	
Syntaxe	Sémantique
DeleteConcept (c)	Supprimer le concept c
Pré-condition	Post-condition
c∈ C	c∉ C

Type de changement : *DeleteAssociatedTerm*	
Syntaxe	Sémantique
DeleteAssociatedTerm (t, c)	Supprimer le terme t associé au concept c.
Pré-condition	Post-condition
$t \in T^C$, c∈ C, dénote (t) = c	$t \notin T^C$, c ∈ C, dénote (t) ≠ c

Une capture d'écran d'EvOnto contenant le 'popup_menu', que nous avons ajouté au panel de la hiérarchie des concepts et qui propose les changements élémentaires, est présenté dans la figure V-6 (rectangle pointillé en couleur rouge). Ce popup_menu présente les différents changements élémentaires que nous avons développés à savoir *CreateTerm*, *CreateAssociatedTerm*, *DeleteAssociatedTerm*, *RenameConcept*, *CreateSubConcept*, *DeleteConcept*, *CreateHierarchieConceptLink* et *DeleteHierarchyConceptLink*.

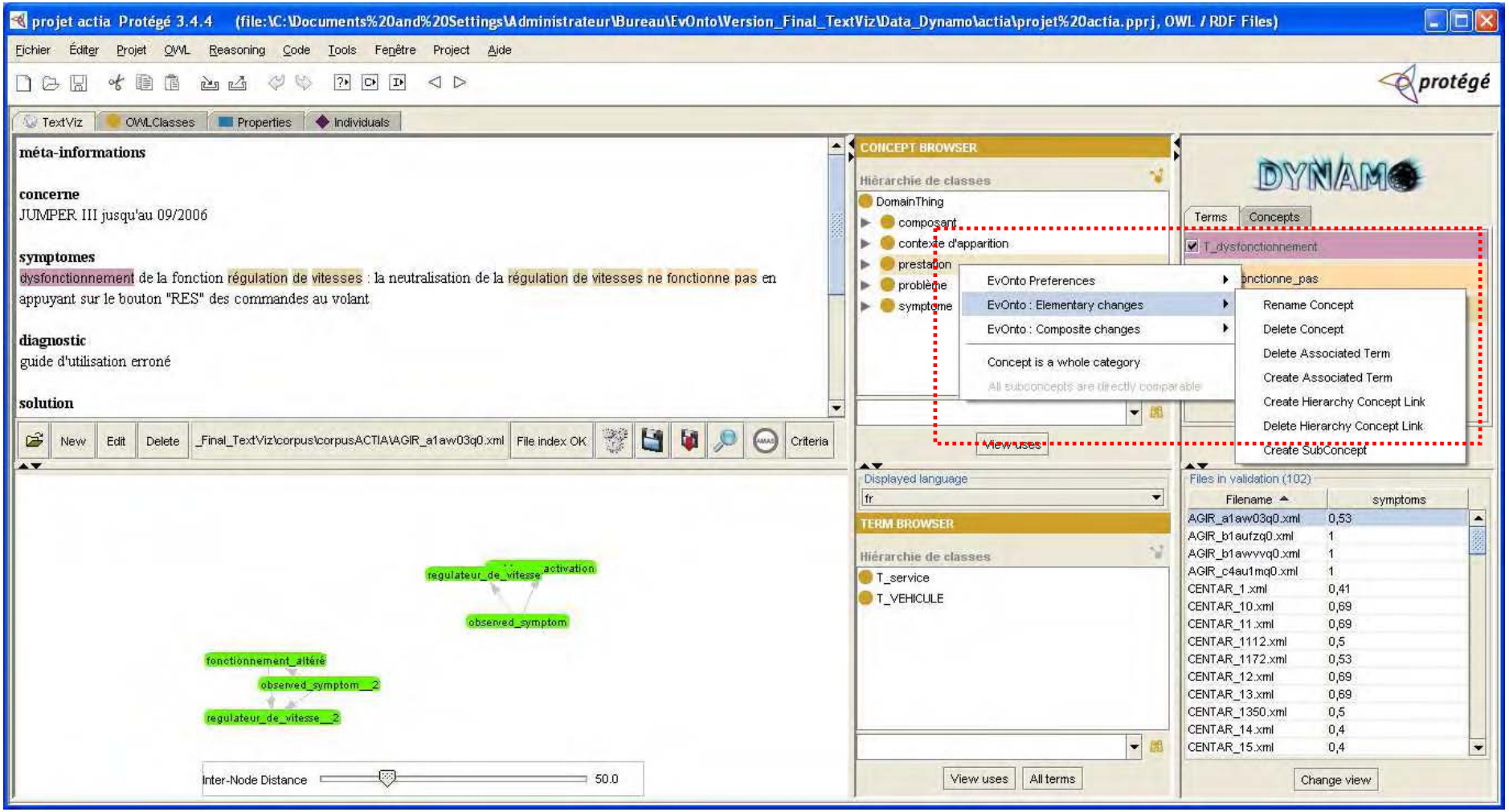


Figure V -6 : Capture d'écran d'EvOnto : présentation des changements élémentaires

5.4.5 Changements complexes

Comme pour les changements élémentaires et en se basant sur le modèle de la RTO décrit dans la section 5.2, nous décrivons pour chaque type de changement la syntaxe, les paramètres, la sémantique de ce changement ainsi que les conditions qui doivent être satisfaites avant et après la mise en œuvre du changement. Nous présentons dans le tableau V-3 deux exemples d'une description du changement, *MergeConcept (c1, c2, Newc)* qui permet de fusionner deux concepts existants et *MoveAssociatedTerm (t, Oldc1, Oldc2)* qui permet de déplacer le lien de dénotation qui associe le terme t au concept Oldc à la nouvelle position pour associer le terme t au nouveau concept Newc.

Tableau V - 3 : Description d'un changement complexe

Type de changement : *MergeConcept*	
Syntaxe	Sémantique
MergeConcept (c1, c2, Newc)	Fusionner les concepts c1 et c2, et les remplacer par un nouveau concept Newc.
Pré-condition	Post-condition
c1 et c2 ∈ C, Newc ∉ C, (c1, c2) ∉ H^C	c1 et c2 ∉ C, Newc ∈ C

Type de changement : *MoveAssociatedTerm*	
Syntaxe	Sémantique
MoveAssociatedTerm (t, Oldc1, Oldc2)	Déplacer le terme t qui dénote le concept Oldc1 à la nouvelle position pour dénoter le concept Oldc2.
Pré-condition	Post-condition
Oldc1 et Oldc2 ∈ C, t ∈ T^C, dénote (t) = Oldc1.	Oldc1 et Oldc2 ∈ C, t∈ T^C, dénote (t) = Oldc2.

Nous établissons une classification des changements complexes permettant de représenter tous les types de changements de RTO possibles sur le concept, la propriété, le terme et le lien de dénotation (Tableau V-4). Cinq changements complexes ont été développés dans notre prototype à savoir *MoveAssociatedTerm*, *MoveConcept*, *MergeConcept*, *SplitConcept* et *GroupConcept*.

Une capture d'écran d'EvOnto qui contient le 'popup_menu' que nous avons ajouté au panel de la hiérarchie des concepts présente les changements complexes est présenté dans la figure V-7 (rectangle pointillé en couleur rouge). La description complète des autres changements élémentaires et complexes est présentée dans la partie Annexe A.

Tableau V - 4 : Liste des changements complexes

Changements complexes	
Opérations appliquées aux concepts	
Syntaxe de l'opération	**Sémantique de l'opération**
MoveConcept (c, OldC, NewC)	Déplacer le concept c qui est le fils du concept OldC à la nouvelle position comme fils du concept NewC.
MergeConceptWith (OldC1, OldC2, NewC)	Fusionner les concepts OldC1 et OldC2, et les remplaçant par un nouveau concept NewC.
SplitConcept (OldC, NewC1, NewC2)	Diviser le concept OldC en deux nouveaux concepts NewC1 et NewC2.
GroupConcept With (NewC, OldC1, OldC2)	Créer un super-concept commun NewC pour les concepts OldC1 et OldC2
CreateMultipleSubConcept (c1, c2, …, cn, c)	Insérer les nouveaux concepts c1, c2,…cn comme fils du concept c
Opérations appliquées aux propriétés	
Syntaxe de l'opération	**Sémantique de l'opération**
MoveProperty (p, OldP, NewP)	Déplacer la propriété p qui est la sous-propriété de OldP à la nouvelle position comme la sous-propriété de NewP
MergeProperty With (OldP1, OldP2, NewP)	Fusionner les propriétés OldP1 et OldP2, et les remplaçant par une nouvelle propriété NewP. Transférer les domaines/co-domaines OldP1 et OldP2 vers New P
SplitProperty (OldP, NewP1, NewP2)	Diviser une propriété OldP en deux nouvelles propriétés NewP1et NewP2, et distribuer les concepts domaines/co-domaines de p entre ces nouvelles propriétés.
Opérations appliquées aux termes	
Syntaxe de l'opération	**Sémantique de l'opération**
MoveAssociatedTerm (OldT, OldC, NewC)	Déplacer le lien de dénotation qui associe le terme OldT au concept OldC à la nouvelle position pour associer le terme OldT au nouveau concept NewC.

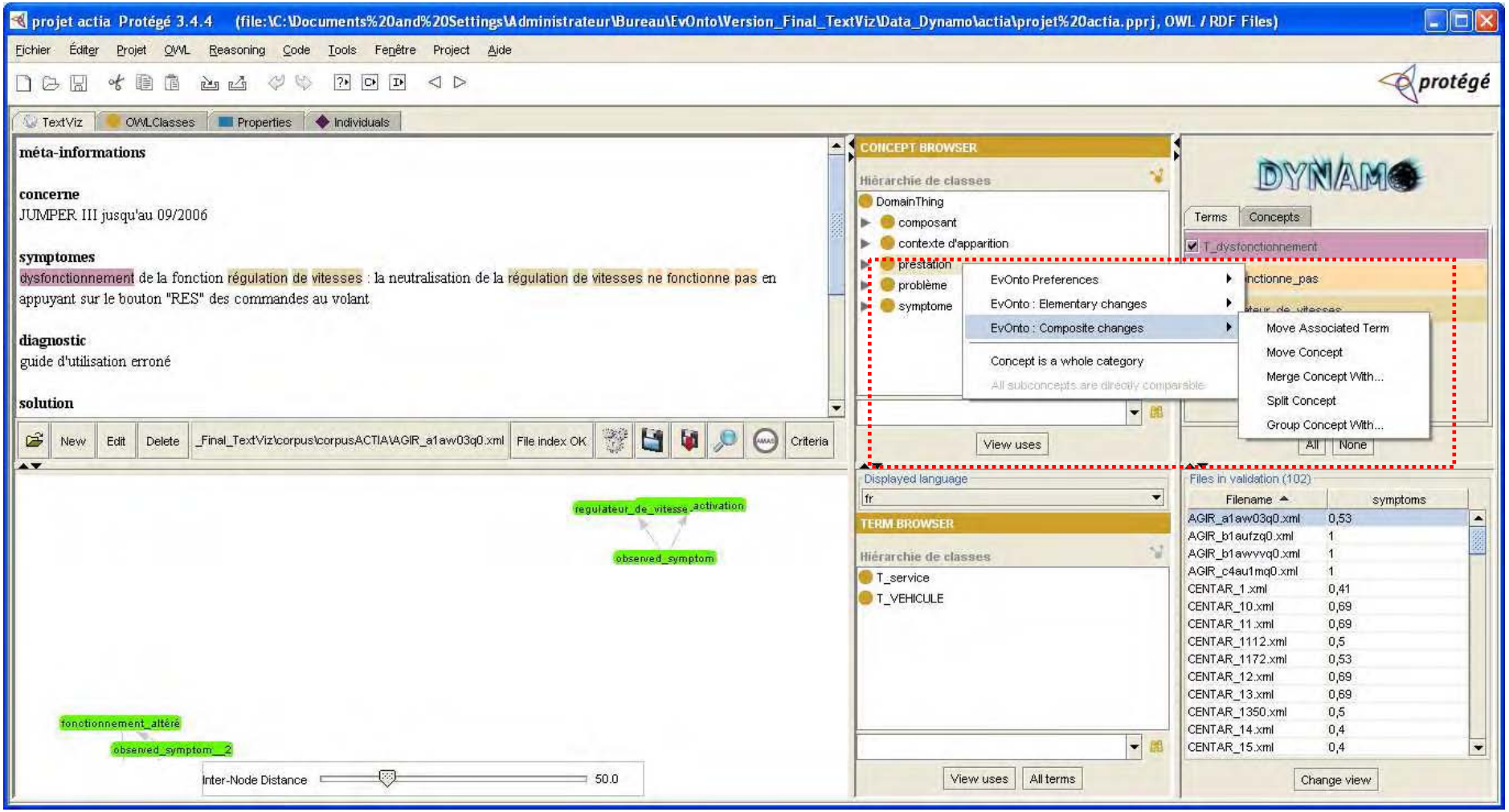


Figure V - 7 : Capture d'écran d'EvOnto : présentation des changements complexes

5.5 Présentation des stratégies et simulation de leurs conséquences

Comme nous l'avons vu dans la section précédente, l'ontologue identifie par le biais de l'éditeur d'ontologies le changement qu'il souhaite mettre en œuvre ainsi que l'élément de la RTO sur lequel il porte.

Chaque changement 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. Afin de permettre à l'ontologue d'anticiper ces conséquences, nous avons défini pour chaque changement un ensemble de stratégies.

En effet pour chaque type d'évolution dans la RTO, il est possible de définir un ensemble de changements supplémentaires qui peuvent être effectués pour garder la ressource dans un état cohérent. Lorsque plusieurs effets sont possibles pour une même évolution, un choix doit être effectué soit arbitrairement par le système de gestion des évolutions dans lequel on aura fixé un des effets possibles, soit par l'ontologue. La manière de faire intervenir l'ontologue pour formuler le choix d'un de ces effets (au cas par cas, une fois pour toutes, …) revient à fixer une stratégie d'évolution au sens de [Stojanovic, 2004]. Dans la section suivante, nous présentons nos stratégies d'évolution de la RTO qui sont différents par rapport aux stratégies de résolutions de Stojanovic

5.5.1 Stratégie d'évolution de la ressource termino-ontologique

Pour chaque type de changement, nous avons identifié différentes stratégies d'évolution, c'est-à-dire différentes 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 (figure V-8). 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 ou détruits. Les stratégies sont présentées à l'ontologue avec leurs conséquences. Pour chaque opération de changement, une stratégie par défaut est affichée à l'ontologue. L'ontologue a bien sûr la possibilité de choisir une autre stratégie s'il le souhaite. Il peut également parcourir l'ensemble des stratégies proposées pour évaluer celle qui lui convient. Pour chaque stratégie sélectionnée, ses conséquences sont simulées sur la RTO. Ce n'est qu'une fois que l'ontologue choisit explicitement d'exécuter la stratégie que le changement et ses conséquences seront effectifs dans l'ontologie.

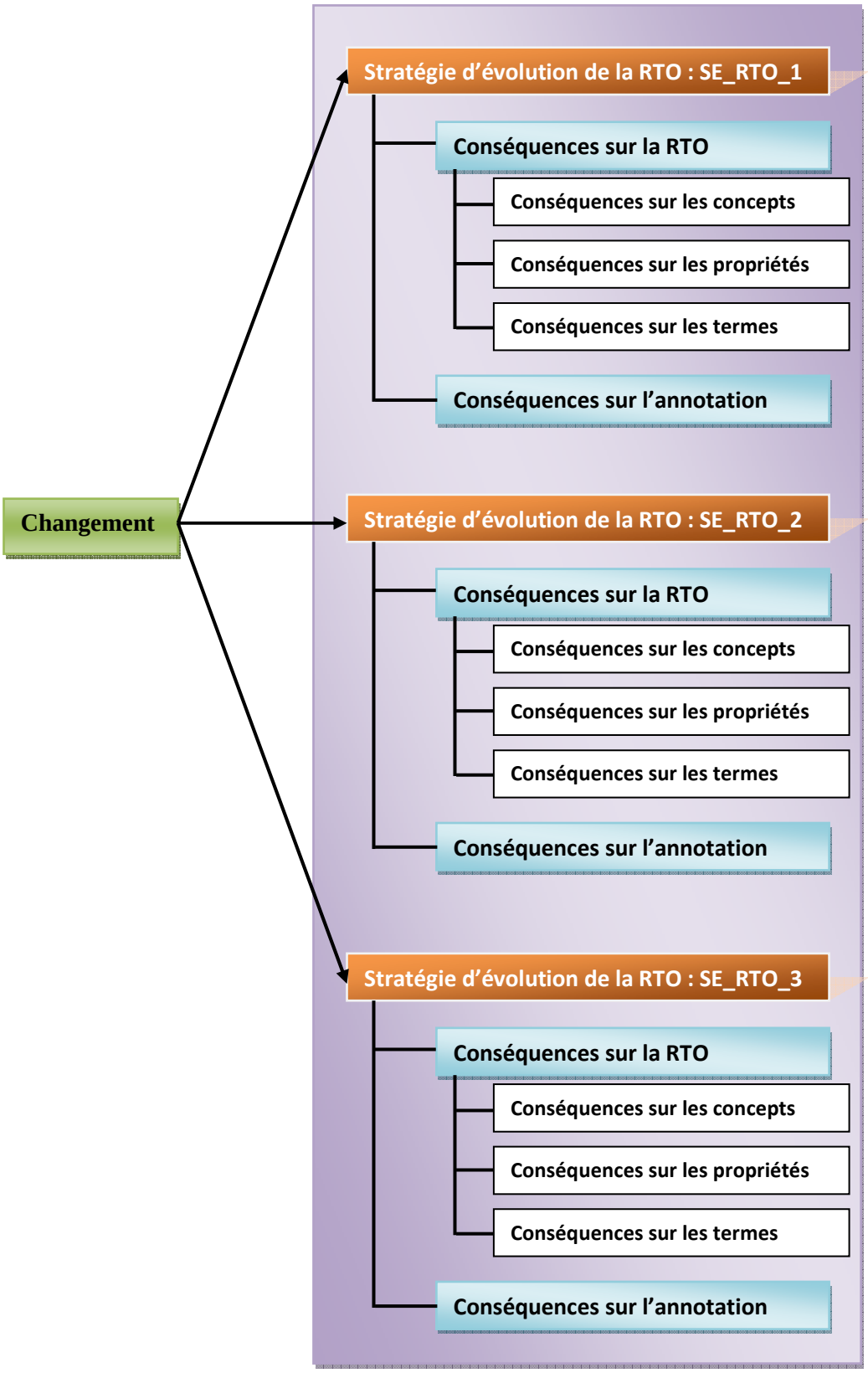


Figure V - 8 : Stratégies d'évolution pour la ressource termino-ontologique

Selon la figure V-8, chaque stratégie (SE_RTO_1, SE_RTO_2, SE_RTO_3) correspond à un ensemble additionnel de changements résolvant l'incohérence en tenant compte d'un ensemble de besoins particuliers de la ressource. Cela veut dire que les différents ensembles de changements additionnels (conséquences sur les concepts, sur les propriétés et sur les termes) seront également générés (si l'ontologue valide le choix de cette stratégie) pour compléter le changement de la RTO initial.

Pour chaque stratégie d'évolution sélectionnée, le système présente à l'utilisateur d'une part les conséquences engendrées par ce changement sur la RTO et d'autre part les effets du changement sur les annotations sémantiques d'un autre côté (chapitre VI). Ces informations lui permettent de choisir la stratégie qui lui semble la mieux adaptée. Par exemple, si l'utilisateur sélectionne une autre stratégie, le système affiche les changements secondaires qui seront appliquées avec le changement de base.

5.5.2 Exemple de stratégie d'évolution

Revenons à l'exemple d'un extrait de l'ontologie ARTAL (figure V-1 et figure V-2), si nous supprimons le concept *Software_Programming_Component*, qui est un concept intermédiaire (cas 4 du tableau V-5), nous pourrions avoir les différentes stratégies suivantes pour traiter les conséquences de la suppression du concept.

Dans l'exemple, trois stratégies sont possibles pour gérer les concepts fils du concept détruit par DeleteConcept (figure V-9, rectangle pointillé A en couleur rouge) :

- ***SE_RTO_1*** : Attacher les sous concepts au concept père de celui détruit (stratégie par défaut)
- ***SE_RTO_2*** : Attacher les sous concepts au DomainThing
- ***SE_RTO_3*** : Supprimer les sous concepts

Les conséquences sur l'ensemble des éléments de la RTO de la stratégie par défaut pour DeleteConcept sont les suivantes (figure V-10, rectangle pointillé B en couleur rouge) :

- Le concept [Component] ne sera plus le super-concept de concept [Software_Programming_Component] ;
- Le concept [Software_Programming_Component] ne sera plus le super-concept des concepts [Manager, Internet_Component[21], Lib, Command, File, …] qui seront maintenant des sous-concepts de [Component] ;
- Les termes [Linux, Procedure, script, Sentence, Tag, Position_Polling_Mechanism, Plugins_Directory] ne seront plus associés au concept [Software_Programming_Component] mais au concept [Lib] ;

Le choix du concept Lib est arbitraire. En fait, le système ne peut pas savoir quel concept ces termes vont dénoter. Le système choisit un concept parmi la liste des concepts fils de façon arbitraire. Pour aider l'ontologue à faire le meilleur choix, nous avons proposé dans EvOnto l'étape d'ajustement (voir section 5.6) des entités de la RTO (concepts, propriétés, termes) qui permet d'ajuster la solution de base proposé par le système.

Si l'ontologue choisit[22] la deuxième stratégie *SE_RTO_2*– Attacher les sous concepts au DomainThing, les conséquences sur l'ensemble des éléments de la RTO de la stratégie sont les suivantes :

- Le concept [Component] ne sera plus le super-concept de concept [Software_Programming_Component] ;

[21] Les concepts Manager et Internet_Component sous concepts de software_Programming_Component qui n'étaient pas présentés dans la figure V-1, sont détaillés dans la figure V-9 (application réel de l'exemple avec EvOnto).

22 On pourrait penser que l'utilisateur ne choisira pas cette stratégie puisque les sous-concepts du concept supprimé C semblent être naturellement sous-concepts du sur-concept de C, ici Component. Mais ce n'est le cas que si la modélisation initiale était correcte et ce n'est peut-être pas le cas. Cela pourrait être même la raison de la suppression de C.

- Le concept [Software_Programming_Component] ne sera plus le super-concept des concepts [Manager, Internet_Component, Lib, Command, File, …] qui seront maintenant des sous-concepts de [DomainThing] ;
- Les termes [Linux, Procedure, script, Sentence, Tag, Position_Polling_Mechanism, Plugins_Directory] ne seront plus associés au concept [Software_Programming_Component] mais au concept [Lib] ;

Si l'ontologue choisit la troisième stratégie *SE_RTO_3* – Supprimer les sous concepts, les conséquences sur l'ensemble des éléments de la RTO de la stratégie sont les suivantes :

- Le concept [Component] ne sera plus le super-concept de la sous hiérarchie du concept [Software_Programming_Component] ;
- Les termes [Linux, Procedure, script, Sentence, Tag, Position_Polling_Mechanism, Plugins_Directory] seront supprimés ;
- Les sous concepts [Manager, Internet_Component, Lib, Command, File, …] seront supprimés ;
- Tous les termes qui sont associées aux concepts de la sous hiérarchie du concept [Software_Programming_Component] seront supprimés ;

Pour guider au mieux l'ontologue, nous avons défini toutes les stratégies d'évolutions correspondant aux changements élémentaires et complexes utilisées dans notre travail. Le Tableau V-5 représente les stratégies proposées pour le changement élémentaire de suppression d'un concept de la RTO. Dans ce tableau, le concept Top égale au DomainThing, pour i de 1 à n, tous les Ci sont différent de Top et pour i de 1 à n, Fi sont les fiches annotées avec n = 3 (F1, F2 et F3). t1, t2, t3 et t4 $\in$ T^C avec T^C ensemble des termes qui désignent les concepts. La fiche F1 contient t1, la fiche F2 contient t1 et t2 et la fiche F3 contient t3 et t4.

La description complète des stratégies d'évolution des changements de la RTO est présentée dans la partie Annexe A.

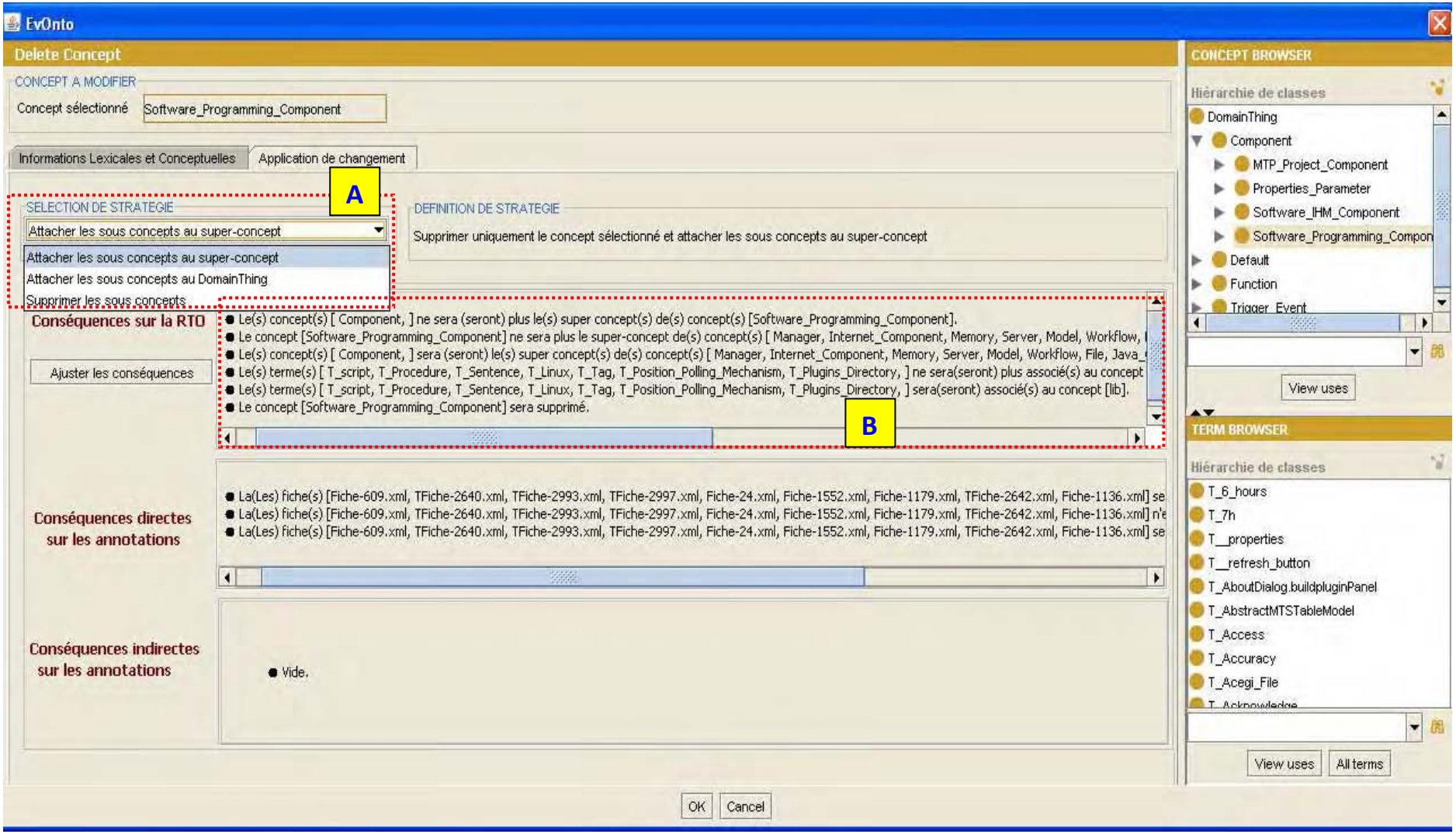


Figure V - 9 : Capture d'écran d'EvOnto : présentation de l'opération DeleteConcept

Tableau V - 5 : Stratégies d'évolution pour le changement DeleteConcept

Changement n° 1	**Fiche de cas du changement**		
Identification de la fiche			
Nom fiche de cas de changements	DeleteConcept		
Syntaxe	DeleteConcept (C1)		
Sémantique	Supprimer le concept C1		
Pré conditions	$C1 \in C$		
Post conditions	$C1 \notin C$		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
Cas 1 : Si C1 est un concept feuille, possédant un concept père qui est Top $C1 \in C, (C1, Top) \in H^C$ $\neg \exists C2 \in C : (C2, C1) \in H^C$ dénote (t1) = C1, dénote (t2) = C1 $p \in P$ $C1 \in$ domaine (p) ou $C1 \in$ co-domaine (p)	Il n'y a pas de stratégie proposée par le système	➢ t1 et t2 ne seront plus associés à C1 mais seront supprimés. ➢ C1 n'appartient plus au domaine ou au co-domaine de p. ➢ p sera supprimée si le domaine ou le co-domaine ne contient que C1. ➢ C1 sera supprimé.	➢ [F1, F2] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par C1; ➢ [F1, F2] seront annotées par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;
Cas 2 : Si C1 est un concept feuille, possédant un concept père C2 différent du Top $C1, C2 \in C, (C1, C2) \in H^C$ $\neg \exists C3 \in C :(C3, C1) \in H^C$ dénote (t1) = C1, dénote (t2) =	Il n'y a pas de stratégie proposée par le système	➢ t1 et t2 ne seront plus associés à C1 mais seront supprimés. ➢ C1 n'appartient plus au domaine ou au co-domaine de p. ➢ p sera supprimée si le domaine ou le co-domaine ne contient que C1. ➢ C1 sera supprimé.	➢ [F1, F2] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par C1; ➢ [F1, F2] seront annotées par C2 ;

C1 p∈P C1 ∈ domaine (p) ou C1 ∈co-domaine (p)			
Cas 3 : Si C1 est un concept intermédiaire, possédant un concept père qui est Top C1, C2 ∈ C (C1, Top) ∈ H^C, (C2, C1) ∈ H^C dénote (t1) = C1, dénote (t2) = C1 p ∈P C1 ∈ domaine (p) ou C1 ∈co-domaine(p) **Note** : *S'il y a d'autres sub-concepts de C1, le traitement de ses sub-concepts est pareil que c1* .	***SE_RTO_1 :** Attacher les sous concepts au Top*	• C1 ne sera plus le super concept du C2. • C2 sera attaché au Top. • t1 et t2 ne seront plus associés à C1 mais seront supprimés. • C1 n'appartient plus au domaine ou au co-domaine de p. • p sera supprimée si le domaine ou le co-domaine ne contient que C1. • C1 sera supprimé.	• [F1, F2] seront invalidées ; • [F1, F2] ne seront plus annotées par C1; • [F1, F2] seront annotées par C2 ;
	***SE_RTO_3:** Supprimer les sous C1*	• C1 ne sera plus le super concept du C2. • t1 et t2 ne seront plus associés à C1 mais seront supprimés. • p sera supprimée si le domaine ou le co-domaine ne contient que C1. • C2 sera supprimé. • C1 sera supprimé. .	• [F1, F2] seront invalidées ; • [F1, F2] ne seront plus annotées par C1; • [F1, F2] seront annotées par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;
Cas 4 : Si C1 est un concept intermédiaire, possédant un concept père différent du Top	***SE_RTO_1 :** Attacher les sous concepts au super concept*	• C1 ne sera plus le super concept du C2. • C2 sera attaché à C. • t1 et t2 ne seront plus associés au concept	• [F1, F2] seront invalidées ; • [F1, F2] ne seront plus annotées par C1 ;

C, C1, C2 ∈ C, (C, Top) ∈ H^C (C1, C) ∈ H^C, (C2, C) ∈ H^C, dénote (t1) = C1, dénote (t2) = C1 p∈P C1 ∈ domaine (p) ou C1 ∈co-domaine(p)		C1 mais à C2. C1 n'appartient plus au domaine ou au co-domaine de p. C2 sera ajouté au domaine de p. C1 sera supprimé.	[F1, F2] seront annotées par C2;
	SE_RTO_2 : *Attacher les sous concepts au Top*	C1 ne sera plus le super concept du C2. C2 sera attaché au Top. t1 et t2 ne seront plus associés à C1 mais à C2. C1 n'appartient plus au domaine ou au co-domaine de p. C2 sera ajouté au domaine de p. C1 sera supprimé.	Les fiches [F1, F2] seront invalidées ; Les fiches [F1, F2] ne seront plus annotées par C1 ; Les fiches [F1, F2] seront annotées par C2;
	SE_RTO_3 : *Supprimer les sous concepts*	C1 ne sera plus le super concept du C2. t1 et t2 ne seront plus associés à C1 mais seront supprimés. C1 n'appartient plus au domaine ou au co-domaine de p. p sera supprimée si le domaine ou le co-domaine ne contient que C1. C2 sera supprimé. C1 sera supprimé. *Note : Traiter ensuite le concept fils C2 de la même façon*	[F1, F2] seront invalidées ; [F1, F2] ne seront plus annotées par C1; [F1, F2] seront annotées par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;

5.6 Choix de la stratégie et ajustement des conséquences

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 la RTO impactés par le changement et les documents annotés par l'élément modifié, mais aussi les documents annotés par les éléments de la RTO impactés par cette modification (figure V-9). 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 guide pour ajuster les conséquences ou revoir la modification demandée.

