Journal of Web Engineering, Vol. 14, No.3&4 (2015) 301-324

A NOVEL ONTOLOGY EVOLUTION METHODOLOGY

ALI RAHNAMA AHMAD ABDOLLAHZADEH BARFOROUSH

Computer Engineering and IT department, Amirkabir University of Technology
424 Hafez Ave., Tehran, Iran

{arahnama, ahmad}@aut.ac.ir

Received November 11, 2013
Revised December 1, 2014

In recent years, ontology engineering has received a great amount of attention and has advanced greatly. Today, ontologies are finding their role in knowledge and information systems. To keep up with the dynamic aspect of knowledge, the need for ontology evolution systems has emerged. Such systems help facilitate the management of changes on the ontology in a systematic way. To define a systematic way of facilitating management of changes, a process model is needed. Therefore in this paper, we present our novel ontology evolution process model which uses ontology change rules to achieve this goal. These change rules are defined via the SWRL rule representation language. This new approach to ontology evolution helps the ontology evolution process by preserving the consistency of the ontology throughout the ontology evolution process. To classify ontology changes, we will also present our novel taxonomy of ontology changes. To test the feasibility of our presented process model, we have implemented the OntoEvol system. It is also presented in this paper.

Key words: Ontology Evolution, Ontology Maintenance, Ontology Changes, Knowledge Management
Communicated by: D. Schwabe & R. Mizoguchi

1 Introduction

Over the past years, ontology engineering has received a great amount of attention and has advanced greatly. Today, ontologies are finding their role in knowledge and information systems. With the application of these ontologies in such systems and the fact that knowledge is not static, the need of keeping the ontology up to date has emerged. To keep up with this dynamic aspect of domain knowledge, the need for ontology evolution systems has emerged. Such systems help facilitate the management of changes on the ontology in a systematic way. Ontology Evolution is the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, and the consistent management/propagation of changes to dependent elements [1].

The distinction between the concepts *ontology management*, *ontology modification*, *ontology evolution*, and *ontology versioning* should be clarified, which are presented in [2] as:

- *Ontology management* is the whole set of methods and techniques that is necessary to efficiently use multiple variants of ontologies from possibly different sources for different

tasks. Therefore, an ontology management system should be a framework for creating, modifying, versioning, querying, and storing ontologies. It should allow an application to work with an ontology without worrying about how the ontology is stored and accessed, how queries are processed, etc.

- *Ontology modification* is accommodated when an ontology management system allows changes to the ontology that is in the use, without considering the consistency.

- *Ontology evolution* is accommodated when an ontology management system facilitates the modification of an ontology by preserving its consistency.

- *Ontology versioning* is accommodated when an ontology system management allows handling of ontology changes by creating and managing different versions of it.

So as stated above, ontology evolution is the activity of facilitating the modification of an ontology by preserving its consistency [3]. To better clarify this definition, we define ontology evolution as follows:

Definition (ontology evolution):

Given the ontology O and the set of ontology change operations $Op \equiv \{Op_1, Op_2, \cdots, Op_x\}$. ontology evolution is the application of change operations $Op_1, Op_2, \cdots, Op_x$ to ontology $O \rightarrow O'$ for which the resulting O' is a consistent ontology.

In other words, ontology evolution is the process of applying a set of changes on an ontology and creating a new version of the ontology which is consistent. In this process, one of the hardest tasks is insuring that the evolved ontology is consistent. For this reason, we decided to find a process that applies the changes on the ontology, but doesn't need to check for its consistency. Our proposed ontology evolution process model facilitates this exact goal by only implementing safe ontology changes to the initial ontology, so that the resulting ontology will not become inconsistent, therefore the check is not needed. Therefore, to help facilitate ontology evolution, we present our novel ontology evolution process model which uses ontology change rules. These change rules are defined via the SWRL rule representation language. This new approach to ontology evolution helps the ontology evolution process by preserving the consistency of the ontology throughout the ontology evolution process. The reason for this is that only changes that preserve the consistency are performed on the ontology. Therefore, the consistency of the ontology is preserved. Also, our approach to ontology evolution can help the ontology engineer by presenting an analysis of the type of changes that are to be performed on the ontology beforehand, so that the ontology engineer can be sure that the intended changes are being performed. We will present an example of it in section 4. In this paper, we will also present the OntoEvol system which is used to test the feasibility of our approach. The reason for this is that in order to show the feasibility of our proposed process model, we need 1) a change taxonomy (to know what kind of changes may occur and control the requested changes), 2) SWRL rules (to provide a means of defining the relations between different changes), and 3) the OntoEvol system (to perform all the phases in the process model and provide the results.). The remainder of this paper is as follows: First, we will present related work and research done in the ontology evolution field. Second, we will present our approach and process model to ontology evolution. Third, we will present our taxonomy of

ontology changes which facilitates the classification of ontology changes. Fourth, we will present our SWRL rules. Fifth, the OntoEvol system is presented. Sixth, we present our results and define future work.

2 Related Work

The ontology evolution field of research has been an active field over the past decade. Many researches have discussed the characteristics of an ontology evolution process [1], [4] and several ontology evolution approaches have been proposed in literature [2], [3], [5], [6].

The first and most referenced work in this field is the work of Stojanovic [2]. Stojanovic presented a six phase process for performing ontology evolution. It is presented in figure 1. Stojanovic defined ontology evolution as "Ontology Evolution is the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artifacts".

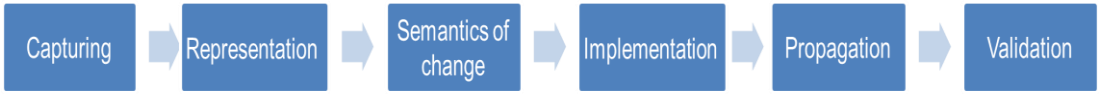


Figure 1 Six phase ontology evolution process presented by Stojanovic

Also, Flouris et. al. [7] define ontology evolution as "the process of modifying an ontology in response to a certain change in the domain or its conceptualization". They also differentiate it from ontology versioning as, "ontology evolution is restricted to the process of modifying an ontology while maintaining its validity, whereas ontology versioning deals with the process of managing different versions of an evolving ontology, maintaining interoperability between versions, and providing transparent access to each version as required by the accessing element (data, service, application or other ontology)". In this paper, we are loyal to Stojanovic's view of ontology evolution.

Another process model that has been presented is Zablith's approach [5], which also uses Stojanovic's view and uses background knowledge to help analyze the *ontological changes* phase and therefore needs less user input in the ontology evolution process. Zablith has also implemented his process model as a plugin in the NeOn toolkit by the name of Evolva [8],[9].

Figure 2 Five phase ontology evolution process presented by Zablith

Also in the NeOn project [3], a process model for ontology evolution which consists of four phases has also been presented. It can be seen in Figure 3. As it can be seen, NeOn project's view is from a very broad and general perspective. The reason for that is that they planned to use this process model as part of the NeOn toolkit therefore, it had to be very general so that the developers using the toolkit would not be restrained.

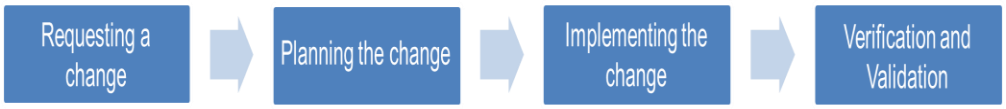


Figure 3 Four phase ontology evolution process presented by the NeOn Project

Another process model that has been proposed is the work of Djedidi [10]. A seven phase ontology evolution process model is presented which can be seen in figure 4. A pattern based consistency resolution approach from both local and global perspectives is followed.

Figure 4 Seven phase ontology evolution process presented by Djedidi

Jaziri in [11] proposes an approach of ontology evolution which facilitates a versioning management process, based on three phases: evolution changes, ontology coherence and versioning management. In it, the first two phases contain two subtasks each: change identification and change representation; and change propagation and coherence analysis respectively. It can be seen in Figure 5. It lets the ontology become inconsistent and then tries to resolve the inconsistencies in the ontology coherence phase. This is done via corrective operations.

