Change Management Patterns (CMP) for Ontology Evolution Process

Rim Djedidi[1], and Marie-Aude Aufaure[2]

[1] Computer Science Department, Supélec Campus de Gif
Plateau du Moulon – 3, rue Joliot Curie – 91192 Gif sur Yvette Cedex, France
rim.djedidi@supelec.fr

[2] MAS Laboratory, SAP Business Object Chair –Centrale Paris
Grande Voie des Vignes, F-92295 Châtenay-Malabry Cedex, France
marie-aude.aufaure@ecp.fr

Abstract. Ontology evolution is an essential research area for the widespread use of ontologies in industrial and academic applications. In this paper, we present Change Management Patterns *CMP* defining three kinds of patterns: *Change Patterns*, *Inconsistency Patterns* and *Alternative Patterns*. CMP patterns are proposed to guide ontology evolution process by driving and controlling change application while maintaining consistency of the evolved ontology.

Keywords: Ontology Evolution, Change Management, Pattern Modeling, Inconsistency Resolution, OWL DL.

1 Introduction

Ontology development is a dynamic process starting with an initial rough ontology, which is later revised, refined and filled in with the details [1]. Even during ontology usage, knowledge of the modeled domain can change and develop. Changes generally aim to make the ontology more accurate or adequate with respect to the domain of discourse, the consistency model of the ontology language and ontology design practices.

Ontology evolution is a complex problem. In our work[1], we focus on issues related to ontology change management in a local context and particularly on consistency maintenance. To guide ontology evolution, we have defined Change Management Patterns CMP. The patterns model the three dimensions: *Change*, *Inconsistency* and *Resolution Alternative*. Based on the modeled patterns and the conceptual links between them, we propose an automated process driving change application while maintaining consistency of the evolved ontology.

[1] This work is founded by the French National Research Agency ANR, as a part of the project DAFOE: Differential and Formal Ontology Editor.

The paper is organized as follows: in section 2, we detail the application context of Change Management Patterns (CMP) which are described and illustrated in section 3. Application of CMP to guide change resolution is presented in section 4. Before concluding and discussing further developments of this work, we report on related work in section 5.

2 CMP as a *Meta-layer* on Top of an Ontology Evolution Process

CMP patterns are proposed as "meta-layer" of an ontology evolution approach – ***ONTO-EVO^{A}L*** – guiding the change management process at three key phases: change specification, change analysis and change resolution (Fig. 1). Three categories of patterns are modeled: *Change Patterns* classifying types of changes, *Inconsistency Patterns* classifying types of logical inconsistencies and *Alternative Patterns* classifying types of inconsistency resolution alternatives.

The starting point of ***ONTO-EVO^{A}L*** process is the specification of the required change by formally describing its semantics (intermediate changes composing it –if any– and their order, involved entities, change values, etc.). This is guided by the instantiation of the corresponding *Change Pattern*, resulting in an explicit change signature. The following phase aims to analyze change impact by explaining and localizing caused inconsistencies. Detected inconsistencies are classified according to the corresponding *Inconsistency Patterns*. In the change resolution phase, the instantiation is not made from process level to pattern level, but rather in the other way i.e. by generating alternative instances from the conceptual links between inconsistency patterns –instantiated in the previous phase– and *Alternative Patterns* resolving them (Section 3). Resolution alternatives represent additional and/or substitutive changes to implement to maintain ontology consistency.