La figure V-10 montre l'interface d'adaptation des conséquences du changement DeleteConcept après plusieurs modifications manuelles. Pour chaque type de données à modifier (sous-concepts, termes, 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.

L'ontologue a le droit de choisir entre la validation de changement de base et les changements supplémentaires directement sans ajustement ou d'ajuster les changements supplémentaires et ensuite, il valide ces changements.

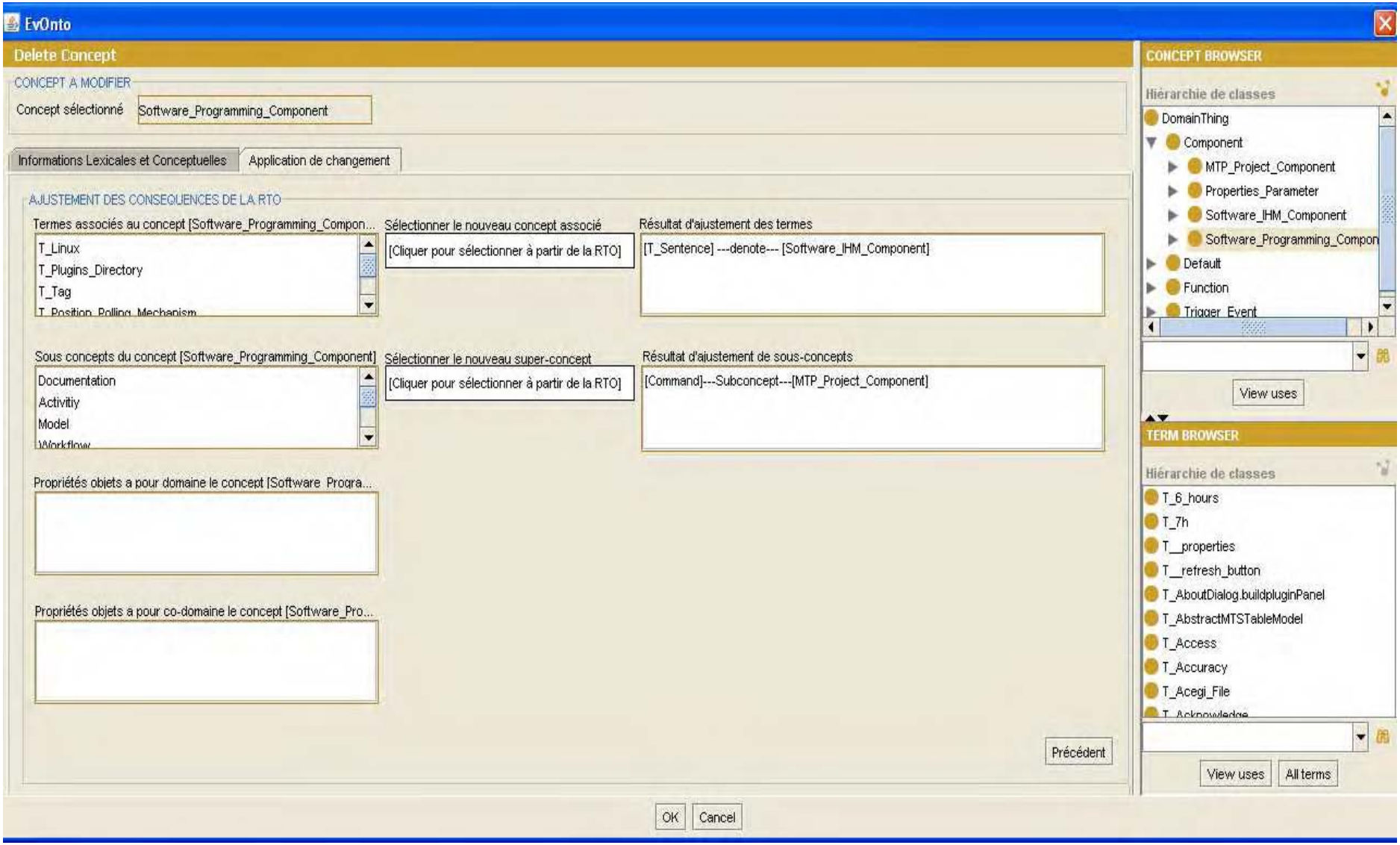


Figure V - 10 : Capture d'écran d'EvOnto pour adapter les conséquences sur d'autres composants de la RTO du changement DeleteConcept

5.7 Application du changement à la RTO

Dans cette phase, la séquence des changements est validée par l'ontologue afin d'obtenir une nouvelle version de la RTO. Si l'ontologue est d'accord pour appliquer ces différentes conséquences, il valide l'opération et le système effectue tous les changements pour obtenir la nouvelle version de la RTO. Si l'utilisateur n'est pas d'accord, le système donne le droit à l'utilisateur d'ajuster les conséquences sur les entités liées à l'objet modifié.

Après avoir appliqué ces changements, nous obtenons la nouvelle version de la RTO dans laquelle certains éléments ont été changés par rapport à son ancienne version.

Comme nous l'avons déjà mentionné dans le chapitre IV, le fait de prendre en compte les termes et les annotations sémantiques utilisant la RTO renouvelle les questions classiques soulevées par la gestion de l'évolution c'est-à-dire que certains triplets dans les annotations deviennent inconsistants (peut causer une perte d'accès aux ressources référencées). Nous verrons dans le prochain chapitre comment analyser les effets d'un changement sur les annotations et les gérer au mieux.

5.8 Conclusion

Dans ce chapitre, nous avons présenté le premier bloc de notre méthodologie d'évolution à savoir le processus d'évolution de la RTO. La tâche la plus importante du processus d'évolution est d'assurer la cohérence de la RTO (assuré si le changement de base et les changements additionnels, qui sont causés par un changement de base, sont appliqués ensemble). Nous avons tout d'abord défini un modèle formel de la RTO. Ensuite, nous avons détaillé le processus d'évolution de la RTO à travers ses quatre phases. En reposant sur ce processus, nous avons établi une typologie de changements qui permet de classer l'ensemble des changements de la RTO en changements élémentaires et complexes. Nous nous sommes

concentrés dans ce chapitre sur le travail de résolution des effets de changements sur la RTO. Nous avons construit une bibliothèque complète de stratégies d'évolutions des changements de la RTO qui permettent de réaliser des changements d'une manière systématique et automatique. Ces stratégies guident l'ontologue dans la modification et le choix de la solution la plus convenable pour chaque type de changement.

Durant le processus de l'évolution, nous avons montré la possibilité de propagation des changements de la RTO vers les annotations sémantiques concernées. La question qui se pose alors, est comment détecter et rectifier les annotations sémantiques qui deviennent incohérentes après la modification de la RTO ? Nous allons examiner ce problème dans le chapitre suivant.

CHAPITRE VI

Processus d'évolution d'annotations sémantiques

Sommaire

6.1 Introduction

Quand la ressource termino-ontologique est utilisée dans un processus d'annotation sémantique, les évolutions doivent se faire de manière cohérente. La tâche de l'évolution de l'annotation sémantique est non seulement de détecter des annotations incohérentes à cause des changements de la RTO mais aussi de les corriger en vue de garantir la cohérence de toute la base d'annotations.

Dans ce chapitre, nous présentons dans une première partie les aspects importants de l'évolution de l'annotation sémantique à travers les trois étapes du processus d'évolution de l'annotation sémantique (deuxième bloc sur la figure IV-9 du chapitre IV). Dans une deuxième partie, nous présentons un module important d'EvOnto qui sert à vérifier la qualité des annotations produites automatiquement par le système.

Le chapitre est organisé comme suit : dans la section 2, nous présentons un exemple illustratif afin de mieux comprendre les différentes étapes du processus d'évolution d'annotation sémantique. Les sections suivantes (sections 6.4.1, 6.4.2 et 6.4.3) présentent ce processus à travers les trois étapes à savoir : (i) détection des annotations incohérentes, (ii) rectification des annotations incohérentes et (iii) mise à jour interactive de la base d'annotation, tout en présentant pour chaque étape les travaux réalisés et leur rôle dans le processus d'évolution. La dernière section du chapitre présente le module de vérification de la qualité des annotations produites par le système.

6.2 Exemple illustratif

Dans la suite, nous nous servons d'un même exemple pour montrer les différentes étapes du processus d'évolution de l'annotation sémantique afin, nous l'espérons, de mieux faire

comprendre ce qui motive notre approche. L'exemple montre un extrait de la RTO évolutive ARTAL (figure VI-1) développée dans le cadre du projet Dynamo. En figure VI-2 et VI-3, nous donnons un exemple de la représentation interne correspondant à un graphe d'annotation d'un document particulier, graphe qui s'appuie sur cette partie de la RTO.

Rappelons que comme nous avons vu dans le chapitre III, les concepts *Trigger_Event*, *Default*, *Function* et *Component* sont les concepts de base (concepts de haut niveau).Sur la figure, nous pouvons voir des relations transverses telles que *Causes*, *Concerns_Function* ou *Affects_Component* et aussi des termes qui dénotent des concepts de la ressource telles que *T_unzoom* et *T_command*. Chaque terme est relié à un concept par la relation dénote.

Supposons que cet extrait de RTO soit modifié en appliquant les deux opérations de changements suivants :

- **Opération 1 :** *SplitConcept (Interface, User_Interface, Screen)* : diviser le concept *Interface* en sous-concepts *User_Interface* et *Screen*.
- **Opération 2 :** *GroupConcept (Unzoom_Function, Zoom_Function, Display_Function)* : créer un super-concept commun *Display_Function* pour les concepts *Unzoom_Function*et *Zoom_Function*.

Après avoir appliqué ces changements, nous obtenons une nouvelle version de la RTO (figure VI-4) dans laquelle certains éléments ont été changés par rapport à l'ancienne version. À cause des changements effectués dans la RTO (division du concept «Interface» et du groupement des concepts «Unzoom_Function» et «Zoom_Function»), certains triplets des annotations sémantiques deviennent maintenant incohérents (lignes en couleur rouge : 2, 3, 4, 6, 10, 11, 12, 14, 18, 19, 28 et 37 concernant l'opération 1, figure VI-5. Lignes 2, 3, 7 et 16 concernant l'opération 2, figure VI-6) du fait de la perte des liens qui font référence à des

concepts correspondants dans la RTO avant sa modification. Nous détaillerons les raisons de cette incohérence et les solutions apportées dans les sections 6.4.1, 6.4.2 et 6.4.3.

Dans la section suivante, nous présentons notre approche d'évolution de l'annotation sémantique et en particulier le deuxième bloc – processus d'évolution de l'annotation sémantique - sur la figure IV-9 du chapitre IV, qui identifie les étapes essentielles à l'évolution et qui décrit le déroulement prévu pour chacune de ces étapes.

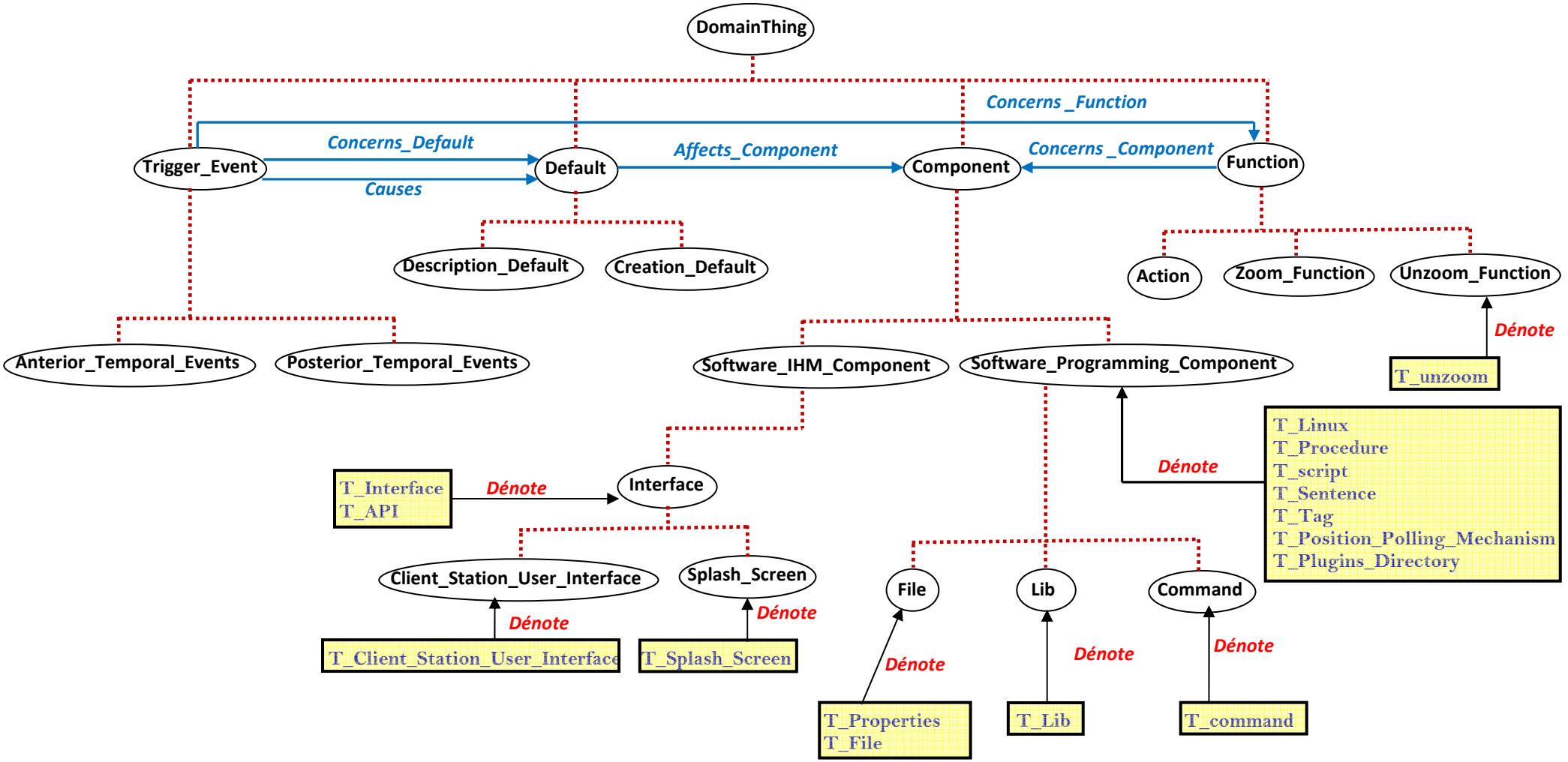


Figure VI - 1 : Extrait de la ressource termino-ontologique pour la maintenance de logiciels avant modification

```
1.  <Document rdf:ID="Onto_Individual_402">
2.  <Concept rdf:ID="Interface">
3.  < Interface  rdf:ID="Onto_Individual_57"/>
4.  </Concept
5.  <Term_en rdf:ID="T_Splash_Screen">
6.  <rdfs:subClassOf  rdf:resource="#TermOccurrence"/>
7.  <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Splash  Screen</texte>
8.  <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">en</language>
9.  <dénote>
10. <Concept rdf:ID=" Splash_Screen">
11. <Splash_Screen rdf:ID="Onto_Individual_98"/>
12. <rdfs:subClassOf rdf:resource="#Interface "/>
13. </Concept> </dénote> </Term_en>
14. <Term_en rdf:ID="T_Client_Station_User_Interface">
15. <rdfs:subClassOf  rdf:resource="#TermOccurrence"/>
16. <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Client Station User Interface </texte>
17. <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">en</language>
18. <dénote>
19. <Concept rdf:ID="Client_Station_User_Interface">
20. <Client_Station_User_Interface rdf:ID="Onto_Individual_99"/>
21. <rdfs:subClassOf rdf:resource="#Interface "/>
22. </Concept>
23. </dénote>
24. </Term_en>
25. …
```

Figure VI - 2 : Extrait d'annotation sémantique basée sur l'extrait de la RTO ARTAL avant d'appliquer l'opération 1

```
1.  <Document rdf:ID="Onto_Individual_402">
2.  <Concept rdf:ID="Zoom_Function">
3.  <Zoom_Function rdf:ID="Onto_Individual_27"/>
4.  <rdfs:subClassOf rdf:resource="#Function"/>
5.  </Concept>
6.  <Term_en rdf:ID="T_unzoom">
7.  <rdfs:subClassOf rdf:resource="#TermOccurrence"/>
8.  <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string">unzoom</texte>
9.  <language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">en</language>
10. <dénote>
11. <Concept rdf:ID="Unzoom_Function">
12. <Unzoom_Function rdf:ID="Onto_Individual_28"/>
13. <rdfs:subClassOf rdf:resource="#Function"/>
14. </Concept>
15. </dénote>
16. </Term_en>
17. …
```

Figure VI - 3 : Extrait d'annotation sémantique basée sur l'extrait de la RTO ARTAL avant d'appliquer l'opération 2

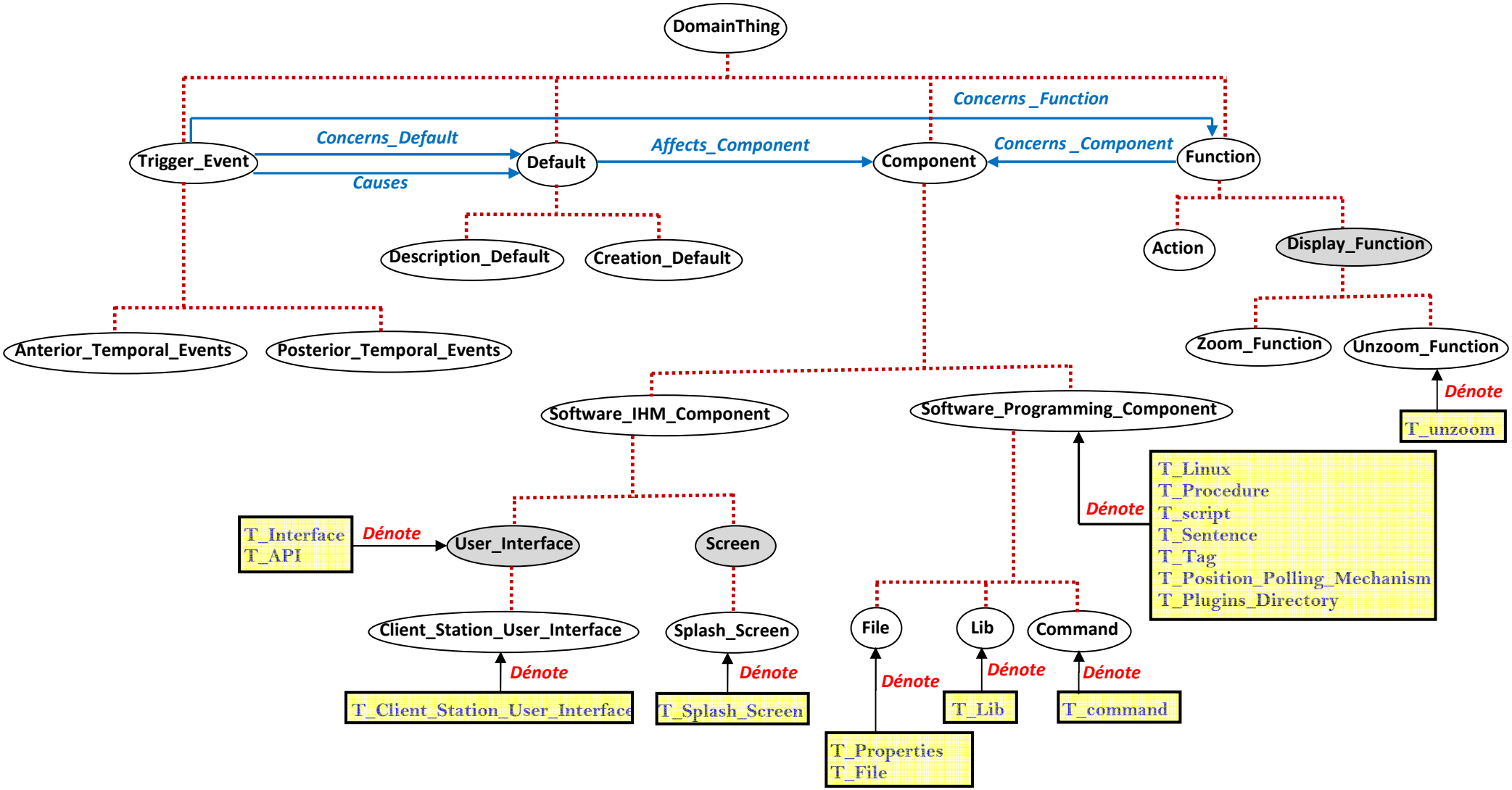


Figure VI - 4 : Extrait de la ressource termino-ontologique pour la maintenance de logiciels après modification

```
1. <Document rdf:ID="Onto_Individual_402"/>
2. <Concept rdf:ID= "Screen">
3. < rdfs:subClassOf rdf:resource="Software_IHM_Component">
4. < Screen rdf:ID= "Onto_Individual_57"/>
5. </Concept>
6. <Term_en rdf:ID="T_Interface">
7. <rdfs:subClassOf rdf:resource="#TermOccurrence">
8. <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Interface</texte>
9. <language rdf:datatype=="http://www.w3.org/2001/XMLSchema#string">en</language>
10. <dénote>
11. <Concept rdf:ID= "User_Interface">
12. < rdfs:subClassOf rdf:resource="Software_IHM_Component">
13. </Concept> </dénote> </Term_en>
14. <Term_en rdf:ID="T_API">
15. <rdfs:subClassOf rdf:resource="#TermOccurrence">
16. <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string">API</texte>
17. <language rdf:datatype=="http://www.w3.org/2001/XMLSchema#string">en</language>
18. <dénote>
19. <Concept rdf:ID= "User_Interface">
20. </Concept> </dénote> </Term_en>
21. <Term_en rdf:ID="T_Splash_Screen">
22. <rdfs:subClassOf rdf:resource="#TermOccurrence">
23. <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Splash Screen</texte>
24. <language rdf:datatype=="http://www.w3.org/2001/XMLSchema#string">en</language>
25. <dénote>
26. <Concept rdf:ID="Splash_Screen">
27. < Splash_Screen rdf:ID= "Onto_Individual_98"/>
28. <rdfs:subClassOf rdf:resource="#Screen">
29. </Concept> </dénote> </Term_en>
30. <Term_en rdf:ID="T_Client_Station_User_Interface">
31. <rdfs:subClassOf rdf:resource="#TermOccurrence">
32. <texte rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Client Station User Interface</texte>
33. <language rdf:datatype=="http://www.w3.org/2001/XMLSchema#string">en</language>
34. <dénote>
35. <Concept rdf:ID="Client_Station_User_Interface">
36. < Client_Station_User_Interface rdf:ID= "Onto_Individual_99"/>
37. <rdfs:subClassOf rdf:resource="#User_Interface">
```

Figure VI - 5 : Extrait d'annotation sémantique concernant l'opération1, basée sur l'extrait de la RTO ARTAL après modification

```
1.  <Document rdf:ID="Onto_Individual_402"/>
2.  <Concept rdf:ID= "Display_Function">
3.  < rdfs:subClassOf  rdf:resource="#Function">
4.  </Concept>
5.  <Concept rdf:ID= "Zoom_Function">
6.  < Zoom_Function rdf:ID= "Onto_Individual_27"/>
7.  < rdfs:subClassOf  rdf:resource="#Display_Function">
8.  </Concept>
9.  <Term_en  rdf:ID="T_unzoom">
10. <rdfs:subClassOf  rdf:resource="#TermOccurrence">
11. <texte  rdf:datatype="http://www.w3.org/2001/XMLSchema#string">unzoom</texte>
12. <language  rdf:datatype="http://www.w3.org/2001/XMLSchema#string">en</language>
13. <dénote>
14. <Concept rdf:ID= "Unzoom_Function">
15. < Unzoom_Function rdf:ID= "Onto_Individual_28"/>
16. < rdfs:subClassOf  rdf:resource="#Display_Function">
17. </Concept>
18. </dénote>
19. </Term_en>
```

Figure VI - 6 : Extrait d'annotation sémantique concernant l'opération 2, basée sur l'extrait de la RTO ARTAL après modification

Quelles que soient les raisons de l'évolution de la ressource termino-ontologique, l'exemple ci-dessus, nous confirme que les annotations sémantiques de documents basés sur la RTO peuvent être affectées par les changements de cette ressource. Nous avons donc besoin, pour exprimer l'évolution, de recenser l'ensemble des modifications que l'on peut apporter aux ressources termino-ontologiques ainsi que les annotations sémantiques qui sont concernées par ces changements et qui doivent être déterminées, analysées et gérées au cours de l'évolution afin d'assurer leur cohérence par rapport à la RTO de base. Après avoir présenté le contexte d'évolution de l'annotation sémantique, nous détaillons dans la section suivante les étapes nécessaires du processus d'évolution de l'annotation sémantique.

6.3 Contexte d'évolution

Nous considérons deux scénarios ci-dessous qui peuvent affecter l'état cohérent des annotations sémantiques.

Scénario 1: A chaque fois que, le système produit une annotation automatique avec l'outil TextViz, une vérification de la qualité des annotations, selon des critères définis par l'utilisateur, est lancée. Si l'annotation ne vérifie pas les critères, l'ontologue doit modifier la RTO, jusqu'à ce que le processus d'annotation automatique produise une annotation correcte. L'ontologue fait des modifications sur la RTO de base. A cause de ces changements, les annotations peuvent être affectées entraînant un état incohérent.

Scénario 2: Suite à l'ajout d'un nouveau document dans le corpus, l'utilisateur décide d'apporter des modifications sur la RTO de base. A cause de ces changements, les annotations peuvent ne plus être en adéquation avec la RTO.

Dans le cadre de Dynamo, si des changements sont effectués sur la RTO, les annotations devront être adaptées à la nouvelle version de la RTO et vice versa (si l'ontologue n'est pas satisfait des annotations produites par le système et décide de modifier la RTO jusqu'à ce que l'annotation soit correcte).

6.4 Propagation des changements aux annotations

Après avoir appliqué les changements sur la RTO, le problème est d'adapter les annotations à la nouvelle version de la RTO.

Pour gérer cette évolution, nous proposons un processus en trois temps (figure VI-7) : Tout d'abord, le processus commence par la phase - *détection des annotations incohérentes* – où le système recherche dans la base d'annotations, celles concernées par les modifications de la RTO ensuite la phase - *rectification des annotations incohérentes* – qui permet de modifier

les annotations devenues incohérentes à cause de ces changements. Enfin la phase - *mise à jour de la base d'annotations*– où l'ontologue valide les fiches concernées par les modifications.

