NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 – "Semantic-based knowledge and content systems"

D 1.5.1 Dynamics of Metadata

Deliverable Co-ordinator: **Diana Maynard**

Deliverable Co-ordinating Institution: **USFD**

Other Authors: Wim Peters (USFD), Mathieu d'Aquin (OU), Marta Sabou (OU), Niraj Aswani (USFD)

Document Identifier:	NEON/2007/D1.5.1/v1.0	Date due:	February 28th, 2007
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	March 30th, 2007
Project start date:	March 1, 2006	Version:	V1.0
Project duration:	4 years	State:	Final
		Distribution:	Report/Public

NeOn Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities, grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta} @open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 28 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Richard Benjamins E-mail address: rbenjamins@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 - 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Asociación Española de Comercio Electrónico (AECE)** C/Alcalde Barnils, Avenida Diagonal 437 08036 Barcelona Spain Contact person: Jose Luis Zimmerman E-mail address: jlzimmerman@fecemd.org
Food and Agriculture Organization of the United Nations (FAO) Viale delle Terme di Caracalla 1 00100 Rome, Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org	**Atos Origin S.A. (ATOS)** Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed writing parts of this document:

USFD

OU

UKARL

Change Log

Version	Date	Amended by	Changes
0.1	22/01/2007	Diana Maynard	Initial draft
0.2	13/02/2007	Diana Maynard	Additions from OU
0.3	19/02/2007	Diana Maynard	Comments from Karlsruhe and OU
0.4	20/02/2007	Wim Peters	Final comments
0.5	08/02/2007	Diana Maynard	Final edits following reviewer's comments
1.0	15/03/2007	Aneta Tumilowicz	QA

Executive Summary

This document briefly describes the key objectives of T1.5 (Dynamics of Metadata), which forms part of NeOn WP1. The overall goal of NeOn's work package 1 is to develop an integrated approach to the evolution process of networked ontologies and related metadata. Within this context, the more specific goal of T1.5 is to capture the evolution of metadata due to changed concepts, relations or metadata in one of the networked ontologies, and to capture changes to the ontology caused by changes to the metadata. After a short discussion of the nature of metadata, we propose a methodology to capture (1) the evolution of metadata and (2) changes to the ontology. This will lead, in a later stage of the project, to the actual implementation of an approach for evolution of metadata related to networked ontologies.

Table of Contents

List of figures

List of tables

1. Introduction

This document briefly describes the key objectives of T1.5 (Dynamics of Metadata), which forms part of NeOn WP1. The overall goal of NeOn's work package 1 is to develop an integrated approach to the evolution process of networked ontologies and related metadata. Within this context, the more specific goal of T1.5 is to capture the evolution of metadata due to changed concepts, relations or related metadata in one of the networked ontologies, and to capture changes to the ontology caused by changes to the metadata. After a short discussion of the nature of metadata, we propose a methodology to capture (1) the evolution of metadata and (2) changes to the ontology. This will lead, in a later stage of the project, to the actual implementation of an approach for evolution of metadata related to networked ontologies.

In the remainder of this section, we introduce some relevant concepts. Section 2 provides some background information about the nature of semantic metadata and the need for mechanisms to represent its evolution. Sections 3 and 4 describe the top-down and bottom-up approaches respectively: the top-down approach describing the required metadata changes to reflect ontology changes, and the bottom-up approach describing the required ontology changes to reflect metadata changes. Finally, Section 5 concludes with a description of future work.

1.1 Terminology: Semantic Metadata

At this stage, we would like to clarify some terminological ambiguity when talking about metadata. This is due to the use of the term in various areas of knowledge engineering. First, there is the notion of metadata as **ontology metadata**. Ontology metadata provides information about the ontology, e.g. who created it, how many concepts it contains, etc. A proposal exists for a standard description for this type of metadata information, the Ontology Metadata Vocabulary (OMV) [1]. Changes in the ontology may affect the value of these standardised attributes: for instance, the addition of another natural language for the labels will lead to a change in the OMV "language" attribute. Ontology metadata in this sense is dealt with in Task 1.3 and is therefore not discussed further here.

The second notion of metadata stems from the area of natural language processing, and is often called **semantic metadata** or **annotations**. This type of metadata concerns concept instantiation in the form of the annotation of textual (or other forms of) data, and therefore contains information about the linguistic content of the ontology. Note that within NeOn, we are concerned primarily with text rather than other forms of media such as images or videos, so where the question of media is left unspecified, we shall be referring to textual forms of media. In this sense of metadata, data annotation concerns the task of adding metadata to text. In this context it concerns the linking of instances in the text to concepts in the ontology, and potentially also finding relations between such concepts. This is known as **semantic metadata creation**, which forms the main link between text and ontologies. It is applied to ontology evolution in the form of bottom-up change discovery and ontology population, which concerns adding instances from the text to concepts in the ontology. In this deliverable, we will be discussing the dynamics of semantic textual metadata: hereafter when we refer to metadata, unless otherwise stated, we mean semantic metadata.

2. Support for metadata evolution

Support for metadata evolution becomes extremely important in a distributed, dynamic environment. Change management should warrant the continuity of data access, i.e. all data previously associated with an older version of an ontology should be accessible and interpretable through the new version. When ontologies change, this change must be propagated to all the information objects that are dependent on them, such as local copies of ontologies and annotated texts. After the detection of changes in conceptual structure between two versions of an ontology, the ontology management system must enable the update of metadata affected by the changes in the ontology, in order to maintain a consistent link between ontology and metadata. This link is defined by a number of attributes, amongst which are the URLs for ontology, annotated text and concept. Similarly, the system must be capable of enabling updates to the ontology which may be necessary when the metadata changes: for example, the processing of new documents might show the emergence of new concepts and new relations, reflecting an evolution of the domain itself, or some concepts may become less significant. Similarly, the mismatch between the results of two different annotators in a collaborative annotation environment might require the merging of two subconcepts in an ontology. In these scenarios, the evolution of ontologies should be guided by changes in the metadata, to keep the knowledge they contain up-to-date with respect to the considered domain.

Before we discuss the support for changing metadata and the methodology we use (section 2.2), we first take a brief look at the creation of semantic metadata, in order to fully understand the role of its creation, existence and need for maintenance.

2.1. Creation of semantic metadata

The Semantic Web aims to add a machine tractable, repurposable layer to complement the existing web of natural language hypertext. In order to realise this vision, the creation of semantic annotation, the linking of web pages to ontologies, and the creation, evolution and interrelation of ontologies must become automatic or semi-automatic processes. An important aspect of the Semantic Web revolution is that it is based largely on human language materials, and in making the shift to the next generation knowledge-based web, human language will remain crucial. In the context of new work on distributed computation, Semantic Web Services go beyond current services by adding ontologies and formal knowledge to support description, discovery, negotiation, mediation and composition. This formal knowledge is often strongly related to informal materials. To make these types of services cost-effective, we need automatic knowledge harvesting from all forms of content that contain natural language text or spoken data.

Semantic annotation is essentially the task of assigning to the entities in the text links to their semantic descriptions. This kind of metadata provides both class and instance information about the entities. Semantic annotations enable many new applications to be performed, such as highlighting, indexing and retrieval, categorisation, generation of more advanced metadata, and a smooth traversal between unstructured text and available relevant knowledge. Semantic annotation can be applied to any kind of text - web pages, regular (non-web-based) documents, text fields in databases, etc. - or even to non-textual forms of data (although, as mentioned earlier, we shall restrict ourselves to textual content). Furthermore, knowledge acquisition can be performed on the basis of the extraction of more complex dependencies, such as analysis of the relationships between entities, event and situation descriptions, and so on.