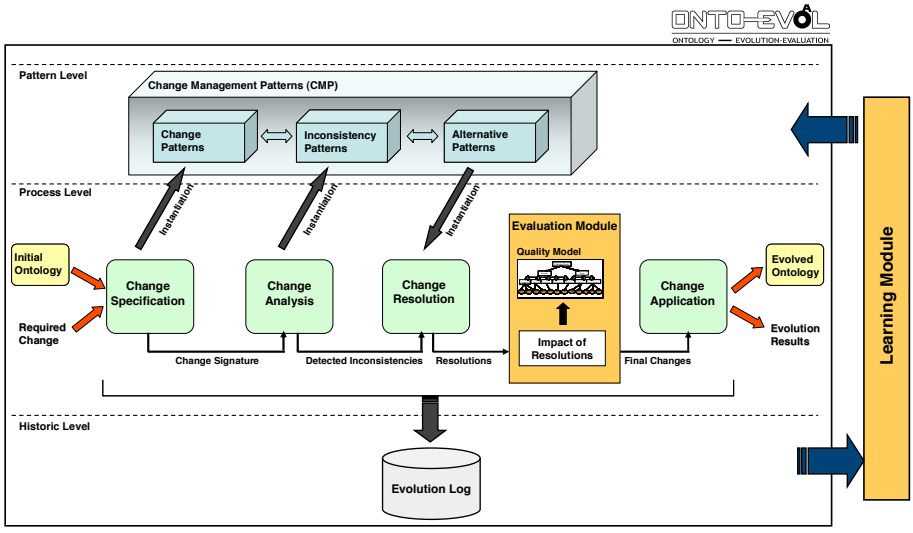


Fig. 1. CMP as a Meta-layer on Top of an Ontology Evolution Process.

In addition, the process includes a *Quality Evaluation Module* assessing the impact of proposed resolutions on ontology quality. The goal of the evaluation module is to help selecting the resolution that preserves ontology quality. If quality level is maintained, required change and its derived changes are directly applied and the ontology evolves. In the case that all the proposed resolutions have negative effect on ontology quality, the results of the different phases are presented to ontology engineer –as a complement to his expertise– so that he can decide about the changes.

All the results of the process are saved in the evolution log to keep ontology evolution historic. Besides controlling ontology evolution, revoking or justifying changes, the evolution log facilitates learning new change management patterns. For a detailed description of ***ONTO-EVO^{A}L*** evolution process, the reader can refer to [2].

3 Change Management Patterns (CMP) Description

In this section, we want to highlight the role of CMP in guiding the evolution process and more precisely change specification, analysis and resolution phases. CMP are proposed as a solution looking for invariances in change management that repeatedly appear when evolving ontologies. Three categories of patterns are distinguished: *Change Patterns*, *Inconsistency Patterns* and *Alternative Patterns*. The goal of CMP modeling is to offer different levels of abstraction, to establish conceptual links between these three categories of patterns (Fig. 2) determining the inconsistencies that could be potentially caused by a type of change and the alternatives that possibly resolve a kind of inconsistency and thus, to guide an automated change management process.

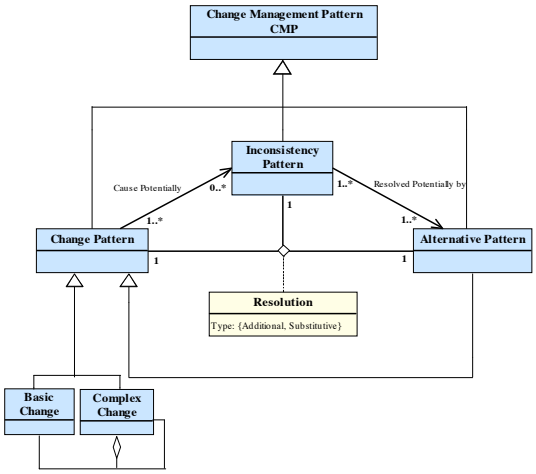


Fig. 2. CMP Conceptual Model.

CMP classify categories of changes based on OWL meta-model, categories of logical inconsistencies considering OWL DL constraints and categories of inconsistency resolution alternatives. To describe the three categories of patterns and

explain theirs relations, we illustrate step-by-step the instantiation of the patterns based on a simple example given as a thread.

Let's consider the OWL ontology *O* defined by the following axioms:

{Animal ⊑ Fauna-Flora, Plant ⊑ Fauna-Flora, Carnivorous-Plant ⊑ Plant, Plant ⊑ ¬Animal}

And let's consider a required change *Ch1* defining *Carnivorous-Plant* class as a sub-class of class *Animal*.