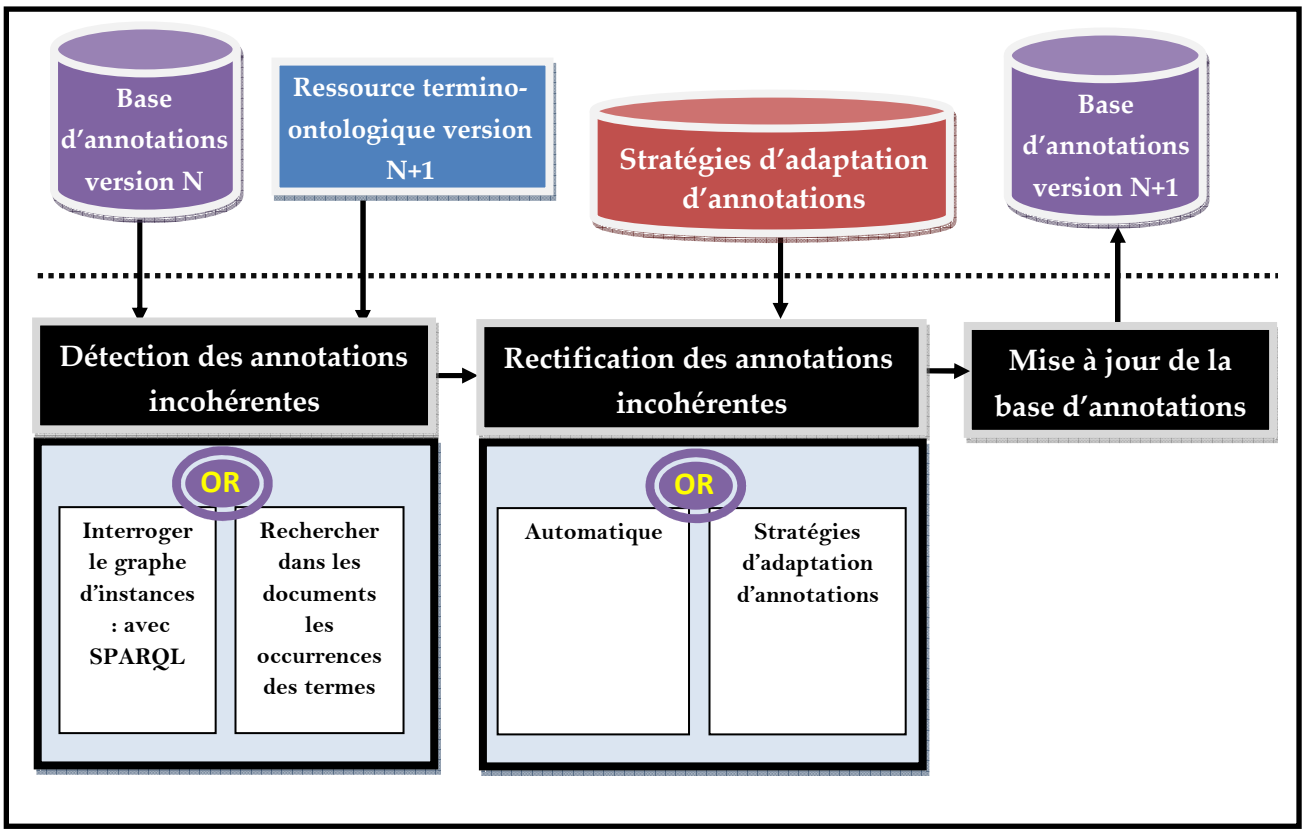


Figure VI - 7 : Processus d'évolution des annotations sémantiques

6.4.1 Détection des annotations incohérentes

La phase de détection des annotations incohérentes permet de capturer une liste d'annotations sémantiques (graphes d'annotations) qui sont incohérentes par rapport à la RTO de base. Pour cette phase, deux méthodes sont utilisables :

6.4.1.1 Interrogation des graphes d'annotations

Le but de cette méthode est de rechercher dans les graphes d'annotation les parties qui subissent un impact des modifications de la RTO. Par exemple, un triplet RDF où apparaît un concept qui a été supprimé doit clairement être remis en cause.

Reprenons notre exemple ci-dessus (la section 6.2) où l'ontologue fait des modifications sur la RTO. Pour garder la RTO dans un état cohérent, un changement peut en nécessiter d'autres : nous appelons ces *changements des changements additionnels.* Dans notre exemple, la division du concept «Interface» impliquent d'autres changements tels que la distribution des concepts fils «Splash_Screen » et «Client_Station_User_Interface» respectivement entre les concepts «Screen » et «User_Interface». Nous présentons dans la suite les changements additionnels à effectuer pour achever l'opération de division du concept «Interface» et le regroupement des concepts «Zoom_Function» et «Unzoom_Function».

Opération 1 : SplitConcept (Interface, User_Interface, Screen)

- Le concept «Interface» ne sera plus le super concept des concepts «Client_Station_User_Interface » et «Splash_Screen».
- Les concepts «User_Interface» et «Screen» seront crées et rattachés au concept «Software_IHM_Component».
- Les concepts «Client_Station_User_Interface» et «Splash_Screen» sont rattachés respectivement aux concepts «User_Interface» et «Screen».
- Les termes «T_Interface » et « T_API » ne seront plus associés au concept «Interface».
- Les termes «T_Interface » et « T_API » seront associés au concept «User_Interface».
- Le concept «Interface» sera supprimé.

Opération 2: GroupConcept (Unzoom_Function, Zoom_Function, Display_Function)

- Le concept «Display_Function» sera crée et rattaché au concept «Function» ;
- Le concept «Function» ne sera plus le super concept des concepts «Zoom_Function» et «Unzoom_Function» ;
- Les concepts «Zoom_Function» et «Unzoom_Function» seront rattachés au nouveau concept «Display_Function» ;

Une fois les changements appliqués sur la RTO, une étape intermédiaire - *récupération des entités modifiées de la ressource termino-ontologique* -, nous permet d'obtenir une liste des entités modifiées de la RTO, à savoir les concepts, les termes et les propriétés.

Pour chercher les graphes d'annotations concernés qui deviennent incohérents par rapport à la RTO à cause de la perte des liens de références, nous utilisons la liste des entités modifiées de la RTO et une requête SPARQL[23] [Prud et Seaborne, 2007] sur l'ensemble de ces graphes correspondant à tous les documents déjà annotés. Dans le cas de l'exemple ci-dessus de la division du concept «Interface», la requête créée automatiquement est présentée dans la figure VI-7.

```
PREFIX table : <http://www.owlontologies.com/Ontology1221232201.owl>
SELECT * where {
GRAPH ?s {
?suj ?prop ?obj
?suj rdf:type ?type
FILTER (?type ~ " Interface")}
```

Figure VI - 8 : Exemple d'une requête en langage SPARQL

Après avoir lancé la requête SPARQL, le système nous renvoie les triplets dont la ressource est de type «Interface», à savoir le triplet contenant *?suj*, *?prop* et *?obj*.

Les triplets des annotations sémantiques incohérents détectés concernant la division du concept « Interface » et le regroupement des concepts «Zoom_Function» et «Unzoom_Function» sont présentés avec des lignes en couleur rouge dans la figure VI-5 et figure VI-5. Les lignes 12 et 21 pour l'opération 1 et les lignes 4 et 13 pour l'opération 2.

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

6.4.1.2 Rechercher les occurrences des termes

Le but de cette méthode est de rechercher directement dans les documents textuels (simple recherche textuelle) les occurrences des termes liés aux éléments de la RTO modifiés (modification d'un terme, d'un concept, d'une propriété).

La différence entre la première méthode – *interrogation de graphes d'annotations*– et la deuxième méthode – *rechercher les occurrences des termes* - réside dans le type de document dans lequel la recherche sera faite. Pour la première méthode, le système utilise le fichier OWL (contient les triplets incohérents par rapport à la RTO et qui constituent par la suite un graphe d'annotation) pour faire la recherche, alors que dans la deuxième méthode, le système utilise la collection des documents annotés pour faire une simple recherche textuelle des occurrences des termes liés aux éléments modifiés (les concepts) de la RTO.

Dans le cas de l'exemple ci-dessus de la division du concept «Interface», le système utilise les termes « T_Interface » et « T_API » qui dénotent le concept « Interface » pour rechercher les occurrences des termes liés aux éléments de la RTO modifiés et par la suite les documents impactés.

Dans cette thèse, nous nous sommes contentés de développer la première méthode.

6.4.2 Rectification des annotations incohérentes

La phase de rectification des annotations incohérentes permet de corriger les incohérences des triplets détectées.

Si la deuxième méthode – *rechercher les occurrences des termes* - a été utilisée, simple recherche des documents impactés, le seul choix possible est la ré-annotation automatique de ces documents. Le temps de calcul reste raisonnable si la base de documents ne comprend que quelques centaines de documents courts. Mais, même dans ce cas, un inconvénient majeur est souligné par nos partenaires dans le contexte de leurs applications, particulièrement pour le diagnostic automobile (société ACTIA). En effet, dans leur cas, les annotations doivent être

vérifiées par l'ontologue, fiche par fiche et le coût en temps devient important. Un de nos objectifs est de réduire ce coût.

Si la première méthode, *interrogation des graphes d'annotations*, a été utilisée, nous appliquons des stratégies d'adaptation d'annotations (SAA) présenté dans la figure VI-9. Chacune d'entre elles est associée à une des stratégies d'évolution de la RTO (SE_RTO) présentées en section 5.5.1 du chapitre V.

Comme nous avons expliquée dans le chapitre IV, la nécessité d'avoir plusieurs stratégies, qui plus est adaptables, correspond à la diversité des raisons pour lesquelles l'utilisateur peut souhaiter effectuer une évolution. Différents exemples nous ont montré que pour une stratégie donnée l'adoption ou non par l'utilisateur de telle ou telle conséquence peut différer d'une situation à une autre.

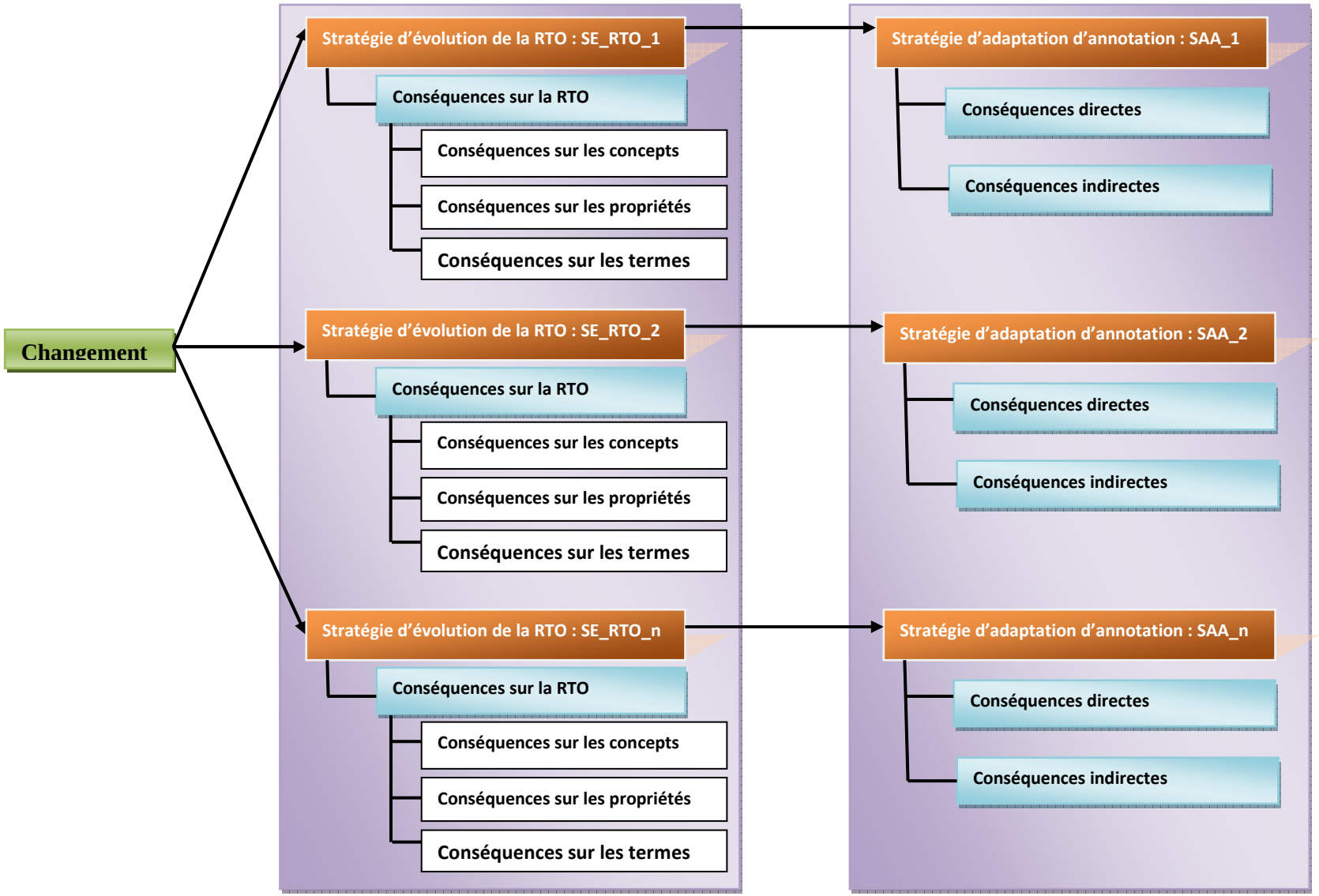


Figure VI - 9 : Stratégies d'adaptation d'annotations sémantiques

Une stratégie d'adaptation d'annotation a pour but de corriger les incohérences que peut produire le changement dans les annotations sémantiques. Selon la figure VI-8, chaque stratégie d'adaptation (SAA_1, SAA_2, SAA_n) correspond à un ensemble de changements directs et indirects (s'il en existe) résolvant l'incohérence des annotations par rapport à la RTO.

Un changement direct est un changement qui affecte directement l'entité modifiée de la RTO utilisée dans l'annotation des documents. Le changement indirect prend en compte les changements additionnels qui peuvent affecter d'autres entités modifiées de la RTO utilisés dans l'annotation des documents. Prenons l'exemple d'un concept C et son concept fils C1 qui sont utilisés pour annoter respectivement les fiches F1 et F2. Dans le cas de suppression du concept C, l'ontologue décide d'appliquer un autre changement additionnel à savoir la suppression du concept C1. Le changement direct sur l'annotation est celui qui affecte l'entité modifiée de la RTO à savoir C qui annote la fiche F1, alors que, le changement indirect sur l'annotation est le changement additionnel à savoir la suppression du concept C1 qui annote la fiche F2.

Pour chaque stratégie d'adaptation sélectionnée, le système avertit l'utilisateur et présente les conséquences directes et indirectes de ce changement sur l'annotation des documents. Ces informations permettent que celui-ci choisisse la stratégie qui lui semble la mieux adaptée avec la RTO et avec l'annotation sémantique des documents. Afin d'illustrer nos stratégies d'évolution, supposons qu'un extrait de RTO ARTAL (figure VI-10) soit modifié en appliquant l'opération de changement qui permet de fusionner les concepts «State_Modification» et «Renaming», et les remplaçant par le nouveau concept «Modification».

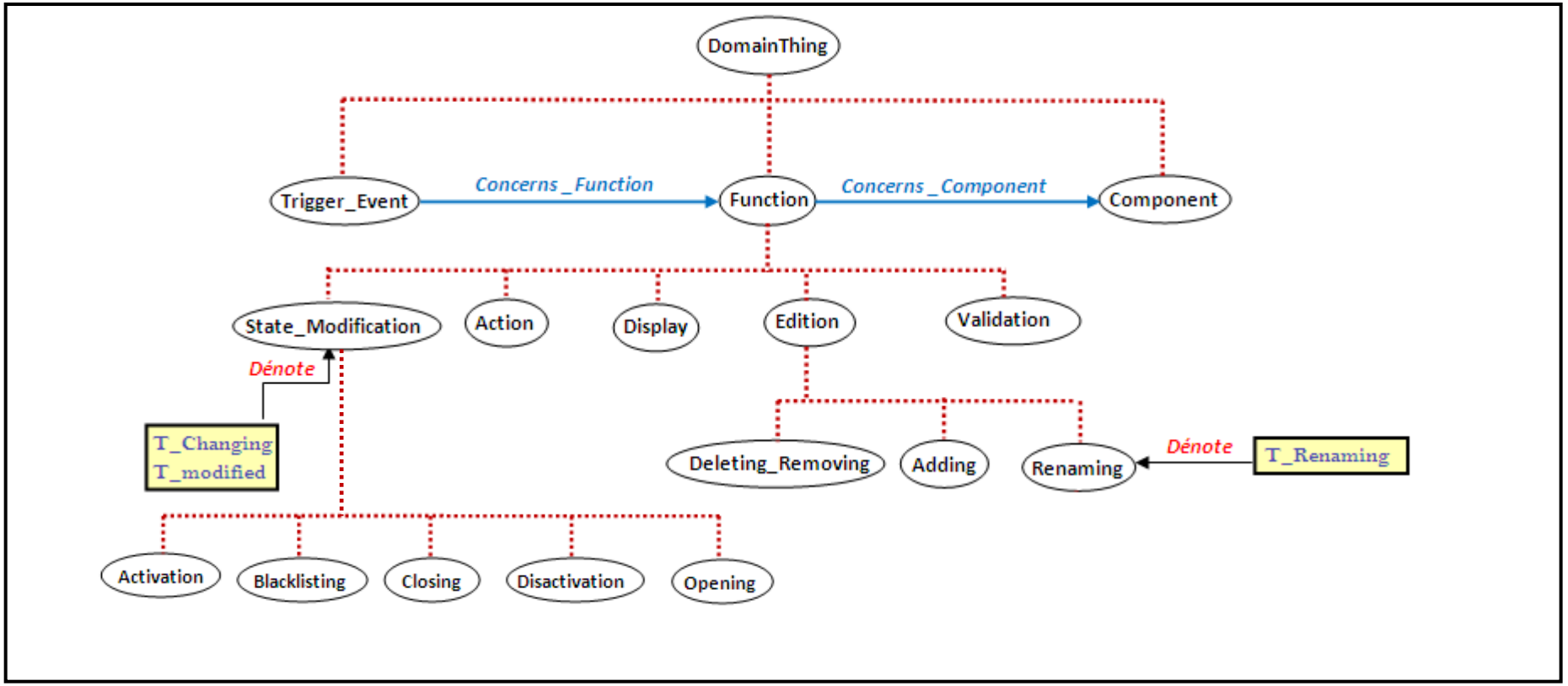


Figure VI - 10 : Extrait de la ressource termino-ontologique pour la maintenance de logiciels

Dans l'exemple suivant, on a vu que deux stratégies sont possibles pour gérer les conséquences sur l'ensemble des éléments de la RTO :

- ***SE_RTO_1*** : Attacher le nouveau concept au super-concept du premier concept sélectionné
- ***SE_RTO_2*** : Attacher le nouveau concept au super-concept du deuxième concept sélectionné

Les conséquences sur l'ensemble des éléments de la RTO de la stratégie SE_RTO_1 sont les suivantes :

- [Modification] sera créé ;
- [Function] sera le super-concept de [Modification];
- [Edition] ne sera plus le super-concept de [Renaming];
- [State_Modification] ne sera plus le super-concept de [Activation, BlackLinsting, Closing, Disactivation, Opening];
- [Modification] sera le super-concept de [Activation, BlackLinsting, Closing, Disactivation, Opening];
- [T_ Renaming] ne sera plus associé à [Renaming] ;
- [T_Changing, T_modified] ne seront plus associés à [State_Modification] ;
- [T_ Renaming, T_Changing, T_modified] seront associés à [Modification] ;
- [Renaming] et [State_Modification] seront supprimés ;

Les conséquences directes sur l'annotation de la stratégie *SE_RTO_1* pour MergeConcept sont les suivantes (figure VI-10, rectangle pointillé en couleur rouge) :

- La fiche [Fiche-565] sera invalidée ;
- Ce fiche ne sera plus annotée par le concept [Renaming] ;
- Ce fiche sera annotée par le concept [Modification] ;

Note : avec ce type de stratégie (SE_RTO_1) il n'y a pas des conséquences indirectes sur les annotations.

Si l'ontologue choisit la deuxième stratégie *SE_RTO_2* – Attacher le nouveau concept au super-concept du deuxième concept sélectionné –, on a vu que les conséquences sur l'ensemble des éléments de la RTO de la stratégie sont les suivantes :

- [Modification] sera créé ;
- [Edition] sera le super-concept de [Modification];
- [Function] ne sera plus le super-concept de [State_Modification];
- [Edition] ne sera plus le super-concept de [Renaming];
- [State_Modification] ne sera plus le super-concept de [Activation, BlackLinsting, Closing, Disactivation, Opening];
- [Modification] sera le super-concept de [Activation, BlackLinsting, Closing, Disactivation, Opening];
- [T_ Renaming] ne sera plus associé à [Renaming] ;
- [T_Changing, T_modified] ne seront plus associés à [State_Modification] ;
- [T_ Renaming, T_Changing, T_modified] seront associés à [Modification] ;
- [Renaming] et [State_Modification] seront supprimés ;

Les conséquences directes sur l'annotation de la stratégie *SE_RTO_2* sont pareils que la Stratégie *SE_RTO_1*.

Rappelons que ces stratégies seront appliquées telles quelles si l'utilisateur les a acceptées sans en modifier les conséquences (sans ajuster les conséquences).

Pour mieux guider l'ontologue dans les évolutions de l'annotation sémantique, nous avons construit toutes les stratégies d'adaptation correspondant aux changements élémentaires et complexes utilisées dans notre travail et qui sont associées aux stratégies d'évolution de la RTO. La description complète des stratégies d'adaptation d'annotations est présentée dans la partie annexe A.

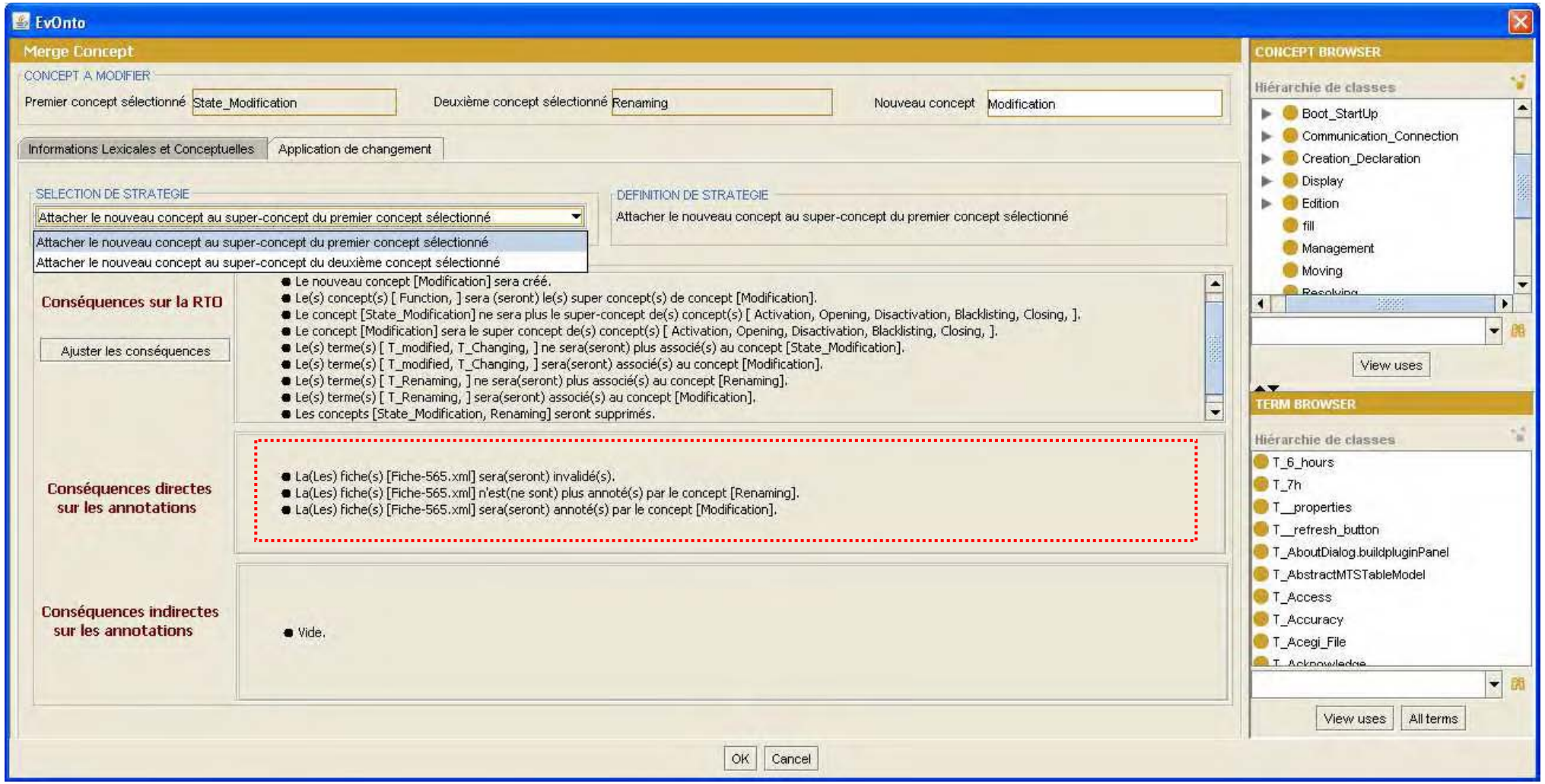


Figure VI - 11 : Capture d'écran d'EvOnto : présentation de conséquences directes sur les annotations

6.4.3 Mise à jour interactive de la base d'annotations

Une fois les annotations de toutes les fiches concernées par la modification rectifiées, la phase de mise à jour de la base d'annotations permet à l'ontologue de valider l'annotation fiche par fiche. En fait, après la rectification des annotations incohérentes et à l'aide de l'outil TextViz, on garde les fiches dans un état invalide, c'est-à-dire, on donne le droit à l'ontologue de vérifier manuellement si les annotations produites après rectifications sont correctes ou non (fiches bien ré-annotés ou non) par rapport à la RTO.

Si l'ontologue n'est pas d'accord avec les solutions proposées, il peut laisser la fiche dans un état invalide et 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 la RTO 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.

Si l'ontologue est d'accord avec des solutions proposées. Il valide l'annotation fiche par fiche.

6.5 Evaluation de la qualité des annotations

6.5.1 Contexte

Un module important d'EvOnto sert à vérifier les annotations produites automatiquement, ce qui est une originalité de notre approche[24]. Une vérification manuelle, basée sur la lecture des annotations par l'ontologue, peut conduire à des oublis et de plus est coûteuse en temps. Pour repérer rapidement et systématiquement des documents mal annotés, EvOnto propose un module d'évaluation de la qualité des annotations.

[24] L'idée a été aussi proposée et mise en œuvre dans la thèse de Reymonnet [Reymonnet, 2008], mais la réalisation est différente par rapport à notre travail.

Ce module demande à l'utilisateur de définir des critères de qualité des annotations propres au corpus et au domaine d'application. Comme pour [Reymonnet, 2008], 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 (indirecte en fait) de chacun de ces concepts, instances éventuellement en relation. En général, ces concepts sont des classes de haut niveau dans la RTO. 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 ACTIA, etc.). Pour chaque document, un score évalue le pourcentage d'éléments reconnus, indiquant dans quelle mesure les critères sont vérifiés et pour chaque document on peut avoir besoin de critères différents pour vérifier leur qualité comme les cas d'ARTAL où plusieurs couvertures sont possibles (voir section 6.5.2).

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 de corriger les annotations et surtout de faire évoluer la RTO.

6.5.2 Aide à l'évaluation d'annotations

Comme nous l'avons expliqué, 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 figure VI_1, les annotations doivent vérifier certains critères comme une couverture minimale du noyau de la RTO. Par exemple, ARTAL a choisi comme critères que l'annotation d'une fiche doit contenir au moins :

Une première couverture minimale :

- Une instance indirecte de *Default* ;
- Une instance indirecte de *Component* ;
- Une instance de la relation *Affects* entre les deux instances indirectes de *Default* et *Component*.
- Une instance de la relation *Located_in* entre les deux instances indirectes de *Default* et *Component*.

Ou bien

Deuxième couverture minimale :

- Une instance indirecte de *Default* ;
- Une instance indirecte de *Component* ;
- Une instance indirecte de *Trigger_Event* ;
- Une instance de la relation *Causes* entre les deux instances indirectes de *Trigger_Event* et *Default* ;
- Une instance de la relation *Relates_to* entre les deux instances indirectes de *Trigger_Event* et *Component* ;

La figure VI-12 présente l'interface qui permet de définir ces critères. Dans le rectangle marqué A, on voit la fenêtre qui permet de choisir les concepts dont des instances directes et/ou indirectes doivent être identifiées dans chaque document. Dans le rectangle marqué B, on a la fenêtre qui permet de donner au système les relations qui doivent être identifiées dans chaque document. Ces relations dans le cas de TextViz sont déterminées par des propriétés simples, essentiellement de proximité des termes associés aux concepts en relation, mais elles pourraient, de manière plus générale, être déterminées à partir du texte, mais non sans difficultés.

Dans la version initiale de TextViz, ces critères étaient fixes pour une application donnée et insérés dans le code. Par contre, dans notre travail, 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, il se peut que le système dispose d'information pour modifier automatiquement les annotations sémantiques. 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 peut créer le lien 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. 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, afin que le document puisse être ré-annoté de façon satisfaisante d'une manière automatique ou en modifiant les annotations sémantiques comme décrit dans la section 6.4.1. Ensuite, les documents à l'origine de cette évolution doivent être ré-annotés et les annotations à nouveau évaluées par rapport aux critères.

Les résultats fournis par le système 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.

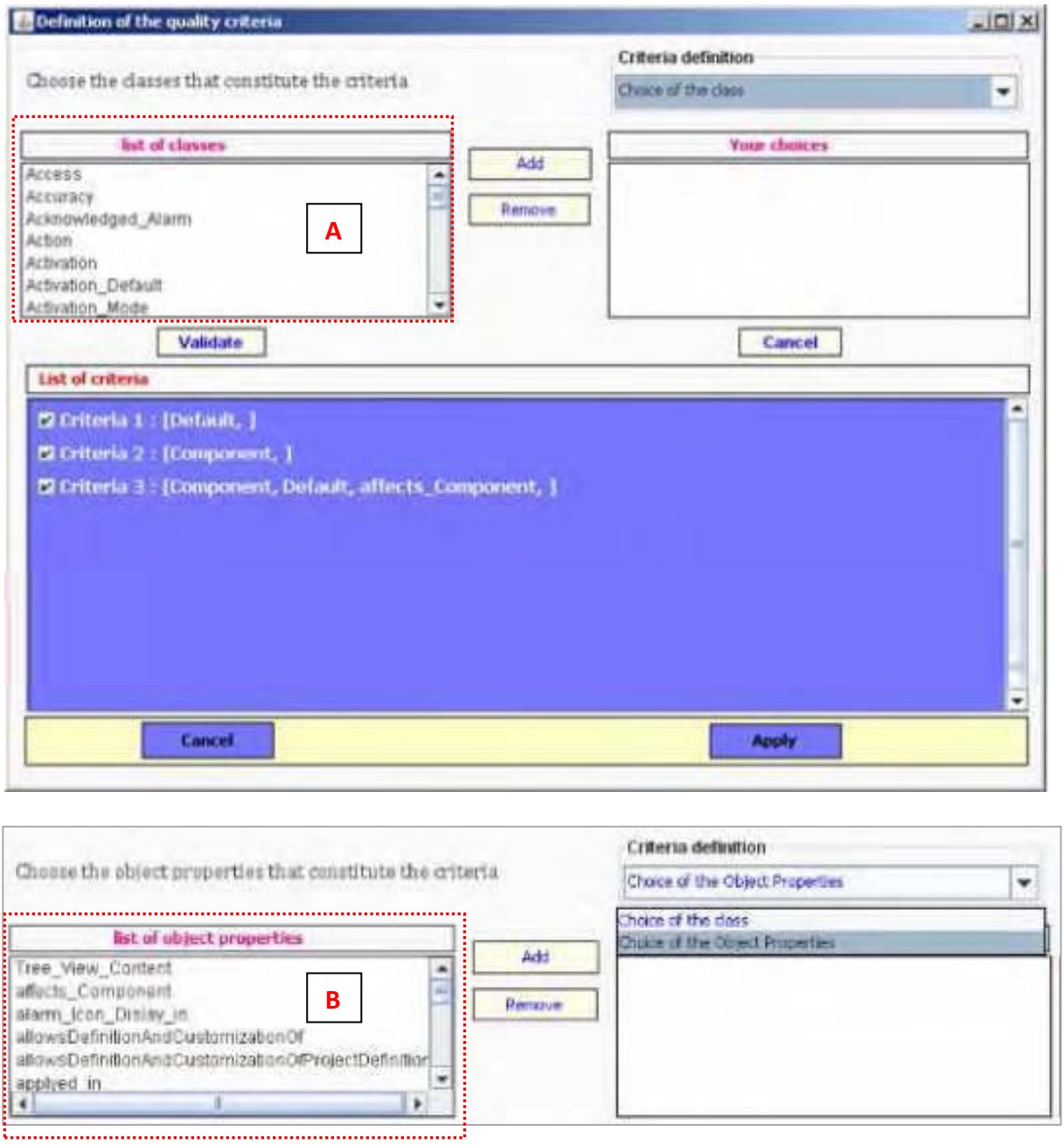


Figure VI - 12 : Définition des critères de qualité des annotations

6.6 Conclusion

Dans ce chapitre, nous avons présenté le deuxième bloc de notre méthodologie d'évolution à savoir le processus d'évolution de l'annotation sémantique. La tâche la plus importante du processus d'évolution est d'assurer la conformité entre la RTO et l'annotation sémantique. Nous avons tout d'abord détaillé le processus d'évolution de l'annotation sémantique à travers ses trois phases. Nous nous sommes concentrés dans ce chapitre sur le travail de résolution

des effets de changements de la RTO sur l'annotation sémantique. Pour cela, nous avons construit une bibliothèque complète de stratégies d'adaptation d'annotations. Chacune d'entre elles est associée à une des stratégies d'évolution de la RTO présentées en section 5.5.1 du chapitre V. Ces stratégies ont pour but de corriger les incohérences que peut produire le changement dans les annotations sémantiques. Durant ce chapitre, nous avons présenté aussi un module qui représente une originalité de notre approche : le module de vérification de la qualité des annotations produites automatiquement.

