Analyzing Impacts of Change Operations in Evolving Ontologies

Yalemisew M. Abgaz[1], Muhammad Javed[2], Claus Pahl[3]

Centre for Next Generation Localization (CNGL),
School of Computing, Dublin City University, Dublin 9, Ireland
{yabgaz[1]|mjaved[2]|cpahl[3]}@computing.dcu.ie

Abstract. Ontologies evolve over time to adapt to the dynamically changing knowledge in a domain. The evolution includes addition of new entities and modification or deletion of obsolete entities. These changes could have impacts on the remaining entities and dependent systems of the ontology. In this paper, we address the impacts of changes prior to their permanent implementation. To this end, we identify possible structural and semantic impacts and propose a bottom-up change impact analysis method which contains two phases. The first phase focuses on analyzing impacts of atomic change operations and the second phase focuses on analyzing impacts of composite changes which include impact cancellation, balancing and transformation due to implementation of two or more atomic changes. This method provides crucial information on the impacts and could be used for selecting evolution strategies and conducting what-if analysis before evolving the ontologies.

Keywords: Ontology evolution, change impact analysis, structural impacts, semantic impacts.

1 Introduction

Semantic web applications widely use ontologies to formally represent and explicitly specify the domain of discourse. Semantic web applications annotate content using ontologies and make inference using reasoners [1] [2] [3]. In various domains, ontologies serve as sources of semantics, which contain commonly agreed meaning of concepts and relationships among concepts [7][8]. Despite the agreement, neither the domain knowledge nor the ontologies are permanent. The dynamics in a domain causes ontologies to evolve frequently over time [4].

Ontologies evolve whenever there is a change in the specification, representation or conceptualization of the domain [4]. These changes are additions of new concepts, modifications of existing ones or removal of obsolete or erroneous representations. The changing entities are concepts, properties, instances and axioms. When ontologies evolve, a change of one entity may cause many unseen and undesired impacts on dependent entities [4] [9]. It is arduous and time consuming to manually catch these impacts. However, they later cause inconsistencies in the ontology [5] [6]. Thus, before permanent implementation of the changes, it is vital to conduct change impact analysis to understand which structural and semantic impacts occur and which entities are affected [10].

Despite the importance of impact analysis in evolving ontologies, only a few approaches proposed. In [11], we proposed a framework for change impact analysis, focussing on empirical observation to understand and define necessary components of the framework. A six-phase evolution framework is widely used to evolve ontologies [12] [13] [14]. The semantics of change phase deals with derived change operations to consistently evolve the ontology. Our work provides detailed analysis of how and why the changes affect the entities. Closely related research [5] [15] focuses on formal approaches for evolving RDF/S ontologies using a formal, well-specified way which is inspired by belief revision principles. [6] discusses evolution of OWL ontologies with the aim of guaranteeing validity of data instances. Our approach takes this further to analyze impacts numerically and compare different implementation results based on different metrics. Others suggest promptDiff [16] to analyze the structural changes between two versions of ontologies. We only require the original version and the change operations to process the potential impacts and allows the user to analyze different scenarios.

Many of these works focus on addressing the changes in terms of consistency of the ontology, focusing less on the structural and semantic impacts of the changes on the entities. In [10], the authors provide a critical evaluation of existing evolution approaches and propose an analysis of after-effects of ontology evolution. [17] highlights assertional effects of ontology editing activities in OWL. Their work is closely related to our work, but focuses more on highlighting the consequences of terminological axioms on the assertion axioms.

Here, we propose a bottom-up approach to analyze the impacts of atomic and composite change operations. Analyzing the impacts of individual change operations and also composite change operations enables us to understand the impact, i.e. entities impacted and the reason for the impact. This provides the user with semantic and structural impacts of a change operation before the change is permanently implemented. It allows us to compare different implementation strategies and suggests the optimal strategy with minimum impact. This feature can be exploited in a fully automated evolution environment. It further makes the evolution process transparent to the user by providing details on the number of changes, the statement types and the additions and deletions of entities.