CMP are modeled as an OWL DL ontology describing CMP catalogue. In addition, a simplified representation is used. It extends the representation of Ontology Design Patterns ODP [3] adopted from a common and accepted format describing design patterns in software engineering [4].

CMP are described by a general template containing the following slots:

- General Properties: *General Information* about the pattern, and *Use Case* of the pattern;
- Specific Properties related to CMP abstraction level including general properties of *Pattern Description* (some independent from pattern category and others extended to the different kind of CMP patterns), *Graphical Representation* of the pattern represented in UML format, and *Implementation* of the pattern in OWL DL language (not presented in this paper for page limit reason);
- Properties about *Relationships* of the pattern with other patterns including *Relations to Other CMP*.

These slots are presented in the following tables (Tab. 1) (Tab. 2) (Tab. 3) (Tab. 4).

3.1 Change Patterns

Change patterns aim to categorize changes, formally define their signification, their scope and their potential implications. Two categories of OWL changes are distinguished in the literature [5]: basic changes and complex changes. Change patterns cover all OWL basic changes and a first core of complex changes.

Basic change patterns describe all OWL basic changes derived from OWL meta-model. They model simple and indivisible OWL changes. To extend the set of defined OWL basic operations with composite and specific change operations, we define complex change patterns. They correspond to composite OWL changes grouping logical sequences of basic and composite changes (e.g. enlarging the range of an object property to its super-class or merging two classes).

Considering the example given above, the change *Ch1* instantiate the basic change pattern: *Add a sub-class*. The description and the instantiation of the pattern are given by the following table (Tab. 1).

Table 1. Basic change pattern: add a sub-class.

Basic change Pattern Example	
General Properties	
General Information	
Name*	Add a sub-class.
Identifier*	BChP_AddSubClass.

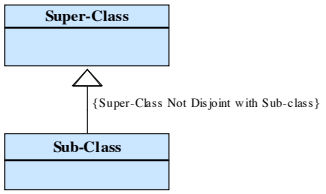
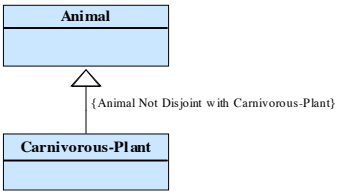
CMP Type*	Change Pattern.
Use Case	
Problem	Define a sub-class relation between two classes.
Examples	Suppose that we need to express that the class Carnivorous-*Plant* is a sub-class of the class *Animal*.
CMP Abstraction Level	
Pattern Description	
Intent	The pattern models an indivisible change that defines a sub-class relation between two classes and notifies if there is any constraint on this subsumption so that it does not affect the logical consistency of the ontology.
Consequences	The pattern defines a subsumption between two classes and notifies that the super-class and the sub-class should not be disjointed.
Scenarios	Define the class *Carnivorous-Plant* as a sub-class of the class *Animal*. Notify that the constraint: {class *Carnivorous-Plant* and class *Animal* should not be disjointed} has to be verified.
Change Abstraction Level	
Pattern Description	
Change Pattern Type*	Basic Change Pattern.
Object Type*	Class.
Involved Entity Type*	Class, Class
Arguments*	
Object*	ID of the sub-class (sub_classID).
Example	Carnivorous-Plant.
Referred Entities*	
Sub-Class ID	ID of the sub-class (sub_classID).
Super-Class ID	ID of the super-class (super_classID).
Examples	
Sub-Class ID	Carnivorous-Plant.
Super-Class ID	Animal.
Constraints	
Constraints	¬(sub_classID disjointWith Super_classID).
Examples	¬(Carnivorous-Plant disjointWith Animal).
Graphical Representation	
Diagram	

Diagram Examples	
Relationships	
Relations to other CMP	
Inconsistency Patterns	Inconsistency disjointness related to subsumption.