Dans le chapitre suivant, nous allons évaluer, auprès d'utilisateurs humains les performances de notre système ainsi que son utilisation sur des corpus et des RTOs issus de différents domaines, corpus en anglais ou en français.

.

Troisième partie

UTILISATION ET ÉVALUATION D'EVONTO

Cette partie va nous permettre d'illustrer la validation de notre modèle à travers le prototype EvOnto. Nous montrons comment EvOnto peut aider l'ontologue à adapter la ressource termino-ontologique au corpus pour produire les annotations les plus précises possible, et à adapter les annotations aux évolutions de la ressource.

CHAPITRE VII

Évaluation du système EvOnto

Sommaire

7.1 Introduction

A partir des idées développées dans les chapitres précédents, nous présentons le système EvOnto ainsi qu'un scénario d'utilisation de cet outil. Ce système a pour objectif de guider interactivement l'ontologue pour formuler une demande de changement, évaluer son impact (effets supplémentaires) sur la qualité de la RTO et aussi sur les annotations sémantiques, et décider ensuite de leur mise en œuvre.

Afin de tester la fiabilité et de montrer la faisabilité de notre modèle, nous allons présenter dans ce chapitre, un ensemble d'expérimentations menées sur notre système d'évolution EvOnto. En effet, il est extrêmement compliqué d'évaluer correctement un outil d'aide à l'utilisateur. Nous avons fait des expérimentations sur deux domaines d'application du projet Dynamo. Nous avons pour cela considéré la RTO du domaine des incidents logiciel «ARTAL» puis la RTO du domaine du diagnostic automobile «ACTIA». Ces deux domaines sont complètement différents. D'une part, la RTO ARTAL est une ressource de la langue anglaise qui est en cours de construction et n'est pas encore achevée. D'autre part, la RTO ACTIA qui est en langue française est mieux adaptée au corpus considéré. Les résultats acquis sont analysés et discutés au fur et à mesure de leur présentation.

7.2 Scénario d'utilisation de l'outil EvOnto

Afin de faciliter l'utilisation du système par l'ontologue, nous avons mis en place dans EvOnto un certain nombre d'interfaces et de fonctionnalités (figure VII-1, figure VII-2) s'appuyant sur TextViz.

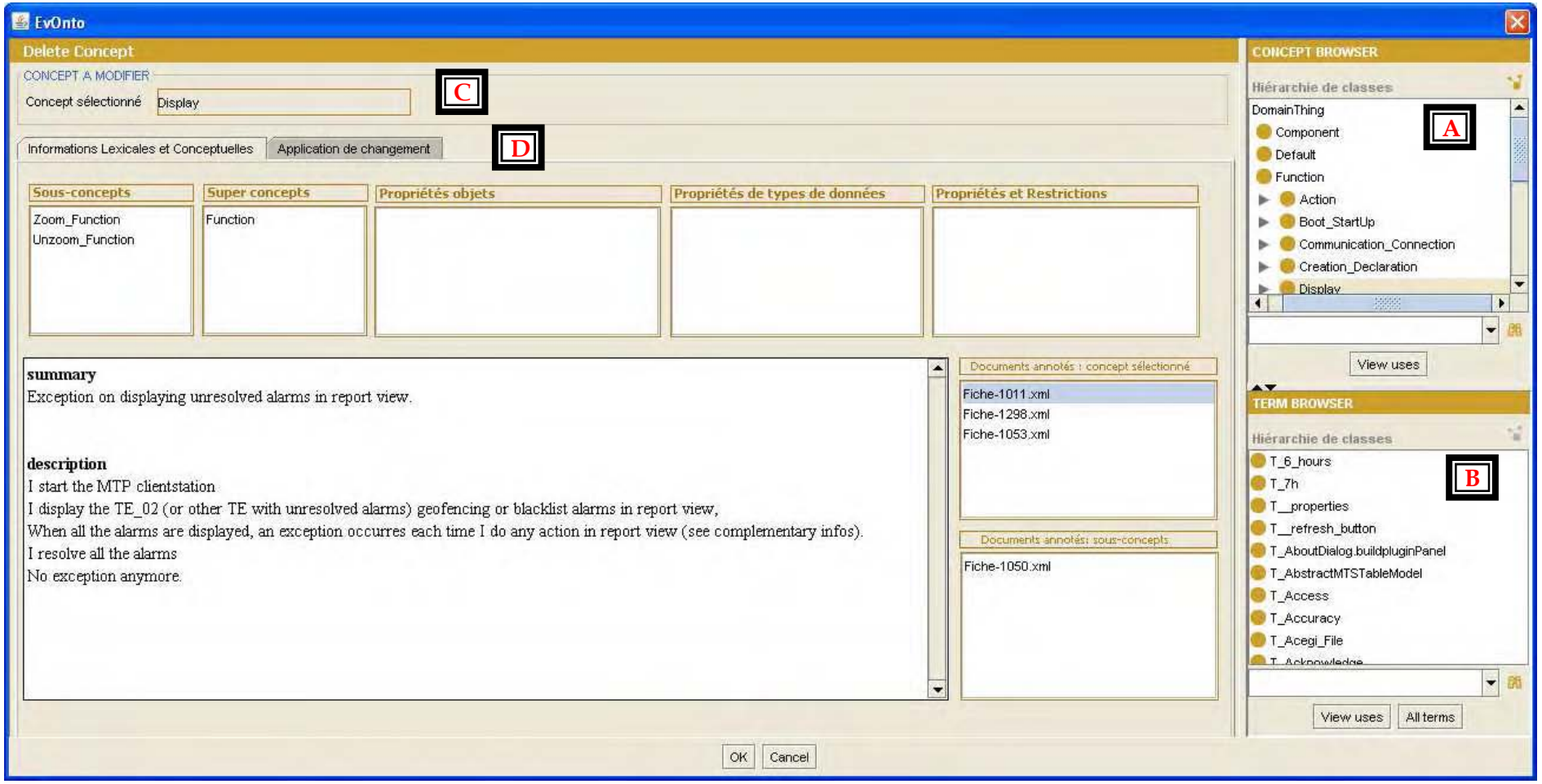


Figure VII - 1 : Copie de l'écran principal d'EvOnto : Onglet informations lexicales et conceptuelles

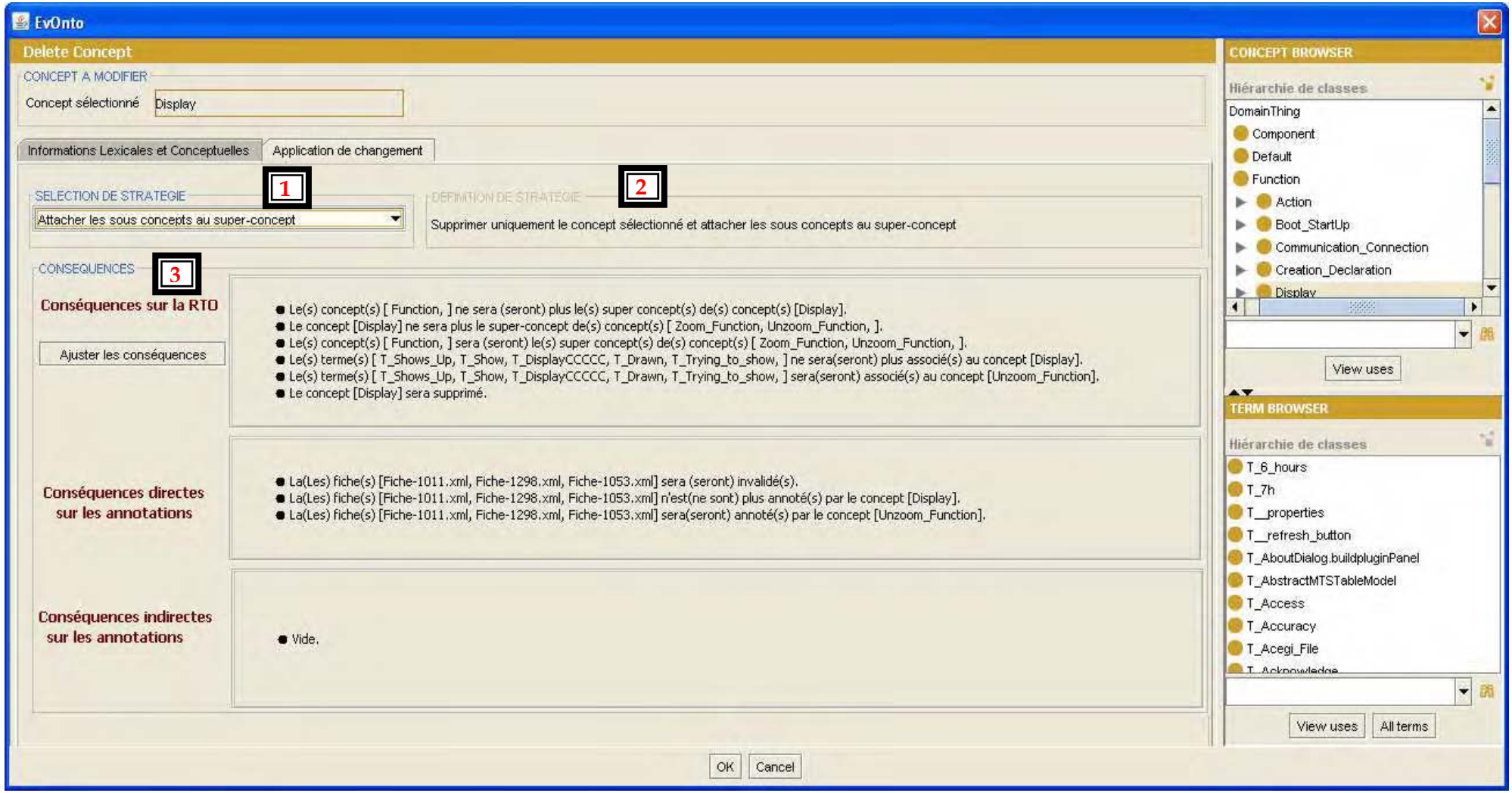


Figure VII - 2 : Copie de l'écran principal d'EvOnto : Onglet application de changement

Dans EvOnto, nous avons implémenté deux parties qui sont dépendantes. La première partie correspond à l'interface d'évolution de la RTO et des annotations sémantiques où l'ontologue peut adapter la RTO au corpus pour produire les annotations les plus précises possibles. Il peut également, adapter les annotations aux évolutions de la ressource. Dans la deuxième partie, l'ontologue peut utiliser d'autres interfaces pour évaluer la qualité des annotations produites par le système TextViz. Dans les deux sections suivantes, nous détaillons ces deux parties.

7.2.1 Dans le contexte d'évolution de la RTO et d'annotations sémantiques

Le processus commence par la sélection de l'entité à modifier, qui se fait comme dans Protégé. Le 'popup_menu' (figure VII-3, figure VII-4) que nous avons ajouté au panel de la hiérarchie des concepts présente séparément les changements élémentaires (*CreateConcept, DeleteAssociatedTerm, DeleteConcept,…*) et les changements composés (*MoveConcept, MergeConcept,…*).

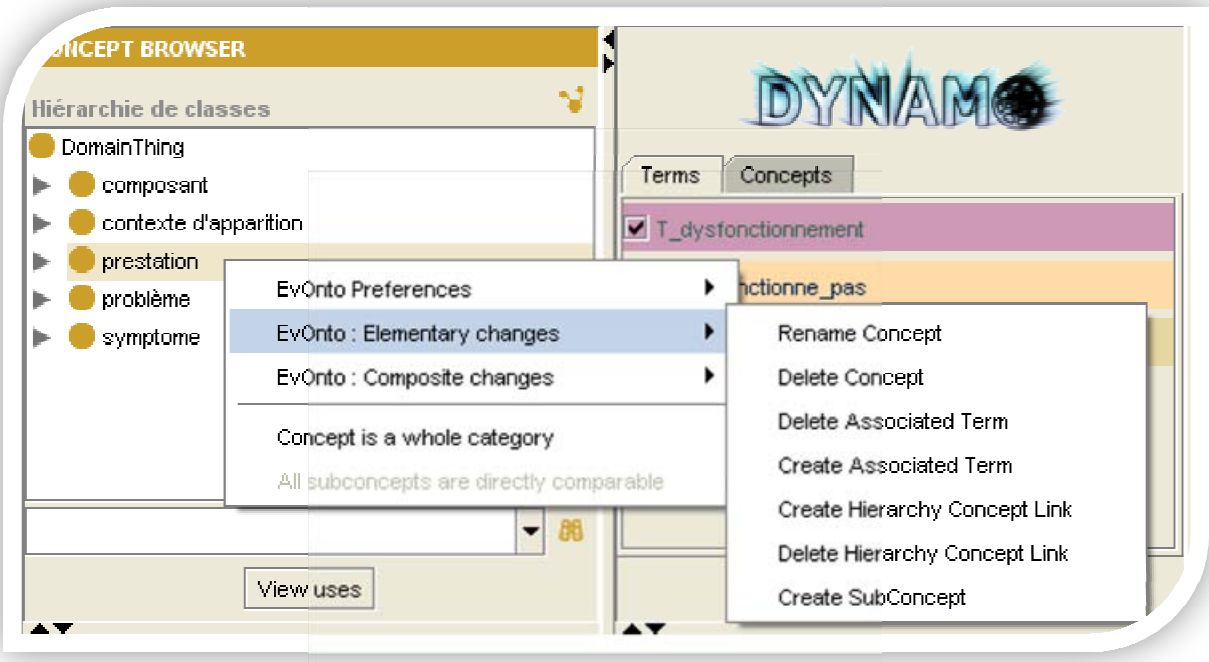


Figure VII - 3 : Popup_menu d'EvOnto : changements élémentaires

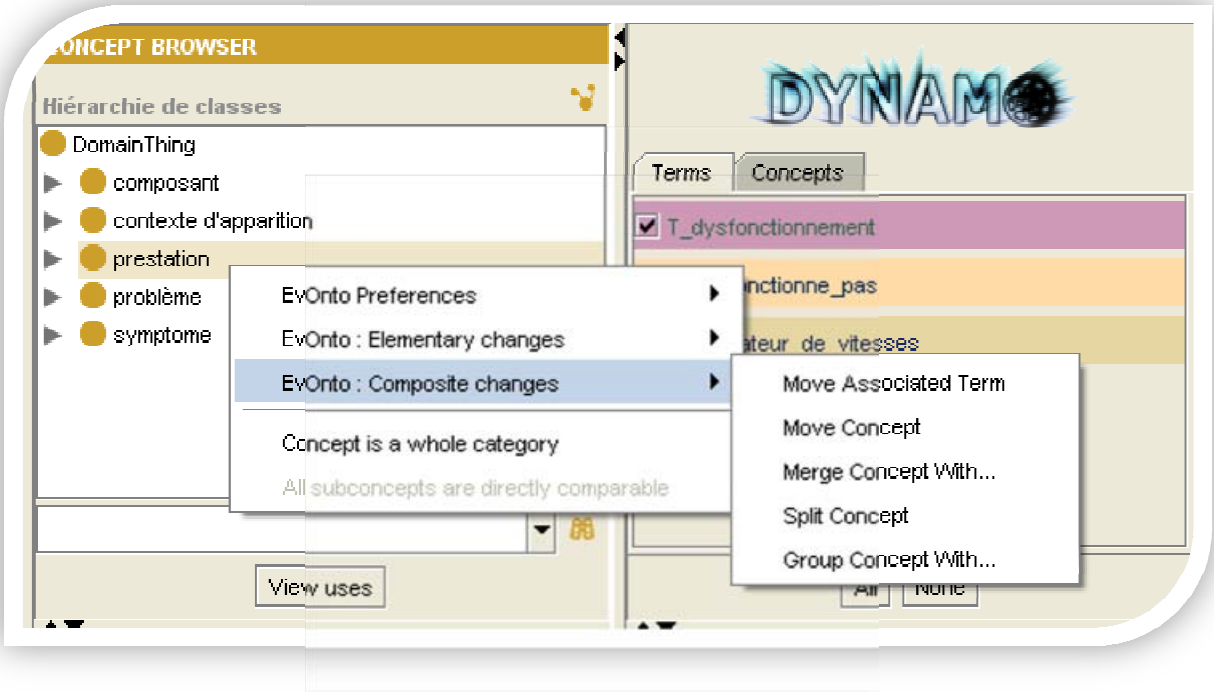


Figure VII - 4 : Popup_menu d'EvOnto : changements complexes

La figure VII-1 montre la fenêtre affichée par EvOnto après la sélection de l'opération du changement (dans la copie d'écran de la figure VII-1, l'opération est «DeleteConcept»). Cette interface contient quatre panels.

A **CONCEPT BROWSER** : affiche la hiérarchie des concepts d'une RTO. Si l'ontologue sélectionne un concept dans le panneau CONCEPT BROWSER, ce concept sera affiché dans le panel C et les termes associés à ce concept sont affichés dans le panneau TERM BROWSER.

B **TERM BROWSER** : affiche la liste des termes d'une RTO. Si l'ontologue sélectionne un terme dans le panneau TERM BROWSER, le concept dénoté est alors présenté dans le panneau CONCEPT BROWSER et aussi dans le panel C.

C **CONCEPT A MODIFIER** : Le panel nommé « CONCEPT A MODIFIER » affiche le nom de l'opération en cours, le concept sélectionné («Display» dans notre exemple).

D **Panel d'onglets** : Le panel d'onglets contient deux onglets dépendants. Le premier onglet nommé «*informations lexicales et conceptuelles*» est celui qui définit et s'affiche par défaut après la sélection de l'opération du changement.

L'onglet «*informations lexicales et conceptuelles*» affiche toutes les entités liées au concept sélectionné : ses concepts fils, son concept père, les propriétés ou restrictions de propriétés dont il est domaine ou co-domaine, les propriétés objets, les propriétés de types de données, les documents annotés et le contenu textuel de chaque document dans la liste des documents.

L'onglet «*application du changement*» présente les conséquences de cette modification selon la stratégie prévue par défaut. Ces conséquences visent à maintenir la cohérence de la RTO et des annotations de documents. L'ontologue peut décider de changer de stratégie, d'ajuster plus finement chacune des conséquences, ou de renoncer à cette modification. L'onglet «*application du changement*» est composé de trois panels (figure VII-2) :

1 **SELECTION DE STRATEGIE** : ce panel propose la stratégie définie par défaut pour ce changement et permet de la modifier en sélectionnant une stratégie alternative.

2 **DEFINITION DE STRATEGIE** : pour mieux comprendre l'objectif de la stratégie, ce panel présente à l'ontologue une définition plus détaillée de chaque stratégie.

3 **CONSEQUENCES** : pour chaque stratégie sélectionnée, le système avertit l'ontologue et présente les conséquences engendrées par ce changement sur la RTO et les effets du changement sur les annotations sémantiques, directes et indirectes. Ces informations lui permettent de choisir la stratégie qui lui semble la mieux adaptée.

Avant de valider le changement et ses conséquences, l'ontologue peut ajuster les conséquences en sélectionnant le bouton «*Ajuster les conséquences*» qui ouvre un nouveau panel nommé « **AJUSTEMENT DES CONSEQUENCES DE LA RTO**».

La figure VII-5 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.

Une interface analogue (pas encore implémentée) doit permettre d'ajuster les conséquences directes et indirectes sur les annotations. Une fois ces différentes conséquences validées (bouton Ok), les changements sont effectués sur la ressource termino-ontologique ainsi que sur les annotations sémantiques. Ce deuxième processus consiste à modifier le moins possible les annotations en remplaçant par de nouveaux triplets ceux touchés par le changement.

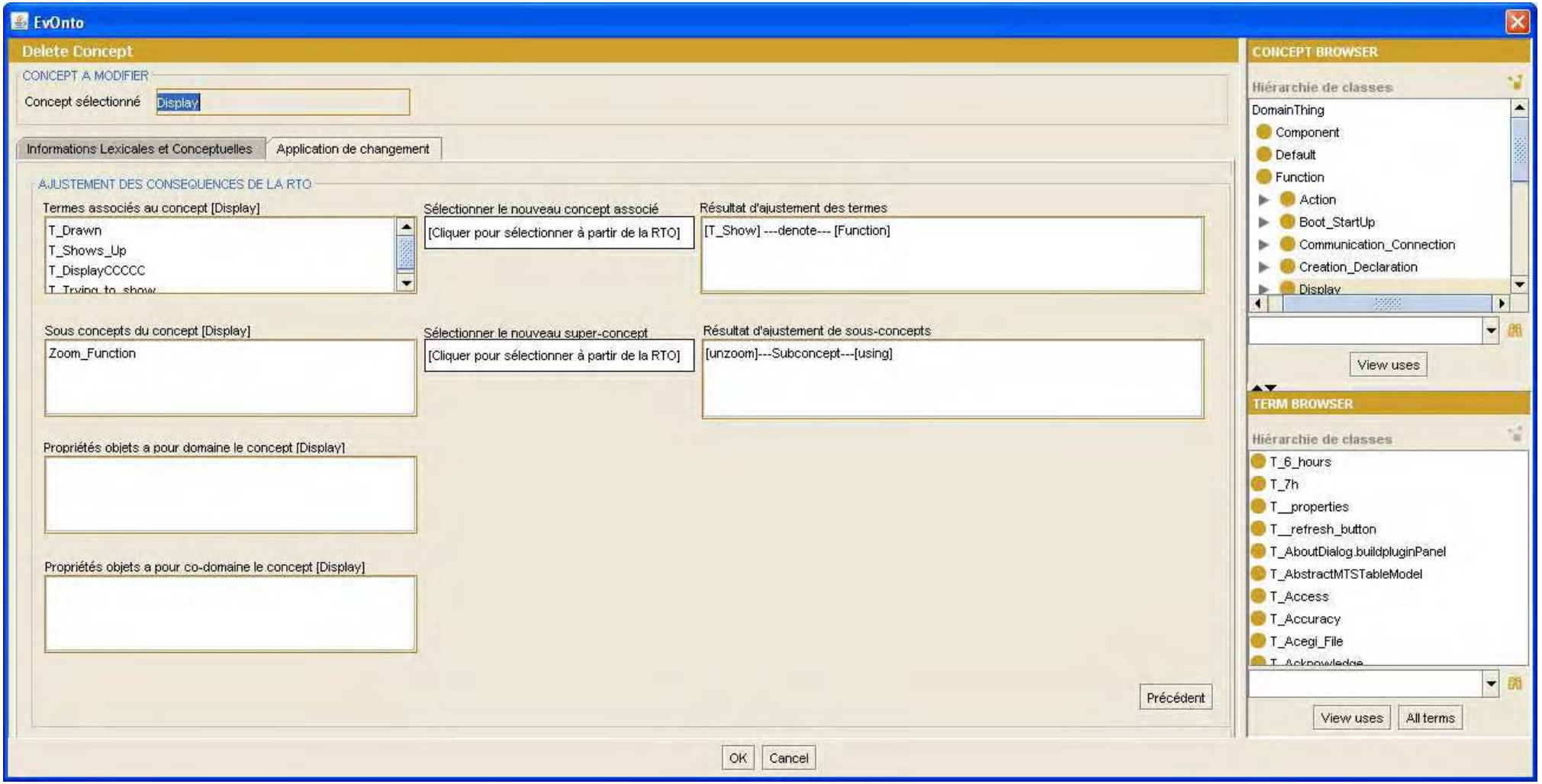


Figure VII - 5 : Interface pour adapter les conséquences sur d'autres composants de la ressource termino-ontologique

7.2.2 Dans le contexte d'évaluation de la qualité d'annotations

Chaque fois que TextViz produit une nouvelle annotation, EvOnto vérifie la qualité de ces annotations, selon des critères définis par l'utilisateur. Nous avons expliqué ces mécanismes et présenté les interfaces correspondant dans la section 6.5.2.

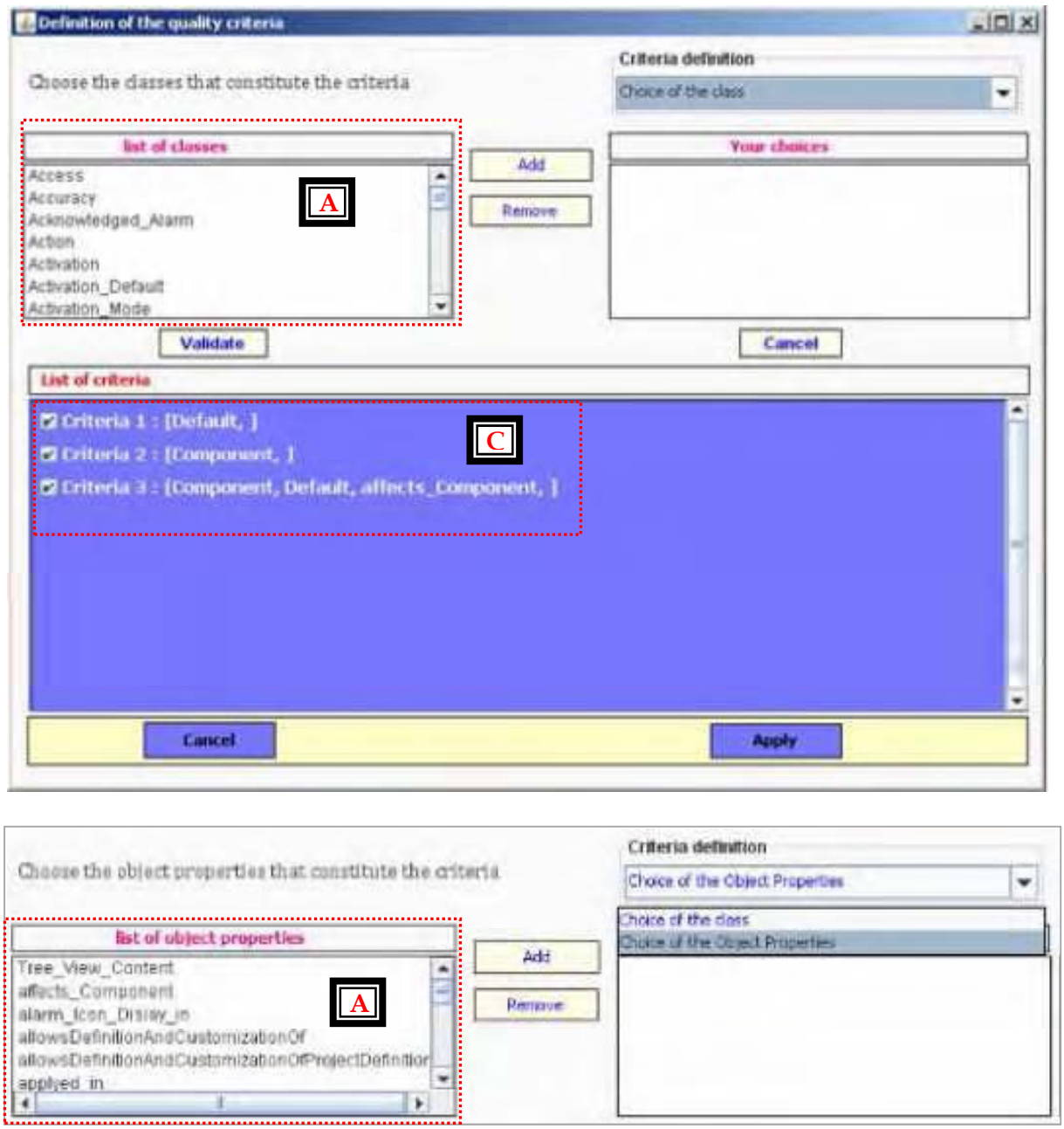


Figure VII - 6 : Définition des critères de qualité des annotations

7.3 Techniques d'évaluation des systèmes

La phase d'évaluation consiste à mesurer l'utilisabilité du système par rapport aux exigences définies lors de la phase d'analyse. Si le système n'atteint pas le niveau d'exigence défini, alors il sera corrigé et à nouveau testé. Il y a plusieurs méthodes pour concevoir et construire l'évaluation d'un prototype existant. Plusieurs questions se posent pour choisir la méthode la plus efficace à savoir, (a) quels sont les critères de cette évaluation ? (b) en fonction de ces critères et de l'état actuel du prototype, quelles méthodes pouvons nous considérer ? et (c) quelle(s) méthode(s) choisir ?

7.3.1 Critères d'évaluation

Pour améliorer l'Interaction Homme Machine (IHM), la norme (ISO 9241-11 1998) définit deux dimensions clés :

- **L'utilité :** qui se définit par l'adéquation de l'objet aux besoins et à la tâche de l'utilisateur.

- **L'utilisabilité :** qui se définit comme le « *degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié* ».