The paper is organized as follows: Section 2 gives background on representation of ontologies and dependencies between entities. Section 3 introduces structural and semantic impacts. We discuss the change impact analysis method in Section 4 and a comparative evaluation in Section 5. We conclude in Section 6.

2 Preliminaries

2.1 Graph-based Representation of the Ontology

In this research we use graphs to represent OWL-DL ontologies. We represent classes, properties and instances as nodes and axioms as edges that connect these nodes. An ontology O is represented by a directed labeled graph $O = (N, E)$ where N is a set of nodes $\{N_1, N_2, \dots N_m\}$ which represents classes, data properties and object properties and instance. The nodes represent named or anonymous classes and instances. They further represent user-defined properties or OWL/RDFS properties such as *rfds:subClassOf* and *owl:disjointClass*. E is a set of labeled edges $\{ E_1, E_2, \dots E_n \}$. An edge E_i is written as (N_1, α, N_2) where $N_1, N_2 \in N$ and α represents class axioms, data property axioms, object property axioms, class assertion axioms and restriction axioms [1]. The label of any edge $E_i = (N_1, \alpha, N_2)$, which is represented by α , is a string given by **label(E_i)**. The type of any node is given by **type(N)** which returns the type of a node as class node, data property node, object property node and instance node. A sample graph representation of university domain ontology in OWL language is give in Fig 1.

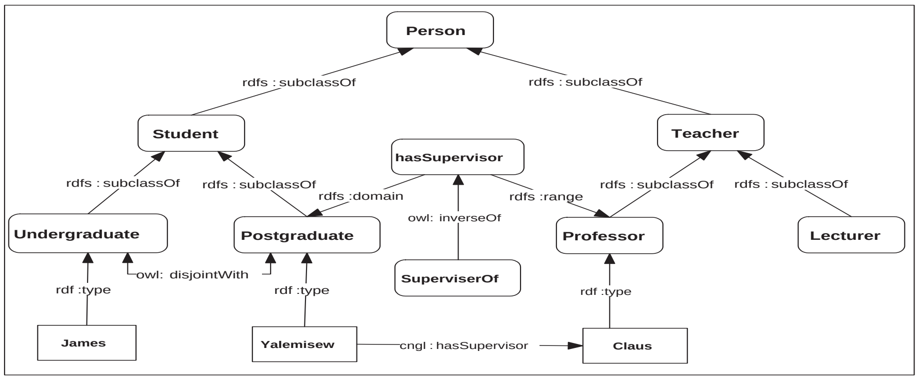

Fig. 1. A Graph representation of university ontology

2.2 Entity Dependency in Ontologies

Understanding the dependencies between entities is a starting point for analyzing the impacts of change operations on dependent entities [18]. Thus, characterization, representation and analysis of dependencies within and among the ontology entities is the major input for impact analysis. We define ***dependency*** as a reliance of one node on another node to get its structural and semantic meanings. The edges between two nodes indicate the dependency of one node on the other node. Given an ontology graph $O = (N, E)$, and two nodes $N_1, N_2 \in N$, N_1 is dependent on N_2 represented by $Dep(N_1, N_2)$, if $\exists\ E_i \in E$ where $E_i = (N_1, \alpha, N_2)$. Where N_1 is the dependent entity and N_2 is the antecedent entity.

The dependency between two entities provides substantial information about how a change in one entity affects the structure and the semantics of the dependent entities. This research exploits different dependency types to analyze impacts and identify affected entities whenever an antecedent entity evolves.

[1] http://www.w3.org/TR/owl2-syntax/

2.3 Types of Dependency

We identify eight different types of dependencies between entities [19]. The dependencies can be direct or indirect. Indirect dependencies are identified by iteratively implementing direct dependencies. Due to space constraint we present only three of the dependencies that we use in consecutive examples. Details of the dependencies can be found in [19].

