Patrons de gestion de changements OWL

Rim Djedidi[1] et Marie-Aude Aufaure[2]

[1] Département Informatique, Supélec Campus de Gif
Plateau du Moulon – 3, rue Joliot Curie – 91192 Gif sur Yvette Cedex, France
rim.djedidi@supelec.fr
[2] Laboratoire MAS, Chaire SAP Business Object – Centrale Paris
Grande Voie des Vignes, F-92295 Châtenay-Malabry Cedex, France
marie-aude.aufaure@ecp.fr

Résumé : Tout au long de leur cycle de vie, les ontologies évoluent pour répondre à différents besoins de changements. Nous nous intéressons particulièrement aux problèmes inhérents à la gestion des changements d'une ontologie dans un contexte local et nous présentons une approche d'évolution d'ontologies à base de patrons. Les patrons modélisés correspondent aux dimensions *changement*, *incohérence* et *alternative de résolution*. Sur la base de ces patrons et des liens conceptuels entre eux, nous proposons un processus automatisé permettant de guider et contrôler l'application des changements tout en assurant la cohérence de l'ontologie évoluée.
La gestion des changements étant fortement liée au modèle dans lequel est représentée l'ontologie, nous nous focalisons sur le langage OWL et nous tenons compte de l'impact des changements sur la cohérence logique de l'ontologie telle que spécifiée dans la couche OWL DL.

Mots-clés : Evolution d'ontologies, Gestion de changements, Patrons, Cohérence, OWL DL.

1 Introduction

Tout au long de leur cycle de vie, les ontologies évoluent pour répondre à différents besoins de changements : la dynamicité de l'environnement où elles sont appliquées, l'évolution du domaine modélisé, la modification des besoins utilisateurs, les corrections et restructurations apportées à leur conceptualisation, et leur réutilisation pour d'autres applications.

L'évolution d'ontologies est une problématique complexe (Stojanovic, 2004) (Klein, 2004). Outre l'identification même des besoins de changements à partir de différentes sources (domaine, environnement d'usage, conceptualisation interne, etc.), la gestion de l'application d'un changement – de sa formulation jusqu'à son application et sa validation finale – nécessite de spécifier le changement requis, d'analyser ses effets sur la cohérence de l'ontologie et les résoudre, de l'implémenter et de valider son application finale. Dans un contexte collaboratif ou distribué, il est aussi nécessaire de propager le changement appliqué localement à l'ontologie, aux artefacts dépendants (les applications et/ou les ontologies dépendantes) et de valider les changements globalement. De plus, la traçabilité des changements doit être gardée

pour pouvoir les justifier, les expliquer, voire les annuler et gérer les différentes versions d'une ontologie.

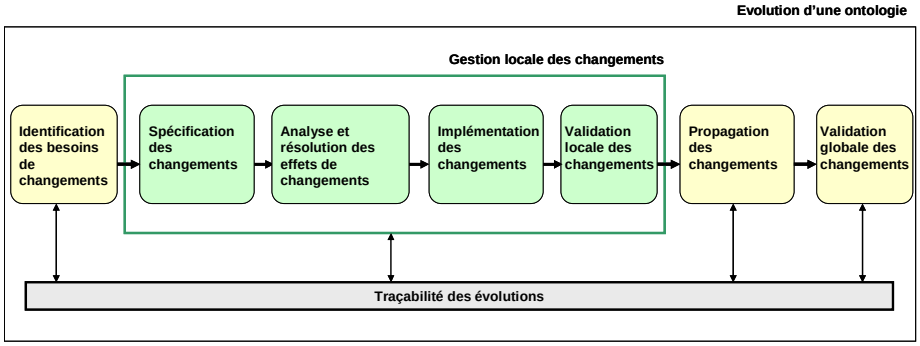

Fig. 1 – Processus global d'évolution d'ontologies

Dans nos travaux[1], nous nous intéressons aux problèmes inhérents à la gestion des changements d'une ontologie dans un contexte local. Conduire l'application des changements tout en maintenant la cohérence de l'ontologie est une tâche cruciale et couteuse en termes de temps et de complexité. Un processus automatisé est donc essentiel. Mais comment conduire l'application d'un changement tout en assurant la cohérence de l'ontologie évoluée ? Comment analyser les effets d'un changement et les résoudre ? Et si plusieurs solutions sont possibles, laquelle choisir et selon quels critères ?