    - **L'efficacité** désigne les capacités de l'objet à permettre d'atteindre des buts, des résultats prévus.
    - **L'efficience** désigne les capacités de l'objet à permettre d'atteindre des buts avec un effort moindre, et en un minimum de temps.
    - **La satisfaction** désigne le ressenti subjectif des utilisateurs sur l'objet. Cette notion renvoie au plaisir d'usage, à la convivialité de l'interaction (agréabilité, attractivité) et à l'acceptabilité de l'objet dans son activité.

7.3.2 Approche qualitative pour l'évaluation du prototype

L'une des principales méthodes utilisées pour l'évaluation du système est le test utilisateur, qui permet de placer un utilisateur final en situation d'usage réel et d'observer les difficultés rencontrées. Dans [Paillé et Mucchielli, 2003], les auteurs ont défini l'analyse qualitative comme une démarche de recherche du sens dans laquelle le chercheur observe des comportements ou interprète des paroles ou des textes, afin de caractériser un objet ou un phénomène et d'en trouver ainsi son sens et sa pertinence.

Plusieurs techniques peuvent servir l'évaluation qualitative d'un système [Mucchielli, 1996] telles que, *le focus groupe*, *les questionnaires de satisfaction*, *l'évaluation par paire* et *la verbalisation*.

- ***Les focus group*** ou groupe de discussion sont un moyen de recueillir les points de vue d'un groupe de personnes sur un sujet bien précis [Slocum, 2006]. Cette technique favorise l'émergence de toutes les opinions des participants et répond aux « pourquoi ? » et aux « comment ? ».
- ***Des questionnaires de satisfaction*** peuvent être proposés aux participants sous forme des listes de questions écrites [Quivy et vanCampenhoud, 1995]. On peut aussi procéder à des entretiens, observations, avec des verbalisations concomitantes (en même temps que se déroule l'observation) et / ou consécutives (suite à l'observation). Il existe trois principaux types de questions: (a) les questions fermées avec une série de réponses à choisir; (b) les questions ouvertes qui ne prévoient aucune réponse, mais demandent aux participants de s'exprimer librement; (c) les questions semi-ouvertes qui sont une combinaison des deux autres.

- ***L'évaluation par paire*** est une technique où les participants sont regroupés en groupe de deux pour réaliser les activités d'évaluation [Gediga *et al.*, 2002] [O'Malley *et al.*, 1984].
- ***La technique de verbalisation*** demande à chaque participant de réfléchir à haute voix (d'exprimer verbalement ce qu'il pense), à exprimer ses raisonnements lors de l'interaction avec le système à évaluer [Jorgensen, 1989].

7.4 Démarche d'évaluation du système EvOnto

Dans cette section, nous présentons les participants et la procédure que nous avons mise en œuvre pour l'évaluation du système EvOnto.

7.4.1 Participants à l'évaluation

Nous avons choisi des participants ayant une expérience dans le domaine suivant :

- La représentation des ontologies ;
- L'utilisation des éditeurs d'ontologies : en particulier Protégé ;
- Le langage de représentation des ontologies : OWL ;
- Les annotations sémantiques : des annotations associées à des documents en utilisant l'ontologie.

Cinq personnes ont été choisies pour évaluer notre prototype (Tableau VII-1) dont trois participants sont des enseignants chercheurs touchant aux spécialités suivantes : représentation et gestion des connaissances, méthodes et outils de construction des ontologies, web sémantique et ontologies. Les deux autres participants sont des doctorants travaillant sur la recherche d'informations textuelles et structurées.

Tableau VII - 1 : Participants choisis pour l'évaluation d'EvOnto

	Niveau d'expérience

Participants	*Représentation des ontologies*	*Éditeurs d'ontologies*	*Langage OWL*	*Annotations sémantiques*
Participant P1	Très bon	Très bon	Très bon	Très bon
Participant P2	Très bon	Bon	Très bon	Bon
Participant P3	Bon	Bon	Moyen	Très bon
Participant P4	Bon	Faible	Moyen	Faible
Participant P5	Moyen	Moyen	Moyen	Moyen

7.4.2 Procédure d'évaluation

Les étapes du processus d'expérimentation du système EvOnto sont au nombre de trois :

7.4.2.1 Présentation d'EvOnto

C'est une étape préliminaire à l'expérimentation qui permet de présenter aux participants notre système EvOnto. Nous avons utilisé le dispositif de projection sur grand écran avec une présentation PowerPoint. L'objectif de cette présentation est de fournir à tous les participants les connaissances nécessaires pour porter des jugements de l'utilité des services proposés par le système EvOnto, des objectifs et de la nature concrète de notre recherche. Cette étape permet aussi de familiariser les participants avec notre système ainsi qu'avec les notions utilisées lors de l'expérimentation, afin de leur permettre de se concentrer davantage sur l'évaluation de l'utilité d'EvOnto.

Dans cette présentation, nous avons évoqué les points suivants :

- Les objectifs, le contexte et la démarche de notre recherche ;
- Les ontologies :
  - Qu'est ce qu'une ontologie ;
  - Représentation des ontologies : le language OWL ;
  - Ontologies et l'éditeur Protégé ;
- La ressource termino-ontologique : représentation des ontologies à composante terminologique ;
- La collection de documents ;

- Les annotations associées à ces documents en utilisant l'ontologie ;
- Le Projet DYNAMO ;
  - Le méta-modèle pour les RTOs ;
  - La représentation du terme ;
  - La modélisation des liens terme - concept ;
  - TextViz ;
- Le Système EvOnto ;
  - Problématique : évolution cohérente de RTO et des annotations sémantiques ;
  - Modèle ;
  - Prototype ;
  - Exemples de modifications : formulation d'une demande de changement et sa mise en œuvre pour deux cas de changements.

7.4.2.2 Phase « d'entrainement » avec EvOnto

Suite à cette présentation et afin de maîtriser l'utilisation d'EvOnto, nous avons prévu une phase « d'entrainement » avec le logiciel. La durée de cette phase de préparation a varié d'une heure à deux heures selon le niveau d'expérience des participants.

Pour guider les participants dans la formulation d'une demande de changement et dans l'évaluation de son impact sur la qualité de la RTO et sur les annotations sémantiques pour décider de leur mise en œuvre, nous avons préparé « un scénario d'entrainement ». Ce dernier contient un ensemble de modifications à effectuer pour montrer aux participants toutes les possibilités du logiciel.

Nous avons proposé aux participants une liste de changements à effectuer. Cette liste contient un ensemble de changements élémentaires et complexes qui affectent les entités de la RTO, à savoir, les concepts et les termes. Dans cette phase d'entrainement, nous avons utilisé

la RTO ARTAL. Pendant le déroulement de cette étape, tous les participants peuvent discuter et réfléchir à voix haute.

- Les changements élémentaires présentés sont:
  - *DeleteConcept* : supprimer un concept de la RTO ;
  - C*reateAssociatedTerm* : créer un nouveau terme et l'associer au concept sélectionné.
- Les changements complexes présentés sont:
  - *MoveConcept* : déplacer le concept OldC qui est le fils du concept OldC1 à la nouvelle position comme fils du concept OldC2 ;
  - *MoveAssociatedTerm* : déplacer le lien de dénotation qui associe le terme OldT au concept OldC à la nouvelle position pour associer le terme OldT au nouveau concept NewC.

Pour mieux guider les participants dans la formulation d'une demande de changement, nous avons défini une succession de quatre étapes qui était formulée et présentée sous format textuel aux participants. Ces étapes sont :

Etape 1 : Expression du changement

Premièrement, le participant doit sélectionner l'entité à modifier. Ensuite, il choisit le type de changement qu'il désire effectuer (voir section 5.4).

Etape 2 : Traitement des effets du changement

Après la sélection de l'opération du changement, EvOnto présente aux participants un panel qui contient deux onglets. Le premier onglet nommé «*informations lexicales et conceptuelles*» affiche toutes les entités liées au concept sélectionné (voir section 5.5). A cette étape, les participants peuvent:

- Donner un avis sur l'utilité des informations présentées ;
- Décider de continuer ou pas l'application du type de changement choisi

Le deuxième onglet nommé «*application du changement*» propose à l'utilisateur une stratégie d'évolution par défaut en présentant ses conséquences. Le participant peut donc décider de changer la stratégie proposée, d'ajuster plus finement chacune des conséquences, ou de renoncer à cette modification (voir section 5.6).

Etape 3 : Propagation des changements dans la RTO

La stratégie choisie par le participant correspond à un ensemble additionnel de changements résolvant l'incohérence des entités de la RTO liées à l'entité modifiée. A ce stade, des changements additionnels (conséquences sur les concepts, sur les propriétés et sur les termes) sont générés automatiquement. Le participant est amené à examiner ces changements puis à valider ou pas cette stratégie afin d'effectuer le changement sur la RTO. Si ces changements ne lui conviennent pas, il a la possibilité de les ajuster manuellement.

Etape 4 : Propagation sur les annotations des fiches

Après avoir appliqué les changements sur la RTO, le problème est d'adapter les annotations à la nouvelle version de la RTO. Selon la stratégie validée par le participant (étape 3), une stratégie d'adaptation d'annotation (qui correspond à un ensemble de changements directs et indirects s'il en existe résolvant l'incohérence des annotations par rapport à la RTO) sera appliquée de manière automatique pour ré-annoter les fiches concernées par ce type de changement.

Le participant doit vérifier la ré-annotation des fiches, s'il n'est pas d'accord avec les solutions proposées, il peut laisser la fiche dans un état invalide et choisir alors d'enrichir ou de modifier la RTO ou bien de modifier manuellement l'annotation.

7.4.2.3 Évaluation du système EvOnto

A) Objectifs visés par l'étape

Dans la section 1.1, nous avons défini les critères d'évaluation d'un système selon la norme « ISO 9241-11 1998 » à savoir, l'utilité et l'utilisabilité. Le but principal est de valider les fonctionnalités d'EvOnto d'abord en termes *d'utilité* mais aussi en termes *d'utilisabilité*. Donc l'objectif du cette étape d'évaluation vise à déterminer l'utilité des fonctionnalités d'EvOnto, qui se traduit par les questions suivantes :

1) Le système possède-t-il les fonctionnalités qui répondent aux besoins des utilisateurs et qui nous permettent d'atteindre les objectifs fixés ?
2) Le système offre-t-il de nouvelles fonctionnalités qu'on trouve appropriées dans le contexte d'évolution des RTOs et des annotations sémantiques ?

Les objectifs fixés au préalable permettent de déterminer les points forts d'EvOnto ainsi que les améliorations indispensables pour son efficacité. Ils sont au nombre de trois :

1) Le premier objectif est de déterminer l'intérêt de l'outil en fonction de ses fonctionnalités globales *pour tous les changements demandées*. Les fonctionnalités globales sont en termes de la présentation des effets possibles (conséquences) des changements de la RTO avant leur application, des fiches impactées, des stratégies d'évolution proposées et de l'assistance fournie par cet outil.

2) Le deuxième objectif est de déterminer *pour chaque type de changement*, l'utilité des stratégies d'évolution proposées et de leur pertinence, l'utilité de présenter les conséquences sur la RTO, les conséquences directes et indirectes sur les annotations, et le besoin d'ajustements de ces conséquences.

3) Le troisième objectif est d'avoir un système qui permet l'atteinte des buts avec un effort moindre et en un minimum de temps.

B) Déroulement de l'étape

Techniques d'évaluation

Nous avons défini un processus d'évaluation qui se base sur la technique du questionnaire permettant aux participants de donner leur avis et de les guider dans les activités d'évaluation.

Données d'expérimentation

Dans l'évaluation du système, nous avons proposé aux participants deux RTOs avec les corpus correspondant fournis par deux partenaires de projet Dynamo à savoir, la RTO *ACTIA* qui comporte 330 concepts et 579 termes et la RTO *ARTAL* qui comporte 582 concepts et 887 termes.

Exigences d'expérimentation

Pour éviter des manipulations n'ayant aucun lien avec nos objectifs d'évaluation, nous présentons certaines exigences générales d'utilisation du système qui doivent être satisfaites pour bien évaluer le fonctionnement de ce dernier.

(1) Pour savoir comment le participant répond aux questions et comment vont être traités les différents types de changements, nous proposons aux participants de répondre aux questions après l'application de chaque type de changement.

(2) L'utilisation d'EvOnto lors de l'évaluation est contrôlée. Nous imposons donc aux participants une RTO, un corpus et une liste de modification à effectuer.

(3) A la fin de l'évaluation, chaque participant décrit son impression par type de changement puis donne un avis général sur l'utilité des fonctionnalités d'EvOnto.

(4) Il faut chronométrer le temps nécessaire pour effectuer chaque type de changement avec EvOnto.

(5) Comparaison aux systèmes existants à savoir, Protégé 4 et TextViz.

Procédure d'évaluation

Comme le montre le tableau VII-2, nous avons défini deux types d'évaluation à savoir :

- Une évaluation par questionnaire en utilisant les données d'ARTAL (évaluation n° 1). Dans ce type d'évaluation, la liste de changements suggérés est différente de la liste de changements suggérés au préalable dans la phase d'entraînement.
- Une évaluation par questionnaire en utilisant les données d'ACTIA (évaluation n° 2);

Pour chaque type d'évaluation, chaque participant est amené à répondre aux questions une fois un type de changement est appliqué parmi la liste des modifications suggérées.

Tableau VII - 2 : Différents types d'évaluation du système EvOnto

Type d'évaluation	Techniques d'évaluations	Données d'expérimentation	Liste des modifications
Evaluation N° 1	Questionnaire	ARTAL	DeleteConcept MergeConcept DeleteAssociatedTerm
Evaluation N° 2	Questionnaire	ACTIA	DeleteConcept MergeConcept DeleteAssociatedTerm

Organisation du questionnaire d'évaluation

Le questionnaire (voir Annexe B) comporte six parties qui sont complémentaires et dépendantes l'une de l'autre, à savoir :

- La première partie nommée « intérêt de l'outil » est remplie par le participant à la fin du questionnaire. Cette partie se compose de cinq questions demandant aux participants s'ils ont l'habitude de travailler avec ce type d'outil, s'ils jugent nécessaire de présenter les effets possibles des changements de la RTO avant leur application, s'il est utile de présenter les différentes stratégies d'évolutions proposées et s'il est nécessaire de connaître les fiches impactées.

- La deuxième partie nommée « utilisation de l'outil » comporte quatre questions demandant aux participants leurs avis sur l'utilisation générale de l'outil, ses fonctionnalités ainsi que les informations présentées dans les interfaces manipulées.

- La troisième partie nommée «Evaluation des changements proposés» comporte neuf questions demandant aux participants, leurs avis sur l'utilité :
  - des informations affichées avant de faire un changement à savoir les informations concernant les éléments de la RTO et les fiches impactées ;
  - des différentes stratégies d'évolutions proposées et choisies ;
  - de la présentation des conséquences directes et indirectes de l'opération en cours ;
  - de l'ajustement manuel.
- La quatrième partie nommée «Comparaison d'EvOnto avec d'autres systèmes» comporte deux questions demandant aux participants leurs avis sur l'utilité de l'assistance permettant de guider l'utilisateur à appliquer un changement et sur le traitement des conséquences de certaines opérations en comparaison avec les autres outils à savoir Protégé 4 et TextViz.

- La cinquième partie nommée « avis général » où le participant est amenée à rédiger un avis général sur les fonctionnalités et la simplicité de l'outil.

- La sixième partie nommée «suggestions» où le participant peut, s'il le souhaite nous proposer des recommandations et des suggestions à propos de l'outil et ses fonctionnalités.

7.4.3 Résultats d'évaluation

7.4.3.1 Expérimentation avec les données d'ARTAL

Dans cette section, nous présentons les différents résultats d'expérimentation en utilisant les données ARTAL ainsi que la technique d'évaluation par questionnaire. Les résultats sont organisés sous forme de tableaux récapitulatifs que nous avons remplis à partir des différentes

réponses au questionnaire de tous les participants. Ces tableaux résument les différents avis des participants suite à l'application de tous les types des changements suggérés. Pour chaque évaluation, quatre tableaux sont proposés et sont organisés en respectant les différentes parties du questionnaire à savoir, (a) *Intérêt de l'outil*, (b) *Utilisation de l'outil*, (c) *Évaluation des changements proposés* et (d) *Comparaison d'EvOnto avec d'autres systèmes*. L'analyse des résultats d'évaluations sera décrite dans la section 7.5.

Dans cette partie, nous présentons les différents résultats d'expérimentation en utilisant les données ARTAL et la technique d'évaluation par questionnaire.

Tableau VII - 3 : Résultat d'évaluation selon le champ «intérêt de l'outil»

Intérêt de l'outil	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
L'habitude de travailler avec ce type d'outil.	Oui	Oui	Oui	Oui	Oui un peu
L'assistance fournie par ce type d'outil.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
La présentation des effets possibles des changements de la RTO avant leur application.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
L'utilité des différentes stratégies d'évolutions.	Oui	Oui	Oui un peu	Oui	Oui
Nécessité de connaître les fiches impactées.	Oui	Oui	Oui	Oui	Oui

Tableau VII - 4 : Résultat d'évaluation selon le champ «utilisation de l'outil»

Utilisation de l'outil	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
L'utilisation générale de cet outil.	Simple	Simple	Simple	Simple	Simple
Nécessité de chercher la place des informations.	Non	Non	Non	Non	Oui un peu
Nécessité de chercher la	Non	Oui un peu	Non	Non	Oui un peu

place des fonctionnalités.					
Manque des fonctionnalités.	Oui un peu	Non	Non	Non	Non

Tableau VII - 5 : Résultat d'évaluation selon le champ «évaluations des changements proposés»

Evaluation des changements proposés	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
Les informations affichées concernant les éléments de la RTO avant de faire le changement.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
Les informations concernant les fiches impactées avant de faire le changement.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
Les stratégies d'évolution pour ce type de changement.	Nécessaire	Nécessaire	Intéressante, mais pas nécessaire	Nécessaire	Nécessaire
La stratégie d'évolution choisie pour un type de changement.	Pertinente	Pertinente	Pertinente	Pertinente	Pertinente
La présentation des conséquences.	Inutile	Utile	Utile	Utile	Utile
La présentation des conséquences directes sur l'annotation.	Utile	Utile	Utile	Utile	Utile
La présentation des conséquences indirectes sur l'annotation.	Utile	Utile	Inutile	Utile	Inutile
Le besoin d'utiliser l'ajustement.	Oui	Oui	Oui	Oui	Oui
L'utilité de l'interface traitant un type de changement.	Satisfaisante	Satisfaisante	Satisfaisante	Insatisfaisante	Satisfaisante

Tableau VII - 6 : Résultat d'évaluation selon le champ «comparaison d'EvOnto avec d'autres systèmes»

Comparaison d'EvOnto avec d'autres systèmes	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
L'utilité des informations	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui
	TextViz :	TextViz :	TextViz :	TextViz :	TextViz : Oui

d'assistance pour guider l'utilisateur à appliquer un changement.	Non	Non	Non	Oui un peu	un peu
	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non
Le traitement des conséquences d'un changement sur la RTO vous a-t-elle paru complète.	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui
	TextViz : Non	TextViz : Non	TextViz : Non	TextViz : Oui	TextViz : Oui
	Protégé 4 : Oui un peu	Protégé 4 : Oui un peu	Protégé 4 : Oui un peu	Protégé 4 : Non	Protégé 4 : Non

7.4.3.2 Expérimentation avec les données d'ACTIA

Dans cette section, nous présentons les différents résultats d'expérimentation en utilisant les données ACTIA ainsi que la technique d'évaluation par questionnaire. Les résultats sont décrites dans des tableaux en appliquons la même organisation que l'expérimentation des données ARTAL.

Tableau VII - 7 : Résultat d'évaluation selon le champ «Intérêt de l'outil»

Intérêt de l'outil	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
L'habitude de travailler avec ce type d'outil.	Oui	Oui	Oui	Oui	Oui un peu
L'assistance fournie par ce type d'outil.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
La présentation des effets possibles des changements de la RTO avant leur application.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
L'utilité des différentes stratégies d'évolutions.	Oui	Oui	Oui un peu	Oui	Oui
Nécessité de connaître les fiches impactées.	Oui	Oui	Oui	Oui	Oui

Tableau VII - 8 : Résultat d'évaluation selon le champ «Utilisation de l'outil»

Utilisation de l'outil	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
L'utilisation générale de cet outil.	Simple	Simple	Simple	Simple	Simple
Nécessité de chercher la place des informations.	Non	Non	Non	Non	Oui un peu

Nécessité de chercher la place des fonctionnalités.	Non	Oui un peu	Non	Non	Oui un peu
Manque des fonctionnalités.	Non	Non	Non	Non	Non

Tableau VII - 9 : Résultat d'évaluation selon le champ «Evaluation des changements proposés»

Evaluation des changements proposés	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
Les informations affichées concernant les éléments de la RTO avant de faire le changement.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
Les informations concernant les fiches impactées avant de faire le changement.	Nécessaire	Nécessaire	Nécessaire	Nécessaire	Nécessaire
Les stratégies d'évolution pour ce type de changement.	Nécessaire	Intéressante, mais pas nécessaire	Intéressante, mais pas nécessaire	Nécessaire	Nécessaire
La stratégie d'évolution choisie pour un type de changement.	Pertinente	Non Pertinente	Pertinente	Non Pertinente	Pertinente
La présentation des conséquences.	Inutile	Utile	Utile	Utile	Utile
La présentation des conséquences directes sur l'annotation.	Utile	Utile	Utile	Utile	Utile
La présentation des conséquences indirectes sur l'annotation.	Utile	Utile	Inutile	Utile	Inutile
Le besoin d'utiliser l'ajustement.	Oui	Oui	Oui	Oui	Oui
L'utilité de l'interface traitant un type de changement.	Satisfaisante	Satisfaisante	Satisfaisante	Insatisfaisante	Satisfaisante

Tableau VII - 10 : Résultat d'évaluation selon le champ «Comparaison d'EvOnto avec d'autres systèmes»

Comparaison d'EvOnto avec d'autres systèmes	Participants				
	Participant P1	***Participant P2***	***Participant P3***	***Participant P4***	***Participant P5***
L'utilité des	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui

informations d'assistance pour guider l'utilisateur à appliquer un changement.	TextViz : Oui un peu	TextViz : Oui un peu	TextViz : Oui un peu	TextViz : Oui un peu	TextViz : Oui un peu
	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non
Le traitement des conséquences d'un changement sur la RTO vous a-t-elle paru complète.	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui	EvOnto : Oui
	TextViz : Oui un peu	TextViz : Oui un peu	TextViz : Oui	TextViz : Oui	TextViz : Oui
	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non	Protégé 4 : Non

7.5 Analyse des résultats

Pour interpréter les résultats d'expérimentation, nous les avons organisées en fonction des différentes parties du questionnaire.

Intérêt de l'outil

Les tableaux VII-3 et VII-7 présentent les résultats d'expérimentation parlant de l'intérêt de l'outil. Tout d'abord, nous remarquons que 80 % des participants ont l'habitude de travailler avec ce genre de logiciel tel que *Protégé 4* et *Evolva*, ce qui fait qu'ils peuvent apporter une perspective sur l'évolution de la RTO et des annotations sémantiques des documents. Ensuite, tous les participants jugent nécessaire l'assistance fournie par EvOnto et soutiennent la nécessité de savoir et de comprendre les effets possibles des changements de la RTO avant leur application et aussi de connaître les fiches impactées. De plus, 80% des participants affirment qu'il est vraiment utile d'utiliser des stratégies d'évolutions. Ces stratégies les guident à bien formuler une demande de changement.

Pour résumer, le résultat obtenu affirme que les objectifs poursuivis dans cette étape à savoir : la nécessité de l'assistance fournie par le système, la nécessité de connaître les changements, les effets possibles des changements de la RTO et les fiches impactées afin de pouvoir se positionner par rapport à la nouvelle version de la RTO, sont atteints.

Utilisation de l'outil

En regardant les tableaux VII-4 et VII-8, nous observons que tous les participants confirment la simplicité de l'outil et aussi la présence des informations à l'endroit attendu. Deux de cinq participants affirment que les fonctionnalités sont bien placées dans l'endroit attendu. En revanche, le reste des participants (3/5) ont tous validé l'importance des fonctionnalités, en les qualifiant « oui un peu » pour leurs emplacements à l'endroit attendu. Deux participants ont argumenté leurs avis en nous suggérant des solutions qui nous permettent de faciliter la mise en œuvre du changement. Un des participants nous a proposé de supprimer le panel d'ajustement des entités liées à la RTO et traiter cette étape en utilisant les informations affichées dans le panel «informations lexicales et conceptuelles» et rendre ces informations modifiables. Un autre participant nous suggère une solution qui n'est pas dans le même sujet de la question (présence des fonctionnalités à l'endroit attendu). Il a précisé qu'il n'aimerait pas un système automatique, qui prend des décisions à sa place dans un domaine aussi sensible que l'annotation sémantique des documents. Il préfère, par contre, avoir le plus de choix possibles et le contrôle sur la modification des annotations, même si cela entraîne des manipulations supplémentaires et un effort cognitif plus intense de sa part. Il pense également que le fait de proposer et visualiser plusieurs solutions pourrait donner aux utilisateurs de nouvelles idées auxquelles il n'a pas pensé auparavant.

Cette dernière argumentation peut être liée à la dernière question (manque de fonctionnalités) de cette partie (utilisation de l'outil) où 80% des participants jugent qu'avec l'aide de l'outil ils peuvent avoir une vue complète et systématique de l'évolution et des différentes fonctionnalités de cet outil.

Pour résumer, un système adéquat est un système qui est, en même temps, *utile* et *utilisable*. Le résultat obtenu affirme que les objectifs poursuivis dans cette étape a savoir : identifier l'utilité du système et ses fonctionnalités, identifier les principaux problèmes d'utilisabilité s'il y a, et proposer des pistes de solutions, donnent des résultats positifs.

Evaluation des changements proposés

En regardant les tableaux VII-5 et VII-9, nous remarquons que tous les participants affirment qu'il est vraiment nécessaire, pour un outil de gestion d'évolution d'ontologie ou même un éditeur d'ontologie, de connaître les informations concernant les éléments de la RTO avant d'effectuer un changement et connaître les informations concernant les fiches impactées, si l'outil traite l'annotation sémantique des documents. Tout d'abord, quatre des cinq participants affirment que les stratégies d'évolution proposées pour les différents types de changements appliquées sont nécessaires. Donc le fait d'avoir accès à plusieurs stratégies d'évolution (plusieurs solutions) a été validé. Les participants ont précisé que, puisque le système EvOnto ne peut pas « savoir ce qui se passe dans la tête des gens », c'est mieux d'offrir aux utilisateurs un large spectre de solutions et de lui donner la possibilité de choisir celle qui semble la plus appropriée pour la modification de la RTO et des annotations sémantiques. L'idée d'utiliser plusieurs solutions pourrait aussi donner aux utilisateurs des nouvelles idées. En revanche, un des cinq participants a changé d'avis lors de la deuxième évaluation avec les données ACTIA. Le participant affirme que les stratégies sont nécessaires en utilisant les données ARTAL et intéressantes mais pas nécessaires au moment de l'évaluation avec les données ACTIA. Ce choix est argumenté par le fait que la RTO du partenaire ACTIA est mieux adaptée au corpus considéré, de plus, il est difficile de comprendre le domaine de diagnostic automobile qui compose cette RTO.

Ensuite, 80% des participants affirment la pertinence des différentes stratégies d'évolution choisie pour la liste des modifications demandées. Un de cinq participants juge qu'il y a des stratégies d'évolutions qui ne sont pas pertinentes à savoir les stratégies associées au changement MergeConcept. Pour ce dernier changement, le participant affirme que les stratégies proposées par le système à savoir : « [Stratégie 1] : attacher le nouveau concept au super-concept du premier concept sélectionné, [Stratégie 2] : attacher le nouveau concept au

super-concept du deuxième concept sélectionné » ne sont pas pertinentes car l'utilisateur est limité à attacher le nouveau concept a un des deux super-concepts des concepts sélectionnés. Le participant nous suggère une autre stratégie qui consiste à attacher le nouveau concept à n'importe quel concept de la hiérarchie de concepts. Le même participant s'interroge sur l'absence des stratégies d'évolutions pour le changement DeleteAssociatedTerm.

De plus, quatre des cinq participants affirment que présenter les conséquences d'un changement sur la RTO est utile. Donc le fait de guider les utilisateurs dans l'application de changements, et en particulier dans le choix qu'ils auront à faire pour résoudre les impacts (conséquences) d'un changement sur la RTO a été validé par les participants. Par exemple, pour la fusion de deux concepts, EvOnto demande à l'utilisateur de cliquer sur les deux concepts qu'il désire fusionner, de donner ensuite un nom au concept résultant, de spécifier son super-concept, de transférer les termes, et de transférer ou effacer les sous-concepts et les propriétés. Dans le même sujet de présentation des conséquences, tous les participants affirment aussi que la présentation des conséquences directes sur les annotations (sur les fiches impactées) est utile. En ce qui concerne la présentation des conséquences indirectes sur les annotations, 60 % des participants jugent utiles ces informations. En revanche, le reste des participants s'interrogent sur l'utilité de présenter les conséquences indirectes sur les annotations.

Ensuite, quatre de cinq participants affirment l'utilité et le besoin d'ajustement des conséquences. L'autre participant a remis en question la présentation de la liste des concepts proposés au moment de l'ajustement en suggérant d'utiliser un code de couleur pour classer ceux-ci par ordre d'importance.

Enfin, 80% des participants jugent que les interfaces de traitement des changements demandés sont satisfaisantes. 20% des participants affirment que les interfaces ne sont pas satisfaisantes sans argumentation de leurs avis.

Nous présentons dans le tableau VII- 11 et VII-12 la durée d'activité de chaque type de changement sur les données ARTAL et ACTIA.

Tableau VII - 11 : durée d'activité de chaque type de changement avec les données ARTAL

Types de changements	Participants				
	Part P1	**Part P2**	**Part P3**	**Part P4**	**Part P5**
DeleteConcept	4 min 49 sec	5 min 36 sec	4 min 58 sec	6 min 30 sec	7 min 12 sec
MergeConcept	3 min 37 sec	2 min 48 sec	3 min 44 sec	3 min 55 sec	4 min 09 sec
DeleteAssociatedTerm	2 min 40 sec	2 min 10 sec	3 min 00 sec	4 min 21 sec	3 min 56 sec
Total de la durée d'activité	***10 min 26 sec***	***10 min 34 sec***	***11 min 02 sec***	***14 min 06 sec***	***15 min 17 sec***

Nous remarquons que le temps d'activité par changement varie d'un participant à l'autre. Cette variation dépend du niveau d'expérience de chaque participant. Nous observons aussi que le temps requis pour appliquer le changement DeleteAssociatedTerm est le plus court. Ce résultat est dû à l'absence des stratégies d'évolutions ce qui implique aussi l'absence de la partie ajustement des conséquences de la RTO. Même avec la phase d'entrainement que nous avons réalisé avec les participants avant de commencer l'évaluation, nous remarquons que les participants P4 et P5 ont pris beaucoup de temps pour appliquer les trois changements. Ce résultat est dû au manque d'expérience et de connaissances dans le domaine de la gestion d'évolution d'ontologie et d'annotations sémantiques.