1. **Concept-Concept Dependency:** for a graph $O = (N, E)$ and concept nodes $C_1, C_2 \in N$, C_1 is dependent on C_2 represented by $\mathbf{CCDep}(C_1, C_2)$, if $\exists\ Dep(C_1, C_2) \wedge (label(E_i) = \text{“}rdfs : subClassOf\text{”}) \wedge (type(C_1) = type(C_2) = \text{“}Class\text{”})$. Concept-concept dependency is transitive.
2. **Concept-Axiom Dependency:** for a graph $O = (N, E)$, a class node C_1, and any node $N_i \in N$ and an edge $E_1 \in E$, E_1 is dependent on C_1 represented by $\mathbf{CADep}(E_1, C_1)$, if $(E_1 = (C_1, \alpha, N_i) \vee E_1 = (N_i, \alpha, C_1)) \wedge (type(C_1) = \text{“}class\text{”})$.
3. **Concept-Instance Dependency:** for a graph $O = (N, E)$ and an instance node I_1 and a concept node $C_1 \in N$, I_1 is dependent on C_1 represented by $\mathbf{CIDep}(I_1, C_1)$ if $\exists\ E_i \in E$ where $E_i = (I_1, \alpha, C_1) \wedge (label(E_i) = \text{“}rdf : type\text{”}) \wedge (type(I_1) = \text{“}instance\text{”}) \wedge (type(C_1) = \text{“}class\text{”})$.

3 Impacts of Change Operations

Impacts of change operations are diverse. For a better analysis, we divide them into structural and semantic impacts. We further studied the change operations that cause the impacts and the reasons why the impacts occur. All entities in an ontology are subject to change and are impacted by change operations in one way or another way. To represent all the constructs of an ontology collectively, we use the term Entity (E). However, to refer to a specific constructs, we replace the term Entity (E) by Class (C), Data Property (DP), Object Property (OP), Instance (I), Axiom (A) and Restriction(R) whenever appropriate.

Structural impacts are impacts that affect the structure of the ontology entities. They occur when there is a structural change in the ontology [4]. From an empirical study [11], we identify the following structural impacts.

$StrImp = \{OC, CCR, OPCR, DPCR, OI, AE, DE\}$

- **Orphan Concept (OC)** occurs when a given concept is introduced without a superclass other than the default “Thing” class. Generally OWL allows orphan classes, but sometimes the application requirement do not.
- **Concept Cyclic Reference (CCR)** occurs when a change operation introduces a cyclic reference between concepts.
- **Object Property Cyclic Reference ($OPCR$)** occurs when a change operation introduces a cyclic reference between object properties.
- **Data Property Cyclic Reference ($DPCR$)** occurs when a change operation introduces a cyclic reference between data properties.

- **Orphan Instances** (OI) occurs when a change operation introduces/makes an instance with noa link to a specific class.
- **Addition of new Entity** (AE) when an entity is added to the ontology.
- **Deletion of existing Entity** (DE) when an entity is removed.

The last two structural impacts directly correspond to the change operations and are straight forward. We consider them as impacts because they are side effects of the evolution process and they play a significant role during composite change impact analysis.

Semantic impacts are impacts that change the interpretation of the entities. Whenever a change occurs, it may change the existing information about the target entity or the dependent entities. The semantic impacts are derived from either structural or semantic changes [6].

$SemImp = \{EMD, ELD, PMR, PLR, AME, ALE, EG, ES, EI, UE, IE\}$

- **Entity More Described** (EMD)={CMD, DPMD, OPMD, IMD} occurs when we add previously unknown semantic (facts) about an entity.
- **Entity Less Described** (ELD)={CLD, DPLD, OPLD, ILD} occurs when we remove an existing semantics (facts) about an entity.
- **Property More Restricted** (PMR)={OPMR, DPMR} occurs when the existing semantics of a property is more restricted.
- **Property Less Restricted** (PLR)={OPLR, DPLR} occurs when the existing semantics of a property is less restricted.
- **Axiom More Expanded** (AME) occurs when an axiom further extends its semantics to other entities.
- **Axiom Less Expanded** (ALE) occurs when an axiom further restricts its semantics to fewer entities.
- **Entity Generalized** (EG)={CG, DPG, OPG, IG} occurs when an entity become more general (move up in the hierarchy).
- **Entity Specialized** (ES)={CS, DPS, OPS, IS} occurs when an entity become more specific (move down in the hierarchy).
- **Entity Incomparable** (EI)={CI, DPI, OPI, II} occurs when a change on an entity neither generalize nor specialize the entity.
- **Unsatisfiable Entity** (UE)={UC, UDP, UOP, UA} occurs when a change on a given entity introduces logical contradiction [20].
- **Invalid Entity** (IE)={II, IIP} occurs when a change on a given instance or instance property(IP) introduces invalid interpretation [6].