Automatic semantic annotation of textual data is generally carried out by means of some kind of ontology-based information extraction (OBIE). While semantic annotation can of course be performed manually, it is a time-consuming and laborious task, which certainly cannot scale to the

demands of real world applications on the web. Therefore at the least, a semi-automatic, if not fully automatic, process is required. Ontology-based IE differs from traditional IE in a number of ways. First, it makes uses of a formal ontology rather than a flat lexicon or gazetteer, which may also require reasoning to be carried out. Second, it not only finds the (most specific) type of the extracted entity, but it also identifies it, by linking it to its semantic description in the instance base. This allows entities to be traced across documents and their descriptions to be enriched through the IE process. This more semantic form of IE is therefore a much harder task than the traditional one: see for example [2], which describes the extension of a traditional IE system into one which performs a more semantic extraction, comparing the two tasks, systems and results.

The automatic population of ontologies with instances from the text requires the existence of an ontology and a corpus. From this, an OBIE application identifies instances in the text belonging to concepts in the ontology, and adds these instances to the ontology in the correct location. It is important to note that instances may appear in more than one location in the ontology, because of the multidimensional nature of many ontologies and/or ambiguities which cannot or should not be resolved at this level. For examples of OBIE applications, see for example [3] and [4].

2.2. Support for metadata evolution

With respect to ontology evolution, we may distinguish two types of changes: top-down and bottom-up [8]. Top-down changes are explicit changes in the ontology, to which the metadata needs to be adapted. Bottom-up changes occur when distributed metadata need to be reflected by changes of an ontology. As mentioned above, we shall not concern ourselves in this task with changes that involve the ontology metadata, only with those that involve semantic metadata.

Top-Down metadata evolution: As metadata creation is an expensive task, it is important that sets of ontology metadata and document annotations are kept in sync with an evolving ontology. As far as possible, we do not want to have to reannotate a whole corpus every time the ontology changes in some way (although in some cases this is inevitable). The evolution of the related metadata has to be synchronised with the evolution of the ontologies for the purpose of preserving instance data and compatibility between ontology versions [7]. Therefore, methods for evolving metadata automatically and in parallel with the networked ontologies are required. In the presence of networked ontologies, this includes the synchronisation of distributed metadata.

Bottom-Up ontology evolution: Textual resources and the associated metadata generally evolve faster than the related ontologies. In that sense, they reflect more accurately the evolution of the domain itself, e.g. progresses realised in scientific fields, new trends or obsolete elements. Even if ontologies are supposed to be a stable conceptualisation of the domain, the evolution of the metadata has to be reflected in the related ontologies, to keep the relation between ontologies and metadata up-to-date, and so that the ontologies evolve in accordance with the current state of the domain they represent. Therefore, the dynamics of metadata have to be captured in a way that it can be related to the adequate ontologies and guide their evolution by suggesting corresponding ontology changes, leading to a metadata-driven maintenance of ontologies. We shall therefore investigate which operations are necessary to cover a number of evolution strategies for bottom-up change discovery, e.g. in the case of a required extension or refinement of the ontological structure of a particular ontology component, or the suggestion of the merge of two classes for a particular application if an automatic classifier is unable to distinguish between the two (or if human agreement is not reached) and the concepts in question are considered relatively similar.

Consequently, changes in ontology structure, rather explicit (top-down) or suggested by metadata changes (bottom-up), will need to be captured by means of evolution operations, and described in some standardised fashion. One solution to this problem is to keep the metadata static and keep track of the (specific) version of the ontology used for the text annotation. In this case, we assume

that annotations are stable but contextual, and thereby manage the evolution of the ontology only at the ontological level by means of links between the old and new versions of the ontology, or the link between different ontologies. In this case we could study how the annotations according to one ontology can be imported into a new ontology (linked to the other ontology in a formal way). However, this approach leads to a high level of complexity if there are many versions of the ontology and also makes automatic processing more difficult (for example, using the populated ontology for tasks such as information retrieval and question answering). Furthermore, it is not evident how this could be achieved when material is added to or deleted from the ontology, as links cannot exist to items which are non-existent. So it would only be useful for certain types of change and not as a complete framework for change (assuming that we wish to avoid data loss).

A potential candidate for the change capture and description phases is [5], who propose a framework that integrates all sources of ontology change information. A so-called transformation set encompasses all changes that have occurred between an old and a new version of the ontology. The changes that can occur in OWL ontologies have been gathered into a change typology and made available on the web. The change typology is an ontology of change operations, and covers basic change operations such as “delete superclass”, and complex operations, e.g. “add an entire subtree”. The ontology of basic change operations contains “add” and “delete” operations for each feature of the OWL knowledge model. Complex operations consist of a combination of basic operations. We envisage evaluating this ontology with respect in the light of the requirements for capturing the dynamics of metadata. For this purpose we shall establish:

1. which change classes are relevant for metadata evolution;

2. what effect such changes have on the metadata, and what action should follow these changes. For instance, if a concept is moved, references to the concept should be found and modified appropriately; if a concept is deleted, annotations referencing the concept should be changed to reference its superclass, etc.

3. what effect the evolution of metadata has on the ontologies, in terms of ontology changes. For instance, if a new prominent term appears from a set of textual resources, a concept should be added in the ontology; if two terms appear to be related in the metadata (e.g. through frequent co-occurrence) then the ontology should be extended with a new relation.

In the following section, we describe in more detail the first task -- capturing metadata changes resulting from ontology changes -- and the methodology we plan to follow. The envisaged end product of this task will be a classification of ontology change types relevant for capturing metadata evolution, and associated operations on the metadata database.

3. Capturing changes to metadata

As mentioned in Section 2.2, one aspect of capturing metadata evolution concerns top-down changes to the ontology. This is where explicit changes in the ontology require adaptation of the semantic metadata in order to remain synchronised with the evolving ontology. For example, if a concept is removed from the ontology, we must consider what should happen to the associated metadata (instances) attached to it. Where possible, we want to automate this evolution process by capturing the regularities that persist. In some cases, re-annotation or manual intervention will be the only solution, but we want to avoid this wherever possible in order to maximise efficiency. We first consider Klein and Noy's framework for capturing ontology evolution (as introduced in the previous section), and then investigate its suitability for our purposes and what adaptations it may require. Note that although Klein and Noy's framework deals with both basic and complex changes, we restrict ourselves here largely to discussing the basic changes, since the complex changes can be inferred from the basic changes.

3.1 Klein and Noy's hierarchy

Klein and Noy's proposed hierarchy for ontology evolution describes a complete support for change management in an ontology environment, given an ontology O and its two versions V1 and V2. It consists of a set of basic change operations, each one modifying a specific feature of the OWL knowledge model, for example, changing the cardinality of a property restriction. The change operations are organised into a hierarchy in order to exploit inheritance and to be able to specify common changes more efficiently, by enabling, for example, the specification of facts for multiple operations only once. They also propose a set of complex change operations, which are composed of multiple basic operations or which incorporate some additional kind of knowledge about the change. This provides a means to group several related basic mechanisms together into a single operation.

Noy notes that the set of change operations for ontologies is much larger than that for database schemas [10], partly because of the richer knowledge model for ontologies, e.g. operations that deal with changes in slot restrictions, and partly because of the use of composite operations, e.g. changing a domain slot from a class to its superclass.

Below we summarise the main aspects of Klein and Noy's set of possible change operations as detailed in their paper. They classify the operations in terms of their information preservation effect, i.e. whether changes are information-preserving (no instance data is lost), translatable (instance data is not lost but must be translated into a new form), and information loss (some instance data may be lost). The full ontology of changes is shown in the Appendix to this document. It differs a little from the change set they describe in their paper, which is more of a frame-based approach. We present the simplified version here for the sake of clarity (as it highlights the most important aspects) and also because they do not specify all the effects for the full OWL-based ontology given in the Appendix.