Tableau VII - 12 : durée d'activité de chaque type de changement avec les données ACTIA

Types de changements	Participants				
	Part P1	**Part P 2**	**Part P3**	**Part P4**	**Part P5**
DeleteConcept	2 min 13 sec	2 min 54 sec	3 min 27 sec	4 min 19 sec	4 min 00 sec
MergeConcept	1 min 51 sec	2 min 24 sec	2 min 07 sec	3 min 05 sec	3 min 31 sec
DeleteAssociatedTerm	2 min 00 sec	1 min 46 sec	2 min 23 sec	1 min 32 sec	2 min 24 sec
Total de la durée d'activité	***6 min 04 sec***	***6 min 24 sec***	***7 min 57 sec***	***8 min 56 sec***	***9 min 55 sec***

En comparant les résultats de ce tableau avec ceux du tableau VII-11, nous remarquons que le temps requis pour appliquer les trois changements est diminué d'une façon remarquable même pour les participants P4 et P5 qui sont caractérisés par un manque de connaissances dans le domaine de gestion d'évolution d'ontologie et d'annotations sémantiques des documents.

Pour résumer, tous les participants disent avoir une meilleure compréhension des effets de changements après avoir consulté les conséquences présentées par EvOnto. Tous les participants affirment que les stratégies d'évolution proposées les guident dans les choix qu'ils auront à faire pour résoudre les impacts d'un changement sur la RTO et aussi sur les annotations. Pour finir, tous les participants trouvent un intérêt à l'ajustement manuel des conséquences de la RTO. Le résultat obtenu affirme que les objectifs poursuivis dans cette étape à savoir : l'utilité d'utiliser des stratégies d'évolution et de ses pertinences, l'utilité de présenter les conséquences sur la RTO, les conséquences directes et indirectes sur les annotations, et le besoin d'ajustements de ses conséquences, sont atteints.

Comparaison d'EvOnto avec d'autres systèmes

En se basant sur les résultats obtenus dans les tableaux VII-6 et VII-10, tout d'abord, nous remarquons que tous les participants affirment l'utilité des informations d'assistance fournies par EvOnto pour guider l'utilisateur à appliquer un changement. En comparant les résultats

EvOnto avec ceux de l'outil TextViz, 60% des participants jugent que les informations d'assistance fournie par TextViz ne sont pas utiles. En revanche, 40% des participants valident l'utilité de ces informations, en les qualifiant « oui un peu ». Ces participants argumentent leurs avis par l'existence des informations d'assistance fournies par TextViz au moment de la ré-annotation des fiches impactées par un changement. Une autre comparaison a été faite avec l'outil Protégé 4, où tous les participants affirment que l'outil ne présente pas des informations d'assistance aux utilisateurs pour les guider à appliquer un changement.

Ensuite, en ce qui concerne le traitement des conséquences d'un changement sur la RTO, tous les participants jugent que le traitement dans EvOnto est complet. Pour l'outil TextViz, 80% des participants affirment la non-complétude des traitements des conséquences d'un changement sur la RTO. 60% des participants valident la complétude de traitement des conséquences avec Protégé 4, en les qualifiant « Oui un peu ». Le reste des participants jugent la non-complétude du Protégé 4 par rapport à ce type de traitement.

Nous présentons dans la suite, le temps d'activité pour appliquer la liste de changements demandés avec les données ARTAL et ACTIA en utilisant les outils EvOnto, TextViz et Protégé.

Tableau VII - 13 : durée d'activité de trois types de changements avec les données ARTAL

Types d'outils utilisés	**Participants**				
	Part P1	Part P2	Part P3	Part P4	Part P5
EvOnto	10 min 26 sec	10 min 34 sec	11 min 02 sec	14 min 06 sec	15 min 17 sec
TextViz	13 min 47 sec	11 min 56 sec	14 min 17 sec	21 min 14 sec	17 min 32 sec
Protégé 4	8 min 56 sec	10 min 23 sec	12 min 34 sec	14 min 49 sec	16 min 41 sec

Tableau VII - 14 : durée d'activité de trois types de changements avec les données ACTIA

Types d'outils utilisés	Participants				
	Part P1	Part P2	Part P3	Part P4	Part P5
EvOnto	6 min 04 sec	6 min 24 sec	7 min 57 sec	8 min 56 sec	9 min 55 sec
TextViz	11 min 23 sec	12 min 09 sec	13 min 33 sec	16 min 19 sec	15 min 41 sec
Protégé 4	6 min 17 sec	9 min 40 sec	12 min 51 sec	12 min 30 sec	15 min 13 sec

En se basant sur les résultats du tableau VII-13 et VII-14, nous remarquons que le temps d'activité avec les trois types de changements varie d'un outil à un autre et aussi d'un participant à un autre. Avec les données ARTAL ou ACTIA, nous observons que la durée d'activité pour appliquer les trois changements est plus courte avec EvOnto. En effet, trois participants affirment qu'il y a des problèmes de traitement des conséquences de la RTO avec TextViz à savoir, un problème de transfert des propriétés vers le nouveau concept une fois que le changement MergeConcept a été validé, problème de validation automatique des fiches ré-annotées sans demande de l'avis des utilisateurs. Il y a aussi un autre problème qui concerne les sous concepts et les instances. Un des participants nous a informé que durant l'exécution de l'opération DeleteConcept avec TextViz, il était obligé de supprimer tous les concepts fils et les instances, ce qui est par fois incohérent. Un autre participant a eu des problèmes avec Protégé 4 en particulier avec les propriétés qui restent sans un concept domaine ou co-domaine au moment de la suppression d'un concept. Enfin, TextViz et Protégé 4 ne traitent pas convenablement les conséquences sur la RTO puisque ils ne les prennent pas en compte.

Pour résumer, le résultat obtenu affirme que l'objectif poursuivi dans cette étape à savoir : avoir un système qui permet l'atteinte des buts avec un effort moindre, c'est-à-dire avec un minimum d'effort et en un minimum de temps identifier, est atteint.

7.6 Bilan d'évaluation de la qualité d'EvOnto

D'une façon générale, et même si l'interface du système EvOnto comporte quelques lacunes, nous remarquons que l'outil permet d'aider l'ontologue dans la maintenance d'une RTO et des annotations sémantiques des documents. Il permet de mieux guider l'ontologue à formuler une demande de changement, évaluer son impact sur la qualité de la RTO et aussi sur les annotations sémantiques, et décider ensuite de leur mise en œuvre. En effet, nous avons obtenu des résultats satisfaisants pour deux domaines complètement différents et deux langues différentes. Les résultats des évaluations avec les données du partenaire ARTAL et ACTIA sont de 80% de pertinence pour les différents objectifs que nous avons énoncés pour d'évaluation du système.

Tout d'abord, tous les participants confirment l'utilité des informations d'assistance fournie par EvOnto pour guider l'ontologue à appliquer un changement sur la RTO. Ensuite, même si tous les participants ont émis un avis positif au sujet de l'utilisation des stratégies, ils ont cependant apprécié différemment son importance. Ainsi, tous les participants ont affirmé connaître déjà une bonne partie des effets présentés sur les fiches impactées. De plus, les participants affirment leurs besoins en ce qui concerne l'ajustement des conséquences. Pour finir, le temps mis pour faire évoluer une RTO avec notre outil est plus court que celui des outils TextViz et Protégé et avec un minimum d'effort.

7.7 Conclusion

Dans ce chapitre, nous avons évalué la qualité de notre système d'évolution de RTO et des annotations sémantiques des documents. Tout d'abord, nous avons décrit les modes d'interaction avec notre système ainsi que les scénarios d'utilisation. EvOnto a été implémenté sous la forme d'un plugin ajouté à l'éditeur d'ontologies Protégé. Il complète l'outil de gestion manuelle de RTO et d'annotations sémantiques de documents TextViz.

Ensuite nous avons présenté le contexte de nos évaluations, nous avons commencé par présenter les critères d'évaluation appliqués ainsi que la procédure adoptée. En outre, nous avons présenté les résultats d'évaluation pour les deux domaines d'ARTAL et ACTIA. Enfin, nous avons analysé les résultats d'évaluation des systèmes afin d'en tirer des conclusions concernant le système développé dans cette thèse et qui permettront de l'améliorer.

L'évaluation qualitative du système EvOnto a permis de montrer sa pertinence et son utilité. Le système permet aux utilisateurs de dégager une vue correcte des changements et de l'aide offerte pour mieux formuler leurs changements. Ces évaluations nous ont également offert la possibilité d'identifier les prochaines étapes pour améliorer EvOnto, principalement l'interface graphique et l'ajout de certaines fonctionnalités, comme l'utilisation du code couleur pour classer les concepts par ordre d'importance, l'intégration de l'interface d'ajustement des conséquences avec l'interface « informations lexicales et conceptuelles », et le développement d'autres stratégies d'évolutions pour donner aux utilisateurs de nouvelles idées et plus de solutions.

CHAPITRE VIII

Conclusion et perspectives

Sommaire

Dans ce chapitre, nous récapitulons les innovations apportées par la thèse et identifions les perspectives d'amélioration et d'extension à court et long terme. Tout d'abord, nous présentons l'originalité de notre contribution, en décrivant ses apports et ses limites. Ce travail est loin d'être achevé et plusieurs perspectives peuvent être envisagées. Ces perspectives seront détaillées dans les sections 8.2 et 8.3.

8.1 L'évolution d'ontologie : une question insuffisamment étudiée

A travers ce travail, nous avons rendu compte des problèmes qui peuvent survenir lorsqu'une ressource termino-ontologique vient à évoluer dans un environnement dynamique. En effet, dans de nombreux domaines, les connaissances associées, les collections et les annotations sémantiques construites avec l'ontologie doivent être modifiées régulièrement et en cohérence pour s'adapter à l'évolution du domaine sur lequel elles portent et des collections documentaires annotées. Suivant le type d'évolution que cette ressource termino-ontologique subit, nous avons apporté des éléments de réponse pour savoir (a) Comment mieux représenter les opérations de changement de la ressource termino-ontologique afin d'aider l'ontologue à choisir une opération adaptée au changement qu'il veut effectuer ? (b) Comment analyser les effets d'un changement et les gérer au mieux (par exemple en les minimisant) ? (c) Comment peut-on détecter une perte de cohérence suite à des changements dans la ressource termino-ontologique ? et (d) Comment résoudre les problèmes de cohérence détectés suite à la modification de ressource termino-ontologique pour assurer la consistance globale du système (au sein de la ressource termino-ontologique mais aussi sur ses utilisations) ?

Les travaux de [Luong, 2007] [Rogozan, 2009] et [Stojanovic, 2004] se sont intéressés au développement d'outils destinés à réaliser l'évolution d'ontologies et d'annotations

sémantiques de façon automatique ou semi-automatique. Toutefois, aucun de ces travaux n'aboutit a des résultats entièrement satisfaisants, soit car seules certaines étapes du processus sont automatisées, soit parce que seule une partie des connaissances peut évoluer.

Notre recherche a débouché sur la mise au point du logiciel « EvOnto » qui aide l'ontologue à adapter une ressource termino-ontologique à un corpus pour produire les annotations les plus précises possible, et à adapter les annotations aux évolutions de la ressource. Il permet de guider interactivement l'ontologue pour formuler une demande de changement, évaluer son impact sur la qualité de la ressource termino-ontologique et aussi sur les annotations sémantiques, et décider ensuite de leur mise en œuvre.

8.2 Synthèse des contributions

Notre contribution consiste à associer l'évolution de la RTO et celle des annotations sémantiques des documents. Nous avons défini pour cela les types de changements applicables (élémentaires ou complexes), ainsi que des stratégies d'évolution que l'utilisateur peut ajuster en définissant les conséquences d'un changement sur cette RTO et les annotations. Cette contribution préserve la cohérence interne de la RTO mais aussi la cohérente de la RTO avec les annotations sémantiques des collections documentaires.

Nous avons défini une approche d'évolution, EvOnto (Evolution d'Ontologie), et un logiciel qui la met enoeuvre dans l'environnement d'annotation automatique de documents (TextViz) défini dans le cadre du projet DYNAMO. EvOnto prévoit plusieurs scénarios d'évolution, qui correspondent à des choix différents dans les changements et leurs conséquences sur la RTO et sur les annotations.

L'originalité d'EvOnto est d'assurer un support à ces différents scénarios, tout en gérant une forte imbrication entre la RTO et les annotations sémantiques. D'une part, toute évolution

de la RTO implique de détecter les annotations devenues incohérentes suite aux 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 de faire évoluer la RTO, qui, après évolution, implique en retour le maintien de la cohérence sur l'ensemble des annotations. L'étude de la qualité des annotations comme moyen de justifier l'évolution de la RTO est une des originalités d'EvOnto. La qualité des annotations est évaluée par la vérification de certaines contraintes, comme le fait de retrouver certains concepts dans les annotations, ou que la majorité des mots du document soient annotés par des concepts. Lorsque ces contraintes ne sont pas vérifiées, alors la RTO doit être modifiée pour produire de meilleures annotations.

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 qu'il recherche dans les documents pour les annoter par un graphe d'instances de concepts en relation. L'évolution de la RTO se déroule selon les étapes classiques de l'évolution des ontologies dans l'état de l'art. En revanche, chacune de ces étapes est innovante car elle traite de manière couplée l'ontologie et sa composante terminologique. De plus, l'étape de propagation des changements aux annotations est particulièrement fouillée, et correspond au processus d'évolution des annotations.

EvOnto est donc une méthode et un outil qui se situent dans la continuité de travaux antérieurs sur l'évolution d'ontologies, et qui les étendent pour traiter des problèmes spécifiques aux RTO : la gestion des termes associés aux concepts et la gestion conjointe des annotations. Pour mieux gérer ces deux facettes, l'approche générale EvOnto est structurée selon deux processus.

8.2.1 Processus d'évolution de la ressource termino-ontologique

Le processus d'évolution de la RTO comporte quatre phases, à savoir : Expression d'un changement, Présentation des stratégies et simulation de leurs conséquences, Choix de la stratégie et ajustement des conséquences et Application du changement à la RTO. Ce processus cherche à préserver la cohérence de la RTO. Il guide l'ontologue de manière interactive pour formuler une demande de changement et évaluer son impact.

8.2.2 Typologie des changements

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. Les changements peuvent être élémentaires ou complexes. Cette typologie élargit la conceptualisation de [Stojanovic, 2004], par l'ajout d'un certain nombre de caractéristiques dont la plus notable est 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 d'évolution dans la RTO, il est possible de définir un ensemble de changements supplémentaires qui doivent être effectués pour garder la ressource dans un état cohérent. Lorsque plusieurs effets sont possibles pour une même évolution, un choix doit être effectué soit arbitrairement par le système de gestion des évolutions dans lequel on aura fixé un des effets possibles, soit par l'ontologue. La manière de faire intervenir l'ontologue pour formuler le choix d'un de ces effets (au cas par cas, une fois pour toutes, …) revient à fixer une stratégie d'évolution.

8.2.3 Stratégies d'évolution et ajustement

Pour chaque type de changement, nous avons identifié différentes stratégies d'évolution, c'est-à-dire différentes manières de gérer les conséquences de ce changement sur les entités

de la RTO liées à l'entité modifiée. Ces stratégies permettent d'anticiper les conséquences possibles d'un changement avant de le rendre effectif dans la RTO. Elles sont présentées à l'ontologue avec un ensemble d'informations afin qu'il puisse décider de la stratégie qu'il souhaite adopter. Une fois sa décision prise, le changement est répercuté sur la RTO et les annotations.

Les stratégies d'évolution sont adaptables. Chacune propose un ensemble de conséquences prédéfinies pour chaque type de changement, mais l'ontologue a la possibilité d'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. Les stratégies sont adaptables pour mieux répondre à la diversité des situations à l'origine d'un changement. En effet, pour une même modification, l'utilisateur peut préférer telle ou telle conséquence selon les documents ou les concepts, alors que le système ne propose que des choix systématiques.

8.2.4 Processus d'évolution des annotations

Pour préserver la cohérence de la nouvelle version de la RTO avec les annotations sémantiques des documents basées sur cette ressource, celles-ci devront être adaptées à la nouvelle version de la RTO. Cette adaptation suit un processus en trois phases : détection des annotations incohérentes, rectification des annotations incohérentes et mise à jour de la base d'annotations.

Pour détecter les annotations à modifier, le système recherche dans les graphes d'annotation les nœuds qui correspondent à des entités de la RTO qui ont été modifiées. Une fois les annotations incohérentes déterminées, elles doivent être corrigées. Nous proposons deux méthodes. Selon la première, l'ontologue relance le module d'annotation automatique

seulement pour les documents concernés. La deuxième méthode consiste à adapter les annotations localement dans le fichier d'annotation, par des requêtes changeant le type des instances et en suivant des stratégies d'adaptation d'annotations. Chacune d'entre elles est associée à une des stratégies d'évolution de la RTO.

Chaque fois que TextViz produit une nouvelle annotation, une vérification de la qualité des annotations est lancée, selon des critères définis par l'utilisateur. A partir de cette évaluation, l'ontologue dispose d'indices pour savoir quels concepts ajouter ou modifier. Il peut alors réaliser dans la RTO les changements nécessaires pour améliorer les annotations.

EvOnto permet de vérifier les annotations produites automatiquement, ce qui est une originalité de notre approche. Dans chaque document de la collection, les annotations produites par le système doivent respecter des critères choisis au préalable par l'ontologue. 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 qui vérifie automatiquement que les annotations satisfont ces critères. Ce module demande à l'utilisateur de définir les critères de qualité des annotations propres au corpus et au domaine d'application. Ensuite, à la demande de l'ontologue, ces critères sont appliqués sur tout le corpus et chaque document annoté est évalué.

8.2.5 Résultat d'évaluation d'EvOnto

Nous avons évalué la qualité de notre système d'évolution de RTO et des annotations sémantiques des documents avec deux jeux de données en utilisant des questionnaires. Les résultats de cette évaluation sont plutôt encourageants et ont permis de montrer la pertinence et l'utilité d'EvOnto. Le système a permis aux utilisateurs de dégager une vue correcte des changements et les aide à mieux les formuler.

8.3 Perspectives à court terme

Réduire l'impact de la modification des annotations : Dans le processus d'évolution des annotations, nous avons proposé un processus en deux temps. La première étape - détection des annotations incohérentes – consiste à rechercher dans la base d'annotations les annotations concernées par les modifications de la RTO. La deuxième - rectification des annotations incohérentes -permet de modifier les annotations devenues incohérentes à cause de ces changements. Pour la première étape, deux méthodes sont possibles, la première - interroger les graphes d'annotation en utilisant des requêtes SPARQL – est celle développée dans la thèse. Nous allons donc développer et automatiser la deuxième méthode - rechercher directement dans les documents les occurrences des termes liés aux éléments de la RTO modifiés – afin de pouvoir comparer les résultats obtenus. En effet, cette solution présente deux avantages : la réduction du temps de calcul, qui permet de déterminer les documents qui sont affectés par le changement ; la ré-annotation automatique de ces documents. Le temps de calcul reste raisonnable si la base de documents ne comprend que quelques centaines de documents courts.

Gérer d'autres types de changements : Nous nous sommes concentrés sur l'ajout, la suppression ou la modification des concepts et des termes d'une RTO. Cependant, il existe d'autres types de changements qui ne sont pas encore définis dans EvOnto, à savoir la modification des relations exprimées par des propriétés d'objets OWL. Nous convenons donc de développer ces types de changements et les stratégies d'évolutions correspondantes appliquées pour la RTO et les stratégies d'adaptations correspondantes pour les annotations.

Gérer des changements complexes : Théoriquement, l'ensemble de changements complexes est infini, parce que le nombre de combinaisons possibles est aussi infini. Ceux qui nous intéressent sont ceux qui sont pertinents pour l'ontologue et qui correspondent à un enchaînement de modifications fréquemment réalisé, comme le fait de regrouper des concepts

ou de diviser un concept. Nous envisageons prochainement de traiter certains changements complexes, de définir les stratégies d'évolution correspondantes pour la RTO et les annotations. Nous avons choisi de gérer les changements suivants : *MergeProperty*, *SplitProperty*, *GroupProperty*, *MoveProperty*, *AddMaxCardinality* et *AddMinCardinality*.

Pour assurer la gestion de l'historique des changements, nous proposons de concevoir un **modèle de journal des modifications**, ainsi qu'un langage de formalisation des changements pour rendre compte des changements dans ce journal.

La première évaluation d'EvOnto sur deux jeux de données nous a permis de recenser les avantages d'EvOnto et de suggérer des améliorations. Cependant nous devons **tester EvOnto sur d'autres jeux de données** offrant d'autres caractéristiques (ontologie plus volumineuse, textes annotés plus longs, etc.). Nous envisageons d'utiliser le corpus et la RTO construits pour le projet MOANO[25]. Ce projet vise à offrir une infrastructure logicielle (plateforme) permettant à tout type de périphérique mobile de capturer des informations contextuelles de nature spatio-temporelles, de lesenrichir afin d'augmenter leur pouvoir sémantique et de faciliter leur indexation en vue d'une exploitation/recherche ultérieure par les applications. Dans ce projet, une ressource termino-ontologique est utilisée pour l'annotation sémantique de documents et la recherche d'information à l'aide de TextViz. Il a été choisi de construire cette RTO à partir de pages web structurées et de l'enrichir grâce à l'annotation sémantique d'un ouvrage au sein duquel sont recherchées des informations. Dans cette perspective, nous prévoyons d'utiliser EvOnto. Ensuite une évaluation comparative entre les résultats d'enrichissement avec EvOnto et les résultats obtenus par analyse de texte s'avère nécessaire pour évaluer la performance de notre système.

[25] "Modèles et Outils pour Applications NOmades de découverte de territoire", projet ANR-2010-CORD-024-03 dont l'IRIT, le LIUPPA, le LIG et le LIFL sont partenaires. http ://moano.liuppa.univ-pau.fr

Actuellement, EvOnto fonctionne avec l'éditeur d'ontologies Protégé et l'outil d'annotation sémantique de documents TextViz. Or ce dernier n'est pas en libre accès, pour cela il est actuellement impossible de mettre en ligne EvOnto. Dans cette perspective, nous prévoyons soit de **remplacer l'outil TextViz par un autre logiciel** en libre accès, soit de développer un nouvel outil d'annotation sémantique à coupler à Protégé et EvOnto.

Pour faciliter l'utilisation d'EvOnto, et pour rendre l'interface plus conviviale et organiser les informations d'une manière plus suggestive, nous proposons l'utilisation d'un code couleur pour représenter les modifications faites par l'ontologue et lui permettre de mieux comprendre ses choix de modifications.

Finalement, dans la perspective de diffuser et valider notre travail à l'échelle internationale, nous visons une publication dans une revue internationale avec un facteur d'impact où nous présentons EvOnto et sa validation sur les études de cas des différents partenaires du projet Dynamo.

8.4 Perspectives à plus long terme pour EvOnto

Les principales perspectives d'amélioration et d'extension à long terme qui apparaissent à l'issue de cette thèse visent à guider plus l'utilisateur d'EvOnto. Afin d'aller plus loin dans l'automatisation et l'optimisation des stratégies d'évolution de la RTO et les stratégies d'adaptation des annotations, il serait possible de s'appuyer sur les critères d'évaluation de la qualité des annotations. L'idée serait de retenir en priorité la stratégie d'évolution de la RTO qui améliorera le plus la qualité des annotations tout en préservant celle de l'ontologie. Pour cela, on classera les stratégies proposées – si plusieurs stratégies sont possibles – pour ne choisir que la meilleure au regard des critères de qualité. Cela évitera de solliciter l'intervention de l'ontologue.

8.4.1 Aide à la découverte de nouveaux termes

Si, dans une RTO, les termes nécessaires à l'indexation d'une collection de documents sont absents, les annotations générées par TextViz seront de mauvaise qualité. Or selon la méthode implémentée dans EvOnto, l'ajout de termes se fait soit au fur et à mesure que sont parcourues les fiches mal annotées, soit à l'initiative de l'utilisateur. Même si les termes à ajouter à la RTO sont connus a priori, leur ajout requiert beaucoup de temps. Ce faisant, l'utilisateur peut commettre des erreurs d'interprétation du sens des termes et les associer à un mauvais concept de la RTO. Dans cette perspective, il serait souhaitable d'envisager une approche complémentaire pour identifier des changements à apporter à une RTO, en considérant des propositions faites automatiquement par l'analyse automatique du contenu des fiches à l'aide de logiciels de traitement automatique des langues (TAL). On peut utiliser pour cela des extracteurs de termes qui fournissent des indices linguistiques de concepts. En effet, en extrayant automatiquement les termes de nouvelles fiches, il est aisé d'identifier ceux qui ne sont pas encore présents dans la RTO, et de les suggérer ensuite pour les y ajouter. La difficulté est de déterminer automatiquement s'ils sont des synonymes de termes existants (et doivent être ajoutés comme termes associés à des concepts existants) ou s'ils renvoient à de nouveux concepts.

8.4.2 Situer les nouvelles entités dans l'ontologie existante

L'étape précédente donne lieu à la création de nouvelles entités ontologiques qui doivent être placées relativement aux connaissances antérieures. Pour cela, il est nécessaire de détecter les relations entre ces nouvelles entités et celles présentes dans la ressource termino-ontologique. Pour ce faire, nous nous proposons d'exploiter encore le contenu des corpus. Les extracteurs de relations permettent de définir de nouvelles relations entre entités. Ces relations guident pour placer les nouveaux concepts par rapport aux concepts existants. Dans le cas d'ajout de concepts ou de termes, des méthodes statistiques permettent de repérer les cas fréquents de cooccurrence en corpus de

termes associés aux entités ajoutées avec des termes de la ressource termino-ontologique. Lorsque les termes co-occurrent souvent, on peut faire l'hypothèse qu'il existe une relation entre eux.Toutefois, cette méthode ne permet pas de qualifier la nature de la relation. D'autres techniques inspirées de la fouille de données s'appuient sur l'étude de séquences plus complexes et identifient plus finement des relations sémantiques et la nature de ces relations. Enfin, le regroupement de termes en classes (clustering) en fonction de leurs voisinages en corpus permet de suggérer des relations entre clusters ou entre les termes d'un cluster. Chaque cluster représente alors la possibilité qu'une relation, non définie, existe entre les concepts qu'il regroupe.

Bibliographie personnelle

Articles de revues nationales / *National journals articles*

Anis Tissaoui, Nathalie Aussenac-Gilles, Nathalie Hernandez, Philippe Laublet. EvOnto : un outil d'évolution d'une ressource termino-ontologique pour l'annotation sémantique. *Technique et Science Informatiques*, Hermès Science, Vol. 32, N. 7-8, p. 817-840, 2013.

Conférences et workshops internationaux / *International conferences articles*

Anis Tissaoui, Nathalie Aussenac-Gilles, Nathalie Hernandez, Philippe Laublet. *EvOnto une approche pour maintenir la cohérence entre une ressource termino-ontologique et des annotations sémantiques* (regular paper). Conférence Internationale sur la Terminologie et l'Intelligence Artificielle (TIA 2011), Paris (F), 08-09/11/2011, K. Kageura, P. Zweigenbaum (Eds.), INALCO, p. 24-30, 2011.

Anis Tissaoui, Nathalie Aussenac-Gilles, Nathalie Hernandez, Philippe Laublet. *EVONTO - Joint evolution of ontologies and semantic annotations. (*short paper). International Conference on Knowledge Engineering and Ontology Development (KEOD 2011*)*, Paris, 26-29/10/2011, J. Dietz (Eds.), INSTICC - Institute for Systems and Technologies of Information, Control and Communication, p. 226-231, 2011.

Anis Tissaoui, Nathalie Aussenac-Gilles, Philippe Laublet, Nathalie Hernandez. *EvOnto : évolution de ressource termino-ontologique pour l'annotation sémantique* (regular paper). 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 (Best paper).

Conférences sans actes publiés / *Conference articles without published proceedings*

Anis Tissaoui, Nathalie Aussenac-Gilles, Philippe Laublet, Nathalie Hernandez. *Méthodologie d'évolution d'une ressource termino-ontologique : EvOnto* (poster). Journées Francophones d'Ingénierie des Connaissances (IC), Nîmes (France), 08-11/06/2010, Sylvie Despres (Eds.).

Anis Tissaoui, Nathalie Aussenac-Gilles, Philippe Laublet, Nathalie Hernandez. *Gestion des évolutions d'une ressource termino-ontologique et des annotations sémantiques*. Atelier "Evolution d'ontologies" des 21emes Journées francophones d'Ingénierie des Connaissances (IC2010), Nîmes (France), 08/06/2010, N. Aussenac-Gilles, N. Hernandez, P. Laublet (Eds.).

Anis Tissaoui. *Typologie de changements et leurs effets sur l'évolution de Ressources Termino-Ontologiques* (poster). IC 2009 : Posters des 20es Journées Francophones d'Ingénierie des Connaissances, Hammamet (Tunisie), 25-29/05/2009, F. Gandon (Eds.).

Rapports / *Reports*

Philippe Laublet, Nathalie Aussenac-Gilles, Valérie Camps, Pierre Glize, Nathalie Hernandez, H. Maurel, Mohamed Mbarki, Josiane Mothe, Bachelin Jhonn Victorino Ralalason, Axel Reymonet, Bernard Rothenburger, Zied Sellami, Jérôme Thomas, Anis Tissaoui. Projet ANR 07 TLOG 004 01 - DYNAMO (DYNAMic Ontology for information retrieval) : *Etat de l'art* - Livrable lot 2. Rapport de contrat, Dynamo 2.1, IRIT, décembre 2009. Accès : ftp://ftp.irit.fr/IRIT/IC3/Rapport_lot2_integre_F7.pdf

Bibliographie

[Amardeilh, 2007] F. Amardeilh. *Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d'une plateforme logicielle*. Thèse de Doctorat, Université Paris X, (2007).

[Amardeilh, 2008] F. Amardeilh. *Semantic Annotation & Ontology Population*. In Semantic Web Engineering in the Knowledge Society, Cardoso J. et Lytras M. D. (Eds), Idea Group Publishing, (2008).

[Amardeilh *et al.*, 2005] F. Amardeilh, P. Laublet et J.-L. Minel. *Document Annotation and Ontology Population from Linguistic Extractions*. In Proceedings of the International Conference on Knowledge Capture (KCAP 2005), Banff, 1-5 octobre (2005).