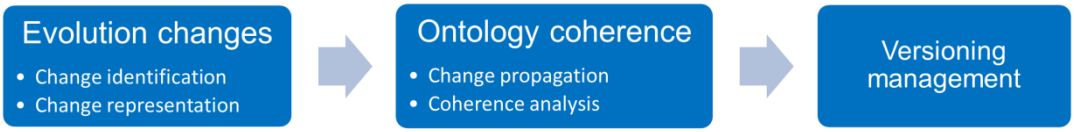


Figure 5 Three phase ontology evolution process presented by Jaziri

Other related work on ontology evolution related fields, mainly maintenance and management are as follows: capturing change requirements [12]–[14]; belief change based approaches [15]; change detection and version logging [16]–[20]; learning based approaches [21], [22]; automatic ontology evolution [5], [52]; formal change specification [1], [4], [16], [23]; change implementation [2], [15], [24], [25]; consistency maintenance [2], [16], [26]–[28]; ontology versioning [4], [11], [17], [29]; change impact analysis and resolution; and pattern based approaches [6], [30]–[35].

In table 1, we have reviewed the proposed process models and their supporting tasks. In table 1, the following definitions have been assumed: 1)*change detection and identification* is the process of collecting and identifying changes to be performed on the ontology (requested changes), 2)*change representation* is the process of specifying and standardizing the requested changes into a format appropriate for use in the ontology evolution process, 3)*change analysis* is the process of analyzing the changes in order to better understand their semantic intentions (this has been achieved differently in each of the process models mentioned, but all of them provided some sort of analysis on the changes.),

4)*change resolution* is the process of deciding which of the requested changes are to be performed on the ontology, 5)*change implementation* is the process of actually performing the requested changes on the ontology, 6)*change propagation* is the process of accessing the effects, side effects, and their scope caused by performing the changes on the ontology, 7)*change validation and verification* are the process of determining whether the changes performed were valid and were they performed correctly, 8)*change recommendation* is the process of suggesting to the user, changes that will improve the quality of the ontology (by implementing good practices and ontology design patterns), 9)*change and ontology versioning* is the process of keeping track of changes done to the ontology and managing the different versions created in the process. As it can be seen, the main differences between our approach and the others is that 1)there is no need for change validation because the consistency is preserved in our process(shown with '-'), 2)there is no need for change verification, because we assume that the user requests correct changes(shown with '-'), 3)there is no need for change propagations, because we in our change resolution phase, evaluate all changes with respect to the ontology change rules therefore, restricting their side effects, and 4)we have added a recommendation phase to the process to help facilitate better use and implementation of ontology design and implementation strategies.

Table 1 Comparison of ontology evolution process models

Change tasks	Stojanovich	NeOn project	Djedidi	Zablith	Jaziri	Our approach
Change detection and identification	×	×	×	×	×	×
Change representation	×		×		×	×
Change analysis	×	×	×	×	×	×
Change resolution	×	×	×	×		×
Change implementation	×	×	×	×		×
Change propagation	×		×	×	×	-
Change validation and verification	×	×	×	×		-
Change recommendation						×
Change and ontology versioning					×	

The analysis of the related works showed that, there is still a need for a process model which can assist and guide the evolution process while preserving the consistency of the ontology being changed. For this reason, we decided to present our novel process model which can help achieve the mentioned goal. We will present our approach in section 3. Also, to help facilitate such a process model, we will present our taxonomy of ontology changes and our ontology change rules in sections 3.1 and 3.2 respectively. The OntoEvol system, which implements our process model, is also presented in section 3.3. And finally in Section 4, we will present and discuss the results of our test case and define future work to be done.

3 A New Approach to Ontology Evolution

After reviewing the different process models presented in section 2, we define our novel process model for ontology evolution. It has been created with all the process models mentioned in mind, so that it can perform like them without losing any main functionality. It consists of six phases: 1)The *change request* phase, in which the changes that are to be done on the ontology are received and collected. 2)The *change identification* phase, in which the change requests are analyzed to determine their types with respect to our presented taxonomy of ontology changes(presented in section 3.1). 3)The *change analysis* phase, in which the identified changes are analyzed furthermore to if possible convert them into their equivalent higher level changes with the help of our presented SWRL rules (presented in section 3.2). 4)The *change resolution* phase in which we decide which of the changes are to be performed so that the consistency of the resulting ontology is preserved. 5)The *change implementation* phase in which the approved changes are performed on the ontology. 6)The *change recommendation* phase in which the updated ontology is evaluated for common pitfalls [36], anti-patterns [37], and compliance with ontology design patterns [38]. If there are no problems the updated ontology is released for use. If any problems are found, they are reported to the user as change recommendations for which the user can define new change request with respect to them. Our proposed ontology evolution process model can be seen in figure 6. We will discuss them further in section 3.3.

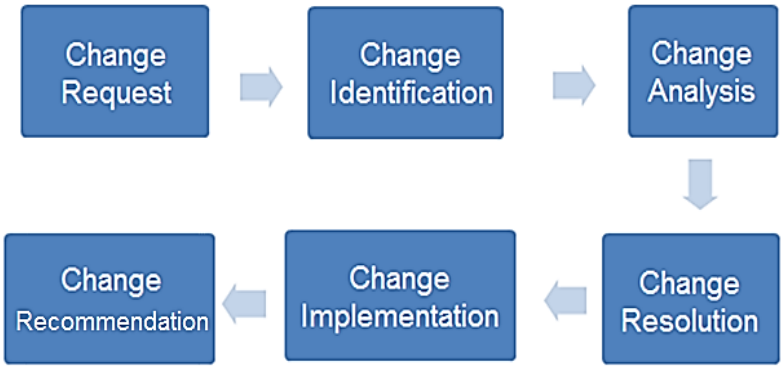


Figure 6 Our ontology evolution process model

3.1 Taxonomy of Ontology Changes

We, in our research, have investigated different literature and have created a taxonomy for ontology change [39]. We believe, to help facilitate better evolution and maintenance of ontologies, a taxonomy of changes is needed. With the existence of such a taxonomy, the set of activities and work tasks of the ontology evolution process can be defined more precisely. In this section, we will present our taxonomy of ontology changes. This taxonomy will be used in phase two (*change identification*) of our process model.

In the past, very little work has been performed on creating a list of ontology changes. Flouris [24] defines ontology changes as "the problem of deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements".

Different researchers have tried to classify ontological changes from different aspects throughout the past. The main classifications defined in literature, deal with KAON [2], [40], [41] and OWL languages [4]. The ontology of KAON changes, classifies KAON changes through three levels of abstraction [2]:

- *Elementary changes* , applying modifications to one single ontology entity
- *Composite changes*, applying modifications to the direct neighborhood of an ontology entity
- *Complex changes*, applying modifications to an arbitrary subset of ontology entities

Also, two change types are considered in [2]:

- *Additive changes*, adding new entities to an ontology without changing the existing ones
- *Subtractive changes*, removing an entity

It being a complete and minimal set of changes, the mentioned change ontology does not include entity modifications. We believe that, there is a need of considering "*modificative*" changes, because that in most conditions renaming a concept does not cause any structural inconsistencies and are less costly than two consecutive subtractive and additive operations. In [4], the *modificative* ontology change operation is considered, but they did not investigate them with respect to the three levels of abstraction mentioned. In it, they have only classified them into basic and complex changes. So to help create a more general taxonomy, we in our research have combined both views in creating our taxonomy of ontology changes. We have also added some of the complex ontology evolution subtasks mentioned in [15], [24]. So, in our proposed taxonomy of ontology changes, we classify changes from both aspects. Also, we have classified the changes based on the elements that they change in the ontology (e.g. concept, taxonomic relations, non-taxonomic relations, axiom, and omain elements). We have discussed this classification of ontological elements in [42]. Our taxonomy of ontology changes can be seen in figures 7 – 8[a].

As it can be seen, in the presented figures the relations between low level change operations (e.g. add,…) and higher level change operations (e.g. merge, alignment,…) are presented. It should be noted that, we have considered some ontology change operations equal from an ontology evolution

[a] Due to limited publication space, an extended version of this paper with more detailed figures of the taxonomy of ontology changes can be accessed at : http://ceit.aut.ac.ir/~87231904/papers/ontoevol-extended.pdf

point of view. For example, the operations *alignmentation* and *matching* have been considered the same because both of them from an ontology evolution point of view perform the same operations; they both create relations, relating similar concepts between two ontologies. Such equal operations have been shown in the figures with an "*or"* between the equal concepts (e.g. *alignmentation* or *matching)*. In the following sections, we will define these ontology change operations in more detail.