Pour répondre à toutes ces questions, nous avons défini une méthodologie de gestion de changements ***Onto-Evo^{a}l*** (*Ontology Evolution-Evaluation*) qui s'appuie sur une modélisation à l'aide de patrons. Ces patrons spécifient des classes de *changements*, des classes d'*incohérences* et des classes d'*alternatives de résolution*. Sur la base de ces patrons et des liens entre eux, nous proposons un processus automatisé permettant de conduire l'application des changements tout en maintenant la cohérence de l'ontologie évoluée. La méthodologie intègre aussi une activité d'évaluation basée sur un modèle de qualité d'ontologies. Ce modèle est employé pour guider la résolution des incohérences en évaluant l'impact des alternatives de résolution proposées sur la qualité de l'ontologie et ainsi choisir celles qui préservent la qualité de l'ontologie évoluée.

La gestion des changements étant fortement liée au modèle dans lequel est représentée l'ontologie, nous nous focalisons sur le langage OWL et nous tenons compte de l'impact des changements sur la cohérence logique de l'ontologie telle que spécifiée dans la couche OWL DL.

L'article est organisé comme suit : dans la section 2, nous détaillons le processus de gestion de changements. Les patrons de gestion de changements sont présentés et illustrés dans la section 3. Avant de synthétiser les différents points de l'approche définie et présenter nos travaux en cours, une discussion et une comparaison avec les travaux existants sont présentées à la section 4.

[1] Ces travaux sont financés par l'Agence Nationale de Recherche dans le cadre du Projet-RNTL DAFOE.

2 Processus de gestion des changements

L'objectif d'un processus de gestion de changements est de conduire de manière automatisée l'application d'un changement tout en assurant la cohérence de l'ontologie. Ceci nécessite de formuler explicitement le changement requis, de détecter les incohérences dues à son application, de proposer des solutions pour les résoudre, de guider l'ingénieur dans le choix des résolutions et de l'assister dans la validation finale (Stojanovic, 2004).

Pour réaliser cet objectif, nous avons défini des patrons de gestion de changements CMP (*Change Management Patterns*). Ces patrons permettent de ressortir et de classer des types de changements en se basant sur le modèle OWL, des types d'incohérences logiques en se référant aux contraintes de OWL DL et des types d'alternatives de résolution d'incohérences.

Le processus de gestion de changements est conduit à travers quatre phases (figure 2): spécification du changement, analyse du changement, résolution du changement et application du changement.