4 Change Impact Analysis Process

Change impact analysis process takes a change request from the user and identifies the impacts of the change operations used to implement the request. It further identifies which entities are impacted by the change. We use a bottom-up approach which identify the possible impacts of individual atomic changes then analyze impacts of composite and domain specific change operations.

4.1 Individual Change Impact Analysis

Individual change impact analysis takes a single and atomic change operation and identifies the structural and semantic impacts together with the impacted entities. This phase uses addition and deletion operations, the target entities (E) and the parameters to determine the impacts of the atomic change operation. We studied and identified the possible impacts and preconditions for the impacts of atomic change operations. We present only five of the change operations and their associated impacts in Table 1.

Table 1. Impacts of selected atomic change operations

No	**Change Operation**	**Impact Type**	**Impact (Entity)**	**Impact Precondition**
1	Add Class (C)	Structural	AC(C), OC(C)	None
2	Add subClass (C_1, C)	Structural	AA (FullAxiom)	None
			CCR(C_1), CCR (C)	$\exists$ **CCDep**(C, C_1)
		Semantic	UC (C_1)	$\exists$ **CCDep**$(C_1, D) \wedge disjointClass(C, D)$
			CMD(C_1), CMD(C)	None
10	Delete Class (C)	Structural	DC (C)	None
		Semantic	UA (a_i)	$\exists$ **CADep**(a_i, C)
11	Delete subclass (C_1, C)	Structural	DA (FullAxiom)	None
			OC (C_1)	$\exists$ **CCDep**$(C_1, C) \wedge \nexists$ **CCDep**$(C_1, D) \wedge C \neq D$
		Semantic	$CLD(C_1)$, CLD (C)	None
22	Add Disjoint $Class(C_1, C_2)$	Structural	AA (FullAxiom)	None
		Semantic	UC (C_1), UC(C_2)	$\exists$ **CCDep**$(C, C_1) \wedge$ **CCDep**(C, C_2)
			II(I)	$\exists$ **CIDep**$(i, C_1) \wedge$ **CIDep**(i, C_2)

Let us use examples from Figure 1 to elaborate the impacts presented in Table 1. The *AddClass*(*Student*) change introduces a new class *AC*(*Student*) and the class becomes orphan *OC*(*Student*). When we use *Add subclassOf(Student, Person)*, we add a new axiom (*AA*) and since the precondition is not satisfied, we do not assign *CCR*(*Student*) and *CCR*(*Person*). The addition of the axiom semantically describes the two classes *CMD*(*Student*) and *CMD*(*Person*). Whenever we delete a class, say *DeleteClass*(*Faculty*), we remove the class *DC*(*Faculty*) and create unsatisfiable axioms which refer to the deleted class. All the three *subcalssOf* axioms referring to the deleted class become unsatisfiable (*UA*). Whenever we delete a subclass axiom, say, *Delete subclass* (*Faculty*, *Person*), we delete the axiom (*DA*) and we may introduce orphan class *OC*(*Faculty*) if *Person* is the only parent class of *Faculty*. But if *Faculty* has another super class, *OC*(*Faculty*) will not occur. However, both classes become semantically less described *CLD*(*Faculty*), *CLD*(*Person*) due to the removal of a previously known fact about *Faculty* and *Person*.

We take another example to further elaborate atomic change impact analysis. The impact of *Delete subclassOf(Lecturer, Faculty)* operation, without additional change operations, is given in Table 2.