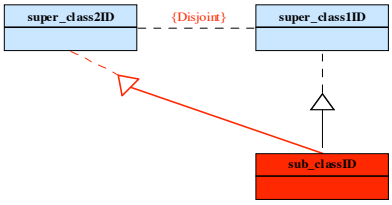
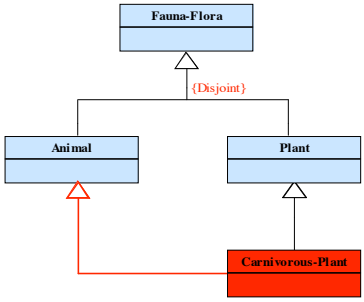
3.2 Inconsistency Patterns

After applying temporary the required change, inconsistencies detected using an ontology reasoner, are classified according to inconsistency patterns. Inconsistency patterns model a sub-set of OWL DL logical inconsistencies: disjointness inconsistencies related to subsumption and instantiation; inconsistencies related to equivalence and complement, inconsistencies related to equivalence and disjointness, inconsistencies related to value restrictions and inconsistencies related to cardinality restrictions.

Reconsidering the example given as a thread, the pattern corresponding to the disjointness inconsistency detected and its instantiation are described by the following table (Tab. 2).

Table 2. Inconsistency pattern: inconsistency of disjointness related to subsumption.

Inconsistency Pattern Example	
General Properties	
General Information	
Name*	Inconsistency disjointness related to subsumption.
Identifier*	InconsP_DisjSub.
CMP Type*	Inconsistency Pattern.
Use Case	
Problem	Analyze and delineate a disjointness inconsistency related to a subsumption relation between two classes.
Examples	Suppose that we need to explain and track a disjointness inconsistency caused by a subsumption relation between the class *Carnivorous-Plant* and the class *Animal*.
CMP Abstraction Level	
Pattern Description	
Intent	The pattern models explicitly the analysis of a disjointness inconsistency related to a subsumption relation between two classes of an ontology.
Consequences	The pattern explains a disjointness inconsistency related to a subsumption relation and gives details on its analysis and localization.

Scenarios	Explain disjointness inconsistency caused by a subsumption relation between the class *Carnivorous-Plant* and the class *Animal* by tracking the classes concerned by this inconsistency and specifying the axioms causing it.
Inconsistency Abstraction Level	
Pattern Description	
Arguments*	
Implicated Entities*	
ID SuperClass1	ID of a first super-class (super_class1ID).
ID SuperClass2	ID of a second super-class (super_class2ID).
ID SubClass	ID of the sub-class (sub_classID).
Examples	
ID SuperClass1	Plant.
ID SuperClass2	Animal.
ID SubClass	Carnivorous-Plant.
Involved Entities*	
ID SuperClass2	ID of the involved super-class (super_class2ID).
ID SubClass	ID of the sub-class (sub_classID).
Examples	
ID SuperClass2	Animal.
ID SubClass	Carnivorous-Plant.
Axioms*	
Involved Axioms	(super_class1ID disjointWith super_class2ID), (sub_classID $\sqsubseteq$ super_class1ID).
Examples	(Plant $\sqsubseteq$ $\neg$Animal), (Carnivorous-Plant $\sqsubseteq$ Plant).
Responsible Axioms	(sub_classID $\sqsubseteq$ super_class2ID).
Examples	(Carnivorous-Plant $\sqsubseteq$ Animal)
Graphical Representation	
Diagram	
Diagram Examples	

Relationships	
Relations to other CMP	
Change Patterns	Add a sub-class, …
Alternative Patterns	- Define Hybrid Class for Resolving Disjointness Subsumption, - Enlarge Class Definition for Resolving Disjointness Subsumption.

3.3 Alternative Patterns

Change resolution is based on the conceptual N-ary relation-class *Resolution* defined between change, inconsistency and alternative patterns (Fig. 2). For each detected inconsistency, based on the corresponding inconsistency pattern instance and the instantiated change pattern specifying the required change, potential alternative patterns are generated and instantiated. An alternative pattern represents an additional change (applied jointly to the required change) or a substitutive change to apply (replacing the required change) so that a logical inconsistency can be resolved. It is described as a change (basic or complex) and it inherits and extends change pattern properties (Fig. 2).