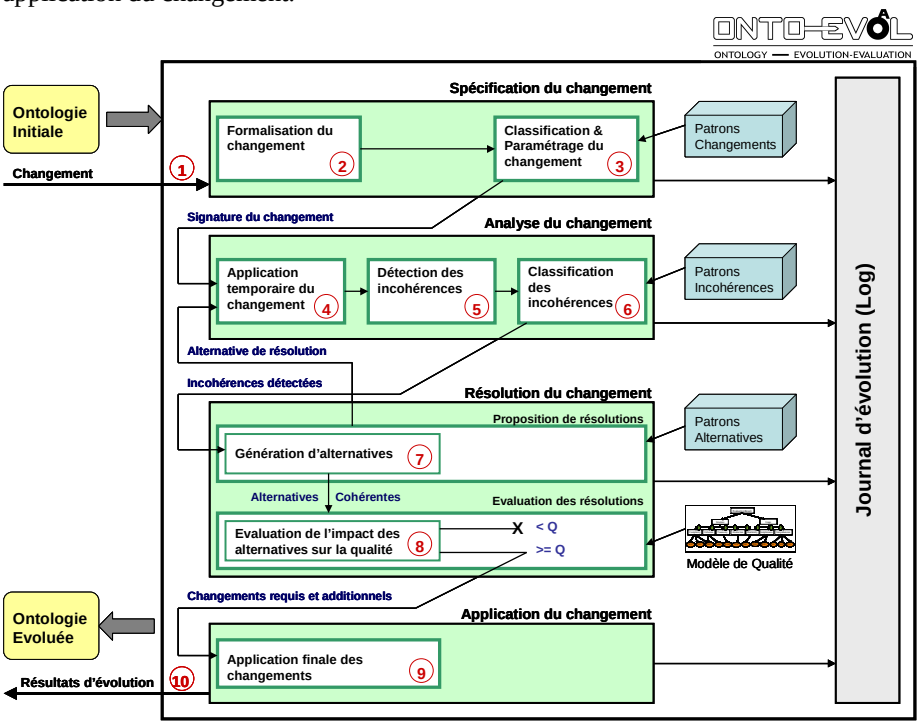

Fig. 2 – Architecture de l'approche de gestion de changements à base de patrons

2.1 Spécification du changement

La phase de spécification est lancée suite à la demande d'un changement à appliquer sur une ontologie initiale supposée cohérente (figure 2 **(1)**). L'utilisateur demande un changement élémentaire ou composé sans avoir à définir et ordonner les opérations de changements intermédiaires nécessaires à son application. Réaliser ces tâches manuellement est très couteux en termes de temps et de risque d'erreurs.

La phase de spécification a un rôle plus large qu'une simple représentation du changement requis dans le langage de l'ontologie. Elle vise à expliciter le changement de manière formelle et compréhensible pour préparer les phases d'analyse et de résolution. Le changement est tout d'abord explicité et formalisé dans le modèle OWL (figure 2 **(2)**). Ensuite, il est classé selon les types prédéfinis par les patrons de changements et enrichi par un ensemble d'arguments permettant de préparer les phases suivantes (figure 2 **(3)**). Le patron correspondant est instancié aux spécificités réelles du changement requis (valeurs, références des entités concernées, etc.). L'ensemble des spécifications dérivées constitue la signature du changement.

2.2 Analyse du changement

Le changement formellement spécifié est appliqué à une version temporaire de l'ontologie pour analyser ses impacts (figure 2 **(4)**). L'étape suivante permet de détecter les incohérences causées (figure 2 **(5)**). Les incohérences détectées sont alors classées selon les types prédéfinis par les patrons d'incohérences (figure 2 **(6)**). L'instanciation des patrons d'incohérences permet de préparer la phase de résolution notamment à travers le lien avec les patrons d'alternatives (figure 3).

Seule la cohérence logique est considérée. La cohérence structurelle – se référant aux contraintes du langage et l'utilisation de ses constructeurs – est vérifiée automatiquement au début du processus. Pour la vérification de la cohérence logique, nous employons le raisonneur Pellet[2] en interfaçage avec le système de gestion de changements que nous avons développé. Pellet supporte aussi bien le niveau terminologique *TBox* (classes et propriétés) que le niveau assertionnel *ABox* (instances) de OWL DL (Sirin et al., 2007). Cependant, certaines incohérences logiques, notamment celles se référant aux propriétés, ne sont pas détectées par Pellet et sont prises en charge par notre système. Par ailleurs, Pellet ne permet pas de préciser les axiomes qui ont causé les incohérences ni comment résoudre les incohérences détectées. L'identification des axiomes causant les incohérences est basée sur les travaux de (Plessers & De Troyer, 2006).

2.3 Résolution du changement

La résolution du changement comprend deux principales activités : proposition de résolutions et évaluation des résolutions.

[2] http://clarkparsia.com/pellet/

2.3.1 Proposition de résolution

A partir des instances d'incohérences détectées, les patrons d'alternatives sont instanciés pour générer les alternatives potentielles de résolution (figure 2 **(7)**). Chaque alternative représente des opérations de changements additionnels à appliquer pour résoudre une incohérence. Cependant, elle ne doit pas causer d'autres incohérences. C'est pourquoi, toutes les alternatives sont vérifiées et résolues selon un mécanisme récursif et seules les alternatives cohérentes sont retenues.

2.3.2 Evaluation des résolutions

Plusieurs solutions peuvent parfois être proposées pour une incohérence. Plutôt que de présenter les différentes alternatives de résolution à l'ingénieur d'ontologies, nous proposons de guider le choix de l'alternative à appliquer en évaluant l'impact de chacune des alternatives sur la qualité de l'ontologie (figure 2 **(8)**). L'évaluation se base sur un modèle de qualité considérant les aspects structure et usage de l'ontologie à travers un ensemble de critères et de métriques. La description détaillée du modèle de qualité ne faisant pas l'objet de ce papier, le lecteur peut se référer à la référence (Djedidi & Aufaure, 2008). L'alternative qui préserve la qualité de l'ontologie, peut être automatiquement choisie. L'évaluation de l'impact sur la qualité participe à l'automatisation du processus en guidant le choix des résolutions à travers des alternatives annotées et évaluées.