Changes to Classes

- create class C
    - effect - no data is lost
- delete class C

  - effect - instances of C become instances of its superclass
- add subclass-superclass link between subclass SubC and a superclass SuperC
  - effect - subC inherits new slots from SuperC
- remove subclass-superclass link between subclass SubC and a superclass SuperC
  - effect - slots from instances of SubC are lost
- declare classes C1 and C2 as disjoint
  - effect - instances that belonged to both classes are now invalid
- move a superclass of class C to a class higher in the hierarchy
  - effect - C no longer has the slots that it inherited from its direct superclass, so any values for such slots become invalid
- move a superclass of class C to a class lower in the hierarchy
  - effect - C may inherit additional slots, but no data is lost
- reclassify an instance I as a class
  - effect: no data is lost
- reclassify a class C as an instance
  - effect: instances of C are less specifically typed
- merge 2 classes
  - effect - no data is lost if slot values are moved; slots of the new class will be the union of the two old classes
- split a class into 2 or more
  - effect - the operation must specify which of the new classes the instances of the old class should belong to

Changes to Slots

- create slot S
  - effect - no data is lost
- delete slot S
  - effect - values of S for all instances are lost
- attach slot S to class C
  - effect - no data is lost
- remove slot S from class C
  - effect - values of S for all instances are lost
- move a slot from class C1 to class C2

  - effect - no data is lost
- define a slot S as transitive or symmetric
  - effect - values that violate these restrictions are invalid
- move slot S from subclass subC to superclass superC
  - effect - class subC still inherits slot S so nothing is lost
- move slot S from superclass superC to subclass subC
  - effect - values of slot S for instances of superC are no longer valid
- widen a restriction for a slot S
  - effect - all existing slot values are still valid
- narrow a restriction for a slot S
  - effect - slot values may become invalid

3.2 Discussion

The main element missing from Klein and Noy's typology is the networked nature of ontologies. For example, the addition of a new ontology to the network would require the addition of new instances and possibly other changes such as relational information linking instances to different parts of the network. This aspect is dealt with in T1.3 (change propagation models), which deals with all change propagation, including changes at the level of the network; however we focus here on any changes which could affect the metadata. For the moment, we concentrate on the changes that stem from one ontology; networked ontology evolution will be part of future work.

The other difference between Klein and Noy's typology and our proposed framework is that they basically developed the change ontology for their own metamodel of OWL, which was built with different assumptions and design considerations that are partially incompatible with those of NeOn. For example the references to slots are more concerned with a frame-based model and are not really appropriate here, although many of them can be translated into changes that affect instances.

Finally, we also aim to provide a different kind of categorisation of the changes: for example, distinguishing between changes that stem from or affect the network, ontology, entity or instance. We also need to characterise the actions associated with the changes, rather than just specifying the information loss: for example, addition of a new concept in the ontology may require additional annotation in order to find the relevant metadata (instances); deletion of a concept may involve an automatic deletion of its relevant instances; merging two concepts requires merging of their instances, and so on.

3.3 Refined framework

We classify the changes according to those which stem principally from the ontology, concept or instance level. First we look at the effect on the instances caused by changes to the ontology / concepts. We describe the change and the effect on the data (largely as established by Klein and

Noy, though with some differences related to the frame-based vs OWL implementation), and propose the actions that should be taken. Note that our aim here is to attempt to specify the changes and actions that should take place – some of these will most likely occur as a matter of course (such as merging associated instances when their respective classes are merged).

It must be borne in mind that the change typology below represents an initial framework for capturing changes in ontologies. Its envisaged role is to serve as the basis for further discussion and a more formalised specification within tasks 1.5 and more importantly 1.3, the latter of which covers the whole spectrum of ontology change.

3.3.1 Changes Originating at the Ontology Level

First we look at some of the changes which involve changes to the relations between existing classes. In general, these do not affect the metadata, because instances are still valid, as shown in the following two scenarios.

- add subclass-superclass link between subclass SubC and a superclass SuperC
    - effect - instances of SubC should still be valid
    - action - no action needed
- remove subclass-superclass link between subclass SubC and a superclass SuperC
    - effect - instances of SubC should still be valid
    - action – no action needed

There are, however, cases where ontology-level changes may affect the instances

- declare classes C1 and C2 as disjoint
    - effect -- instances that belonged to both classes are now invalid
    - action – re-annotation will be necessary in order to decide which classes such instances should belong to
- declare classes C1 and C2 as equivalent
    - effect – no data loss
    - action – instances need to be mapped to the equivalent class and duplicates removed

3.3.2 Changes Originating at the Property Level

Similarly, most changes originating at the property level do not really affect the metadata as such: for example, changing a property means that a new property will be attached to the instance, but does not affect the instance per se.

- add a new property

    - effect - no data is lost
    - action – reannotation necessary to acquire new instances
- remove property
    - effect: data loss
    - action: instance should automatically inherit the superproperty

3.3.3 Changes Originating at the Concept Level

Changes occurring at the concept level are the ones most likely to influence the metadata: in some cases, the instances may have to be moved to new classes; in other cases, reannotation may be required in order to acquire new instances (e.g. when new classes are added to the ontology). This is described further in Section 3.4

- add superclass C
    - effect - no data is lost
    - action – reannotation necessary to acquire new instances of C
- add subclass C
    - effect - no data is lost
    - action – reannotation necessary to acquire new instances of C
- add disjoint class C
    - effect - no data is lost
    - action - reannotation necessary to acquire new instances of C
- remove disjoint class C
    - effect - instances attached to the class are lost
    - action – move instances to superclass
- modify disjoint class C to superclass
    - effect - no data is lost
    - action – move instances to superclass
- modify disjoint class C to subclass
    - effect - no data is lost
    - action – move instances to subclass
- add equivalent class C
    - effect - no data is lost
    - action - instances belonging to equivalent class must be automatically added to C

- modify equivalent class C to superclass
    - effect - no data is lost
    - action – instances must be removed from C (but not from its equivalent class)
- modify equivalent class C to subclass
    - effect - no data is lost
    - action - instances must be removed from C (but not from its equivalent class) Reannotation will also be necessary
- remove equivalent class C
    - effect - no data is lost
    - action - automatic - instances belonging to C should be removed
- remove superclass C
    - effect - instances of C are lost
    - action - instances of C must be moved to their superclass; properties inherited from C are removed.
- remove subclass C
    - effect - instances of C are lost
    - action - instances of C must be moved to their superclass
- modify superclass to subclass
    - effect - no data loss
    - action - instances must be moved to the subclass
- modify subclass to superclass
    - effect - no data loss
    - action - instances must be moved to the superclass
- reclassify a class C as an instance
    - effect – instances of class C are lost
    - action – instances of class C must be moved to the superclass
- merge 2 classes
    - effect - instances of these classes are lost
    - action – instances must be moved to the new class and any duplicates removed
- split a class into 2 or more

- effect - instances must be moved
- action – instances must be reclassified via reannotation

3.3.4 Changes Originating at the Instance Level

Note that these represent the class "Individual_change" in Klein's ontology.

- add an instance I
  - effect -- none
  - action -- none
- delete an instance I
  - effect – data loss
  - action – remove any equivalent instances
- reclassify an instance I as a class
  - effect – no data is lost
  - action – re-annotation to acquire instances of the new class
- add an equivalent instance I
  - effect: no data lost
  - action: none
- remove an equivalent instance I
  - effect: no data lost
  - action: none

3.3.5 Complex operations

We do not specify here the complex operations, as the relevant actions can be derived from the actions for the basic operations.

3.4 Planned Implementation

The actions shown in section 3.3 resulting from changes to metadata can be divided into a set of 3 possible types:

1. no action is needed;
2. a manual process is required;
3. some automatic process should take place.

The first action is self-explanatory and requires no further discussion. In some cases, a degree of data loss is inevitable.