Table 2. An Example of impact of single atomic change operation

Change Operation	**Structural Impact**	**Semantic Impact**
Delete subclassOf(*Lecturer*,*Faculty*)	DA(*FullAxiom*)	CLD(*Lecturer*)
	OC(*Lecturer*)	CLD(*Faculty*)

Single atomic change operations are frequently used by ontology engineers. But despite their simplicity and atomicity, they may have complex impacts which are difficult to identify manually. Thus, individual impact analysis serves this purpose and is used as an input for composite change impact analysis phase.

4.2 Composite Change Impact Analysis

Ontology engineers compose two or more atomic change operations to perform a single task. In such cases, they group related atomic changes and implement them as a transaction. Two or more atomic change operations that are executed together as a transaction are called composite change operations [4]. When a composite change operation is implemented, the impacts of the composite change may not be the same as the aggregation of the impacts of its constituent individual atomic change operations. Composite change impact analysis identifies techniques to analyze the impacts of composite change operations. To analyze these impacts we employ novel techniques such as impact cancellation, impact balancing and impact transformation.

Impact Cancellation is applied on impacts of two change operations when the impact of one operation cancels or overrides the impact of the other operation. This means, if the impact of a given change operation nullifies the impact caused by another operation, or when one impact subsumes the other impact, we say one impact cancels the other. We use heuristics to identify impacts that cancel each other. The heuristic rules are derived from our observation of case studies and are validated using experiments [11].

Rule 1. When a target entity is affected by a given change operation but if that target entity is deleted by another change operation, all the structural and semantic impacts of the first operation will be canceled by the structural impact(DE) of the second operation. Based on this we identify canceling and canceled impacts. Using Rule 1, the delete entity $DE(x)$ impact overrides or cancels all the semantic and structural impacts of the entity x.

Rule 2. When a change operation is executed, if it introduces an impact, but if another change operation falsifies the precondition of the previous impact, the first impact of the entity will be canceled. For example, the impact of $addclass(x)$ is $OC(x)$, because x is not yet linked to a parent class. But if the composite change contains $addsubclass(x, y)$, this statement falsifies the precondition of the first statement, thus we cancel the $OC(x)$ impact of the first change operation due to the second change operation which makes x no more orphan.

A typical characteristics of cancellation is, it occurs between two additions or two deletion operations, but one acts on a node (e.g. class) and the other on the edge (e.g. subclassOf) linked to that node. Impact cancellation is used to filter out impacts which are overridden or nullified by other impacts.

Table 3. Impact cancellation using heuristic Rule-1

No	**Change Operation**	**Before cancellation**		**After cancellation**	
		StrImp	SemImp	StrImp	SemImp
1	Delete SubClassOf (*Faculty, Person*)	OC (*Faculty*)	CLD (*Faculty*) CLD (*Person*)	None	CLD (*Person*
2	Delete Class(*Faculty*)	DC (*Faculty*)	None	DC (*Faculty*)	None

The example in Table 3 elaborates how the impact cancellation process is used to analyze impacts of composite change operations. The first change operation deletes the subclassOf axiom, and introduces OC (*Faculty*), CLD (*Faculty*) and CLD (*Person*) impacts. However the following change operation deletes the class *Faculty* and causes DC (*Faculty*) impact. Based on Rule 1, the OC (*Faculty*) and CLD (*Faculty*) impacts are canceled from the first operation because the class *Faculty* is removed. Note that the CLD (*Person*) impact is still there.

Impact Balancing The impacts of two change operations balance each other when one change operation introduces an impact to an entity and another change operation removes the impact from the entity. Unlike impact cancellation, impact balancing only occurs between an addition and a deletion operation with the same target entity (class with class and subclass with subclass).

Rule 3. When a given change operation affects the target entity with an impact, and when another change operation affects the same entity with a counter impact or vice versa, the two change operations can be balanced.