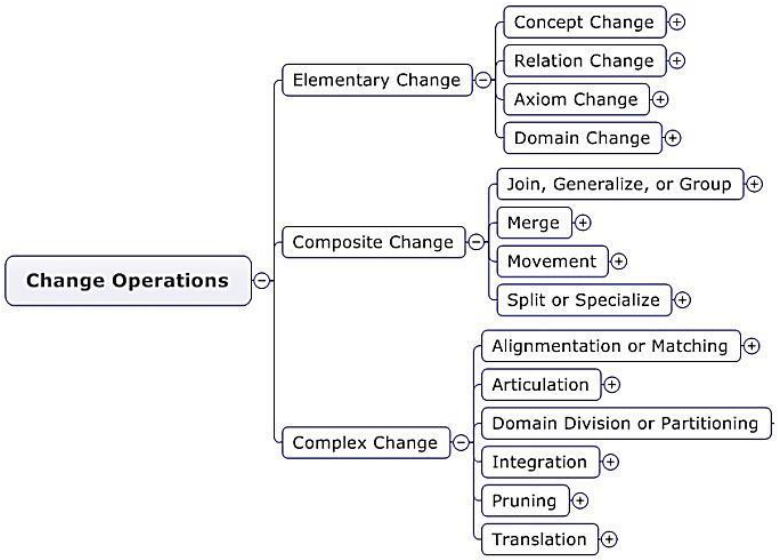


Figure 7 Our taxonomy of ontology changes - top levels

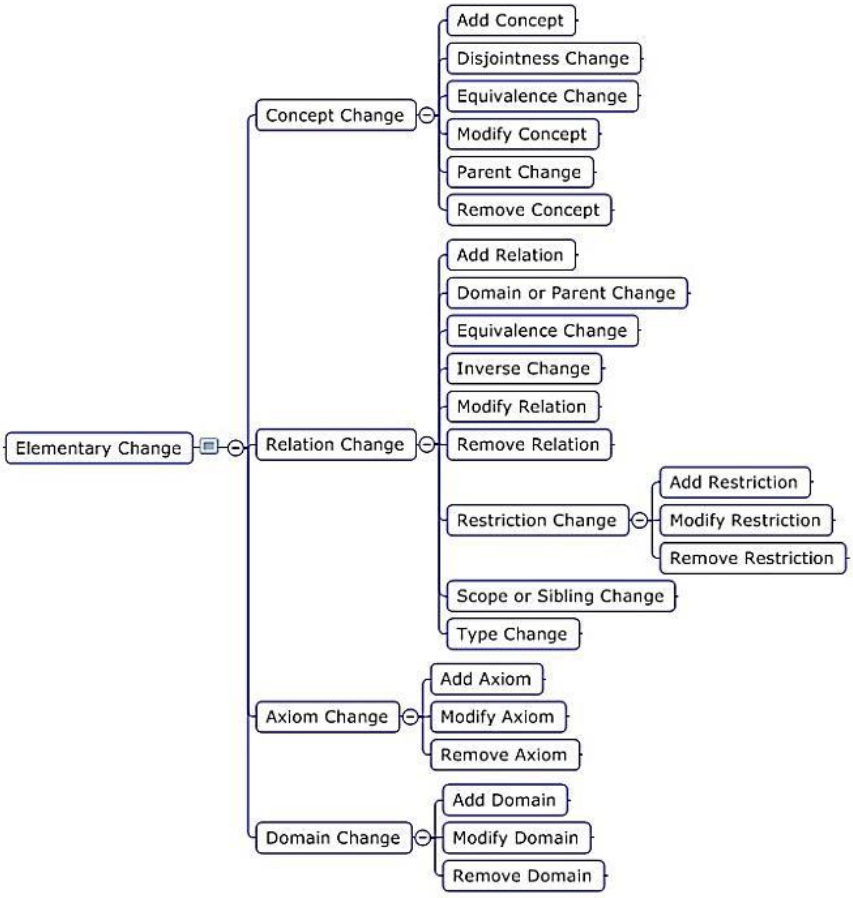
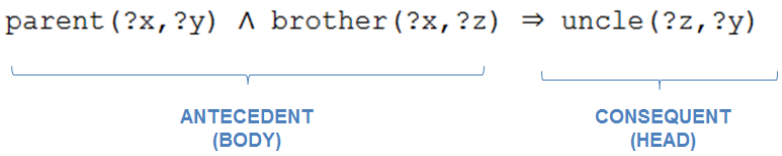


Figure 8 Our taxonomy of ontology changes - elementary changes section

3.2 Ontology Evolution Rules

In order to better understand and use the different change types discussed in section 3.1, we use the SWRL (Semantic Web Rule Language) standard [43] for defining the relations between the different levels of changes (elementary, composite, and complex changes). A SWRL rule consists of two parts:
1) *Antecedent* or *Body*
2) *Consequent* or *Head*
The *Antecedent* consists of a set of atoms (an atom is an expression featuring a set of arguments) that have to be true be able to substitute it with the *Consequent* section of the rule. An example of such a rule defining the *uncle* relation can be seen as the following:



For our ontology changes, for example we define "*joinConcepts*" as:

addConcept(?c3,?o), addSibling (?c1,?c3), addSibling (?c2,?c3) -> joinConcept(?c1,?c2,?c3)

which means that a concept join operation for two concepts *c1* and *c2* is defined as creating a new concept *c3* in ontology *o* and adding the two concepts *c1* and *c2* as its siblings. It should be stated here that, for simplicity and limited publication space, we have excluded the primitive element definitions from the left hand side of the SWRL rules e.g. *concept(?c3),ontology(?o),…* used in defining *joinConcepts*.

As another example, we define "*conceptMerge*" as follows:

addAllSiblingsOf(?c2,?c1),removeConcept(?c2) -> mergeConcepts(?c1,?c2)

which means that a concept merge operation for two elements *c1* and *c2* is defined as adding all the siblings of concept *c2* to concept *c1* and then removing the useless concept *c2* from the ontology.

As it can be seen in the examples presented, there is a delicate difference between the concept join and merge operations. In the former, a new concept is added to the ontology while in the latter, a concept is removed from the ontology. This slight difference is very important in ontology evolution because, if we would want to create an ontology which can be safely roll backed to an earlier version, we would only allow concept joins and not concept merges (because in the merge process, we remove and loose concepts which we can't rollback).

The mentioned SWRL rules can be seen as follows in Tables 2-4[b]:

Table 2 Some of the SWRL definitions of elementary ontology changes used in the taxonomy of ontology changes

Change Operation	SWRL Rule
Add Concept	addConcept(?c,?o)
Equivalence Change	equivalentConcepts(?c1,?c2)
Modify Concept	modifyConcept(?c1,?c2)
Sibling Change	addSibling(?c2,?c1)
Remove Concept	removeConcept(?c)
Add Relation	addRelation(?r,?c)

[b] Due to limited publication space, an extended version of this paper with more detailed SWRL definition tables of the ontology changes can be accessed at : http://ceit.aut.ac.ir/~87231904/papers/ontoevol-extended.pdf

Table 3 Some of the SWRL definitions of composite ontology changes used in the taxonomy of ontology changes

Change Operation	SWRL Rule
Concept Join	addConcept(?c3,?o)^addSibling(?c1,?c3)^addSibling(?c2,?c3)=>joinConcepts(?c1,?c2,?c3)
Axiom Join	addAxiom(?a,?o), modifyAxiom(?a,?a1), modifyAxiom(?a,?a2), removeAxiom(?a1,?o), removeAxiom(?a2,?o) -> joinAxioms(?a1,?a2)
Concept Merge	addAllSiblingsOf(?c2,?c1)^removeConcept(?c2)=>mergeConcepts(?c1,?c2)
Concept Split	addSibling(?c11,?c3)^addSibling(?c12,?c3)^addSomeSiblingsOf(?c1,?c11)^addOtherSiblingsOf(?c1,?c12)^addConcept(?c11,?o)^addConcept(?c12,?o)^removeConcept(?c1)=>splitConcept(?c1,?c3)
Relation Split	addRelation(?r12,?c)^addRelation(?r11,?c)^removeRelation(?r1,?c)=>splitRelation(?r1,?r11,?r12)