The second action almost always requires some kind of re-annotation of the corpus. This can be for several reasons:

i. Existing information in the ontology needs to be **reclassified**. This could be the case where a new subclass is added. Imagine we have an ontology which contains the class comestible and the subclass food. Now imagine we add a new subclass of comestible, drink. Before this subclass was added, all instances of drink that were not also instances of food would have been classified simply as comestible, because that was the most specific class to which they could belong in the ontology. Once drink is added to the ontology, such instances need reclassifying under the drink class. However, this is almost impossible to do automatically because we have no way of knowing which instances of comestible should be moved and which should not, unless we return to the text for further analysis.

ii. Information is **missing** from the ontology. This could be the case when a new top class is added to the ontology, or when an instance is reclassified as a class. Imagine that drink was previously in the ontology as an instance of comestible, and is changed to be a subclass of comestible[1]. In this case, there would probably have been no types of drink (such as wine, water etc.) present so far in the ontology, otherwise drink would not have been classified as an instance. So these instances, if present in the data, would need to be added as new instances of drink.

iii. It may also be a combination of the two factors. For example, in the case where a new superclass is added to the ontology, some information may be missing and some may need reclassification.

While missing instances can simply be added to the appropriate place in the ontology when found in the text, reclassification is a little more tricky because it consists of a two-stage process: first the system must find the instances in the text and recognise that they are currently not classified in the most appropriate way in the ontology, and second, it must follow the automatic procedure for reclassification as specified in action (3) below.

The third action requires a set of procedures to be followed for automating the reclassification of instances in the ontology. Below we give an example of some such possible procedures: naturally the exact procedures will be implementation-specific according to the ontology model used.

- When a class is added
  - It will automatically inherit from its superclasses a set of properties
- When an equivalent class is added
  - It will inherit all the instances from its equivalent class
  - These instances will all have a sameIndividualAs statement added
- When an instance is added
  - It will automatically inherit from its superclasses a set of properties.
- When a class is deleted
  - A list of all its superclasses is obtained. For each class in this list, a list of its subclasses is obtained and the deleted class is removed from it.
  - All subclasses of the deleted class are moved to subclasses of the parent of the deleted superclass. A list of all its disjoint classes is obtained. For each subclass in

[1] We imagine that *drink* was originally (mis)classified as an instance and we wish to now change it to be a class.

this list, a list of its disjoint classes is obtained and the deleted class is removed from it.
    - A list of all its sameAs classes is obtained. For each class in this list, a list of its sameAs classes is obtained and the deleted class is removed from it.
    - A list of all its differentFrom classes is obtained. For each class in this list, a list of its differentFrom classes is obtained and the deleted class is removed from it.
    - All instances of the deleted class are moved to their direct superclass in the ontology.
- When an instance is deleted
    - A list of all its sameIndividualAs instances is obtained. For each instance in this list, a list of its sameIndividualAs instances is obtained and the deleted instance is removed from it.
    - A list of all its differentFrom instances is obtained. For each instance in this list, a list of its differentFrom instances is obtained and the deleted instance is removed from it.
- When a property is deleted
    - A list of all its super properties is obtained. For each property in this list, a list of its sub properties is obtained and the deleted property is removed from it.
    - All sub properties of the deleted property are removed from the ontology.
    - A list of all its sameAs properties is obtained. For each property in this list, a list of its sameAs properties is obtained and the deleted property is removed from it.
    - A list of all its differentFrom properties is obtained. For each property in this list, a list of its differentFrom properties is obtained and the deleted property is removed from it.
    - All instances of the ontology are checked to see if they have the deleted property set on them. If so the respective property is deleted.

4. Ontology evolution implied by changes in the metadata

We consider here the scenarios in which a set of documents associated with metadata are evolving. Metadata is generally used to index documents and resources and to describe their content, in order to facilitate their search, exploitation and management. In these scenarios, the use of ontologies allows for more expressive descriptions by relying on the semantics of the employed terms, enabling powerful mechanisms for metadata management and exploitation (content-based access, disambiguation, semantic search, etc.) Unfortunately, documents and metadata can evolve faster than ontologies, which are supposed to be stable representations of the considered domain: new documents are added, containing new concepts and relations between them, or documents become obsolete thus reducing the need for certain ontology elements that are only characteristic for them. These evolutions of the metadata lead to an important gap between the elements that need to be described and the knowledge contained in the ontologies. In this section, we focus on the definition of a methodology for capturing this gap and the proposal of the necessary changes, thereby providing a metadata-driven evolution, or maintenance, of ontologies.

In this section, we consider the notion of metadata according to a general definition: information about the content of a resource or a document. Resources and documents can take different forms, (texts, images, etc.), and the associated metadata can be either manual annotations (e.g. folksonomy tags given by the author of an image) or automatically extracted (e.g. using tools for information extraction from texts). In the case of semantic annotations, metadata is represented according to ontology elements. The goal of this section is to show how ontologies can evolve in accordance with the evolution of metadata associated with documents of the domain. The basis of the proposed mechanism consists of relating evolving (non-semantic) metadata with the considered ontologies, in order to assess the insufficiencies of these ontologies in representing the considered metadata as semantic annotations. The process of suggesting changes in an ontology based on changes in the underlying annotated dataset was defined as data-driven change in [6]. It can be argued that the methodology described here is data-driven rather than metadata-driven, in the sense that, in most of the cases, metadata evolves as a consequence of the evolution of the underlying data. However, in the case of manual annotation, for example, the annotator may change the metadata associated with a document, without changing the document itself.

The proposed methodology can be considered as a *bottom up* approach for two reasons. First, the basic principle is that changes in metadata would guide the evolution of the ontologies. Second, the definition of the methodology itself is based on “experimentations” using real life datasets: by studying the evolution of these datasets through the associated metadata, we aim at defining general principles for a metadata-driven ontology evolution. Therefore, we plan to apply this methodology to two concrete case studies using two different datasets and two different forms of metadata: the evolution of ontologies depending on changes in folksonomies and depending on changes in textual documents in a given domain. The methodology itself is presented in Section 4.1 and the considered case studies are described in the following sections (4.2 and 4.3). The actual experiments and tools will be implemented as a part of the next deliverable on dynamics of metadata (D1.5.2).

4.1 Methodology

This section presents our approach for studying changes in metadata and deriving suggestions of changes in the corresponding ontologies. More precisely, we describe a method to capture (1) the evolution of different types of metadata and (2) the implied changes in the related (networked) ontologies. The proposed methodology relies on a bottom-up approach for capturing the relations between metadata evolution and ontology changes. First, the data contained in a set of existing

documents/resources has to be automatically related to the considered ontologies, in order to form an exploitable corpus of ontology-based metadata. Metadata changes can then be captured on the basis of the generated structure, and the study of the implications of these changes on ontologies can be carried out, with the aim of deriving suggestions of corresponding ontology evolutions.

Obtaining this exploitable corpus of metadata, linking the resources to the ontologies, leads to several common, generic or domain-specific issues:

- **Pre-processing:** First, the considered documents, resources and metadata can take different forms, and may contain noise, redundancies, etc. A pre-processing step is generally required in order to obtain an exploitable corpus, including domain (or data)-dependent tasks such as filtering, transformation, etc. The result of this step should be a common, simple, and easy to process representation of the metadata. The simplest possible form of representation, applicable in most of the cases, is *sets of terms*. Therefore, the *pre-processing* step results in sets of filtered terms associated with the corresponding documents (or occurrences in the documents). Specific filtering and transformation tasks are described in the following sections for the considered case studies.
- **Conceptual Organisation:** Once a set of descriptive terms is obtained for each of the considered documents, we need to identify from among these terms those that represent important domain concepts, which may be contained in the ontologies. More importantly, potential relations between these terms, or more precisely between the corresponding concepts, have to be detected. One possibility is to group (or cluster) the terms according to their relations in the documents, suggesting in this way potential relations in the ontologies. In the next section (4.2), a clustering technique is proposed to group the tags of folksonomies according to their co-occurrence. Similarly, many ontology learning methods, after a pre-processing step which leads to a set of initial terms, apply a variety of methods to organise these terms into taxonomies and to derive relations between them [9].
- **Ontology Matching:** Finally**,** the links between the obtained conceptual structure and the considered ontologies have to be established. This can be realised by using generic ontology matching techniques [10], mapping the organised terms to ontology concepts, and relating them according to ontology relations. It is worth noticing that these mappings already provide indications of missing knowledge in the ontologies: the terms and relations for which there is no mapping suggest missing concepts and relations.