Table 4. Candidate impacts for balancing

Impacts	**Counter-Impact**
Entity More Described (EMD)	Entity Less Described (ELD)
Axioms More Expanded (AME)	Axioms Less Expanded (ALE)
Object Property Less Restricted (OPLR)	Object Property More Restricted (OPMR)
Addition of new Entity (AE)	Deletion of existing Entity (DE)

Let us take two atomic change operations from the previous example. Add subclass (*Professor, person*) and delete subclass (*Professor, Faculty*). When we delete *Faculty*, depending on our strategy we may link the subclasses of *Faculty* with the class *Person*. Thus, when we execute the two change operations as part of a composite change, we balance the impacts as follows.

Table 5. Impact balancing using Rule-3

No	**Change Operation**	**Before balancing**		**After balancing**	
		StrImp	SemImp	StrImp	SemImp
1	Add SubClassOf (*Professor, Person*)	AA (*FullAxiom*)	CMD (*Professor*) CMD (*Person*)	None	None
2	Delete SubClassOf (*Professor, Faculty*)	DA (*FullAxiom*)	CLD (*Professor*) CLD (*Faculty*)	None	CLD (*Faculty*)

After balancing of the operations in Table 5, we remove the CLD and CMD semantic impacts and the AA and DA structural impacts. However, when two change impacts balance each other, they may introduce a higher level impact which is caused by composite change operations. Some change operations may introduce impacts such as specialization or generalization of the entities, more restriction or less restriction on cardinalities of properties. Thus, the original change impacts are transformed to create another change impacts. In such situations, we move to impact transformation step.

Impact Transformation When one impact is balanced with another impact, it may introduce another set of impacts that are created due to more than one change operation. The balancing of two or more impacts transform existing impacts to another impact set which are not observed at atomic change levels. Two balancing change operations may introduce generalization, specialization or incomparability of entities. We present one of the heuristics that is used to check transformation of impacts related to entity hierarchies.

Rule 4. When impacts of two change operations balance and if the operations are applied to subclass, subDataProperty, subObjectProperty or classAssertion axioms, the balancing impacts transform to generalization, specialization or incomparable impact depending on the new location of the entity in the hierarchy.

For all balancing impacts, we need to check whether those balancing impacts further indicate other impacts. In Table 5, class *Professor* has CMD(*Professor*) and CLD(*Professor*) impacts by the the two change operations. However, the concept *Professor* is generalized CG(*Professor*) and becomes a direct subclass of *Person*. Generalization means that the entity moves up the hierarchy; specialization when it moves down.

Table 6. Impact transformation using Rule-4

No	Change Operation	Before Transformation		After Transformation	
		StrImp	SemImp	StrImp	SemImp
1	Add SubClassOf (*Professor, Person*)	AA (*FullAxiom*)	CMD (*Professor*) CMD (*Person*)	None	CG(*Professor*)
2	Delete SubClassOf (*Professor, Faculty*)	DA (*FullAxiom*)	CLD (*Professor*) CLD (*Faculty*)	None	CLD (*Faculty*)

In this case, the second structural impact of the second change operation will be reduced due to impact balancing. However, the semantic impact of the first change operation will be transformed to another impact and the transformation is determined by the current location of the target entity. Thus, the semantic impact is generalization, as *Professor* class goes up the hierarchy.

4.3 Exploiting Impact Analysis to Evolve Ontologies

Change impact analysis method is used to identify possible impacts of change operations and helps to make an informed decision about the impacts of the change on both the structure and the semantics of the ontology. It further allows us to compare different implementation strategies and select the one with a minimum impact on the ontology. Now let us take the *Delete class (Faculty)* operation and see how the change impact analysis can be exploited.

When we delete *Faculty*, we need to make a decision either to link its subclasses to *Person* class, delete all the subclasses together with the class or just delete the class *Person* and its references from the ontology. Thus, “Attach” strategy attaches the subclasses to *Person* and “Cascade” strategy cascades the deletion to dependent subclasses and “No Action” strategy takes no action except deleting the class and its references. We generate the respective composite change operations and analyze their impacts following the approach discussed above. The results are discussed in Table 7.