2.4 Application du changement

Cette phase correspond à l'application finale du changement. Elle est optimisée par l'emploi de techniques d'évaluation de qualité permettant de guider la résolution des incohérences et de minimiser la dépendance à l'utilisateur. Ainsi, si les résolutions préservent la qualité, le changement requis et ses changements dérivés seront directement validés et appliqués (figure 2 **(9)**) et l'ontologie évoluée. Si par contre, les alternatives ont un impact négatif sur la qualité, les résultats des différentes phases du processus seront présentés à l'ingénieur d'ontologies en complément à son expertise pour qu'il décide du changement (figure 2 **(10)**). Cette phase permet de garder conjointement à l'automatisme du processus, une certaine flexibilité permettant à l'utilisateur de contrôler et de décider du changement et de sa validation finale.

L'ensemble des traitements d'analyse et de maintenance de la cohérence est appliqué à une version temporaire de l'ontologie qui peut être abandonnée si les changements sont finalement annulés, l'ontologie initiale sera alors préservée. Dans le cas contraire, une version modifiée de l'ontologie est définie.

Notons que tout au long du processus, les différents résultats sont sauvegardés dans le journal d'évolution. Le journal d'évolution est une structure permettent de conserver l'historique des évolutions de l'ontologie et les détails de traitement de ces évolutions sous forme de séquences chronologiques d'informations. Il facilite le suivi de l'évolution, le retour arrière, la justification des changements, la gestion des versions et dans une perspective future l'apprentissage de nouveaux patrons de changements, d'incohérences et d'alternatives.

3 Patrons de gestion de changements

Les *Design Patterns* (patrons de conception) représentant des directives (guidelines) partagées qui aident à la résolution de problèmes de conception, ont aussi été adoptés en ingénierie logiciel et ontologique (Gangemi, 2005). Tout comme l'idée de modéliser des *Design Patterns* pour la construction d'ontologies OWL (Gangemi et al., 2007), les patrons de gestion de changements CMP –*Change Management Patterns*– sont proposés comme une solution permettant de ressortir des invariances observées répétitivement lors d'un processus d'évolution d'ontologies. De la modélisation des CMP en ressort trois catégories de patrons : des patrons de *changements*, des patrons d'*incohérences* et des patrons d'*alternatives de résolution*. L'intérêt de cette modélisation est d'offrir différents niveaux d'abstraction, d'établir des liens entre ces trois catégories de patrons déterminant les incohérences qui peuvent être potentiellement causées par un type de changements et les alternatives de résolution possibles pour un type d'incohérences et par là, d'assurer un processus automatisé de gestion de changements.

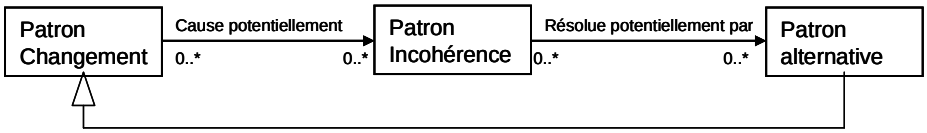

Fig. 3 – Modèle conceptuel des patrons de gestion de changements

3.1 Patrons de changements

Les patrons de changements sont définis sur la base du modèle OWL DL. L'idée est de catégoriser les changements, définir formellement leur signification, leur portée et leurs implications potentielles. Les changements OWL sont classés selon deux grandes catégories : des changements basiques et des changements complexes (Klein, 2004). Les patrons de changements couvrent tous les changements OWL basiques et un premier noyau de changements complexes.

Les composants d'un patron de changement sont:

- Le type des entités concernées correspondant aux primitives conceptuelles de OWL (classe, propriété, instance) ;
- Les arguments regroupant l'ensemble des paramètres nécessaires à l'application du changement. Le contenu varie selon le type du changement et le type des entités concernées et peut comprendre entre autres :
  - La référence des entités réelles sur lesquelles porte le changement (leurs identifiants dans l'ontologie),
  - La référence (les identifiants) des entités intermédiaires impliquées dans le changement (par exemple les superclasses d'une classe à ajouter),

- Les valeurs du changement (par exemple les valeurs d'une restriction de cardinalité).