These three tasks provide the basic structure on which we rely for capturing metadata changes and the related ontology evolutions.

4.1.1 Capturing metadata changes

The evolution of the metadata can be captured through the changes occurring in the corresponding set of terms, conceptual structures, and mappings. Documents can be added or removed, leading to a different (either extended or reduced) set of terms. The comparison of the results of the conceptual organisation of the terms obtained at different times allows the consideration of changes from a more abstract (conceptual) point of view, indicating for example previously unknown relations between terms, or ones that have become obsolete. It is worth noticing here that these conceptual structures are not ontologies, but rather intermediary descriptions of the metadata, used to guide the integration (the matching) with the considered ontologies, and facilitating the study of metadata evolution. Finally, new mappings to the ontologies may appear from the conceptual structure, and some may be modified or reconsidered. Basically, by comparing the processed metadata at different levels and different time points, we can trace, capture and study these evolutions in an abstract, ontology-related representation. The outcome

here is a characterisation of the changes in the metadata that may lead to particular evolution of ontologies.

4.1.2 Suggesting ontology changes

Capturing the evolution of the metadata is only the first part of the problem. Each evolution may lead to an extension of the existing gap between the metadata and the ontology coverage. For example, if some newly added terms have no correspondence in the ontologies, an extension with additional concepts and relations may be required. On the other hand, if the set of terms generated at any moment in time is smaller than the set of terms generated previously (e.g., because some data has been deleted or because some terms are less significantly represented) then this can lead to the pruning of the ontologies in such a way that they do not contain obsolete concepts/relations. Our goal is to provide general principles for suggesting changes in ontologies in order to fill this additional gap. More precisely, studying the evolution of real-life datasets would allow us to associate suggestions of changes in the ontology to "typical" changes in the captured metadata, thus providing a basis for guiding the maintenance of ontologies according to the related metadata.

4.1.3 Summary of the methodology

Figure 1 summarises the proposed methodology in a schematic view. The left part concerns the establishment of mappings between the metadata and the ontologies. The right part shows how, by studying the changes occurring in the metadata and their consequences on the mappings, we can suggest changes in the ontologies. This method relies on two main elements: the initial metadata set and a set of ontologies that are supposed to be related to the considered metadata. First, a mapping is created between the metadata and the ontologies, using the three steps described above: pre-processing, conceptual organisation, and ontology matching. By using the same procedure, the changes occurring in the metadata can be captured in terms of changes in the derived conceptual structure between the extracted terms, and changes in the mappings with the ontologies. We can then study how the evolution of the metadata has extended (or maybe reduced) the gap between the knowledge contained in the ontologies and the elements described by the metadata, suggesting possible evolutions of the ontologies in order to fill this gap.

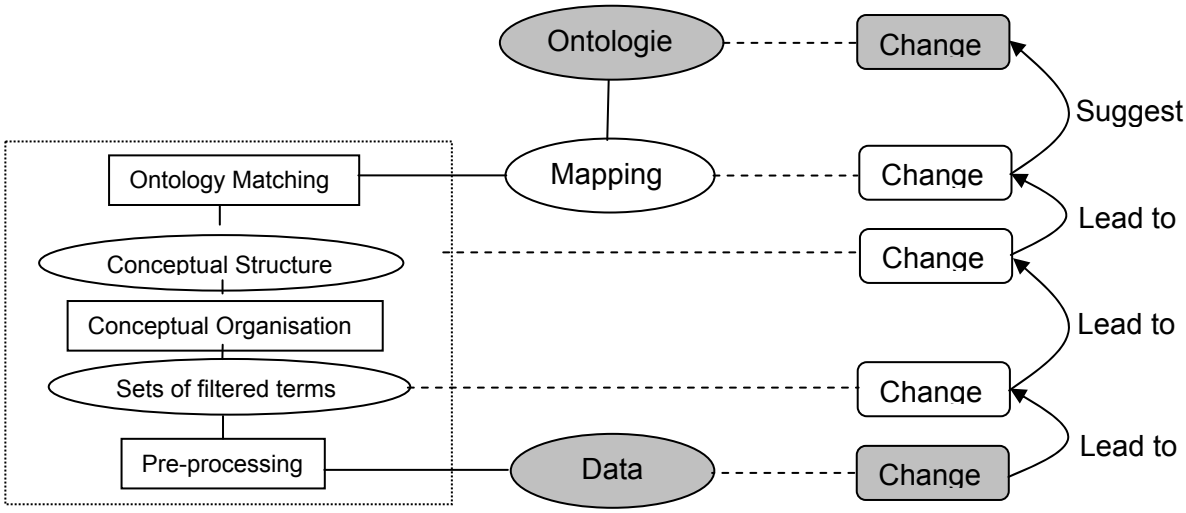


Figure 1: Schematic view of the bottom-up approach for ontology evolution

4.2 Case Study 1: Evolving ontologies through linking them to folksonomies

Social tagging systems such as Flickr (http://www.flickr.com/), for photo-sharing, and Del.icio.us (http://del.icio.us/), for social bookmarking, are becoming more and more popular, nowadays covering a wide range of resources and communities, with a huge number of participants sharing and tagging a large number of resources. For example, Del.ici.ous was said to have more than a million registered users in September 2006, who had been posting more than 100,000 bookmarks each day (http://deli.ckoma.net/stats). The collection of all tags present in a tagging system forms a folksonomy.

Due to their popularity, folksonomies are changing rapidly as users add new resources and tags. Because they are updated continuously, folksonomies are up-to-date with respect to the vocabulary used by a wide range of people, thus reflecting new terms that appear. Unlike folksonomies, ontologies are built at a much slower rate and therefore they often lag behind the novel terminology in a given domain. A solution for automatically enriching (and hence evolving) ontologies is to align them to folksonomies and modify them so that they reflect the changes in the folksonomies. This alignment is also beneficial for folksonomies. Current tag sets lack any semantic relations and therefore are hard to use during searching. The alignment to ontologies allows the enrichment of folksonomies with semantic relations so that more semantic searches can be performed.

In this first case study, we investigate how ontology changes can be derived from changes in folksonomies. The process of linking folksonomies to ontologies has been described in [11] and it will be used in the first part of the case study (as it covers the first three major steps of our methodology). For this reason, we briefly summarise here the techniques proposed in [11].

4.2.1 Data Set

We investigate the tag sets in Flickr and Del.icio.us, due both to their popularity (with a large number of resources, users, and tags) and availability. In our experiments, we use the same Del.ici.ous tags as [12] and Flickr tags for photos posted between 01-02-2004 and 01-03-2006. The numbers of resources, users, and tags in both datasets, as well as the distinct number of each of these elements, are shown in Table 1.

	Total		**Distinct**		
	users / resources	*tags*	*users*	*resources*	*tags*
Delicious	19,605	89,978	7,164	14,211	11,960
Flickr	49,087	167,130	6,140	49,087	17,956

Table 1: Original numbers of users, resources and tags extracted from Del.icio.us and Flickr

4.2.2 Pre-processing

The pre-processing stage proposed in [11] reduces the set of all tags to a set of normalised, representative tags. For this a set of tags are discarded if (i) they are unusual (non-alphabetic), (ii) they are morphologically similar to other tags (by grouping tags such as *cat* and *cats*, *san_francisco*, *sanfrancisco* and *san.francisco*), (iii) they appear less than a certain number of times (infrequent tags).