Table 4 Some of the SWRL definitions of complex ontology changes used in the taxonomy of ontology changes

Change Operation	SWRL Rule
Concept Alignment	equivalentConcepts(?co1,?co2)=>ConceptAlignment(?o1,?o2)
Concept Articulation	ConceptAlignment(?o1,?o2),ConceptAlignment(?o2,?o3)=>ConceptArticulation(?o1,?o3)
Relation Articulation	RelationAlignment(?r1,?r),RelationAlignment(?r,?r3)=>RelationArticulation(?r1,?r3)
Concept Integration	ConceptAlignment(?o1,?o2),mergeConcepts(?co1,?co2),moveConceptGroup(?co1,?co2),removeConcept(?co2,?o2)=>ConceptIntegration(?o1,?o2)
Concept Pruning	removeConcept(?c,?o)=>ConceptPruning(?c,?o)
Relation Pruning	removeRelation(?r,?c)=>RelationPruning(?r,?c)
Axiom Pruning	removeAxiom(?a,?o)=>AxiomPruning(?a,?o)
Concept Translation	modifyConcept(?c1,?c2)=>ConceptTranslation(?c1,?c2)

The following variables have been used in Tables 2-4 :

- *?c,?r,?a,?d,?o* represent any concept, relation, axiom, domain, ontology respectively.
- $?c_i$ represents any concept, where c_i is the concept that changes and there are more than one concept in the rule and i can be any natural positive non-zero number.
- $?c_{ij}$ represents any concept, where c_i is the concept that is split and the resulting concept are c_{ij} and c_{ij+1}. i and j can be any natural positive non-zero number.
- $?r_i$ represents any relation, where r_i is the relation that changes and there are more than one relation in the rule and i can be any natural positive non-zero number.
- $?r_{ij}$ represents any relation, where r_i is the relation that is split and the resulting relation are r_{ij} and r_{ij+1}. i and j can be any natural positive non-zero number.

- *?a_i* represent any axiom, where a_i is the axiom that changes and there are more than one axioms in the rule and *i* can be any natural positive non-zero number.
- *?a_{ij}* represents any axiom, where a_i is the axiom that is split and the resulting axioms are a_{ij} and a_{ij+1}. *i* and *j* can be any natural positive non-zero number.
- *?d_i* represents any domain, where d_i is the domain that changes and there are more than one domain in the rule and *i* can be any natural positive non-zero number.
- *?d_{ij}* represents any domain, where d_i is the domain that is split and the resulting domain are d_{ij} and d_{ij+1}. *i* and *j* can be any natural positive non-zero number.
- *?o_i* represents any ontology, where o_i is the ontology that changes and there are more than one ontology in the rule and *i* can be any natural positive non-zero number.
- *?o_{ij}* represents any ontology, where o_i is the ontology that is split and the resulting ontology are o_{ij} and o_{ij+1}. *i* and *j* can be any natural positive non-zero number.
- *?co_i,?ro_i,?ao_i,?do_i* represent any concept, relation, axiom, domain of an ontology *o* respectively and *i* can be any natural positive non-zero number.

Table 5 Formal context representation of composite change operations

Objects\Attributes	addConcept(?c,?o)	removeConcept(?c,?o)	addDomain(?d,?o)	removeDomain(?d,?o)	addAxiom(?a,?o)	removeAxiom(?a,?o)	disjointConcepts(?c1,?c2)	equivalentConcepts(?c1,?c2)	parentConcept(?c1,?c2)	siblingConcept(?c2,?c1)	equivalentRelation(?r1,?r2)	inverseRelation(?r1,?r2)	addRelation(?r,?c)	removeRelation(?r,?c)	parentRelation(?r,?c)	siblingRelation(?r,?c)	addRestriction(?r,?c)	removeRestriction(?r,?c)	modifyConcept(?c1,?c2)	modifyAxiom(?a1,?a2)	modifyDomain(?d1,?d2)
Concept Join	×									×											
Axiom Join					×	×														×	
Concept Merge		×								×											
Relation Merge													×	×							
Concept Group Move										×				×	×	×					
Concept Move Up									×	×				×							
Concept Move Down									×	×				×							
Concept Split	×	×								×											
Relation Split													×	×							
Axiom Split					×	×															
modifyRelation													×	×							
modifyRestriction																	×	×			

Table 6 Formal context representation of complex change operations

Objects\Attributes	Complex Changes: Concept Alignment	Relation Alignment	Axiom Alignment	Domain Alignment
Concept Alignment				
Relation Alignment				
Axiom Alignment				
Domain Alignment				
Concept Articulation	x			
Relation Articulation		x		
Axiom Articulation			x	
Domain Articulation				x
Domain Division				
Concept Integration	x			
Relation Integration		x		
Axiom Integration			x	
Domain Integration				x
Concept Pruning				
Relation Pruning				
Axiom Pruning				
Concept Translation				
Relation Translation				
Axiom Translation				
Domain Translation				

Objects\Attributes	Composite Changes: joinConcepts(?c1,?c2)	joinAxioms(?a1,?a2)	mergeConcepts(?c1,?c2)	mergeRelation(?r1,?r2)	moveConceptGroup(?c2,?c3)	moveConceptUp(?c3,?c2)	moveConceptDown(?c2,?c3)	splitConcept(?c1,?c2)	splitRelation(?r1,?c)	splitAxiom(?a1,?o)	modifyRelation(?r1,?r2)	modifyRestriction(?r1,?r2)
Concept Alignment												
Relation Alignment												
Axiom Alignment												
Domain Alignment												
Concept Articulation												
Relation Articulation												
Axiom Articulation												
Domain Articulation												
Domain Division					x							
Concept Integration			x		x							
Relation Integration				x								
Axiom Integration		x										
Domain Integration												
Concept Pruning												
Relation Pruning												
Axiom Pruning												
Concept Translation												
Relation Translation											x	
Axiom Translation												
Domain Translation												

Objects\Attributes	Elementary Changes: addConcept(?c,?o)	removeConcept(?c,?o)	addDomain(?d,?o)	removeDomain(?d,?o)	addAxiom(?a,?o)	removeAxiom(?a,?o)	disjointConcepts(?c1,?c2)	equivalentConcepts(?c1,?c2)	parentConcept(?c1,?c2)	siblingConcept(?c2,?c1)	equivalentRelation(?r1,?r2)	inverseRelation(?r1,?r2)	addRelation(?r,?c)	removeRelation(?r,?c)	parentRelation(?r,?c)	siblingRelation(?r,?c)	addRestriction(?r,?c)	removeRestriction(?r,?c)	modifyConcept(?c1,?c2)	modifyAxiom(?a1,?a2)	modifyDomain(?d1,?d2)	equivalentAxiom(?a1,?a2)	equivalentDomain(?d1,?d2)
Concept Alignment								x															
Relation Alignment											x		x										
Axiom Alignment					x																	x	
Domain Alignment					x																		x
Concept Articulation																							
Relation Articulation																							
Axiom Articulation																							
Domain Articulation																							
Domain Division			x	x																	x		
Concept Integration		x																					
Relation Integration													x	x									
Axiom Integration					x	x																	
Domain Integration			x	x																			
Concept Pruning		x																					
Relation Pruning														x									
Axiom Pruning						x																	
Concept Translation																			x				
Relation Translation																							
Axiom Translation																				x			
Domain Translation																					x		

By defining the change operation, the relation between the low level changes and higher level changes can be seen clearly. It should also be noted here that, in some of the complex changes because the SWRL language did not contain the "forAll" or "someOf" expressions, they are not shown in the rules. For example, in *concept pruning* the "someOf" relation is needed to define that some of the concepts are being removed. Also, in *concept translation* the "forAll" relation is needed to show that in translation of an ontology, all concepts have to be modified to the destination language. We have fixed this lack of expressiveness in our implementation of OntoEvol. It should be noted that, there is a hidden semantic hierarchical relation between the three types of changes (elementary, composite, and complex changes). Which is, some elementary change types are used to define composite changes and some elementary and composite changes are used to define complex changes. In other words, elementary changes are primitive elements in defining composite changes and both of them are primitive elements in defining the complex relations. For example in defining the "*moveConceptUp*" composite change, we have included the "*siblingRelation*", "*removeRelation*", and "*modifyRelation*" elementary changes. This relation helps us in clustering the elementary changes into higher composite and complex changes. These relations can be better seen in tables 5 and 6.