• Les contraintes à satisfaire pour que le changement puisse être appliqué sans altérer la cohérence logique de l'ontologie. Ce sont des pré-conditions qui peuvent être vérifiées avant l'application du changement. Elles préparent la prévision des incohérences pouvant être causées par un changement. Par exemple, à la demande d'un ajout d'une relation d'équivalence entre deux classes, la pré-condition serait que ces deux classes ne soient pas disjointes dans leurs hiérarchies respectives.

3.1.1 Patrons de changements basiques

Les patrons de changements basiques correspondent à des changements indivisibles qui ne modifient qu'une seule caractéristique du modèle de connaissances de l'ontologie (tel que la suppression d'une relation « *is-a* »).

Exemple1. Soit le patron d'un changement basique correspondant à l'ajout d'une relation de sous-classe. Ce patron est décrit comme suit (table 1) :

Table 1. Exemple de patron de changement basique

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, prenons un exemple simple d'une ontologie OWL *O* définie par les axiomes suivants :

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

Et soit le changement *Ch1* définissant la classe *PlanteCarnivore* comme sous-classe de la classe *Animal*. L'instanciation du patron (table 1) par *Ch1* permet de spécifier la signature suivante (table 2) :

Table 2. Exemple d'instanciation d'un patron de changement basique

Type	**Entités concernées**	**Arguments**	**Contraintes**	**Axiomes OWL DL**
P_Chgt_Bas_ Ajouter_ Sous_Classe	Classe, Classe	Animal, PlanteCarnivore	¬(PlanteCarnivore disjointWith Animal)	SubClassOf (PlanteCarnivore, Animal)

3.1.2 Patrons de changements complexes

Les patrons de changements complexes correspondent à des changements composites et riches renfermant des séquences logiques de changements basiques et incorporant des informations sur leur implication (tel qu'élargir le co-domaine d'une propriété à la superclasse de la classe qui le spécifiait).

Les patrons de changements complexes regroupent plus de détails puisqu'ils décrivent un ensemble de changements intermédiaires. Si un changement requis consiste à ajouter une classe en spécifiant sa place dans la hiérarchie à travers la liste de ses superclasses et/ ou en définissant une collection d'unions ou d'intersections de classes agrémentée d'un ensemble de restrictions, alors c'est un changement complexe correspondant à un patron de changement complexe lui-même décrit par une séquence de patrons de changements basiques (figure 4) et/ou complexes.

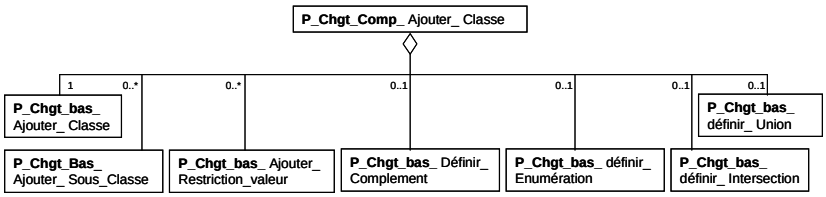

Fig. 4 –Modèle conceptuel d'un patron de changement complexe

3.2 Patrons d'incohérences

Durant la phase d'analyse, le système vérifie les contraintes spécifiées dans la signature du changement, les liens entre le patron de changement instancié et les patrons des incohérences qu'il peut potentiellement causer (figure 3) tout en tenant compte de toutes les incohérences logiques concrètement détectées par le raisonneur Pellet. Toutes les incohérences constatées sont alors classées selon les patrons d'incohérences pour être résolues. Le module de classification se base sur l'interprétation des résultats de Pellet ainsi que les compositions d'axiomes et les dépendances entre axiomes pour identifier les axiomes causant les incohérences.