4.2.3 Conceptual Organisation

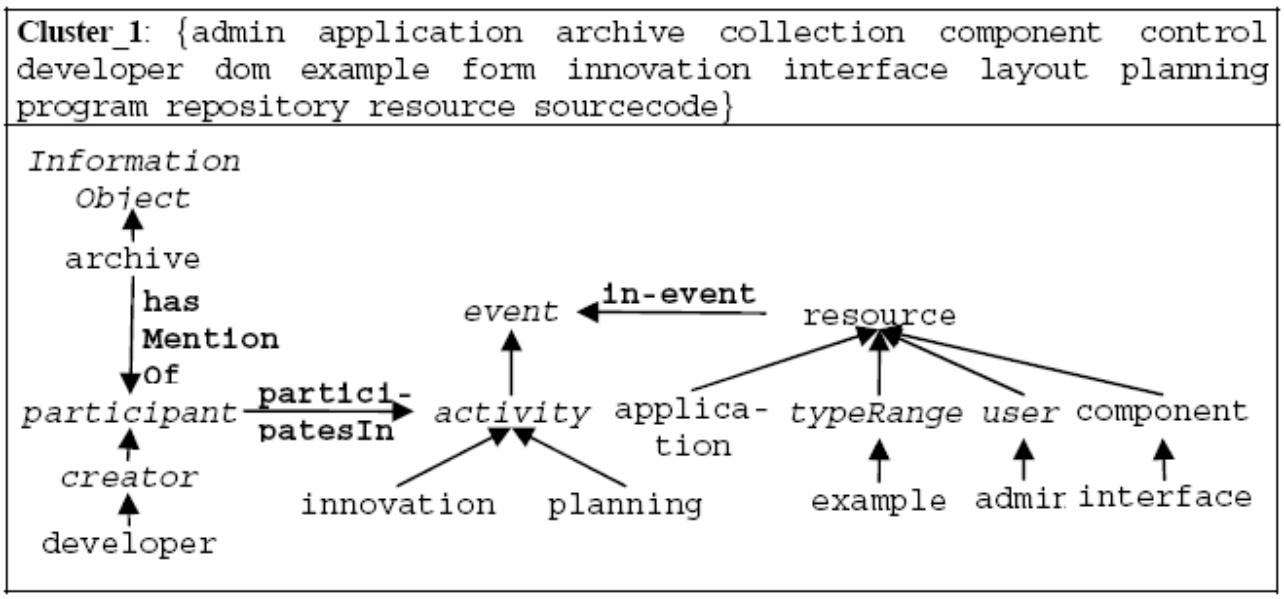
Once relevant tags are identified, the next step is to perform a statistical analysis of the tag space in order to identify groups, or clusters, of possibly related tags. Clustering is based on the similarity between tags, given by their co-occurrence. The result of this step is a set of clusters each containing tags that are likely to be connected because they appear often in similar contexts. Table 2 (reproduced from [11]) provides some examples of tag clusters.

Del.icio.us data
{author books literature}
{bicycle bike courier cycling}
{bingo blackjack casino gamble gambling keno poker roulette slots}
{bookmark bookmarking folksonomy social tagging tags}
{browse extension firefox mozilla thunderbird}
{aim chat gtalk icq instant jabber messaging messenger msn yahoo}
Flickr data
{activism anarchism banner brutality demonstration eu globalization gothenburg police protest riots summit syndicalism worker}
{backpacking hot humid iguacu lush rainforest waterfall wilderness}
{damage flooding hurricane katrina Louisiana}
{bosnia europe herzegovina Sarajevo}
{apple ibook mac ipod macintosh powerbook}

Table 2: Examples of clusters found for Del.icio.us and Flickr data

4.2.4 Ontology Matching

The clusters (conceptual structures) obtained previously are matched to ontologies available online in order to establish semantic relations between the tags. Examples of semantic relations obtained between the tags of clusters obtained by querying several online ontologies are shown below in Figure 2 (reproduced from [11]). The mapping should take into account structural properties of mappable elements, such as compounding, and semantic ambiguity of source or target concept.



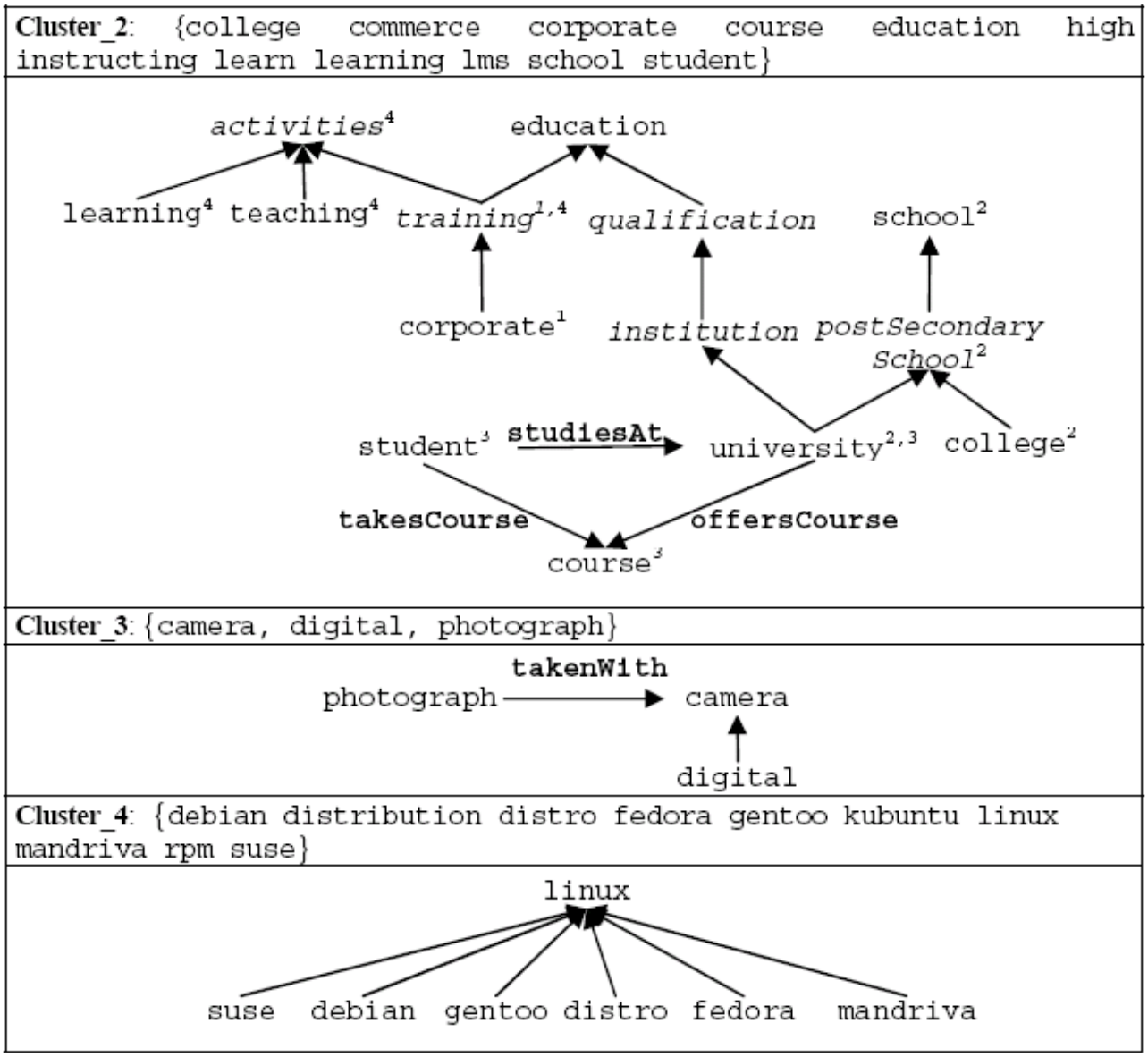


Figure 2: Examples of semantic relations obtained

4.2.5 Capturing metadata changes

While the steps described before (dealing mainly with deriving the metadata) will be implemented using work described in [11], the step of capturing the way metadata changes will require novel work. Our plan is to re-run the process described above at different points in time and compare the output (i.e., see how the obtained clusters differ). We will identify clusters that have been added or clusters which have disappeared. At a more fine-grained level, for each cluster we will monitor changes within the cluster, namely additions or elimination of some tags. All these types of changes have important influences on the evolution of the ontology. Our task will be to derive a wide range of metadata changes taking into account the metadata derived in our experiments.