At this stage of the research, we use number of impacts to compare the strategies by analyzing the number of impacts. However, strategy selection can be analyzed using different metrics such as the severity of the impacts and the amount of change in the inferred ontology using a reasoner. Using the number of impacts, a preferable strategy is the one with minimum number of impacts.

Table 7. Comparison of impacts of different implementation strategies

Impact	Frequency		
	Attach to parent	**Cascade**	**No Action**
Semantic Impacts			
Class Less Described (CLD)	1	3	3
Object Property Less Described (OPLD)	0	1	0
Instance Less Described (ILD)	0	1	0
Concept Generalized (CG)	2	0	0
Structural Impacts			
Orphan Class (OC)	0	0	2
Deleted Class (DC)	1	3	1
Deleted Axiom (DA)	1	6	3
Deleted Instance (DI)	0	1	0
Total structural and semantic Impacts	5	15	9
Additional analysis			
Addition Operation	2	0	0
Deletion Operation	4	10	4
TBox Statements	6	7	4
ABox Statements	0	3	0
Time required in milliseconds	100	150	75

"Attach" strategy ranks first with 5 impacts and the "No Action" strategy ranks second with 9 impacts. Thus, in terms of number of impacts, the first strategy is better. However, if we compare "cascade" and "no action" strategies, the latter strategy introduces orphan class. If we don't allow orphan classes, this strategy becomes less preferable in terms of keeping the consistency of the ontology. If a strategy introduces impacts such as OC, UE and IE, in general they are less preferable. However, this comparison requires further analysis on the severity of the impacts in a given ontology. The additional information can be used to further enhance the selection by evaluating the additions, deletions and the statement types that are affected by a given strategy.

5 Evaluation

We build a prototype to implement our method and compare it with the functionalities of existing tools. We selected widely used editors (Protege and Neon) and compare them with our prototype based on availability of structural and semantic impacts, transparency of the evolution, available strategy, optimal strategy suggestion and reversibility [21].

For the purpose of the evaluation, we use an ontology built for software help management which contains 80 classes, 8 data properties, 10 object properties and more than 500 axioms. We use 10 change operations representing different change scenarios. Two ontology evolution experts and two ontology users participate in the comparative evaluation.

The results in Table 8 show that our approach is capable of filling the gap observed in existing ontology editors. The available tools do not support impact analysis. But, we introduce change impact analysis with structural and semantic impacts capable of comparing and selecting best strategies. It shows each change, detailed impacts and the causes of the impacts. This feature provides a transparent evolution process. The change impact analysis compares implementation strategies based on their structural and semantic impacts. The analysis results can be used to automatically select implementation strategies. Our approach preserves features of other editors such as undo and redoes functionalities.

Table 8. Comparison of impacts of different implementation strategies

Criteria	**Protege**	**Neon**	**Ours**
Structural Impact	does not show details of structural impact	shows structural changes	shows changes, impacts, impacted entities and gives explanation
Semantic Impact	does not show semantic impact before implementation	does not show semantic impact before implementation	shows impacts, impacted entities and gives explanation
Transparency	user don't know which entities are affected before the change	structural impacts are transparent but not semantic impacts	users are able to see detailed impacts of atomic or composite changes (how and why)
Implementation strategy	Delete target entity and delete entity and its reference	allow composition by adding or removing atomic changes	allows attach all, cascade, no action for TBox and ABox whenever applicable
Optimal strategy suggestion	Not available	Not Available	compares and shows the optimal strategy
Reversibility	provides undo and redo	provides undo and redo	provides undo and redo

6 Conclusion and Future Work

Changes in ontologies cause different impacts on entities and dependent systems. It is difficult to manually identify the structural and semantic impacts of these changes. We presented a change impact analysis method which begins with atomic changes and move to composite change operations to analyze semantic impacts including unsatisfiable statements and invalid instances. The proposed approach benefits users by enabling them to view the actual impacts of change operations and the causes of the impacts. It further enables them to compare implementation strategies in terms of impact. It allows transparent evolution by providing the impacts of changes as individual and/or composite operations. The process allows users to conduct a what-if analysis before they permanently implement changes. This approach has a potential for automatic selection of optimal implementation strategies which ensure consistent evolution whenever there is an alternative implementation strategy. Exploiting change impact analysis for the selection of optimal strategy based on criteria such as severity of impacts, number of change operations, statement types, incremental and decremental changes is our future task.