Les composants des patrons d'incohérences peuvent varier d'un type d'incohérences à un autre mais les principaux composants d'un patron d'incohérence sont :

- Les identifiants de toutes les entités impliquées directement ou indirectement dans l'incohérence logique. Ces informations permettent d'expliquer l'incohérence et facilitent sa localisation ;
- Les identifiants des entités concernées par l'incohérence. Ces informations facilitent la détermination des axiomes causant l'incohérence détectée et préparent la proposition de résolution ;
- Les axiomes concernés par l'incohérence.

Exemple2. Reprenons l'exemple de changement basique *Ch1* (exemple 1), l'instanciation du patron d'incohérence de disjonction correspondant est décrite comme suit (table 3) :

Table 3. Exemple d'instanciation d'un patron d'incohérence de disjonction

Type	**Entités Impliquées**	**Entités Concernées**	**Axiomes OWL DL concernés**
P_Incons _Disj	Animal, Plante, PlanteCarnivore,	Animal, Plante	`Plant ⊑ ¬Animal, PlanteCarnivore ⊑ Plant`

3.3 Patrons d'alternatives

Dans la phase de résolution du changement, la proposition de résolutions se base sur les liens entre le patron d'incohérence instancié et les patrons d'alternatives qui peuvent potentiellement le résoudre (figure 3). Un patron d'alternative représente un changement additionnel à appliquer pour résoudre une incohérence logique. Il est décrit comme un changement (basique ou complexe) et hérite des propriétés d'un patron de changement (figure 3), ce qui implique qu'il peut lui-même causer des incohérences (figure 2). D'autres informations peuvent aussi décrire les patrons d'alternatives telles que les pré-conditions à satisfaire pour choisir le patron comme solution de résolution (table4).

Exemple3. Reprenant l'exemple de changement basique *Ch1* (exemple1), deux alternatives de résolution sont possibles pour résoudre l'incohérence de disjonction causée (exemple2). Leurs patrons et instanciations respectifs sont décrits comme suit :

Table 4. Exemple de patron d'alternative résolvant une disjonction (*al1*)

P_Alt_Disj_ Chgt_Bas_ Ajout_ Sous_Classe **(*al1*)**				
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)`

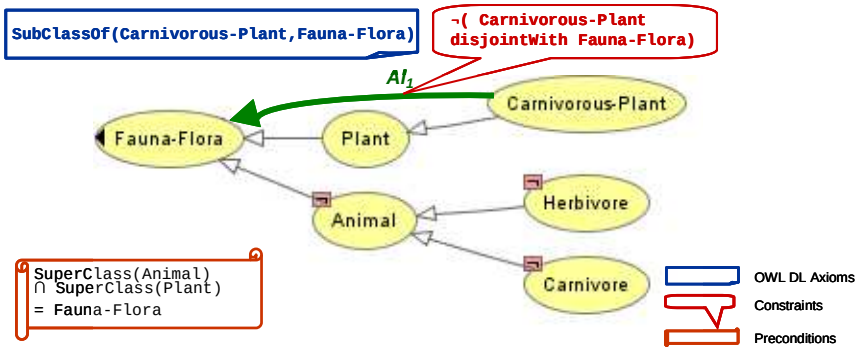

Fig. 5 –Instanciation du patron de l'alternative *al1*

Table 5. Exemple de patron d'alternative résolvant une disjonction (*al2*)

P_Alt_Disj_Chgt_Comp_ Rattacher_ Classe_Hybride **(*al2*) (version synthétisée)**		
Entités Concernées	**Arguments**	**Composants Intermédiaires : Axiomes OWL DL**

Classe,
Classe
	Id_HybridClass,
	Id_sub_class,
	Id1_cls_disj, Id2_cls_disj
		`Class(Id_HybridClass,`
		`UnionOf(Id1_cls_disj,Id2cls_disj))`
		`SubClassOf(Id_HybridClass, Id_sub_class)`