3.2.1 Ontology Change Operations and FCA

In this section, in order to better clarify the relations between the different ontology change operations and better analysis of the relations, we use FCA representation techniques [44] to represent the

ontology change operations. Formal Concept Analysis (FCA) is a formal method for data analysis and knowledge representation. FCA can help to identify binary relationships between the data. The relationship is used to form a formal context according to a formal concept lattice. The formal context tables for composite and complex changes can be seen in tables 5 and 6 respectively. Also, the formal context table and the formal concept lattice of each are presented in figures 9 and 10 respectively.

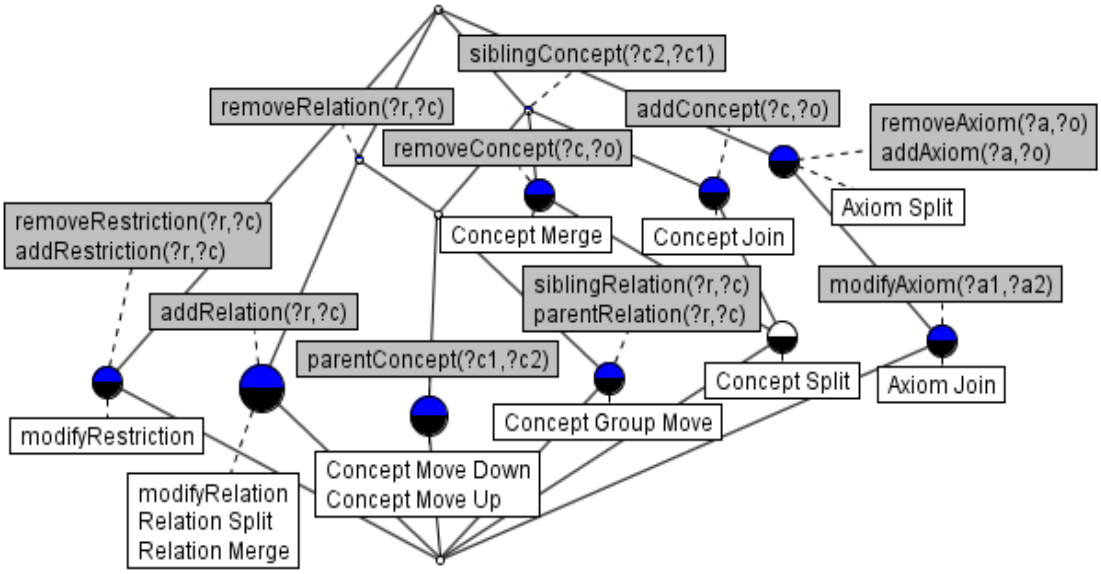


Figure 9 Formal concept lattice representation of composite change operations

As it can be seen, the formal concept lattice shows the relations between the higher and lower level change operations even more clearly. The intention of this section was to illustrate and elaborate the relations between the higher and lower level change operations. These relations will be used in the *change analysis* phase of the evolution process model presented in section 3.3. To help facilitate better understanding of the changes that are taking place in the evolution process. For example, when we see a sequence of *addConcept* and *addSibling* operations, we can deduce that it is the *joinConcepts* operation. In other words, the evolution that is happening is the joining of two or more concepts. The essence of knowing the higher level operations is that, it helps the ontologist determine possible errors or mistakes made in user ontology evolution. For example, if after a *joinConcept* evolution operation we see orphan nodes, we are sure that something has gone wrong. For another example, when we see that the complex *conceptTranslation* operation is being performed and at the end, some of the concepts have not been translated, then we can inform the user that he/she may have forgotten those unchanged concepts therefore providing better management of the ontology evolution process. In the next section, we will present our ontology evolution system (OntoEvol) which uses the above mentioned rules, taxonomy, and proposed process model to help facilitate ontology evolution.

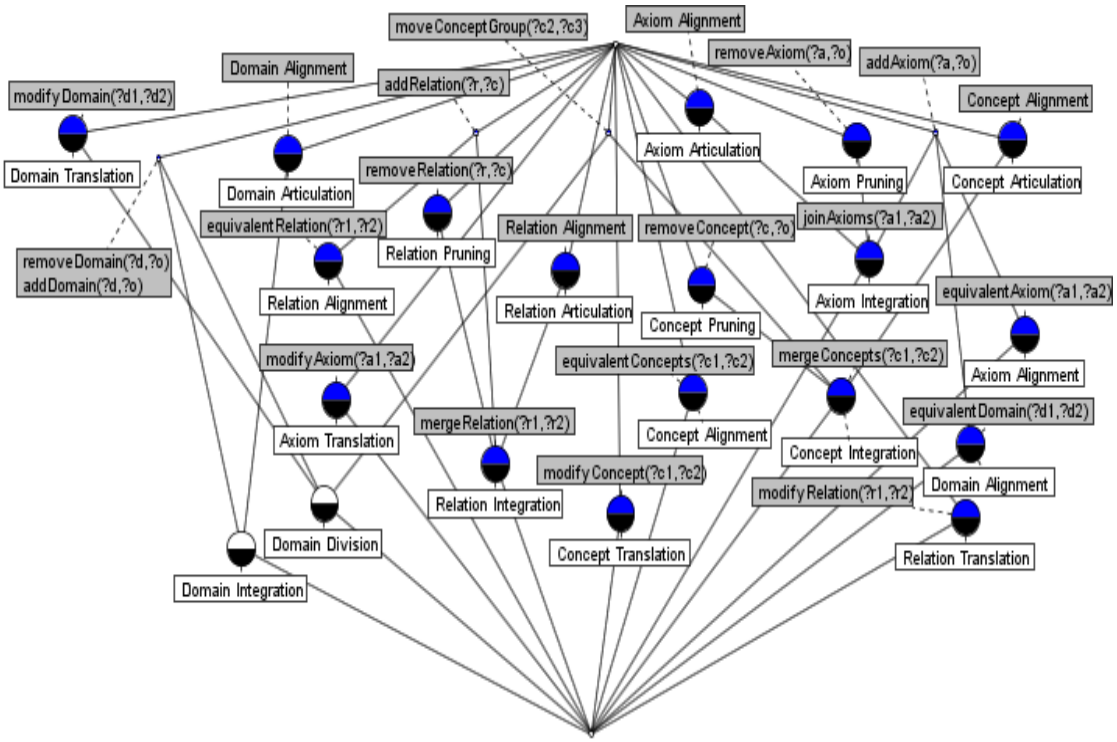


Figure 10 Formal concept lattice representation of complex change operations

3.3 OntoEvol – An Ontology Evolution System

Our OntoEvol system facilitates the ontology evolution process with the help of ontology change rules. These change rules are defined via the SWRL rule representation language which was presented in section 3.2. This new approach to ontology evolution helps the ontology evolution process by preserving the consistency of the ontology throughout the ontology evolution process. The reason for this is that, only changes that preserve the consistency are performed on the ontology (*change resolution* phase). Therefore, the consistency of the ontology is preserved. In this section, we will present our OntoEvol system which implements our process model and uses our presented taxonomy and change operation rules. The overall architecture of the OntoEvol system can be seen in figure 11. In the following subsections, we will present each of our ontology process phases as they are implemented.