Several resolution alternatives can be proposed for an inconsistency. To resolve the inconsistency described above in the example, two alternatives can be proposed: the first one (Tab. 3) is a substitutive resolution extending a complex change. The second one is a substitutive resolution extending a basic change (Tab.4).

Table 3. Alternative pattern: define hybrid class for resolving disjointness related to subsumption.

Alternative Pattern Example	
General Properties	
General Information	
Name*	Define Hybrid Class for Resolving Disjointness_Subsmption.
Identifier*	AltP_DefHybClsResolDisjSubs
CMP Type*	Alternative Pattern.
Use Case	
Problem	Resolve disjointness –related to a subsumption– by defining a hybrid class.
Examples	Suppose that we need to resolve a disjointness inconsistency caused by subsuming the class *Animal* by the class Carnivorous-*Plant*.
CMP Abstraction Level	
Pattern Description	
Intent	The pattern models a resolution alternative resolving disjointness inconsistency –related to a subsumption– by creating a hybrid class.
Consequences	The pattern resolves a disjointness inconsistency –related to a subsumption–by defining a hybrid class based on the definition of disjoint classes implicated in the inconsistency, and redistributing correctly sub-class relations between classes implicated in the inconsistency, the hybrid class, and the most

	specific common super-class of the disjoint classes implicated.
Scenarios	Define a hybrid class *Animal_Plant* based on the definition of the two disjoint classes involved in the inconsistency: *Animal* and *Plant*. Then, create a sub-class relation between the hybrid class created and a the most specific common super-class of the classes *Animal* and *Plant*. And finally, substitute the sub-class relation between the classes *Animal* and *Carnivorous-Plant* by a subsumption between the classes *Carnivorous-Plant* and *Animal_Plant*.
Alternative Abstraction Level	
Pattern Description	
Process	1) The pattern defines a hybrid class as a union of the definitions of the disjoint classes implicated in the inconsistency to be resolved; 2) The pattern defines a subsumption between the most specific common super-class of the disjoint classes implicated in the inconsistency and the hybrid class created; 3) The pattern defines a subsumption between the hybrid class and the sub-class involved in the inconsistency.
Examples	1) The pattern defines a class *Animal_Plant* as a union of the definitions of the disjoint classes *Animal* and *Plant*; 2) The pattern defines a subsumption between the most specific common super-class of the disjoint classes *Fauna-Flora* and the hybrid class created *Animal_Plant*; 3) The pattern defines a subsumption between the defined hybrid class *Animal_Plant* and the sub-class *Carnivorous-Plant* involved in the inconsistency.
Change Abstraction Level	
Pattern Description	
Change Pattern Type*	Complex change pattern.
Object Type*	Class.
Involved Entity Type*	Class, Class, Class.
Arguments*	
Object*	ID of the hybrid class (HybridClassID).
Example	Animal_Plant.
Referred Entities*	
Sub-Class ID	ID of the sub-class (sub_classID).
1st Disjoint Class ID	ID of a first disjoint class (Disjoint_Class1ID).
2sd Disjoint Class ID	ID of a second disjoint class (Disjoint_Class2ID).
Examples	
Sub-Class ID	Carnivorous-Plant.
1st Disjoint Class ID	Animal.
2sd Disjoint Class ID	Plant.
Intermediate Entities*	
Common Super-Class ID	ID of the most specific common super-class of the disjoint classes implicated (Common_super_classID).
Examples	Fauna-Flora.
Complex Change Abstraction Level	
Pattern Description	