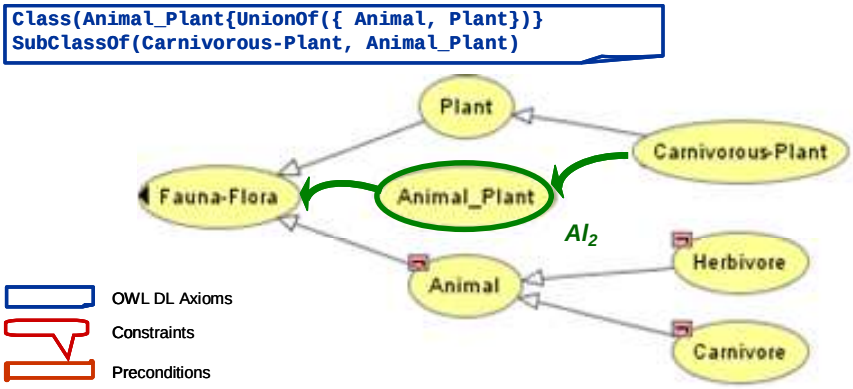

Fig. 6 –Instanciation du patron de l'alternative *al2*

4 Discussion et travaux existants

La modélisation par patrons a été adoptée ces dernières années, dans la conception d'ontologies pour le Web afin de proposer des guides de bonnes pratiques et de fournir des catalogues de composants ontologiques réutilisables[3]. La notion de patrons de conception d'ontologies a été introduite par (Gangemi et al., 2004), (Rector & Roger, 2004), (Svatek, 2004). D'un point de vue théorique, les CMP (*Change Management Patterns*) se rapprochent des ODP (*Ontology Design Patterns*), plus particulièrement des patrons logiques (*Logical Ontology Patterns LOP*). Tous les deux sont appliqués dans le cycle d'ingénierie ontologique. Tout comme les CMP, les LOP sont indépendants du domaine modélisé par l'ontologie et peuvent être appliqués plus d'une fois dans une ontologie pour résoudre un problème d'évolution (CMP) ou de conception (LOP). L'implémentation des CMP et les expressions formelles des LOP sont toutes les deux issues du modèle de OWL DL. Mais d'un point de vue pragmatique, ils sont assez différents : les CMP proposent des solutions réutilisables pour guider la gestion des changements dans un processus d'évolution locale d'une ontologie. Les ODP sont conçus comme des guides de bonnes pratiques réutilisables dans un contexte de conception collaborative d'ontologies en réseau et les LOP représentent des compositions de constructions logiques pour résoudre des problèmes d'expressivité (Presutti et al., 2008).

[3] Exemple : http://odps.sourceforge.net/

Pour les travaux existants en évolution d'ontologies, nous nous positionnons particulièrement par rapport à ceux qui traitent les ontologies OWL, la gestion des changements étant dépendante du modèle de représentation de l'ontologie. Dans (Haase & Stojanovic, 2005), les auteurs ont introduit des stratégies de résolution basées sur les contraintes de OWL Lite. La résolution des incohérences logiques se limite à la détermination des axiomes causant ces incohérences et devant être supprimés et à leur présentation à l'utilisateur pour qu'il puisse décider. Dans notre approche, nous tendons à minimiser les solutions de suppression d'axiomes en proposant des patrons d'alternatives résolvant les incohérences en fusionnant, divisant, généralisant ou spécialisant les concepts pour préserver les connaissances existantes et nous assistons l'ingénieur dans le choix des alternatives à appliquer en faisant intervenir l'évaluation. Dans (Plessers et al., 2006), les auteurs définissent une approche de détection de changements, les changements sont exprimés sous forme de requêtes temporelles appliquées sur un journal de versions. Dans (Plessers et De Troyer, 2006), les auteurs définissent un algorithme localisant les axiomes causant les incohérences dans une ontologie OWL DL que nous adoptons dans notre phase d'analyse de changements. Une autre approche intéressante a aussi été appliquée dans la résolution d'incohérences (Parsia et al. 2005), (Wang et al., 2005) : c'est le débogage d'ontologies. L'objectif est d'offrir à l'ingénieur des explications plus compréhensibles des incohérences que celles fournies par les raisonneurs standards. Deux types de techniques sont distingués : la technique *black-box* considérant le raisonneur comme une boite noire et appliquant des inférences pour localiser les incohérences et la technique *glass-box* qui modifie le mécanisme interne du raisonneur pour expliciter les incohérences et compléter les résultats des raisonneurs.