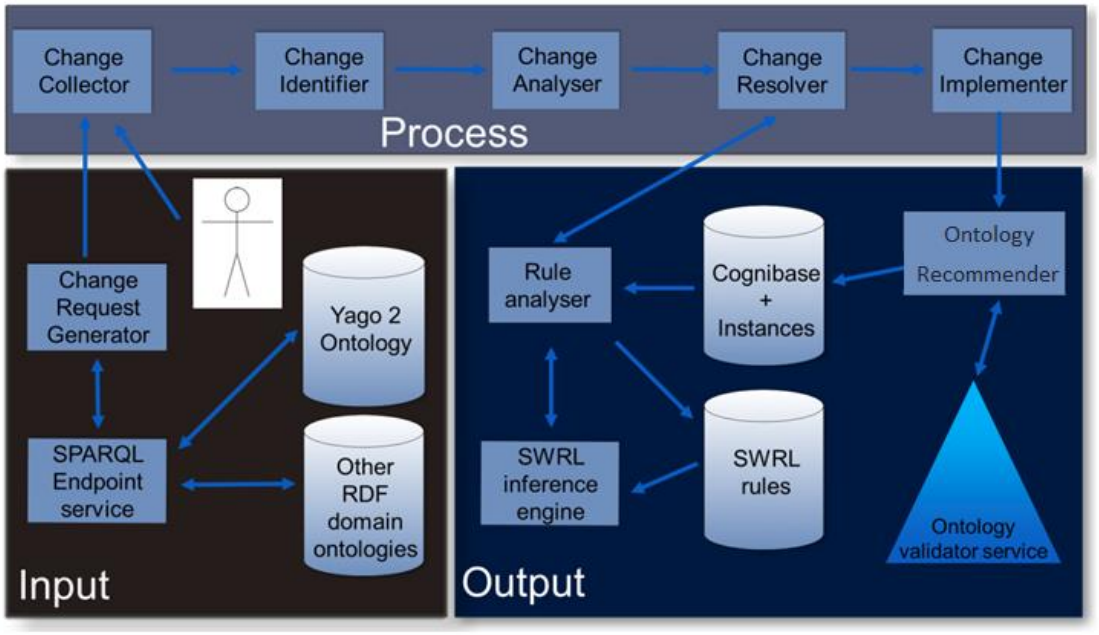


Figure 11 Overall architecture of the OntoEvol system

3.3.1 Change Collection Phase

In this phase, requested changes from different sources are gathered. We have provided interfaces for both user and online resources. User changes are the main changes that are to be performed. The online resources are used to provide suggestions to the user in case of conflict. For querying the online resources, the SPARQL query language and its associated endpoints are used. Some of these endpoints are Yago2 [45], [46], Wikipedia [47], DBpedia [48], etc. A complete list of such endpoints can be seen in [49]. These query results will be presented to the user as recommendations in the change recommendation phase which we will discuss later on. All inputs are stored in triple format.

3.3.2 Change Identification Phase

In this phase, the user change requests are evaluated and classified with respect to our presented taxonomy of ontology changes presented in section 3.1. This is done to ensure that no unidentified changes are injected into the ontologies. This prevents the ontology from going into an inconsistent state therefore needing repair.

3.3.3 Change Analysis Phase

In this phase, the changes are analyzed further to specify if it is possible to convert them into their equivalent higher level changes with the help of our presented SWRL rules which were presented in section 3.2. The reason for this is that, it helps us better understand what the user is planning to

perform on the ontology. Also, if the ontology changes do not represent a higher level change, the closest higher level change is recommended to the user and the additional changes that have to be performed are presented to the user to be accepted. This helps greatly in keeping the ontology consistent. The reason for this is that if a group of changes do not match a higher level change, it shows that there are some changes missing or forgotten by the user. Therefore, if those missing changes are appended to the initial changes then the higher level change is completed. These missing changes can be recommended to the user in the change recommendation phase. In this phase, the changes from the online resources mentioned in section 3.3.1 can be used to provide further recommendations to the user.

3.3.4 Change Resolution Phase

In this phase, we decide which of the proposed change operations from previous phase are to be performed. The reason for this is that, in some cases, the changes to be done can be reduced by combining them into higher level changes. This increase in granularity provides better understanding and maintenance during the ontology evolution process. Also, this helps in finding the mistakes the user may have done. The changes to be performed may be approved by the user. For example, a user may have forgotten to add one of the *subClassOf* relations in a concept join operation, which results in an inconsistent ontology, by checking them with the SWRL rules then such a mistake can be resolved and the ontology is consistent again. In other words, in this phase, we choose the set of changes to be performed that preserve consistency. Therefore at the end of this phase, one specific list of changes to be performed is created for implementation in the next phase.

3.3.5 Change Implementation Phase

In this phase, we perform the changes on the ontology. For accomplishing this task, we find the equivalents of each change with its respective OWL axioms and add them to the updated ontology. This is done by a simple matching table which matches the ontology change requests to their OWL equivalents.

3.3.6 Change Recommendation Phase

In this phase, the updated ontology is evaluated for common pitfalls [36], [50], anti-patterns [37], and compliance with ontology design patterns [34], [38]. If any issues are found, they are reported to the user as recommendations to decide on what to do with them. If the user decides to apply them to the ontology, he/she can create a new set of ontology changes to be applied to the ontology and sends them to the change collection phase. Also, a list of similar concepts and relations collected in the *Change collection phase* are presented to the user so that they can choose to add those relations to the new ontology.

4 Evaluation and Future Work

For testing our process model and the OntoEvol system, we performed the following tasks:

1. We used an initial university ontology, which we created by hand as the input ontology to be evolved.

2. To create user changes, we created a questionnaire about what students believe a university is and handed those questionnaire to over 100 computer science and computer engineering undergraduate students which had passed software engineering and data modeling courses to fill out.

3. The questionnaires were read and the information to be added to the ontology was extracted by hand in form of triples. These additions were used as user ontology change requests. The reason for this was, since our OntoEvol system was in its early steps and the user input interface had not been completed, we could not receive user changes in an interactive fashion.

4. The initial ontology and the user changes were given to the system as input and the resulting ontology was created in the implementation phase.

Table 7 number of elementary requested changes

Elementary Change	Number of requested changes
Add Concept	240
Disjointness Change	-
Equivalence Change	-
Modify Concept	30
Parent Change	3
Sibling Change	119
Remove Concept	557
Add Relation	198
Domain or Parent Change	3
Equivalence Change	-
Inverse Change	-
Modify Relation	629
Remove Relation	260
Add Restriction	-
Modify Restriction	-
Remove Restriction	-
Scope or Sibling Change	119
Type Change	-
Add Axiom	7
Modify Axiom	1
Remove Axiom	2
Add Domain	-
Modify Domain	-
Remove Domain	-

For our test case, in the *change collection* phase, we applied the 2100 changes created to our initial university ontology which came from the users' change requests regarding the university ontology mentioned above. In the *change identification* phase, they were compared with our change types and were filtered into three sets of changes: 1)538 additive changes 2)848 subtractive changes 3)660 modificative changes. Also, 54 changes were removed due to being duplicates. Then, they were compared in relation with the type of ontological element they effected (concept, relation (taxonomic and non-taxonomic), axioms, and domain elements) resulting in 827 concept (240 additive and 557 subtractive, and 30 modificative concept changes), 1209 relation (122 taxonomic (53 additive and 60 subtractive relation changes) and 1087 non-taxonomic relation (198 additive, 260 subtractive, and 629 modificative relation changes) changes), 10 axiom (7 additive, 2 subtractive, and 1 modificative axiom changes) and 0 domain changes. The reason that there were no domain changes was that, the focus of the ontology evolution was on the university domain therefore, there were no domain related changes. And, at last they were sorted in relation to the ontology changes taxonomy elements (shown in Table 7).

After evaluating the elementary changes mentioned in table 7, in the *change analysis* phase, the changes were evaluated with respect to the SWRL change rules. From the 30 composite and complex changes, with the elementary changes requested, only 16 rules were candidate rules (the other rules could not be chosen because, one or more of the atoms in the left hand side or antecedent were not available). From those candidate rules, the changes were classified into the 14 composite and complex changes which can be seen in table 8.