4.2.6 Suggesting ontology changes

Once a set of changes in the metadata has been identified, these changes can be used to suggest corresponding updates to the ontologies. While this typology of changes will be derived when the experiments are run and therefore will be grounded in actual data, we can already predict some typical changes. For example:

A new cluster is added: then find an ontology that contains elements (concepts/relations) with the same label as the tags in the cluster. If the ontology only covers a subset of the terms then find methods to extend the ontology with concepts corresponding to the missing tags.

A new tag is added to a cluster: if no corresponding concept exists in the ontology to which the cluster was aligned, then insert the tag in the ontology.

4.3 Case Study 2: Data Driven Ontology Learning on FAO data

The second case study is performed on a different type of data set from the previous case study. In this case, we investigate how ontologies can be evolved through applying ontology learning methods to textual data. The data that we are working on are made available by FAO and subject to experimentation in WP7.

4.3.1 Data Set

The data set used is a collection of textual data used for the experiments in task T7.3. There are several collections, including news items, fact sheets on species and internal documentation.

4.3.2 Pre-processing

We envisage that pre-processing will be performed using existing software packages. In particular, we will experiment with three tools:

TermExtractor (University of Roma La Sapienza) - a web service for the extraction of domain relevant terminology

GATE [4] (University of Sheffield) - a suite of natural language applications for document annotation.

Text2Onto [13] (University of Karlsruhe) – an ontology learning tool that provides a variety of algorithms needed for the entire process of ontology learning. Some of these algorithms deal with the simplest task in ontology learning, that of extracting relevant terms from a corpus.

4.3.3 Conceptual Organisation

The conceptual organisation step aims at deriving some meaningful structure between the identified terms. Again, we envisage the possibility to use two different kinds of packages to perform this task:

Clustering algorithms – when TermExtractor is used, we can use the extracted terms and their co-occurrence information from the analysed documents as input for the clustering mechanism described in Section 4.2.3. This would lead to clusters similar to those obtained in case study 1.

Text2Onto – we can use the ontology learning algorithms from Text2Onto to derive an ontological structure between the identified terms (including subsumption relations, mereological relations, general relations).

4.3.4 Ontology Matching

Depending on the conceptual structure derived previously, we can use a variety of ontology matching approaches to align these structures to ontologies. Simple, string-similarity based methods can be used to find correspondences between terms in a cluster and the labels of ontology elements. If the derived structures are richer than sets of terms, we can also employ structural matching methods to find alignments between the derived ontologies and the ontologies (ontology networks) that need to be updated.

4.3.5 Capturing Metadata Changes and Suggesting Ontology Changes

Capturing metadata changes and linking them to updates in the corresponding ontologies will be the topic of our investigation after running the above mentioned experiments. We can already say, however, that these types of changes will somehow depend on the kind of conceptual structure that the metadata takes. If we derive clusters of terms, then we can provide a similar typology of changes as in case study 1. If the conceptual structures derived are ontologies, then we can re-use and extend Klein's typology of changes to capture changes between ontologies derived at different moments in time and then suggest adequate changes in the ontologies.

5. Conclusions and Future Work

In this deliverable we have described a framework and methodology to capture the dynamics of metadata, which is the main task of T1.5. This consists of two main approaches: the evolution of metadata when an ontology changes, and changes to the ontology resulting from metadata evolution. The first approach involves using the Klein and Noy ontology as a starting point, and proposing changes in order to make it more suitable for our needs. We then discuss how this will be implemented in NeOn. In Section 3.4 we have detailed the current state of the implementation of the approach and outlined our future plans for integrating our framework in GATE.

In section 4, we have described a methodology for capturing changes in metadata with the aim of suggesting changes in the related ontologies. We adopt a bottom-up approach, studying the evolution of real-life datasets to come up with general principles for metadata-driven ontology evolution. In this line of idea, we proposed to apply this methodology on two concrete case studies, using two different datasets, and two different forms of metadata: tags of folksonomies and texts related to the agricultural domain (provided by FAO). Therefore, the obvious next step of this work is to run the experiments corresponding to these case studies, capturing the evolution of the metadata by relating the considered resources to the considered ontologies at different points in time. We believe that the study of these ontology-based metadata evolutions will allow us to understand and potentially capture the implications of metadata changes on the related ontologies. The concrete outcome of this work should be general rules for suggesting ontology changes to reflect metadata changes, providing a basis for a metadata-driven ontology evolution.

As we have discussed, the first part of this deliverable is concerned with the metadata changes led by ontology changes, i.e. the inverse process of the one described in the second part. In the first part, we rely on a typology of ontology changes as shown in section 3. Since metadata changes lead to ontology changes, and ontology changes lead to metadata changes, the interactions between these two processes have to be considered. We can therefore imagine an integrated process that would alternatively suggest changes in ontologies and metadata until a stable version of both elements is obtained.

Other directions for the continuation of this work concern the link to other studies realised within NeOn:

- **WP1: Task 1.1.** Several other tasks in WP1 are also related to T1.5. For example, changes suggested according to metadata evolution should be represented in a way that matches the NeOn meta-model produced in Task 1.1, in particular for the part concerning *versioning*.
- **WP1: Task 1.3:** The first version of the change typology described in Section 3 is very relevant for T1.3 in general, because it impacts on the change propagation models developed there. Efforts will therefore be harmonised in these two tasks, as the change typology developed here should be formalised and implemented fully into the NeOn model in T1.3. Similarly, the work presented in Section 4, which can be formulated as the study of the propagation of changes from metadata to ontologies, is clearly also relevant for T1.3 and should be integrated.
- **WP2:** The work on folksonomies described in the first case study (Section 4.2) is closely related to the data re-engineering work in WP2, where tag clusters obtained from folksonomies will be semantically enriched (i.e. semantic relations between tags will be discovered). The proposed work in the current task, on the other hand, will reuse tag clusters generated at different points in time and attempt to evolve ontologies based on the differences between these clusters. So while in WP2 the main emphasis will be on the

enrichment methods, here the focus will be on the development of the typology of changes and derived ontology suggestions, based on the case study.

- **WP7:** Our second case study is obviously related to the NeOn case study considered in WP7, in particular concerning the life-cycle management of the fishery ontology. We plan to provide tools for the maintenance of the existing ontology, on the basis of an evolving document repository owned by FAO. Moreover, this task may rely on tools – in particular, ontology learning systems – that are currently being experimented with as part of Task 7.3.

6. References

[1] J. Hartmann, Y. Sure, P. Haase, R. Palma, M. del Carmen Suárez-Figueroa (2005), OMV - Ontology Metadata Vocabulary In Chris Welty, ISWC 2005 - In Ontology Patterns for the Semantic Web. November 2005.

[2] D.Maynard, K. Bontcheva and H. Cunningham. Towards a semantic extraction of named entities. *Recent Advances in Natural Language Processing*, Bulgaria, 2003.

[3] D. Maynard, M. Yankova, A. Kourakis and A. Kokossis (2005). Ontology-based information extraction for market monitoring and technology watch, ESWC Workshop "End User Apects of the Semantic Web", Heraklion, Crete, 2005.

[4] K. Bontcheva, V. Tablan, D. Maynard, H. Cunningham (2004). Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*. **10** (3/4), pp. 349-373. 2004.