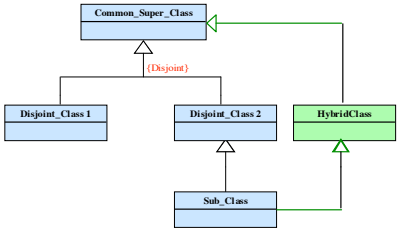
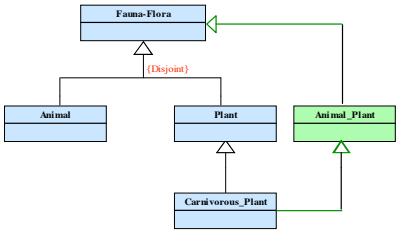
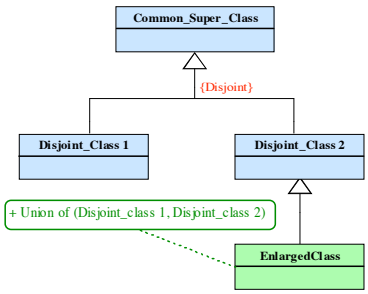
Sequence*	1) BChP_AddClass (HybridClassID, Collection(Disjoint_Class1ID, Disjoint_Class2ID), Operator(Union)) ; 2) BChP_AddSubClass (HybridClassID, Common_super_classID) 3) BChP_AddSubClass (sub-ClassID, HybridClassID)
Examples	1) BChP_AddClass (Animal_Plant, Collection(Animal, Plant), Operator(Union)) ; 2) BChP_AddSubClass (Animal_Plant, Fauna-Flora) 3) BChP_AddSubClass (Carnivorous-Plant, Animal_Plant)
Graphical Representation	
Diagram	
Diagram Examples	
Relationships	
Relations to other CMP	
Change Patterns	Add a class, Add a sub-class.
Inconsistency Patterns	Inconsistency disjointness related to subsumption.

Table 4. Alternative pattern: enlarge class definition for resolving disjointness related to Subsumption (Synthetic version)

Alternative Pattern Example	
General Properties	
General Information	
Name*	Enlarge Class Definition for Resolving Disjointness_Subsumption.
Identifier*	AltP_EnlarClsDefResolDisjSubs.
CMP Type*	Alternative Pattern.
Use Case	
Problem	Resolve disjointness –related to a subsumption– by enlarging a class definition.
Examples	Suppose that we need to resolve a disjointness inconsistency caused by subsuming the class *Animal* by the class *Carnivorous-Plant*.
CMP Abstraction Level	

Pattern Description	
Intent	The pattern models a resolution alternative resolving disjointness inconsistency –related to a subsumption– by enlarging the definition of a class.
Consequences	The pattern resolves a disjointness inconsistency –related to a subsumption–by enlarging the definition of the sub-class involved in the inconsistency, based on the definition of disjoint classes implicated in the inconsistency.
Scenarios	Enlarge the definition of the class *Carnivorous-Plant* based on the definition of the classes *Animal* and *Plant*.
Alternative Abstraction Level	
Pattern Description	
Process	1) The pattern enlarges the definition of the sub-class involved in the inconsistency by adding –in its description– a union of the definitions of disjoint classes implicated in the inconsistency.
Examples	1) The pattern enlarges the definition of the sub-class *Carnivorous-Plant* by adding –in its description– a union of the definitions of the disjoint classes implicated in the inconsistency: the classes *Animal* and *Plant*.
Change Abstraction Level	
Pattern Description	
Change Pattern Type*	Basic change.
Object Type*	Class.
Involved Entity Type*	Class, Class Description.
Arguments*	
Object*	ID of the class to enlarge (classID).
Example	Carnivorous_Plant.
Referred Entities*	
ID Class	ID of the class to enlarge (classID).
ID(s) Collection	ID(s) of the classes of the collection (Disjoint_Class1ID, Disjoint_Class2ID).
Examples	
ID Class	Carnivorous-Plant.
ID(s) Collection	Animal, Plant.
Operator*	
Operator	Union
Graphical Representation	
Diagram	

4 Change Resolution Guided by CMP Application

The primary objective of ontology pattern modeling is to provide shared and reusable guidelines [3], it is therefore necessary to use a readable and understandable formalism to present CMP patterns (Section 3). However, to fulfill the purpose of providing a meta-layer for ontology evolution process guiding change management in an automated way, CMP have to be formally specified in a well expressive language to facilitate explicit interpretation of their semantics. For this reason, we have formalized CMP patterns as an OWL DL ontology. Applying CMP in ***ONTO-EVO^{A}L*** process is thus, based on the identification of class, property and axiom matching according to given change and inconsistencies.