[Aussenac-Gilles *et al.*, 2005] N. Aussenac-Gilles, B. Biébow et S. Szulman. *Modélisation du domaine par une méthode fondée sur l'analyse de corpus*. Ingénierie des Connaissances. R. Teulier, P. Tchounikine et J. Charlet (Eds.), Paris : L'Harmattan, 40-72, (2005).

[Aussenac-Gilles *et al.*, 2008] N. Aussenac-Gilles, S. Despres et S. Szulman. *The TERMINAE Method and Platform for Ontology Engineering from texts*. P. Buitelaar, P. Cimiano (eds), Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text, IOS Press. P. 199 – 223, Volume 167, (2008).

[Banerjee *et al.*, 1987] J. Banerjee, W. Kim, H.J. Kim et H. Korth. *Semantics and implementation of schema evolution in object-oriented databases*. ACM SIGMOD Record: Vol. 16(3), pp. 311-322, New York, USA: ACM, (1987).

[Berners-Lee, 2005] T. Berners-Lee. Uniform Resource Identifier (URI): Generic Syntax. From http://www.gbiv.com/protocols/uri/rfc/rfc3986.html (2005).

[Bloehdorn *et al.*, 2006] S. Bloehdorn, P. Haase, Y. Sure et J. Voelker. *Ontology evolution*. In J. Davies, R. Studer, & P. Warren (Ed.s), Semantic Web Technologies, Trends and research in Ontology-based Systems, pp. 51-70. John Wiley & Sons Publication. (2006).

[Breche et Wörner, 1995] P. Breche et M. Wörner. *How to remove a class in an object database system*. Proceedings of the 2nd international conference on applications of databases (ADB-95) p. 235-248, (1995).

[Castano, 2006] S. Castano. *Ontology evolution: The BOEMIE approach*. BOEMIE Workshop, at the conference EKAW 06, (2006).

[Castano *et al.*, 2007] S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, S. Melzer. *Ontology dynamics with multimedia information: The BOEMIE evolution methodology*. In G. Flouris, et M. d'Aquin (Eds.) Proceedings of the International Workshop on Ontology Dynamics (IWOD-07) at ESWC 07 Conference, pp. 41-54. (2007).

[Corcho et Gomez-Pérez, 2000] O. Corcho et A. Gomez-Pérez. *A roadmap for ontology*

specification languages. In 12th international conference on knowledge engineering and knowledge management (EKAW-2000), Springer, Juan-les-Pins, France, (2000)

[D'Aquin *et al.*, 2008] M. d'Aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez et D. Guidi. *Towards a New Generation of Semantic Web Applications*. IEEE Intelligent Systems, Vol. 23, Issue 3, May/June (2008).

[Djedidi *et al.*, 2007] R. Djedidi, H. Abboute, M.-A Aufaure. *Evolution d'ontologies : Validation des changements basée sur l'évaluation.* Proceedings des 1ères journées francophones sur les ontologies "Les ontologies : mythes, réalités et perspectives" (JFO 2007), édition du centre de Publication Universitaire, F. Gargouri, D. Benslimane et P. Bourque, Sousse, Tunisie, (2007).

[Djedidi et Aufaure, 2008a] R. Djedidi et M.-A. Aufaure. *Gestion des changements d'une ontologie : Résolution d'inconsistances guidée par l'évaluation de la qualité*. Actes de l'atelier Modélisation des Connaissances (ModCo'08), 8èmes journées francophones "Extraction et Gestion des Connaissances" (EGC'08), INRIA Sophia Antipolis, France, 29 janvier-1er février (2008).

[Djedidi et Aufaure, 2008b] R. Djedidi et M.-A. Aufaure. *Enrichissement d'ontologies : maintenance de la consistance et évaluation de la qualité.* 19èmes journées francophones d'Ingénierie des Connaissances (IC'08), Nancy, France, (2008).

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

[Djedidi et Aufaure, 2009] R. Djedidi et M.-A. Aufaure. *Change Management Patterns (CMP) for Ontology Evolution Process*. Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD2009) in ISWC 2009, M. D'Aquin & G. Antoniou (Eds.), CEUR Workshop Proceedings, Vol. 519, Washington DC, USA, 26 Octobre, (2009).

[Djedidi et Aufaure, 2010] R. Djedidi et M.-A. Aufaure. *Onto-Evoal an Ontology Evolution Approach Guided by Pattern Modelling and Quality Evaluation.* Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2010), Sofia, Bulgaria, (2010).

[Fensel, 2001] D. Fensel. *Ontologies, a Silver Bullet for Knowledge Management & Electronic Commerce*. Springer-Verlag, 147 pages, (2001).

[Flouris, 2006] G. Flouris. *On Belief Change and Ontology Evolution*. Thèse de doctorat, University of Crete, Department of Computer Science. (2006).

[Flouris *et al.*, 2005] G. Flouris, D. Plexousakis, et G. Antoniou. *On applying the AGM theory to DLs and OWL*. In Y.Gil, E. Motta, V. Benjamins, & M. Musen, (Eds.), LNCS: Vol. 3729. The Semantic Web – ISWC 2005, pp. 216-231. Berlin, Germany: Springer, (2005).

[Flouris *et al.*, 2006b] G. Flouris, Z. Huang, J.Z. Pan, D. Plexousakis, H. Wache. *Inconsistencies, negations and changes in ontologies*. In Proceedings of AAAI'06. (2006).

[Flouris et Plexousakis, 2005] G. Flouris et D. Plexousakis. *Handling ontology change: Survey and proposal for a future research direction*. Technical Report FORTH-ICS/TR-362, Institute of Computer Science, FORTH, September (2005).

[Gargouri, 2008] Y. Gargouri. Maintenance des ontologies à partir d'analyses textuelles. Thèse de doctorat. Université du Québec à Montrél. (2008).

[Gediga *et al.*, 1993] G. Gediga, K. Hamborg, et 1. Düntsch. *Evaluation of Software Systems*. In A. Kent et J. G. Williams (Eds.), Encyclopedia of Library and Information Science, Vol. 72, pp. 166-192. (1993).

[Gruber, 1993] R. Gruber Thomas. *A Translation Approach to Portable Ontology Specifications*. Knowledge Acquisition, Vol. 5, No. 2. (June 1993), pp. 199- 220. (1993).

[Guizzardi, 2008] G. Guizzardi. *On ontology, ontologies, conceptualizations, modeling languages, and (meta) models*. Databases and Information Systems IV: Selected Papers from the Seventh International Conference. (2008).

[Haase *et al.*, 2005] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. *A framework for handling inconsistency in changing ontologies*. In Proc. of 4th International Semantic Web Conference (ISWC'05), pages 353–367. Springer, (2005).

[Haase et Stojanovic, 2005] P. Haase et L. Stojanovic. *Consistent Evolution of OWL Ontologies*. In A.Gomez-Perez, J. Euzenat (Eds.), LNCS, vol.3532. The Semantic Web: Research and Applications (pp. 182-197). Berlin, Germany: Springer, (2005).

[Haase et Völker, 2008] P. Haase et J. Völker. *Ontology learning and reasoning dealing with uncertainty and inconsistencies*. In P. C. G. Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz et al. (Eds.), LNCS: Vol. 5327. Uncertainty Reasoning for the Semantic Web I (pp. 366-384). Berlin, Germany: Springer. (2008).

[Handschuh *et al.*, 2001] S. Handschuh, S. Staab et A. Maedche. *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. (2001).

[Handschuh, 2007] S. Handschuh. *Semantic Annotation of Resources in the Semantic Web*. In R. Studer, Grimm, S., Abecker, A., (Ed.), Semantic Web Services: Concepts, Technologies, and Applications, pp. 135-155. Berlin: Springer. (2007).

[Heflin et Hendler, 2000] J. Heflin et J. A. Hendler. *Dynamic Ontologies on the Web*. Proceedings of the 7th National Conference on Artificial Intelligence, Menlo Park, CA, August 2000, pp 443-449. AAAI/MIT Press, Cambridge, MA (2000).

[Heflin, 2001] J. Heflin. *Towards the Semantic Web: Knowledge Representation in a Dynamic Distributed Environment*. Thèse de doctorat. Faculty of the Graduate School of the University of Maryland. (2001).

[Jorgensen, 1989] Jorgensen, A. H.. *Using the thinking-aloud method in system development*. In G. Salvendy et M. J. Smith (Eds.), Desingning and Using Human-Computer Interfaces and Knowledge Based Systemes, pp. 743-750. Amsterdam: Elsevier. (1989).

[Klein et Fensel, 2001] M. Klein et D. Fensel. *Ontology versioning for the semantic web*. In Proceedings of the International Semantic Web Working Symposium (SWWS'01), Stanford University, California, USA, August (2001).

[Klein, 2002] M. Klein. *Ontology versioning and change detection on the web*. The 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Siguenza, Spain, (2002).

[Klein *et al.*, 2002] M. Klein, D. Fensel, A. Kiryakov, N. F. Noy et H. Stuckenschmidt. *Versioning of distributed ontologies*. Rapport technique, Université de Vrije, Amsterdam, Décembre (2002).

[Klein, 2004] M. Klein. *Change Management for Distributed Ontologies*. Thèse de doctorat, Université de Vrije, Amsterdam, (2004).

[Klein et Noy, 2003] M. Klein et N. Noy. *A component-based framework for ontology evolution*. In Proceedings of the IJCAI'03 Workshop: Ontologies and Distributed Systems, Acapulco, Mexico, (2003).

[Leenheer et Mens, 2007] P. D. Leenheer et T. Mens. *Ontology Management. Semantic Web, Semantic Web Services, and Business Applications*, chapter Ontology Evolution. State-of-the-art and Future Directions. Springer, (2007).

[Laublet *et al.*, 2009] P. Laublet, N. Aussenac-Gilles, V. Camps, P. Glize, N. Hernandez, H. Maurel, M. Mbarki, J. Mothe, B. J. V. Ralalason, A. Reymonet, B. Rothenburger, Z. Sellami, J. Thomas, A. Tissaoui : Projet ANR 07 TLOG 004 01 - DYNAMO (DYNAMic Ontology for information retrieval) : *Etat de l'art* - Livrable lot 2 Rapport de contrat, Dynamo 2.1, IRIT, (2009).

[Liang, 2006] Y. Liang. *Mini-Thesis: Enabling Active Ontology Change Management within Semantic Web-based Applications*. Thèse de doctorat, Université de Southampton, (2006).

[Liang *et al.*, 2006] Y. Liang, H. Alani, D. Dupplaw, and N. Shadbolt. *An approach to cope with ontology changes for ontology-based applications*. In: Second Advanced Knowledge Technologies DTA Symposium, Aberdeen, (2006).

[Luong, 2007] P. H. Luong. *Gestion de l'évolution d'un web sémantique d'entreprise*. Thèse de doctorat, école doctorale sciences et technologie de l'information et de la communication, école des mines de Paris, (2007).

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

[Maynard *et al.*, 2007] D. Maynard, W. Peters, M. Sabou, and M. d'Aquin. *Change management for metadata evolution*. In International Workshop on Ontology Dynamics (IWOD) ESWC 2007 Workshop, (2007).

[Maedche et Staab, 2003] A. Maedche et S. Staab. *KAON: The Karlsruhe Ontology and Semantic Web Meta Project*. Künstliche Intelligenz (3): 27-30. Special issue on Semantic Web. (2003)

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

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

[Mucchielli, 1996] A. Mucchielli. Dictionnaire des methods qualitative en sciences humaines et sociales. *Paris, Armand Colin,* (1996).

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

[Noy et Klein, 2003] N. F. Noy and M. C. A. Klein. *Tracking complex changes during ontology evolution*. In ISWC-2003 Poster Proceedings, Sanibel Island, Florida, (2003).

[Noy et Klein, 2004] N. F. Noy et M. Klein. *Ontology evolution: Not the same as schema evolution*. Knowledge and Information Systems, Vol. 6, No. 4, 428–440, July (2004).

[Noy *et al.*, 2003] N. Noy, S. Kunnatur, M. Klein, et M. Musen. *Tracking changes during ontology evolution*. Third International Conference on the Semantic Web, Hiroshima, Japan, Springer-Verlag Berlin Heidelberg, pages 259-273, (2004).

[Noy et Musen, 2002] N.F. Noy et M. A. Musen. *PROMPTDIFF: A fixed-point algorithm for comparing ontology versions*. Proceedings of the 18th National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence. Edmonton, Alberta, Canada: AAAI Press. (2002).

[Noy et Musen, 2003] N.F. Noy et M. A. Musen. *The PROMPT suite: Interactive tools for ontology merging and mapping*. International Journal of Human-Computer Studies, 59(6), 983-1024. (2003).

[Noy et Musen, 2004] N.F. Noy et M. A. Musen. *Ontology Versioning in an Ontology Management Framework*. IEEE Intelligent Systems, Vol. 19, No. 4, July –August, (2004).

[Noy *et al.*, 2006] N. Noy, A. Chugh, W. Liu, et M. Musen. *A framework for ontology evolution in collaborative environments.* In International Semantic Web Conference, pages 544–558, (2006).

[O'Malley *et al.*, 1984] O'Malley, C. E., Draper, S. W., et Riley, M. S.. *Constructive interaction: A method for studying human-computer-human interaction*. In B. Shackel (Ed.), Humancomputer interaction -INTERACT'84, pp. 269-274 : Elsevier. (1984).

[Ognyanov et Kiryakov, 2002] D. Ognyanov et A. Kiryakov. *Tracking changes in rdf(s) repositories.* The 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, pages 373–378, London, UK, Springer-Verlag, (2002).

[Paillé et Mucchielli, 2003] P. Paillé et A. Mucchielli. *L'analyse qualitative en sciences humaines et sociales*. Paris: Armand Colin.

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

[Plessers, 2006] P. Plessers. *An approach to web-based ontology evolution*, Ph.D. Thesis, University of Brussels, Belgium, (2006).

[Plessers et De Troyer, 2006] P. Plessers et O. De Troyer. *Resolving inconsistencies in evolving ontologies*. In Y. Sure, & J. Domingue (Eds.), LNCS: Vol.4011. The Semantic Web: Research and Applications, Proceedings of the 3rd European Semantic Web Conference. Pp 200-214. Berlin, Germany: Springer, (2006).

[Plessers *et al.*, 2007] P. Plessers, O. De Troyer, et S. Casteleyn. *Understanding ontology evolution: A change detection approach*. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier Publication, 5, 39-49, (2007).

[Petasis *et al.*, 2007] B. Petasis, V. Karkaletsis et G. Paliouras. D4.3: *Ontology Population and Enrichment: State of the Art*. BOEMIE Deliverable. (2007).

[Quivy et vanCampenhoud, 2007] R. Quivy et L. vanCampenhoud. *Manuel de recherche en sciences sociales*: Dunod. (2007).

[Reymonet *et al.*, 2009] A. Reymonet, J. Thomas et N. Aussenac-Gilles. *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. (2009).

[Reymonet, 2008] A. Reymonet. *Modélisation de connaissances à partir de textes pour une recherche d'information sémantique.* Thèse de doctorat. Institut de Recherche en Informtique de Toulouse, Université Paul Sabatier, Toulouse, France. (2008).

[Roddick, 1996] J. F. Roddick, *A survey of schema versioning issues for database systems*. Information and Software Technology, 37(7), 383-393, (1996).

[Rogozan, 2008] D. C Rogozan. *Gestion de l'évolution des ontologies : méthodes et outils pour un référencement sémantiques évolutif fondé sur une analyse des changements entre versions d'ontologie*. Thèse de doctorat. Télé-Université du Québec. (2008).

[Rogozan et Paquette, 2005] D. C Rogozan et G. Paquette. *Managing ontology changes on the semantic web*. In WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, pages 430–433, Washington, DC, USA, (2005).

[Sabou *et al.*, 2008] M. Sabou, M. d'Aquin et E. Motta. *Exploring the Semantic Web as Background Knowledge for Ontology Matching*. Journal on Data Semantics, (2008).

[Slocum, 2006] N. Slocum. *Focus-groupe*. In Fondation Roi Baudouin et viWTA (Eds.), Méthodes participatives. Un guide pour l'utilisateur (2006).

[Stuckenschmidt et Klein, 2003] H. Stuckenschmidt et M. C. A. Klein. *Integrity and change in modular ontologies*. In International Joint Conference on Artificial Intelligence (IJCAI), pages 900–908, (2003).

[Stojanovic *et al.*, 2002a] L. Stojanovic, N. Stojanovic, et S. Handschuh. *Evolution of the metadata in the ontology-based knowledge management systems*. In Proceedings of the 1st German Workshop on Experience Management, pages 65–77. GI, (2002).

[Stojanovic *et al.*, 2002b] N. Stojanovic, L. Stojanovic, and S. Handschuh. *Evolution in the ontology-based knowledge management systems*. In Proceedings of the 10th European Conference on Information Systems, Information Systems and the Future of the Digital Economy, Gdansk, Poland, (2002).

[Stojanovic *et al.*, 2002c] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. *Userdriven ontology evolution management*. In Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW2002, pages 285–300. Springer, (2002).

[Stojanovic *et al.*, 2003] L. Stojanovic, M. Maedche, N. Stojanovic et R. Studer. *Ontology evolution as reconfiguration-design problem Solving*. Proceedings of the 2nd international conference on Knowledge capture K-CAP'03, pp.162-171. New York, USA: ACM. ISBN: 1-58113-583-1, (2003).

[Stojanovic, 2004] L. Stojanovic. *Methods and Tools for Ontology Evolution*. Thèse de doctorat, Université de Karlsruhe (TH), Allemagne, Août (2004).

[Szulman *et al.*, 2002] S. Szulman, B. Biébow et N. Aussenac-Gilles. *Structuration de Terminologies à l'aide d'outils d'analyse de textes avec TERMINAE*. TAL, Numéro spécial sur la Structuration de Terminologie., 43(1), 103–128. (2002).

[Szulman et Biébow, 2004] S. Szulman et B. Biébow. *OWL et Terminae*. Actes de la 15° journée francophone d'Ingénierie des Connaissances, Lyon (F.), Presses Universitaires de Grenoble: 41-52, (2004).

[Zablith *et al.*, 2009] F. Zablith, M. Sabou, M. d'Aquin et E. Motta. *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. (2009).

[Zablith *et al.*, 2010] F. Zablith, Z. Sellami, M. D'Aquin, N. Aussenac-Gilles, N. Hernandez. *Vers la combinaison de deux techniques d'évolution d'ontologies à partir de ressources générales et de ressources linguistiques*. Atelier "Evolution d'ontologies" des 21e Journées francophones d'Ingénierie des Connaissances, Nîmes, Juin (2010).

[Zhang *et al.*, 2003] Z. Zhang, L. Zhang, C.X. Lin, Y. Zhao et Y. Yu. *Data Migration for Ontology Evolution.* In Poster Proceedings of the 2nd International Semantic Web Conference (ISWC-03), (2003).

Annexe A

STRATEGIES D'EVOLUTION DE LA RESSOURCE TERMINO-ONTOLOGIQUE ET D'ADAPTATION D'ANNOTATION SEMANTIQUE

Dans cette annexe nous allons présenter les fiches qui correspondent aux stratégies d'évolutions proposées, s'il y en a, pour les changements élémentaires et complexes. Nous décrivons pour chaque type de changement la syntaxe, les paramètres, la sémantique de ce changement, les conditions qui doivent être satisfaites avant et après la mise en œuvre du changement, les stratégies d'évolution proposées s'il y a et les conséquences sur la ressource termino-ontologique et les annotations sémantiques de documents.

Liste des fiches

Changements élémentaires	***Changements complexes***
Changement 1. DeleteConcept **Changement 2.** CreateSubConcept **Changement 3.** RenameConcept **Changement 4.** CreateHierarchyConceptLink **Changement 5.** DeleteHierarchyConceptLink **Changement 6.** CreateAssociatedTerm **Changement 7.** DeleteAssociatedTerm **Changement 8.** CreateTerm	**Changement 9.** MoveConcept **Changement 10.** SplitConcept **Changement 11.** MergeConcept **Changement 12.** GroupConcept **Changement 13.** MoveAssociatedTerm

Définition des concepts utilisés :

1. Top = DomainThing ;
2. Pour i de 1 à n, tous les C_i sont différent de Top ;
3. P : Ensemble des propriétés
4. Pour i de 1 à n, F_i sont les fiches annotées avec n = 3 (F1, F2 et F3);
5. t1, t2, t3 et t4 $\in T^C$ avec T^C ensemble des termes qui désignent les concepts ;
   - La fiche F1 contient t1 ;
   - La fiche F2 contient t1 et t2 ;
   - La fiche F3 contient t3 et t4 ;

I. Changements élémentaires

1.1. Changement « DeleteConcept »

Changement n° 1	Cas du changement
Identification	
Nom du changement	DeleteConcept
Syntaxe	DeleteConcept (C1)
Sémantique	Supprimer le concept C1
Pré conditions	$C1 \in C$
Post conditions	$C1 \notin C$

Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique

Cas de changement	Stratégies d'évolutions proposées	Conséquences sur la ressource termino-ontologique	Conséquences sur l'annotation sémantique
Cas 1 : Si C1 est un concept feuille, possédant un concept père qui est Top $C1 \in C, (C1, Top) \in H^C$ $\neg \exists C2 \in C : (C2, C1) \in H^C$ dénote (t1) = C1, dénote (t2) = C1 $p \in P$ C1 ∈ domaine (p) ou C1 ∈ co-domaine (p)	Il n'y a pas de stratégie proposée par le système	➢ t1 et t2 ne seront plus associés à C1 mais seront supprimés ; ➢ C1 n'appartient plus au domaine ou au co-domaine de p ; ➢ p sera supprimée si le domaine ou le co-domaine ne contient que C1 ; ➢ C1 sera supprimé ;	➢ [F1, F2] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par C1; ➢ [F1, F2] seront annotées par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;
Cas 2 : Si C1 est un concept feuille, possédant un concept père C2 différent du Top $C1, C2 \in C, (C1, C2) \in H^C$ $\neg \exists C3 \in C : (C3, C1) \in H^C$	Il n'y a pas de stratégie proposée par le système	➢ t1 et t2 ne seront plus associés à C1 mais seront supprimés ; ➢ C1 n'appartient plus au domaine ou au co-domaine de p ; ➢ p sera supprimée si le domaine ou le co-domaine ne contient que C1 ;	➢ [F1, F2] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par C1; ➢ [F1, F2] seront annotées par C2 ;

dénote (t1) = C1, dénote (t2) = C1 p∈P C1 ∈ domaine (p) ou C1 ∈co-domaine (p)		➢ C1 sera supprimé ;	
Cas 3 : Si C1 est un concept intermédiaire, possédant un concept père qui est Top C1, C2 ∈ C (C1, Top) ∈ H^C, (C2, C1) ∈ H^C dénote (t1) = C1, dénote (t2) = C1 p ∈P C1 ∈ domaine (p) ou C1 ∈co-domaine(p) **Note** : *S'il y a d'autres sub-concepts de C1, le traitement de ses sub-concepts est pareil que c1* .	***SE_RTO_1 :*** *Attacher les sous concepts au Top*	➢ C1 ne sera plus le super concept du C2. ➢ C2 sera attaché au Top ; ➢ t1 et t2 ne seront plus associés à C1 mais seront supprimés ; ➢ C1 n'appartient plus au domaine ou au co-domaine de p ; ➢ p sera supprimée si le domaine ou le co-domaine ne contient que C1 ; ➢ C1 sera supprimé ;	➢ [F1, F2] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par C1; ➢ [F1, F2] seront annotées par C2 ;
	SE_RTO_3: *Supprimer les sous C1*	➢ C1 ne sera plus le super concept du C2 ; ➢ t1 et t2 ne seront plus associés à C1 mais seront supprimés ; ➢ p sera supprimée si le domaine ou le co-domaine ne contient que C1 ; ➢ C2 sera supprimé ; ➢ C1 sera supprimé ; .	➢ [F1, F2] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par C1; ➢ [F1, F2] seront annotées par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;

Cas 4 : Si C1 est un concept intermédiaire, possédant un concept père différent du Top C, C1, C2 ∈ C, (C, Top) ∈ H^C (C1, C) ∈ H^C, (C2, C) ∈ H^C , dénote (t1) = C1, dénote (t2) = C1 p∈P C1 ∈ domaine (p) ou C1 ∈co-domaine(p)	***SE_RTO_1 :*** *Attacher les sous concepts au super concept*	C1 ne sera plus le super concept du C2 ; C2 sera attaché à C ; t1 et t2 ne seront plus associés au concept C1 mais à C2 ; C1 n'appartient plus au domaine ou au co-domaine de p ; C2 sera ajouté au domaine de p ; C1 sera supprimé ;	[F1, F2] seront invalidées ; [F1, F2] ne seront plus annotées par C1 ; [F1, F2] seront annotées par C2;
	SE_RTO_2 : *Attacher les sous concepts au Top*	C1 ne sera plus le super concept du C2 ; C2 sera attaché au Top ; t1 et t2 ne seront plus associés à C1 mais à C2 ; C1 n'appartient plus au domaine ou au co-domaine de p ; C2 sera ajouté au domaine de p ; C1 sera supprimé ;	Les fiches [F1, F2] seront invalidées ; Les fiches [F1, F2] ne seront plus annotées par C1 ; Les fiches [F1, F2] seront annotées par C2;
	SE_RTO_3 : *Supprimer les sous concepts*	C1 ne sera plus le super concept du C2 ; t1 et t2 ne seront plus associés à C1 mais seront supprimés ; C1 n'appartient plus au domaine ou au co-domaine de p ; p sera supprimée si le domaine ou le co-domaine ne contient que C1 ; C2 sera supprimé ; C1 sera supprimé ; *Note : Traiter ensuite le concept fils C2 de la même façon*	[F1, F2] seront invalidées ; [F1, F2] ne seront plus annotées par C1; [F1, F2] seront annotées par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;

1.2. Changement « CreateSubConcept »

Changement n° 2	Cas du changement		
Identification			
Nom du changement	CreateSubConcept		
Syntaxe	CreateSubConcept (NewC)		
Sémantique	Créer le nouveau concept NewC comme fils du concept OldC.		
Pré conditions	NewC$\notin$ C, OldC$\in$ C, (NewC, OldC) $\notin H^C$		
Post conditions	NewC$\in$ C, OldC$\in$ C, (NewC, OldC) $\in H^C$		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
OldC $\in$ C p $\in$ P OldC $\in$ domaine (p)	Il n'y a pas de stratégie proposée par le système	↝ NewC sera créé ; ↝ OldC sera le super-concept du NewC ;	↝ Pas de changement sur l'annotation

1.3. Changement «RenameConcept »

Changement n° 3	Cas du changement		
Identification			
Nom du changement	RenameConcept		
Syntaxe	Rename Concept (OldC, NewC)		
Sémantique	Remplacer le nom du concept modifié OldC par le nouveau nom NewC		
Pré conditions	OldC$\in$ C, Label(OldC) $\in$ Label(C), NewC $\notin$ Label(C) *Note : Label ( ) est la fonction qui renvoie le nom du concept*		
Post conditions	Label(OldC) $\notin$ Label(C), NewC $\in$ Label(C)		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**

OldC ∈ C dénote (t1) = OldC, dénote (t2) = OldC p ∈P OldC ∈ domaine (p)	Il n'y a pas de stratégie proposée par le système	↷ Le label du concept OldC sera NewC ;	↷ Pas de changement sur l'annotation

1.4. Changement «CreateHierarchyConceptLink»

Changement n° 4	**Cas du changement**		
Identification			
Nom du changement	CreateHierarchyConceptLink		
Syntaxe	Create Hierarchy Concept Link (OldC2, OldC1)		
Sémantique	Créer un lien de subsomption entre les concepts OldC1et OldC2		
Pré conditions	OldC1, OldC2 ∈ C, (OldC2, OldC1) ∉ H^C		
Post conditions	OldC1, OldC2 ∈ C, (OldC2, OldC1) ∈ H^C		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
OldC1, OldC2 ∈ C p ∈P OldC1 ∈ domaine (p)	Il n'y a pas de stratégie proposée par le système	↷ OldC1 sera le super-concept de OldC2 ;	↷ Pas de changement sur l'annotation

1.5. Changement «DeleteHierarchyConceptLink»

Changement n° 5	Cas du changement		
Identification			
Nom du changement	DeleteHierarchyConceptLink		
Syntaxe	Delete Hierarchy Concept Link (OldC2, OldC1)		
Sémantique	Supprimer le lien de subsomption entre les concepts OldC1et OldC2		
Pré conditions	OldC1, OldC2 ∈ C, (OldC2, OldC1) ∈ H^C		
Post conditions	OldC1, OldC2 ∈ C, (OldC2, OldC1) ∉ H^C		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
Cas 1 : Si OldC2 est un concept feuille et OldC1 possédant un concept père qui est Top, avec OldC1 appartenant au domaine d'une propriété p. OldC1, OldC2∈ C, (OldC1, Top) ∈ H^C (OldC2, OldC1) ∈ H^C ¬ ∃ OldC3∈ C :(OldC3, OldC2) ∈ H^C p ∈P OldC1∈domaine (p)	***SE_RTO_1 :*** *Attacher le deuxième concept au Top*	- OldC1 n'est plus le super-concept de OldC2 ; - Top sera le super-concept de OldC2 ;	- Si OldC1 appartenait au domaine de la propriété p (avant la suppression du lien entre OldC1 et OldC2) mais pas OldC2 directement, alors supprimer tous les triplets contenant à la fois la propriété p et une ressource du type OldC2, exemple : les triplets (r p v) avec (r type OldC2).

Cas 2 : Si OldC2 est un concept feuille et OldC1 possédant un concept père différent du Top, avec OldC1 appartenant au domaine d'une propriété p. OldC1, OldC2, OldC3∈ C, (OldC1, OldC3) ∈ H^C (OldC2, OldC1) ∈ H^C ¬ ∃ OldC4∈ C : (OldC4, OldC2) ∈ H^C p ∈P OldC1∈ domaine (p) ***Note*** : *Si OldC2 est un concept intermédiaire, on laisse les sous-classes de OldC2 en place. Donc il n'y a pas des effets sur la RTO et sur les annotations.*	***SE_RTO_1 :*** *Attacher le deuxième concept au super-concept du premier concept*	↷ OldC1 n'est plus le super-concept OldC2 ; ↷ OldC3 sera le super-concept de OldC2;	↷ Si OldC1 appartenait au domaine de la propriété p (avant la suppression du lien entre OldC1 et OldC2) mais pas OldC2 directement, alors supprimer tous les triplets contenant à la fois la propriété p et une ressource du type OldC2, exemple : les triplets (r p v) avec (r type OldC2).
	SE_RTO_2 : *Attacher le deuxième concept au Top*	↷ OldC1 n'est plus le super-concept du OldC2 ; ↷ Top sera le super-concept de OldC2 ;	↷ Si OldC1 appartenait au domaine de la propriété p (avant la suppression du lien entre OldC1 et OldC2) mais pas OldC2 directement, alors supprimer tous les triplets contenant à la fois la propriété p et une ressource du type OldC2, exemple : les triplets (r p v) avec (r type OldC2).