Table 8 number of composite and complex requested changes

Composite/Complex Changes	Number of requested changes
Concept Join	38
Axiom Join	-
Concept Merge	2
Relation Merge	62
Concept Group Move	16
Concept Move Up	17
Concept Move Down	30
Concept Split	41
Relation Split	68
Axiom Split	2
Concept Pruning*	512
Relation Pruning	5
Axiom Pruning	-
Concept Translation	30
Relation Translation*	561
Axiom Translation	1

Also, the rules used for matching the two complex changes *concept pruning* and *relation translation* (shown with a *) were very sensitive (by sensitive we mean that, the shortest possible match was accepted) therefore, in the *concept pruning* rule, every *removeConcept* that didn't match the other rules was accepted as a *concept pruning,* amounting in 512 *concept prunings.* The same is for *relation translation*. Therefore, those two amounts can be reduced if a greedier heuristic is chosen. We didn't investigate using the longest possible match, but we intend to perform this in the future.

The following 3 *addAxiom* and 6 *addSibling* changes were not matched:

- addAxiom(Professor"teach"course@Department)
- addAxiom(Student"eat"lunch@Cafeteria)
- addAxiom(GradStudent"publish"ConferencePaper@Conference)
- addSibling (PhdStudent, GradStudent)
- addSibiling(Pizza, Food)
- addSibiling(Hamburger, Sandwich)
- addSibiling(ATM, Bank)
- addSibiling(ConferencePaper, Publications)
- addSibiling(GradCourse, Course)

In the changes above, we have used the format Professor"teach"course@Department (representing Subject1"Relation"Object@Subject2) to represent the axiomatic fact "A Professor teaches a Course at the Department". When taking a closer look at why they had not been matched, we saw that all these changes were leaf nodes in our ontology therefore they did not have the appropriate atoms needed for a complete composite or complex change rule. One of the main points of ontology evolution errors is at the leaf nodes in an ontology. These leaf nodes have the potential to be entered and later forgotten to be populated in the future. This would cause the ontology knowledge to be sparse in some parts and crowded in other sections, therefore causing inconsistent coverage of the domain. It should be noted here, the fact that they were not matched is not a bad thing. It only shows that these changes did not constitute a higher level change and they are elementary changes, but it helps the ontology engineer to notice those changes and act accordingly to populate those sections.

From the results achieved, in the *change resolution* phase all the changes in table 8 were implemented into the ontology and only the 9 unmatched changes were not implemented (*change implementation* phase). They could have been implemented, but we wanted to make sure that they were not mistakenly entered into the ontology or if the user could complete them so that they matched a higher level change. In the *change recommendation* phase, those changes were reported to the user for conformation to be added.

By looking at table 8, an ontology engineer can clearly see what has been going on in the ontology evolution process. It can be seen in figure 12 that, the main changes performed on the ontology were relation translation (40.5%), concept pruning (37%), relation split (4.9%), and relation merge (4.5%). This interprets that, the users were mainly involved with removing concepts and renaming and fixing relations used in the ontology. If the main purpose of the evolution was not the mentioned, the ontology engineer can carefully evaluate the requested changes the users performed in order to find the reason of the mistakes. This can be very helpful in situations where the users have an interactive ontology editor (e.g. Protégé ontology editor) which, the users can use a mouse and drag

and drop concepts. In such cases, some users may accidentally drag a concept under another concept therefore, changing a great amount of underlying subclass relations (*ConceptGroupMove*), so if the ontology engineer sees a spike in *ConceptGroupMove* changes, but he is anticipating a spike in *ConceptJoin* changes, he can detect there has been a mistake.

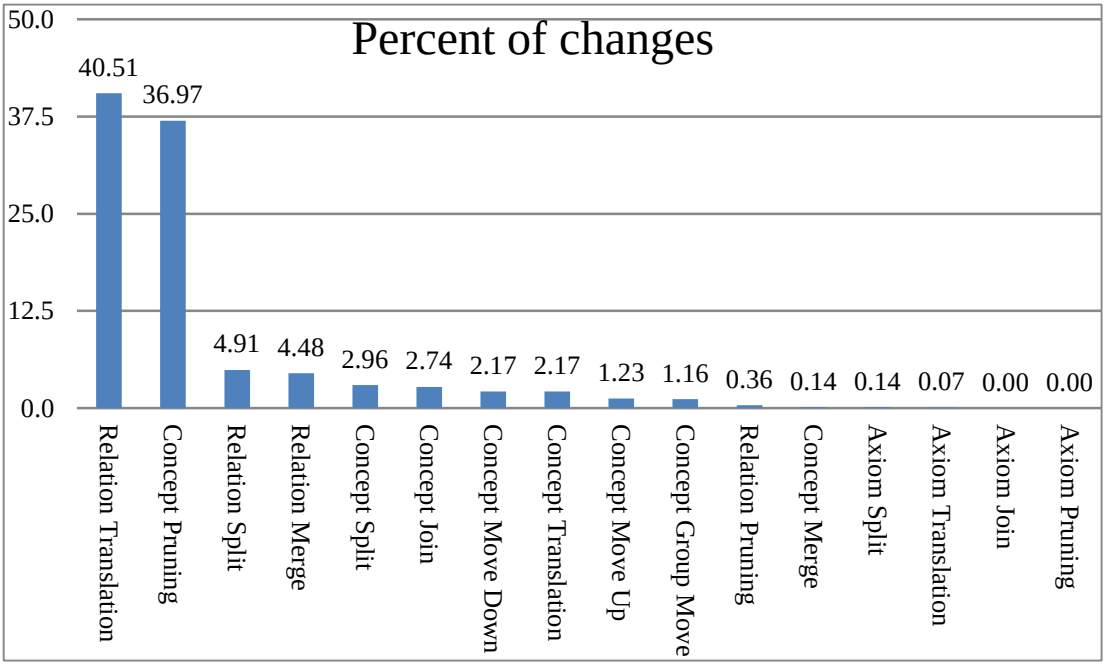


Figure 12 percentage of changes requested

It should be noted here that, our presented methodology has shown better performance than the other methodologies mentioned in section 2 by reducing the search domain needed for finding mistakes and inconsistencies. This is achieved by analyzing the changes before implementing them in the ontology. Therefore, we only need to search the list of proposed changes rather than searching the whole resulting ontology. The reason we chose the aforementioned process for testing our system was that, we could not located a standard or gold standard ontology which had user changes in it. In other words, no set of changes and resulting ontologies were found to be used as a basis for testing our results, so we had to create one by the help of the mentioned students. All the ontologies available on the Web were final versions and none of them had specifications or details about their changes from previous versions. Only a few had very general descriptions of change done, for example in DBpedia[48], they stated that from version 3.7 to 3.8, they "Cleaned up handling of URIs / IRIs" which didn't give us much information on what the exact changes may have been [51]. As future work, first we intend to complete and release the OntoEvol system as a plugin in the NeOn framework so that it can be used and tested further. Second, we intend to extend our testbed and create a standard testbed for ontology evolution.

References

[1] L. Stojanovic, N. Stojanovic, and R. Volz, "Migrating data-intensive web sites into the semantic web," in *Proceedings of the 2002 ACM symposium on Applied computing*, 2002, pp. 1100–1107.

[2] L. Stojanovic, "Methods and tools for ontology evolution," Ph.D. thesis, University of Karlsruhe, 2004.

[3] P. Haase, H. Lewen, R. Studer, D. T. Tran, M. Erdmann, M. d' Aquin, and E. Motta, "The neon ontology engineering toolkit," *WWW*, 2008.

[4] M. C. A. Klein, "Change management for distributed ontologies," Ph.D. thesis, Vrije Universiteit Amsterdam, 2004.

[5] F. Zablith, "Harvesting Online Ontologies for Ontology Evolution," Ph.D. thesis, The Open University, UK, 2011.

[6] R. Djedidi and M. A. Aufaure, "ONTO-EVOAL an Ontology Evolution Approach Guided by Pattern Modeling and Quality Evaluation," in *Foundations of Information and Knowledge Systems (FoIKS 2010)*, 2010, pp. 286–305.

[7] G. Flouris and D. Plexousakis, "Handling Ontology Change: Survey and Proposal for a Future Research Direction," *Inst. Comput. Sci. FORTH Greece Tech. Rep. TR-362 FORTH-ICS*, 2005.

[8] F. Zablith, M. Sabou, M. d' Aquin, and E. Motta, "Ontology evolution with Evolva," in *The Semantic Web: Research and Applications*, Springer, 2009, pp. 908–912.