In the change specification phase (Fig. 1), an appropriate change pattern is selected and instantiated to specify explicitly a required change. The temporary application of the change is then, analyzed to detect caused inconsistencies. This phase is performed by employing *Pellet Reasoner* [6]. Pellet supports both terminological level *TBox* (classes and properties) and assertional level *ABox* (individuals) of OWL DL and provides entailment justifications. However, it does not precise axioms that cause inconsistencies neither how to resolve the detected inconsistencies. Inconsistency localization is driven as a *black-box* approach [7]. Localization algorithm is implemented as a top layer, independent from the reasoner, calling it a linear number of times. Localization algorithm extends the algorithm presented in [8], determining the minimal inconsistent sub-ontology O' as $O' \subseteq O$ (O the analyzed ontology) and $\forall O'' \subset O'$, O'' is a consistent sub-ontology in O. The principle is to start by OWL DL axioms corresponding to the instantiated change pattern, as an input of a selection function called iteratively to select a larger sub-set of axioms and constitute the minimal inconsistent sub-ontology. Axiom selection is based on structural connectedness defined in [8].

Inconsistency detection and localization prepare inconsistency pattern selection and instantiation. Localized inconsistencies are matched to inconsistency patterns to be classified. The matching process consists in identifying correspondences by considering inconsistency pattern type and structure, and also the semantics of pattern arguments and axioms (Tab. 2), which define the *pattern interface* that has to be instantiated. Guided by the instantiated change pattern and its constraints (Tab. 1), the matching process targets firstly, inconsistency patterns that could be potentially caused (Fig. 2). Two types of indicator are considered: structural information (matching of sub-hierarchies), and axiomatic information related to localized inconsistencies (localization context, entities declared as inconsistent, axioms related to these entities, etc.).

Once localized inconsistencies are classified, potential alternative patterns are identified and generated based on N-ary relation classes – *Resolution* – defining semantic resolution relations between instantiated, change and inconsistency patterns. Then, the instantiation of the generated alternative patterns is adapted to the sub-ontology concerned by its application.

Alternative instances proposed for the different inconsistencies are combined to derive global potential resolutions for the required change. Each resolution defines a set of derived changes but should not cause other inconsistencies. Therefore, all

derived global resolutions are verified using Pellet reasoner, and only consistent ones are accepted for the evaluation step (Fig.1).

5 Discussion and Related Work

Discussion with related work is tackled through three parts: pattern modeling, ontology evolution approaches, and inconsistency diagnosis and repair.

Pattern modeling was adopted in web ontology design to propose guidelines and provide reusable ontological component catalogue[2]. CMP patterns are close to Ontology Design Patterns[3] ODP, particularly Logical Ontology Patterns *LOP* and Content Ontology Patterns *COP* [9]. *Change Patterns* of CMP can be considered as *COP* patterns for ontology domain i.e. ontology design patterns solving modeling problems of the domain 'ontology'. *Alternative Patterns* of CMP can be defined as an *LOP* pattern resolving a problem of logical inconsistency.

Concerning OWL ontology evolution approaches, in [10], a pattern-driven approach was adopted for ontology evolution. The patterns determine the evolution operation to be performed: population (adding new instances) or enrichment (extension by new concepts and properties). In [8], authors have introduced resolution strategies based on OWL Lite model. The resolution is limited to the identification of axioms that should be removed to resolve inconsistencies and, their presentation to the user. In ***ONTO-EVO^{A}L*** approach, we tend to minimize axiom removing solutions by proposing alternatives that merge, divide, generalize or specialize classes and properties and redistribute instances to preserve existent knowledge.