1.6. Changement «CreateAssociatedTerm»

Changement n° 6	Cas du changement
Identification	
Nom du changement	CreateAssociatedTerm
Syntaxe	Create Associated Term (NewT, OldC)
Sémantique	Créer le nouveau terme NewT et l'associer au concept sélectionné OldC (ajouter un nouveau terme à partir de la sélection d'un concept de la hiérarchie de concepts).
Pré conditions	OldC∈ C, NewT ∉ T^C,
Post conditions	OldC∈ C, NewT∈ T^C, dénote (NewT) = OldC
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique	

Cas de changement	Stratégies d'évolutions proposées	Conséquences sur la ressource termino-ontologique	Conséquences sur l'annotation sémantique
OldC∈ C F2 et F3 contiennent NewT	Il n'y a pas de stratégie proposée par le système	NewT sera créé ; NewT sera associé à OldC ;	[F2, F3] seront invalidées ; [F2, F3] seront annotées par le OldC;

1.7. Changement «DeleteAssociatedTerm»

Changement n° 7	Cas du changement
Identification	
Nom du changement	DeleteAssociatedTerm
Syntaxe	Delete Associated Term (t2, OldC)
Sémantique	Supprimer le terme t2 associé au concept OldC
Pré conditions	t2 ∈ T^C, OldC∈ C, dénote (t2) = OldC
Post conditions	t2 ∉ T^C, OldC∈ C,
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique	

Cas de changement	Stratégies d'évolutions proposées	Conséquences sur la ressource termino-ontologique	Conséquences sur l'annotation sémantique
OldC∈ C dénote (t2) = OldC	Il n'y a pas de stratégie proposée par le système	↝ t2 ne sera plus associé au concept OldC ; ↝ t2 sera supprimé ;	Traiter les triplets contenant les ressources du type t2 ou OldC: ↝ [F2] sera invalidée ; ↝ [F2] ne sera plus annotée par OldC ; ↝ [F2] sera annotée par un ou plusieurs autres concepts selon l'indication de l'utilisateur ;

1.8. Changement «CreateTerm»

Changement n° 8	**Cas du changement**		
Identification			
Nom du changement	CreateTerm		
Syntaxe	Create Term (NewT, (OldC or NewC))		
Sémantique	Créer le nouveau terme NewT et l'associer à un concept existant OldC ou un nouveau concept NewC (ajouter un nouveau terme à partir de la sélection d'un terme dans une fiche).		
Pré conditions	OldC∈ C, NewC ∉ C, NewT ∉ T^C,		
Post conditions	OldC∈ C, NewT ∈ T^C, dénote (NewT) = OldC (dans ce cas, on utilise un concept existant : OldC) dénote (NewT) = NewC (dans ce cas, on utilise un nouveau concept : NewC)		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
Cas 1 : On utilise un concept existant OldC : OldC∈ C F2 et F3 contiennent NewT	Il n'y a pas de stratégie proposée par le système	↝ NewT sera créé ; ↝ NewT sera associé au concept OldC ;	↝ [F2, F3] seront invalidées ; ↝ [F2, F3] seront annotées par le OldC;

Cas 2 : Créer un nouveau concept NewC : F2 et F3 contiennent NewT	Il n'y a pas de stratégie proposée par le système	↝ NewC sera créé ; ↝ NewT sera créé ; ↝ NewT sera associé à NewC ;	↝ [F2, F3] seront invalidées ; ↝ [F2, F3] seront annotées par le NewC;

II. Changements complexes

2.1. Changement «MoveConcept»

Changement n° 9	**Cas du changement**		
Identification			
Nom du changement	MoveConcept		
Syntaxe	Move Concept (C1, OldC1, OldC2)		
Sémantique	Déplacer le concept C1 qui est le fils du concept OldC1 à la nouvelle position comme fils du concept OldC2		
Pré conditions	C1, OldC1et OldC2 ∈ C, (C1, OldC1) ∈ H^C, (C1, OldC2) ∉ H^C		
Post conditions	C1, OldC1et OldC2 ∈ C, (C1, OldC1) ∉ H^C, (C1, OldC2) ∈ H^C		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
Cas 1 : Si C1 est un concept feuille et le concept OldC1 appartenant au domaine d'une propriété p. C1, OldC1 et OldC2 ∈ C, (C1, OldC1) ∈ H^C ¬ ∃ OldC3 ∈ C : (OldC3, C1) ∈ H^C p et p1∈P OldC1 ∈ domaine (p)	Il n'y a pas de stratégie proposée par le système	↝ OldC1 n'est plus le super-concept de C1 ; ↝ OldC2 sera le super-concept de C1 ;	↝ Si tous les concepts OldC1 et OldC2 appartiennent au domaine de p, alors on ne change pas les triplets contenant à la fois la propriété p et une ressource de type C1. ↝ Si le concept OldC2 n'appartient pas au domaine de p, alors on supprime les triplets contenant à la fois la propriété p et la ressource de

OldC2 ∈ domaine (p1)			type C1.
Cas 2 : Si C1 est un concept intermédiaire et le concept OldC1 appartenant au domaine d'une propriété p. C1, OldC1, OldC2 et OldC3 ∈ C, (OldC1, OldC3) ∈ H^C (C1, OldC1) ∈ H^C ∃ OldC4∈ C : (OldC4, C1) ∈ H^C p et p1∈P OldC1∈ domaine (p) OldC2 ∈ domaine (p1)	***SE_RTO_1 :*** *Associer le concept sélectionné et sa sous-hiérarchie au nouveau concept*	OldC1 n'est plus le super-concept de C1 et sa sous hiérarchie ; OldC2 sera le super-concept de C1 et sa sous hiérarchie ;	Si tous les concepts OldC1 et OldC2 appartiennent au domaine de p, alors on ne change pas les triplets contenant à la fois la propriété p et la ressource de type C1. Si le concept OldC2 n'appartient pas au domaine de p, alors on supprime les triplets contenant à la fois la propriété p et la ressource de type C1.
	SE_RTO_2 : *Associer la sous-hiérarchie au super-concept du concept sélectionné*	OldC1 n'est plus le super-concept de C1 ; OldC2 sera le super-concept de C1 ; OldC3 sera le super-concept de OldC4 ;	Si tous les concepts OldC1 et OldC2 appartiennent au domaine de p, alors on ne change pas les triplets contenant à la fois la propriété p et la ressource de type C1. Si le concept OldC2 n'appartient pas au domaine de p, alors on supprime les triplets contenant à la fois la propriété p et la ressource de type C1.

2.2. Changement «SplitConcept»

Changement n° 10	Cas du changement		
Identification			
Nom du changement	SplitConcept		
Syntaxe	Split Concept (OldC, NewC1, NewC2)		
Sémantique	Diviser un concept OldC en deux nouveaux concepts NewC1 et NewC2.		
Pré conditions	OldC ∈ C, NewC1 et NewC2 ∉ C, NewC1 ≠ NewC2		
Post conditions	OldC ∉ C, NewC1 et NewC2 ∈ C, NewC1 ≠ NewC2		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
Cas 1 : Si OldC est un concept feuille avec OldC appartenant au domaine d'une propriété p. OldC, OldC1 ∈ C, (OldC, OldC1)∈ H^C ¬ ∃ OldC2 ∈ C : (OldC2, OldC) ∈ H^C dénote (t1) = OldC, dénote (t2) = OldC p∈P OldC∈ domaine (p)	Il n'y a pas de stratégie proposée par le système	• NewC1 et NewC2 seront créés ; • OldC1 sera le super-concept de NewC1 et NewC2 ; • t1 et t2 ne seront plus associés à OldC ; • t1 et t2 seront associés à NewC1 (choix arbitraire) ; • OldC sera supprimé ;	• [F1, F2] seront invalidées ; • [F1, F2] ne seront plus annotées par OldC ; • [F1, F2] seront annotées par NewC1 ;

Cas 2 : Si OldC est un concept intermédiaire avec OldC appartenant au domaine d'une propriété p. OldC, OldC1 ∈ C, (OldC, OldC1)∈ H^C ∃ OldC2∈ C : (OldC2, OldC) ∈ H^C dénote (t1) = OldC, dénote (t2) = OldC p∈P OldC ∈ domaine (p)	***SE_RTO_1 :*** *Attacher la sous hiérarchie du concept sélectionné au premier nouveau concept*	↝ NewC1 et NewC2 seront créés ; ↝ OldC1 sera le super-concept de NewC1 et NewC2 ; ↝ NewC1 sera le super-concept de OldC2 ; ↝ t1 et t2 ne seront plus associés à OldC ; ↝ t1 et t2 seront associés à NewC1 ; ↝ OldC sera supprimé ;	↝ [F1, F2] seront invalidées ; ↝ [F1, F2] ne seront plus annotées par OldC; ↝ [F1, F2] seront annotées par NewC1;
	SE_RTO_2 : *Attacher la sous hiérarchie du concept sélectionné au deuxième nouveau concept*	↝ NewC1 et NewC2 seront créés ; ↝ OldC1 sera le super-concept de NewC1et NewC2 ; ↝ NewC2 sera le super-concept de OldC2 ; ↝ t1 et t2 ne seront plus associés à OldC ; ↝ t1 et t2 seront associés à NewC2 ; ↝ OldC sera supprimé ;	↝ [F1, F2] seront invalidées ; ↝ [F1, F2] ne seront plus annotées par OldC ; ↝ [F1, F2] seront annotées par NewC2 ;

2.3. Changement «MergeConcept»

Changement n° 11	**Cas du changement**		
Identification			
Nom du changement	MergeConcept		
Syntaxe	Merge Concept (OldC1, OldC2, NewC)		
Sémantique	Fusionner les concepts OldC1 et OldC2, et les remplaçant par un nouveau concept NewC. Transférer toutes les propriétés et instances de OldC1 et OldC2 vers NewC		
Pré conditions	OldC1 et OldC2 ∈ C, NewC ∉ C, (OldC1, OldC2) ∉ H^C		
Post conditions	OldC1 et OldC2 ∉ C, NewC ∈ C		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**

Cas 1 : Si OldC1 et OldC2 ont le même super-concept OldC3 avec OldC1 appartenant au domaine d'une propriété p et OldC2 appartenant au domaine d'une propriété p1. OldC1, OldC2 et OldC3∈ C, (OldC1, OldC3) ∈ H^C (OldC2, OldC3) ∈ H^C ∃ OldC4 ∈ C : (OldC4, OldC1) ∈ □H^C ∃ OldC5 ∈ C : (OldC5, OldC2) ∈ □H^C dénote (t1) = OldC1, dénote (t2) = OldC1 dénote (t3) = OldC2, dénote (t4) = OldC2 p et p1∈P OldC1∈ domaine (p) OldC2∈ domaine (p1)	Il n'y a pas de stratégie proposée par le système	➢ NewC sera créé ; ➢ OldC3 sera le super-concept de NewC ; ➢ OldC1 ne sera plus le super-concept de OldC4 ; ➢ NewC sera le super-concept de OldC4 ; ➢ OldC2 ne sera plus le super-concept de OldC5 ; ➢ [NewC] sera le super-concept de OldC5 ; ➢ t1 et t2 ne seront plus associés à OldC1 ; ➢ t1 et t2 seront associés à NewC ; ➢ t3 et t4 ne seront plus associés à OldC2 ; ➢ t3 et t4 seront associés à NewC ; ➢ OldC1 n'appartient plus au domaine de p ; ➢ OldC2 n'appartient plus au domaine de p1 ; ➢ NewC sera ajouté au domaine de p et p1 ; ➢ OldC1 et OldC2 seront supprimés ;	Traiter les triplets contenant les ressources du type OldC1 ou OldC2 : Remplacer tous les triplets (r type OldC1) ou (r type OldC2) par (r type NewC) ➢ [F1, F2, F3] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par OldC1 ; ➢ [F3] ne plus annotée par OldC2 ; ➢ [F1, F2, F3] seront annotées par NewC ;

Cas 2 : Si OldC1 et OldC2 ont deux super-concepts différents OldC3et OldC4, avec OldC1 appartenant au domaine d'une propriété p et OldC2 appartenant au domaine d'une propriété p1. OldC1, OldC2, OldC3 etOldC4 ∈ C, (OldC1, OldC3) ∈ H^C (OldC2, OldC4) ∈ H^C ∃ OldC5∈ C : (OldC5, OldC1) ∈ □H^C ∃ OldC6∈ C : (OldC6, OldC2) ∈ □H^C dénote (t1) = OldC1, dénote (t2) = OldC1 dénote (t3) = OldC2, dénote (t4) = OldC2 p et p1∈P OldC1∈ domaine (p) OldC2∈ domaine (p1)	***SE_RTO_1 :*** *Attacher le nouveau concept au super-concept du premier concept sélectionné*	➢ NewC sera créé ; ➢ OldC3 sera le super-concept de NewC ; ➢ OldC1 ne sera plus le super-concept de OldC5 ; ➢ NewC sera le super-concept de OldC5 ; ➢ OldC2 ne sera plus le super-concept de OldC6 ; ➢ NewC sera le super-concept de OldC6 ; ➢ t1 et t2 ne seront plus associés à OldC1 ; ➢ t1 et t2 seront associés à NewC ; ➢ t3 et t4 ne seront plus associés à OldC2 ; ➢ t3 et t4 seront associés à NewC ; ➢ OldC1 n'appartient plus au domaine de p ; ➢ OldC2 n'appartient plus au domaine de p1 ; ➢ NewC sera ajouté au domaine de p et p1 ; ➢ OldC1 et OldC2 seront supprimés ;	Traiter les triplets contenant les ressources du type OldC1 ou OldC2 : Remplacer tous les triplets (r type OldC1) ou (r type OldC2) par (r type NewC) ➢ [F1, F2, F3] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par OldC1 ; ➢ [F3] ne plus annotée par OldC2 ; ➢ [F1, F2, F3] seront annotées par NewC ;
	SE_RTO_2 : *Attacher le nouveau concept au super-concept du deuxième concept sélectionné*	➢ NewC sera créé ; ➢ OldC4 sera le super-concept de NewC ; ➢ OldC1 ne sera plus le super-concept de OldC5 ; ➢ NewC sera le super-concept de OldC5 ; ➢ OldC2 ne sera plus le super-concept de OldC6 ; ➢ NewC sera le super-concept de OldC6 ; ➢ t1 et t2 ne seront plus associés à OldC1 ; ➢ t1 et t2 seront associés à NewC ; ➢ t3 et t4 ne seront plus associés à OldC2 ; ➢ t3 et t4 seront associés à NewC ; ➢ OldC1 n'appartient plus au domaine de p ; ➢ OldC2 n'appartient plus au domaine p1 ;	Traiter les triplets contenant les ressources du type OldC1 ou OldC2 : Remplacer tous les triplets (r type OldC1) ou (r type OldC2) par (r type NewC) ➢ [F1, F2, F3] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par OldC1 ; ➢ [F3] ne plus annotée par OldC2 ; ➢ [F1, F2, F3] seront annotées par NewC ;

		➢ NewC sera ajouté au domaine de p et p1 ; ➢ OldC1 et OldC2 seront supprimés ;	

2.4. Changement «GroupConcept»

Changement n° 12	**Cas du changement**		
Identification			
Nom du changement	GroupConcept		
Syntaxe	Group Concept (OldC1, OldC2, NewC)		
Sémantique	Créer un super-concept commun NewC pour les concepts OldC1 et OldC2, et lui transférer les propriétés communes entre OldC1 et OldC2		
Pré conditions	OldC1 et OldC2 $\in$ C, NewC $\notin$ C, (OldC1, OldC2) $\notin$ H^C		
Post conditions	OldC1, OldC2 et NewC$\in$ C, (OldC1, NewC) $\in$ H^C, (OldC2, NewC) $\in$ H^C		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
Cas 1 : Si OldC1 et OldC2 ont le même super-concept OldC3 avec OldC1et OldC2 appartenant au domaine d'une propriété commune p. OldC1, OldC2 et OldC3$\in$ C, (OldC1, OldC3) $\in$ H^C (OldC2, OldC3) $\in$ H^C $\exists$ OldC4 $\in$ C : (OldC4, OldC1) $\in$ H^C	Il n'y a pas de stratégie proposée par le système	➢ NewC sera créé ; ➢ OldC3 sera le super-concept de NewC; ➢ OldC3 ne sera plus le super-concept de OldC1 ; ➢ OldC3 ne sera plus le super-concept de OldC2 ; ➢ NewC sera le super-concept de OldC1 et OldC2 ; ➢ OldC1 et OldC2 n'appartiennent plus au domaine de p ; ➢ NewC sera ajouté au domaine de p ;	Traiter les triplets contenant les ressources du type OldC1 ou OldC2 et la propriété p: remplacer le nom des ressources du type OldC1, OldC2 par le nom du type du concept NewC ➢ [F1, F2, F3] seront invalidées ; ➢ [F1, F2] ne seront plus annotées par OldC1 ; ➢ [F3] ne plus annotée par OldC2 ; ➢ [F1, F2, F3] seront annotées par NewC ;

∃ OldC5 ∈ C : (OldC5, OldC2) ∈ H^C dénote (t1) = OldC1, dénote (t2) = OldC1 dénote (t3) = OldC2, dénote (t4) = OldC2 p∈P OldC1∈domaine (p) OldC2∈domaine (p)			
Cas 2 : Si OldC1 et OldC2 ont deux super-concepts différents OldC3et OldC4, avec OldC1et OldC2 appartenant au domaine d'une propriété commune p. OldC1, OldC2, OldC3 etOldC4 ∈ C, (OldC1, OldC3) ∈ H^C (OldC2, OldC4) ∈ H^C ∃ OldC5 ∈ C : (OldC5, OldC1) ∈ H^C ∃ OldC6 ∈ C : (OldC6, OldC2) ∈ H^C dénote (t1) = OldC1, dénote (t2) = OldC1 dénote (t3) = OldC2, dénote	***SE_RTO_1 :*** *Attacher le nouveau concept au super-concept du premier concept sélectionné*	• NewC sera créé ; • OldC3 sera le super-concept de NewC ; • OldC3 ne sera plus le super-concept de OldC1 ; • OldC4 ne sera plus le super-concept de OldC2 ; • NewC sera le super-concept de OldC1 et OldC2 ; • OldC1 et OldC2 n'appartiennent plus au domaine de p ; • NewC sera ajouté au domaine de p ;	Traiter les triplets contenant les ressources du type OldC1 ou OldC2 et la propriété p : remplacer le nom des ressources du type OldC1, OldC2 par le nom du type du concept NewC • [F1, F2, F3] seront invalidées ; • [F1, F2] ne seront plus annotées par OldC1 ; • [F3] ne plus annotée par OldC2 ; • [F1, F2, F3] seront annotées par NewC ;
	SE_RTO_2 : *Attacher le nouveau concept au super-concept du deuxième concept sélectionné*	• NewC sera créé ; • OldC4 sera le super-concept de NewC ; • OldC3 ne sera plus le super-concept de OldC1 ; • OldC4 ne sera plus le super-concept de OldC2 ;	Traiter les triplets contenant les ressources du type OldC1 ou OldC2 et la propriété p : remplacer le nom des ressources du type OldC1, OldC2 par le nom du type du concept NewC • [F1, F2, F3] seront invalidées ;

(t4) = OldC2 p∈P OldC1 ∈ domaine (p) OldC2 ∈ domaine (p)		↝ NewC sera le super-concept de OldC1 et OldC2 ; ↝ OldC1 et OldC2 n'appartiennent plus au domaine de p ; ↝ NewC sera ajouté au domaine de p ;	↝ [F1, F2] ne seront plus annotées par OldC1 ; ↝ [F3] ne plus annotée par OldC2 ; ↝ [F1, F2, F3] seront annotées par NewC ;

2.5. Changement «MoveAssociatedTerm»

Changement n° 13	**Cas du changement**		
Identification			
Nom du changement	MoveAssociatedTerm		
Syntaxe	Move Term (t3, OldC1, OldC2)		
Sémantique	Déplacer le terme t3 qui dénote le concept OldC1 à la nouvelle position pour dénoter le nouveau concept existant OldC2		
Pré conditions	OldC1 et OldC2 ∈ C, t3 ∈ T^C, dénote (t3) = OldC1		
Post conditions	OldC1 et OldC2 ∈ C, t3 ∈ T^C, dénote (t3) = OldC2		
Stratégies d'évolutions pour les éléments dépendants de la ressource termino-ontologique			
Cas de changement	**Stratégies d'évolutions proposées**	**Conséquences sur la ressource termino-ontologique**	**Conséquences sur l'annotation sémantique**
OldC1 et OldC2 ∈ C dénote (t3) = OldC1	Il n'y a pas de stratégie proposée par le système	↝ t3 ne sera plus associé à OldC1 ; ↝ t3 sera associé à OldC2 ;	↝ [F3] sera invalidée ; ↝ [F3] ne sera plus annotée par OldC1 ; ↝ [F3] sera annotée par OldC2 ;

Annexe B

QUESTIONNAIRE D'ÉVALUATION DE LA RESSOURCE TERMINO-ONTOLOGIQUE

Vous

1- Identité :

2- Fonction : 3- Spécialité :

I. Intérêt de l'outil (A remplir à la fin)

1. Avez-vous l'habitude de ce type d'outil :
- Oui ☐
- Oui un peu ☐
- Non jamais ☐

Si oui, lequel ?

……
……
……

2. L'assistance fournie par ce logiciel vous a-t-elle paru :
- Nécessaire ☐
- Intéressante, mais pas nécessaire ☐
- Pas intéressante ☐

3. Jugez-vous nécessaire la présentation des effets possibles des changements de la RTO avant leur application ?
- Nécessaire ☐
- Intéressante, mais pas nécessaire ☐
- Pas intéressante ☐

4. Jugez-vous utiles les différentes stratégies d'évolutions ?
- Oui ☐
- Oui un peu ☐
- Non ☐

5. Jugez-vous nécessaire de connaître les fiches impactées ?
- Oui ☐
- Oui un peu ☐
- Non ☐

Quel autre type d'information auriez-vous aimé trouver pour vous aider dans la gestion l'évolution d'une ontologie ? Sous quelle forme et à quel moment ?

……
……
……

II. Utilisation de l'outil

6. Qu'avez-vous pensé de l'utilisation générale de cet outil ?
- Simple ☐
- Compliquée ☐
- Très compliquée ☐

7. Vous est-il arrivé de chercher des informations car elles n'étaient pas à l'endroit attendu ?
- Oui ☐
- Oui un peu ☐
- Non ☐

Lesquelles ? dans quel contexte ?

……
……
……

8. Vous est-il arrivé de chercher des fonctionnalités car elles n'étaient pas à l'endroit attendues ?
- Oui ☐
- Oui un peu ☐
- Non ☐

Lesquelles ? dans quel contexte ?

……
……
……

9. Pensez-vous que l'outil manque des fonctionnalités ?
- Oui ☐
- Oui un peu ☐
- Non ☐

Si Oui, quelles sont à votre avis les fonctionnalités manquantes ?

……
……
……

III. Evaluation des changements proposés :

Remarque : Merci de bien vouloir mesurer le temps d'activité pour effectuer chaque changement.

Changement numéro 1 : ***DeleteConcept***

Concept sélectionné :……..

Contexte d'évolution : ……..……..……
……..……..……..……
……..……..……..……..

Mesure de la durée d'activité :

Durée d'activité	

10. Les informations concernant les fiches impactées avant de faire ce changement, vous ont-elles paru :

Nécessaires ☐
Intéressantes, mais pas nécessaires ☐
Pas intéressantes ☐

Si Non, quelles sont à votre avis les informations inutiles ou manquantes?

.....
.....
.....

11. Les stratégies d'évolution pour ce type de changement, vous ont-elles paru :

Nécessaires ☐
Intéressantes, mais pas nécessaires ☐
Pas intéressantes ☐

12. La stratégie d'évolution choisie pour ce type de changement, vous a-t-elle paru :

Pertinente ☐
Non pertinente ☐

Si non pertinente, pourquoi ?

.....
.....
.....

13. La présentation des conséquences de l'opération Delete Concept sur la RTO vous a-t-elle paru:

Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

.....
.....
.....

14. La présentation des conséquences directes (pour le concept lui-même) de l'opération DeleteConcept sur les fiches vous a-t-elle paru:

Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

.....
.....
.....

15. La présentation des conséquences indirectes (pour les concepts fils) de l'opération DeleteConcept sur les fiches vous a-t-elle paru:

Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

……
……
……

16. Avez-vous eu besoin d'utiliser l'ajustement manuel ?
Oui ☐
Non ☐

Si Oui, comment ?

Son utilisation a-t-elle été

Satisfaisante ☐
Insatisfaisante ☐

17. Que pensez-vous de l'interface traitant ce type de changement :
simple ☐
compliquée ☐
très compliquée ☐

Changement numéro 2 : *MergeConcept*

Concept sélectionné 1:…….

Concept sélectionné 2:…….

Nouveau concept :……

Contexte d'évolution : ……..……..……
……..……..……..……
……..……..……..……..

Mesure de la durée d'activité :

Durée d'activité	

18. Les informations concernant les fiches impactées avant de faire ce changement, vous ont-elles paru :
Nécessaires ☐
Intéressantes, mais pas nécessaires ☐
Pas intéressantes ☐

Si Non, quelles sont à votre avis les informations inutiles ou manquantes?

.....
.....
.....

19. Les stratégies d'évolution pour ce type de changement, vous ont-elles paru :
Nécessaires ☐
Intéressantes, mais pas nécessaires ☐
Pas intéressantes ☐

20. La stratégie d'évolution choisie pour ce type de changement, vous a-t-elle paru :
Pertinente ☐
Non pertinente ☐

Si non pertinente, pourquoi ?

.....
.....
.....

21. La présentation des conséquences de l'opération MergeConcept sur la RTO vous a-t-elle paru:
Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

.....
.....
.....

22. La présentation des conséquences directes (pour les concepts sélectionnés) de l'opération MergeConcept sur les fiches vous a-t-elle paru:
Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

.....
.....
.....

23. La présentation des conséquences indirectes (pour les concepts fils) de l'opération MergeConcept sur les fiches vous a-t-elle paru:
Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

.....
.....
.....

24. Avez-vous eu besoin d'utiliser l'ajustement manuel ?
Oui ☐
Non ☐

Si Oui, comment ?

Son utilisation a-t-elle été

Satisfaisante ☐
Insatisfaisante ☐

25. Que pensez-vous de l'interface traitant ce type de changement :
simple ☐
compliquée ☐
très compliquée ☐

Changement numéro 3 : *DeleteAssociatedTerm*

Terme sélectionné :…….

Contexte d'évolution : ……..……..……
……..……..……..……..……
……..……..……..……..

Mesure de la durée d'activité :

Durée d'activité	

26. Les informations concernant les fiches impactées avant de faire ce changement, vous ont-elles paru :
Nécessaires ☐
Intéressantes, mais pas nécessaires ☐
Pas intéressantes ☐

Si Non, quelles sont à votre avis les informations inutiles ou manquantes?
……
……
……

27. La présentation des conséquences de l'opération DeleteAssociatedTerm sur la RTO vous a-t-elle paru:
Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?
……
……
……

28. La présentation des conséquences directes de l'opération DeleteAssociatedTerm sur les fiches vous a-t-elle paru:
Utile ☐

Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

……
……
……

29. La présentation des conséquences indirectes de l'opération DeleteAssociatedTerm sur les fiches vous a-t-elle paru:

Utile ☐
Inutile ☐

Si Inutile, pourquoi ? quelles autres informations vous auraient paru plus utiles ?

……
……
……

30. Que pensez-vous de l'interface traitant ce type de changement :

simple ☐
compliquée ☐
très compliquée ☐

Question sur les trois types de changements

31. Les informations affichées concernant les éléments de la RTO avant de faire les trois types de changements, vous ont-elles paru :

Nécessaires ☐
Intéressantes, mais pas nécessaires ☐
Pas intéressantes ☐

Si Non, quelles sont à votre avis les informations inutiles ou manquantes?

……
……
……

IV. Comparaison d'EvOnto avec d'autres systèmes :

Remarque : Merci de bien vouloir mesurer le temps d'activité pour chaque changement.

Changement numéro 1 : *DeleteConcept*

Mesure de la durée d'activité pour chaque système :

Logiciels	Durée d'activité
EVOnto	
TextViz	
Protégé 4	

1. Pensez-vous que les informations d'assistance pour guider l'utilisateur à appliquer ce changement sont utiles:

	EvOnto	TextViz	Protégé 4
Oui	☐	☐	☐
Oui un peu	☐	☐	☐
Non	☐	☐	☐

2. Le traitement des conséquences de l'opération DeleteConcept sur la RTO vous a-t-elle paru complète:

	EvOnto	TextViz	Protégé 4
Oui	☐	☐	☐
Oui un peu	☐	☐	☐
Non	☐	☐	☐

Si Incomplète, quelles sont à votre avis les conséquences manquantes?

TextViz : .....

Protégé 4 : .....

Changement numéro 2 : *MergeConcept*

Mesure de la durée d'activité pour chaque système :

Logiciels	Durée d'activité
EVOnto	
TextViz	
Protégé 4	

3. Pensez-vous que les informations d'assistance pour guider l'utilisateur à appliquer ce changement sont utiles:

	EvOnto	TextViz	Protégé 4
Oui	☐	☐	☐
Oui un peu	☐	☐	☐
Non	☐	☐	☐

4. Le traitement des conséquences de l'opération MergeConcept sur la RTO vous a-t-elle paru complète:

	EvOnto	TextViz	Protégé 4
Oui	☐	☐	☐
Oui un peu	☐	☐	☐
Non	☐	☐	☐

Si Incomplète, quelles sont à votre avis les conséquences manquantes?

TextViz : .....

Protégé 4 : .....

Changement numéro 3 : *DeleteAssociatedTerm*

Mesure de la durée d'activité pour chaque système :

Logiciels	Durée d'activité
EVOnto	
TextViz	
Protégé 4	

5. Pensez-vous que les informations d'assistance pour guider l'utilisateur à appliquer ce changement sont utiles:

	EvOnto	TextViz	Protégé 4
Oui	☐	☐	☐
Oui un peu	☐	☐	☐
Non	☐	☐	☐

6. Le traitement des conséquences de l'opération DeleteAssociatedTerm sur la RTO vous a-t-elle paru complète:

	EvOnto	TextViz	Protégé 4
Oui	☐	☐	☐
Oui un peu	☐	☐	☐
Non	☐	☐	☐

Si Incomplète, quelles sont à votre avis les conséquences manquantes?

TextViz : .....

Protégé 4 : .....

Avis général

Suggestions