[5] M. Klein and N. F. Noy. (2003), A Component-Based Framework For Ontology Evolution. In: Proceedings of the Workshop on Ontologies and Distributed Systemsm, IJCAI '03, Acapulco, Mexico, August 9, 2003

[6] D. Maynard, W. Peters, Y. Li (2006). Metrics for Evaluation of Ontology-based Evaluation In: Proceedings of the 4th International EON workshop on the Evaluation of Ontologies on the Web, Edinburgh, May 22, 2006

[7] N. F. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems, 5, 2003.

[8] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic (2003). User-driven ontology evolution management. In European Conf. Knowledge Eng. and Management (EKAW 2002), pages 285--300. Springer-Verlag, 2002.

[9] J. Völker, D. Vrandečić, Y. Sure (2006), Data-driven Change Discovery – Evaluation, SEKT Deliverable D3.3.2

[10] J. Banerjee et al. (1987). Semantics and implementation of schema evolution in object-oriented databases. *Proc of SIGMOD Conference*, 1987.

[11] L.Specia, E.Motta. Integrating Folksonomies with the Semantic Web. Submitted for peer-review (2007)

[12] Mika, P. (2005). Ontologies are us: A unified model of social networks and semantics. ISWC'2005.

[13] P. Cimiano, J. Volker. Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery In Andres Montoyo, Rafael Munoz, Elisabeth Metais, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of Lecture Notes in Computer Science, pp. 227-238. Springer, Alicante, Spain, June 2005.

[14] P. Shvaiko and J. Euzenat. A Survey of Schema-based Matching Approaches. Journal on Data Semantics, IV, 2005.

7. Appendix: Klein and Noy's Full Ontology

Change

- Addition
  - Add_Comment
  - Add_Disjoint_Class
  - Add_Domain
  - Add_Equivalent_Class
  - Add_Equivalent_Individual
  - Add_Equivalent_Property
  - Add_Higherbound
  - Add_Inverse_Property
  - Add_Label
  - Add_Lowerbound
  - Add_Range
  - Add_Restriction
  - Add_Superclass
  - Add_Superproperty
- Atomic_Change
  - Class_Change
    - Class_Disjointness_Change
      - Add_Disjoint_Class
      - Modify_Disjoint_Class
        - Modify_Disjoint_Class_To_Subclass
        - Modify_Disjoint_Class_To_Superclass
      - Remove_Disjoint_Class
    - Class_Equivalence_Change
      - Add_Equivalent_Class
      - Modify_Equivalent_Class
        - Modify_Equivalent_Class_To_Subclass
        - Modify_Equivalent_Class_To_Superclass
      - Remove_Equivalent_Class
    - Restriction_Change
      - Add_Restriction
      - Cardinality_Change
        - Extend_Cardinality
        - Higherbound_Change
          - Add_Higherbound
          - Modify_Higherbound
            - Decrease_Higherbound
            - Increase_Higherbound
          - Remove_Higherbound
        - Lowerbound_Change
          - Add_Lowerbound
          - Modify_Lowerbound
            - Decrease_Lowerbound
            - Increase_Lowerbound
          - Remove_Lowerbound
        - Restrict_Cardinality
      - Modify_Restriction_Filler
        - Modify_Restriction_Filler_To_Subclass
        - Modify_Restriction_Filler_To_Superclass
      - Remove_Restriction
      - Restriction_Type_Change
        - Change_To_Existential_Restriction
        - Change_To_Universal_Restriction

- Superclass_Change
  - Add_Superclass
  - Modify_Superclass
    - Modify_Superclass_To_Subclass
    - Modify_Superclass_To_Superclass
  - Remove_Superclass
- Individual_Change
  - Individual_Equivalence_Change
    - Add_Equivalent_Individual
    - Modify_Equivalent_Individual
    - Remove_Equivalent_Individual
  - Modify_Indivual_Type
    - Modify_Indivual_Type_To_Subclass
    - Modify_Indivual_Type_To_Superclass
- Ontology_Change
  - Ontology_Addition
    - Add_Class
    - Add_Individual
    - Add_Property
  - Ontology_Removal
    - Remove_Class
    - Remove_Individual
    - Remove_Property
- Property_Change
  - Domain_Change
    - Add_Domain
    - Modify_Domain
      - Extend_Domain
      - Restrict_Domain
    - Remove_Domain
  - Property_Equivalence_Change
    - Add_Equivalent_Property
    - Modify_Equivalent_Property
      - Modify_Equivalent_Property_To_Subproperty
      - Modify_Equivalent_Property_To_Superproperty
    - Remove_Equivalent_Property
  - Property_Inverse_Change
    - Add_Inverse_Property
    - Modify_Inverse_Property
      - Modify_Inverse_Property_To_Subproperty
      - Modify_Inverse_Property_To_Superproperty
      - Remove_Inverse_Property
  - Property_Type_Change
    - Change_To_DatatypeProperty
    - Change_To_ObjectProperty
    - Set_Functionality
    - Set_InverseFunctionality
    - Set_Symmetry
    - Set_Transtivity
    - Unset_Functionality
    - Unset_InverseFunctionality
    - Unset_Symmetry
    - Unset_Transtivity
  - Range_Change
    - Add_Range
    - Modify_Range
      - Extend_Range
      - Restrict_Range
    - Remove_Range
  - Superproperty_Change

- Add_Superproperty
- Modify_Superproperty
  - Modify_Superproperty_To_Subproperty
  - Modify_Superproperty_To_Superproperty
- Remove_Superproperty
- Resource_Change
  - Comment_Change
    - Add_Comment
    - Modify_Comment
    - Remove_Comment
  - Label_Change
    - Add_Label
    - Modify_Label
    - Remove_Label

- Composite_Change
- Modification
  - Extend_Cardinality
  - Modify_Comment
  - Modify_Disjoint_Class
    - Modify_Disjoint_Class_To_Subclass
    - Modify_Disjoint_Class_To_Superclass
  - Modify_Domain
    - Extend_Domain
    - Restrict_Domain
  - Modify_Equivalent_Class
    - Modify_Equivalent_Class_To_Subclass
    - Modify_Equivalent_Class_To_Superclass
  - Modify_Equivalent_Individual
  - Modify_Equivalent_Property
    - Modify_Equivalent_Property_To_Subproperty
    - Modify_Equivalent_Property_To_Superproperty
  - Modify_Higherbound
    - Decrease_Higherbound
    - Increase_Higherbound
  - Modify_Indivual_Type
    - Modify_Indivual_Type_To_Subclass
    - Modify_Indivual_Type_To_Superclass
  - Modify_Inverse_Property
    - Modify_Inverse_Property_To_Subproperty
    - Modify_Inverse_Property_To_Superproperty
  - Modify_Label
  - Modify_Lowerbound
    - Decrease_Lowerbound
    - Increase_Lowerbound
  - Modify_Range
    - Extend_Range
    - Restrict_Range
  - Modify_Restriction_Filler
    - Modify_Restriction_Filler_To_Subclass
    - Modify_Restriction_Filler_To_Superclass
  - Modify_Superclass
    - Modify_Superclass_To_Subclass
    - Modify_Superclass_To_Superclass
  - Modify_Superproperty
    - Modify_Superproperty_To_Subproperty
    - Modify_Superproperty_To_Superproperty
  - Restrict_Cardinality
  - Restriction_Type_Change
    - Change_To_Existential_Restriction
    - Change_To_Universal_Restriction

- Removal
  - Remove_Comment
  - Remove_Disjoint_Class
  - Remove_Domain
  - Remove_Equivalent_Class
  - Remove_Equivalent_Individual
  - Remove_Equivalent_Property
  - Remove_Higherbound
  - Remove_Inverse_Property
  - Remove_Label
  - Remove_Lowerbound
  - Remove_Range
  - Remove_Restriction
  - Remove_Superclass
  - Remove_Superproperty
- Utility_Classes
  - Role