Concerning inconsistency diagnosis and repair, several debugging services and strategies are proposed in the literature [11] [12] [13] [14] [7]. They provide support to ontology developers by explaining the main causes for unsatisfiable classes or contradictions. However, they provide little support for proposing solutions for them. Solutions are always limited to two choices: removing part of the existing axioms or replacing a class by one of its super-classes. We claim that it is possible to provide additional support to ontology developers, based on the identification and modeling of common patterns of inconsistencies caused when applying some types of changes, and the proposition of some typical alternatives that could potentially resolve them, which can be combined with the use of existing reasoning tools in order to make this task more effective.

6 Conclusion and Future Work

In this paper, we have presented CMP patterns proposed as "meta-layer" of an ontology evolution approach – ***ONTO-EVO^{A}L*** – to guide and control the change management process at three key phases: change specification, change analysis and change resolution.

Currently, we are developing a learning module enriching and enhancing *CMP* by considering evolution log information, new change compositions not yet supported by

[2] Example: http://sourceforge.net/projects/odps/
[3] http://ontologydesignpatterns.org/

Change Patterns, detected inconsistencies not yet classified by *Inconsistency Patterns*, new potential resolution alternatives not yet modeled by *Alternative Patterns*, and also new possible relation instantiations between CMP.

References

1. Noy, N. F., McGuinness, D.: Ontology development. 101: a guide to creating your first ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, (2001).
2. Djedidi, R., Aufaure, M-A.: Ontology Change Management. In: A. Paschke, H. Weigand, W. Behrendt, K. Tochtermann, T. Pellegrini (Eds.), I-Semantics 2009, Proceedings of I-KNOW '09 and I-SEMANTICS '09, ISBN 978-3-85125-060-2, pp. 611--621, Verlag der Technischen Universitt Graz. (2009)
3. Gangemi, A., Gomez-Perez, A., Presutti, V., Suarez-Figueroa, M.C.: Towards a Catalog of OWL-based Ontology Design Patterns, CAEPIA 07, Neon project publications (http://www.neon-project.org), (2007).
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, (1995).
5. Klein, M.: Change Management for Distributed Ontologies. Ph.D. Thesis, Dutch Graduate School for Information and Knowledge Systems, (2004).
6. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics, 5(2), (2007).
7. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P. A Modularization-based Approach to Finding All Justifications for OWL DL Entailments. ASWC'08. (2008).
8. Haase, P., Stojanovic, L.: Consistent Evolution of OWL Ontologies, In Gomez-Perez, A., Euzenat, J. (Eds.) ESWC 2005. LNCS, vol.3532, pp. 182--197. Springer, Heidelberg, (2005).
9. Presutti V., Gangemi A., David S., Aguado De Cea G., Suarez-Figueroa M., Montiel-Ponsoda E., Poveda M.: Library of design patterns for collaborative development of networked ontologies. Deliverable D2.5.1, NeOn project, (2008).
10. Castano, S., Espinosa, S., Ferrara, A., Karkaletsis, V., Kaya, A., Melzer, S., Moller, R., Montanelli, S., Petasis, G.: Ontology Dynamics with Multimedia Information: The BOEMIE Evolution Methodology. In Proceedings of International Workshop on Ontology Dynamics (IWOD-07), ISWC 2007 Conference, (2007).
11. Schlobach, S. Debugging and Semantic Clarification by Pinpointing. ESWC 2005. LNCS Vol. 3532, pp: 226-240. (2005).
12. Wang, H., Horridge, M., Rector, A., Drummond, N., & Seidenberg, J. Debugging OWL-DL ontologies: A heuristic approach. In Y. Gil, E. Motta, V. R. Benjamins, & M. A. Musen (Eds.), LNCS: Vol. 3729. The Semantic Web – ISWC 2005 (pp. 745-757). Berlin, Germany: Springer. (2005).
13. Qi, G., Liu, W., Bell, D. A revision-based approach to handling inconsistency in description logics. Journal of Artificial Intelligence Review, 26(1-2): 115-128. (2006).
14. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E. Finding All Justifications of OWL DL Entailments. Proceedings de ISWC/ASWC'2007. LNCS, Vol. 4825. pp: 267-280. (2007).