[9] F. Zablith, "Evolva: A comprehensive approach to ontology evolution," in *The Semantic Web: Research and Applications*, Springer, 2009, pp. 944–948.

[10] R. Djedidi and M. A. Aufaure, "Ontology Evolution: State of the Art and Future Directions," in *Ontology Theory, Management and Design: Advanced Tools and Models*, IGI Global Publishing, 2010, pp. 179–207.

[11] W. Jaziri, "A methodology for ontology evolution and versioning," in *Advances in Semantic Processing, 2009. SEMAPRO'09. Third International Conference on*, 2009, pp. 15–21.

[12] P. Cimiano and J. Völker, "A framework for ontology learning and data-driven change discovery," in *Natural language processing and information systems: 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, Alicante, Spain, June 15-17, 2005: proceedings*, 2005, pp. 227–238.

[13] L. Stojanovic, N. Stojanovic, J. Gonzalez, and R. Studer, "OntoManager–a system for the usage-based ontology management," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Springer, 2003, pp. 858–875.

[14] S. Bloehdorn, P. Haase, Y. Sure, and J. Voelker, "Ontology evolution," *Semantic Web Technol. Trends Res. Ontol.-Based Syst.*, pp. 51–70, 2006.

[15] G. Flouris, "On belief change in ontology evolution," *AI Commun.*, vol. 19, no. 4, pp. 395–397, 2006.

[16] P. Plessers, O. De Troyer, and S. Casteleyn, "Understanding ontology evolution: A change detection approach," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 5, no. 1, pp. 39–49, 2007.

[17] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov, "Ontology versioning and change detection on the web," in *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, Springer, 2002, pp. 197–212.

[18] N. F. Noy, S. Kunnatur, M. Klein, and M. A. Musen, "Tracking changes during ontology evolution," in *The Semantic Web–ISWC 2004*, Springer, 2004, pp. 259–273.

[19] J. Eder and K. Wiggisser, "Change detection in ontologies using DAG comparison," in *Advanced Information Systems Engineering*, 2007, pp. 21–35.

[20] H.-A. Santoso, S.-C. Haw, and C.-S. Lee, "Change detection in ontology versioning: a bottom-up approach by incorporating ontology metadata vocabulary," in *Database Theory and Application, Bio-Science and Bio-Technology*, Springer, 2010, pp. 37–46.
[21] A. Abdollahzadeh Barforoush and A. Rahnama, "Ontology Learning: Revisited," *J. Web Eng. JWE*, vol. 11, no. 4, pp. 269–289, Dec. 2012.
[22] D. Benz, "Collaborative ontology learning," Master's thesis, University of Freiburg, 2007.
[23] G. Flouris, Z. Huang, J. Z. Pan, D. Plexousakis, and H. Wache, "Inconsistencies, negations and changes in ontologies," in *Proceedings of the National Conference on Artificial Intelligence*, 2006, vol. 21, p. 1295.
[24] G. Flouris, D. Plexousakis, and G. Antoniou, "A classification of ontology change," in *Poster Proceedings of the 3rd Italian Semantic Web Workshop, Semantic Web Applications and Perspectives (SWAP-06)*, 2006.
[25] L. Stojanovic, A. Maedche, N. Stojanovic, and R. Studer, "Ontology evolution as reconfiguration-design problem solving," in *Proceedings of the 2nd international conference on Knowledge capture*, 2003, pp. 162–171.
[26] P. Haase and L. Stojanovic, "Consistent evolution of owl ontologies," presented at the ESWC 2005, 2005, vol. 3532, pp. 182–197.
[27] P. Haase and J. Völker, "Ontology learning and reasoning—dealing with uncertainty and inconsistency," *Uncertain. Reason. Semantic Web I*, pp. 366–384, 2008.
[28] A. Mikroyannidis and B. Theodoulidis, "Ontology management and evolution for business intelligence," *Int. J. Inf. Manag.*, vol. 30, no. 6, pp. 559–566, 2010.
[29] M. C. Klein and D. Fensel, "Ontology versioning on the Semantic Web.," in *SWWS*, 2001, pp. 75–91.
[30] M. Javed, M. Abgaz, and C. Pahl, "A layered framework for pattern-based ontology evolution," presented at the 3rd International Workshop on Ontology-Driven Information System Engineering (ODISE), 2011.
[31] M. Javed, Y. Abgaz, and C. Pahl, "A pattern-based framework of change operators for ontology evolution," in *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, 2009, pp. 544–553.
[32] A. Auger and C. Barrière, "Pattern-based approaches to semantic relation extraction: A state-of-the-art," *Terminol. Int. J. Theor. Appl. Issues Spec. Commun.*, vol. 14, no. 1, pp. 1–19, 2008.
[33] E. Blomqvist, A. Gangemi, and V. Presutti, "Experiments on pattern-based ontology design," in *Proceedings of the fifth international conference on Knowledge capture*, 2009, pp. 41–48.
[34] E. Blomqvist, "OntoCase - A Pattern-Based Ontology Construction Approach," *Move Meaningful Internet Syst. 2007 CoopIS DOA ODBASE GADA IS*, vol. 4803, pp. 971–988, 2007.
[35] R. Djedidi, M. A. Aufaure, R. Qi, V. Letort, M. Kang, P. Cournède, P. de Reffye, T. Fourcaud, R. Neji, A. Besbes, and others, "Change Management Patterns (CMP) for Ontology Evolution Process," in *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD 2009) in ISWC*, 2009.
[36] M. Poveda, M. Suárez-Figueroa, and A. Gomez-Perez, "Common pitfalls in ontology development," in *Current Topics in Artificial Intelligence, , CAEPIA 2009 Selected Papers*, 2010, vol. 5988, pp. 91–100.
[37] C. Roussey, O. Corcho, and L. M. Vilches-Blázquez, "A catalogue of OWL ontology antipatterns," in *Proceedings of the fifth international conference on Knowledge capture*, 2009, pp. 205–206.
[38] A. Gangemi and V. Presutti, "Ontology design patterns," *Handb. Ontol.*, pp. 221–243, 2009.
[39] A. Rahnama and A. Abdollahzadeh Barforoush, "Taxonomy of Ontology Changes - from an Ontology Evolution Perspective," presented at the 19th National CSI Computer Conference, Tehran, Iran, 2014, pp. 1138–1143.

[40] E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, and L. Stojanovic, "KAON—Towards a large scale semantic web," in *E-Commerce and Web Technologies*, Springer, 2002, pp. 304–313.

[41] B. Motik and R. Studer, "KAON2–A Scalable Reasoning Tool for the Semantic Web," in *Proceedings of the 2nd European Semantic Web Conference (ESWC'05), Heraklion, Greece*, 2005.

[42] A. Rahnama and A. Abdollahzadeh Barforoush, "Cognibase: a new representation model to support ontology development," in *IADIS International Conference Information Systems (IS 2011)*, Avila, Spain, 2011.

[43] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Memb. Submiss.*, vol. 21, p. 79, 2004.

[44] B. Ganter, G. Stumme, and R. Wille, *Formal Concept Analysis: foundations and applications*, vol. 3626. springer, 2005.

[45] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 6, no. 3, pp. 203–217, 2008.

[46] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia," *Artif. Intell.*, vol. 194, pp. 28–61, 2013.

[47] "Wikipedia." [Online]. Available: http://www.wikipedia.org/. [Accessed: 22-Aug-2013].

[48] "DBpedia." [Online]. Available: http://dbpedia.org/About. [Accessed: 22-Aug-2013].

[49] "SparqlEndpoints - W3C Wiki." [Online]. Available: http://www.w3.org/wiki/SparqlEndpoints. [Accessed: 22-Aug-2013].

[50] "OOPS! - OntOlogy Pitfall Scanner! - Catalogue," *OOPS! - OntOlogy Pitfall Scanner! - Catalogue*, 2013. [Online]. Available: http://oeg-lia3.dia.fi.upm.es/oops/catalogue.jsp.

[51] "DBpedia Changelog." [Online]. Available: http://wiki.dbpedia.org/Changelog. [Accessed: 22-Aug-2013].

[52] T. M. Akinsola, "Automated Ontology Evolution," Masters of Science informatics thesis, University of Edinburgh, Scotland (2008).