Acknowledgment. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/CE/I1142 as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University (DCU).

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
2. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: SAC '05: Proceedings of the 2005 ACM symposium on Applied computing. (2005) 1634–1638
3. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management:requirements and survey of the state of the art. Web Semantics: Science, Services and Agents on World Wide Web. **4**(1) (2006) 14–28

4. Stojanovic, L.: Methods and tools for ontology evolution. PhD thesis, University of Karlsruhe (2004)
5. Konstantinidis, G., Flouris, G., Antoniou, G., Christophides, V.: A formal approach for rdf/s ontology evolution. In: Proceedings 18th European Conference on Artificial Intelligence ECAI 2008. (2008) 70–74
6. Qin, L., Atluri, V.: Evaluating the validity of data instances against ontology evolution over the semantic web. Inf and Softw Techn. **51**(1) (2009) 83–97
7. Holohan, E., Melia, M., McMullen, D., Pahl, C.: The generation of e-learning exercise problems from subject ontologies. Sixth International Conference on Advanced Learning Technologies. (2006) 967–969.
8. Pahl, C., Giesecke, S., Hasselbring, W.: An ontology-based approach for modelling architectural styles In: European Conference on Software Architecture ECSA'2007. Springer-Verlag (LNCS Series) (2007) 60–75
9. Gruhn, V., Pahl, C., Wever, M.: Data Model Evolution as Basis of Business Process Management. In: 14th International Conference on Object-Oriented and Entity Relationship Modelling O-O ER95. Springer-Verlag (LNCS Series). (1995)
10. Khattak, A., Pervez, Z., Lee, S., Lee, Y.K.: After effects of ontology evolution. In: 5th International Conference on Future Information Technology. (2010) 1–6
11. Abgaz, Y., Javed, M., Pahl, C.: A framework for change impact analysis of ontology-driven content-based systems. In: On the Move to Meaningful Internet Systems: OTM 2011 Workshops. Lecture Notes in Computer Science. (2011)
12. Stojanovic, L., Stojanovic, N., Handschuh, S.: Evolution of the metadata in the ontology-based knowledge management systems. In: Proceedings of the 1st German Workshop on on Experience Management: Sharing Experiences about the Sharing of Experience, GI (2002) 65–77
13. Noy, N.F., Chugh, A., Liu, W., Musen, M.A.: A framework for ontology evolution in collaborative environments. In: 5th International Semantic Web Conference, Springer-LNCS (2006) 544–558
14. Plessers, P., De Troyer, O., Casteleyn, S.: Understanding ontology evolution: A change detection approach. Web Semant. **5**(1) (March 2007) 39–49
15. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. The Knowledge Engineering Review **23**(02) (2008) 117–152
16. Noy, N.F., Musen, M.A.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: AAAI/IAAI'2002. (2002) 744–750
17. Pammer, V., Serafini, L., Lindstaedt, S. In: Highlighting assertional effects of ontology editing activities in OWL. Volume 519. CEUR Workshops (2009)
18. Cox, L., Harry, D., Skipper, D., Delugach, H.S.: Dependency analysis using conceptual graphs. In: Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001, Springer (2001)
19. Abgaz, Y., Javed, M., Pahl, C.: Dependency analysis in ontology-driven content-based systems. In Rutkowski, L. et al., eds.: Artificial Intelligence and Soft Computing. LNCS 7268. (2012) 3–12
20. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA (2003)
21. Stojanovic, L., Motik, B.: Ontology evolution within ontology editors. Proceedings of the OntoWebSIG3 Workshop **68** (2002) 568–580
22. Javed, M., Abgaz, Y.M., Pahl, C.: A pattern-based framework of change operators for ontology evolution. In: On the Move to Meaningful Internet Systems: OTM Workshops. Volume 5872 of LNCS., Springer (2009) 544–553.