5 Conclusion et travaux en cours

Dans cet article, nous présentons une approche d'évolution ***Onto-Evo^{a}l*** dont l'objectif est d'optimiser et automatiser la gestion des changements tout en assurant la cohérence et la qualité de l'ontologie évoluée. La principale contribution de nos travaux est d'intégrer une modélisation par patrons dans un processus de gestion des changements d'ontologies permettant de définir et de classer des types de changements, d'incohérences et d'alternatives et de faire ressortir des liens entre ces patrons pour guider et contrôler l'analyse et la résolution des impacts de changements. De plus, l'évaluation de l'impact sur la qualité permet de mieux conduire le processus de gestion des changements et de préserver la qualité de l'ontologie évoluée.

Les patrons définis couvrent les changements OWL DL basiques et un sous-ensemble de changements complexes, un premier noyau d'incohérences logiques et d'alternatives pouvant les résoudre. Un module d'apprentissage est envisagé pour compléter et enrichir ces patrons à travers l'application de l'approche sur des ontologies tests. En effet, il n'est pas évident de fournir un système de gestion complet qui gère tous les types de changements.

Références

DJEDIDI R. & AUFAURE M.A. (2008). Enrichissement d'ontologies : maintenance de la consistance et évaluation de la qualité, 19èmes journées francophones d'Ingénierie des Connaissances (IC'08), Nancy.

GANGEMI A. (2005). Ontology Design Patterns for Semantic Web Content, In. Y. Gil, E. Motta, V..R. Benjamins, M. A. Musen (Eds.) (ISWC'05), Publication Springer-Verlag, LNCS 3729, pp. 262—276. Springer.

GANGEMI A., CATENACCI C. & BATTAGLIA M. (2004). Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations. In D.M. PISANELLI (Ed.) *Ontologies in Medicine. IOS Press*, Amsterdam.

GANGEMI A., GOMEZ-PEREZ A., PRESUTTI V. & SUAREZ-FIGUEROA, M.C. (2007). Towards a Catalog of OWL-based Ontology Design Patterns, CAEPIA 07, Publications du projet Neon (http://www.neon-project.org).

HAASE P. & STOJANOVIC L. (2005). Consistent Evolution of OWL Ontologies, European Conference on Semantic Web Proceedings (ECSW'05), Lecture Notes in Computer Science, (vol. 3532):182-197.

KLEIN M. (2004). Change Management for Distributed Ontologies. Thèse de doctorat, Dutch Graduate School for Information and Knowledge Systems.

PARSIA B., SIRIN E. & KALYANPUR A. (2005). Debugging OWL ontologies. 14ème conférence internationale sur le World Wide Web (WWW2005), Chiba, Japan.

PLESSERS P. & DE TROY O. (2006). Resolving Inconsistencies in Evolving Ontologies. European Conference on Semantic Web Proceedings (ECSW'06), Lecture Notes in Computer Science.

PLESSERS P., DE TROYER O. & CASTELEYN S. (2006). Understanding ontology evolution: A change detection approach. *Journal of Web Semantics*.

PRESUTTI V., GANGEMI A., DAVID S., AGUADO DE CEA G., SUAREZ-FIGUEROA M., MONTIEL-PONSODA E. & POVEDA M. (2008). Library of design patterns for collaborative development of networked ontologies. Deliverable D2.5.1, NeOn project.

RECTOR A. & ROGERS J. (2004). Patterns, properties and minimizing commitment: Reconstruction of the Galen upper ontology in owl. In A. GANGEMI & S. BORGO (Eds.), EKAW'04 Workshop on Core Ontologies in Ontology Engineering. CEUR.

SIRIN E., PARSIA B., CUENCA GRAU B., KALYANPUR A. & KATZ Y. (2007). Pellet: A practical OWL DL reasoner. *Journal of Web Semantics*, 5(2):51-53.

STOJANOVIC L (2004). *Methods and Tools for Ontology Evolution*. Mémoire de thèse, Université de Karlsruhe.

SVATEK V. (2004). Design patterns for semantic web ontologies: Motivation and discussion. 7ème conférence Business Information Systems, Poznan.

WANG H., HORRIDGE M., RECTOR A., DRUMMOND N. & SEIDENBERG J. (2005). Debugging OWL-DL ontologies: A heuristic approach. In Y. GIL, E. MOTTA, V..R. BENJAMINS, M. A. MUSEN (Eds.) (ISWC'05), Publication Springer-Verlag, ISBN 978-3-540-29754-3, Galway, Ireland.