The Knowledge Engineering Review, Vol. 00:0, 1–29. © 2007, Cambridge University Press
DOI: 10.1017/S0000000000000000 Printed in the United Kingdom

Ontology change: classification and survey

GIORGOS FLOURIS[1,2], DIMITRIS MANAKANATAS[2], HARIDIMOS KONDYLAKIS[2,3], DIMITRIS PLEXOUSAKIS[2,3] and GRIGORIS ANTONIOU[2,3]

[1] *Istituto della Scienza e delle Tecnologie della Informazione, C.N.R.*
Via Giuseppe Moruzzi, 1, 56124, Pisa, Italy
E-mail: flouris@isti.cnr.it
[2] *Institute of Computer Science, FO.R.T.H.*
P.O. Box 1385, GR 71110, Heraklion, Greece
E-mail: fgeo@ics.forth.gr, manakan@ics.forth.gr, kondylak@ics.forth.gr, dp@ics.forth.gr, antoniou@ics.forth.gr
[3] *Computer Science Department, University of Crete*
P.O. Box 2208, GR 71409, Heraklion, Greece
E-mail: kondylak@csd.uoc.gr, dp@csd.uoc.gr, ga@csd.uoc.gr

Abstract

Ontologies play a key role in the advent of the Semantic Web. An important problem when dealing with ontologies is the modification of an existing ontology in response to a certain need for change. This problem is a complex and multifaceted one, because it can take several different forms and includes several related subproblems, like heterogeneity resolution or keeping track of ontology versions. As a result, it is being addressed by several different, but closely related and often overlapping research disciplines. Unfortunately, the boundaries of each such discipline are not clear, as the same term is often used with different meanings in the relevant literature, creating a certain amount of confusion. The purpose of this paper is to identify the exact relationships between these research areas and to determine the boundaries of each field, by performing a broad review of the relevant literature.

1 Introduction

Originally introduced by Aristotle, *ontologies* are formal models about how we perceive a domain of interest and provide a precise, logical account of the intended meaning of terms, data structures and other elements modeling the real world. As such, they are often viewed as the key means through which the Semantic Web vision (Berners-Lee *et al.* 2001) can be realized and have already found several applications in the area of Knowledge Representation (KR) and in the Semantic Web. Ontologies are so important in the Semantic Web because they provide a means to formally define the basic terms and relations that comprise the vocabulary of a certain domain of interest (Lambrix & Edberg 2003), enabling machines to process information provided by human agents. As a result, ontologies can help in the representation of the content of a web page in a formal manner, so as to be suitable for use by an automated computer agent, crawler, search engine or other web service. The importance of ontologies in current Artificial Intelligence (AI) research is also emphasized by the interest shown by both the research and the enterprise community to various problems related to ontologies and ontology manipulation (McGuiness *et al.* 2000).

Ontologies are often large and complex structures, whose development and maintenance give rise to several interesting research problems. One of the most important such problems is *ontology change*, which refers to the generic problem of changing an ontology in response to a certain need; in this paper, the term will be used in a broad sense, covering any type of change, including changes to the ontology in response to external events, changes dictated by the ontology engineer, changes forced by the need to translate the ontology in a different language or terminology and so on. The importance of this problem is emphasized by recent studies, which suggest that change, rather than ontologies, should be the central concept of the Semantic Web (Tzitzikas & Kotzinos 2007).

In order to cope with the many different aspects of the problem of ontology change, several related research disciplines have emerged (such as ontology evolution, versioning, merging, mapping, matching etc), each dealing with a different facet of the problem. These areas are greatly interlinked, and have common elements and subtasks (Noy & Musen 2004). As a result, several research efforts and systems deal with more than one of these topics creating significant confusion to a newcomer (see table 2 in the appendix). This confusion is further increased by the fact that certain terms are often used (some would say abused) with different meanings in the relevant literature, denoting similar, but not identical, research directions or concepts. For examples of such confusing and overused terms refer to (Flouris & Plexousakis 2005), (Pinto *et al.* 1999).

We believe that this lack of a standard terminology constitutes a major bottleneck for the ontology change community, causing an unnecessary confusion as well as misunderstandings. The purpose of this work is the introduction of a terminology, following the most common uses of the various terms in the literature. Fixing this terminology will allow us to determine the boundaries of each field as well as to get a grip on their interactions and connections.

To do that, we perform a literature review on the field of ontology change and introduce a broadly accepted terminology that will, hopefully, serve as a point of reference for the ontology change community. The focus of this review is on classification and breadth of coverage, rather than on depth of analysis; our purpose is to give a clear overall picture of each relevant subfield and determine the boundaries, interactions and overlaps between the various research areas. The interested reader is referred to the numerous bibliographic references that will appear throughout this paper for more details on each area or deeper results; a more compact reference guide is table 2 in the appendix, where a summary of the referenced works and their relation to the various ontology change subfields can be found.

2 Ontologies and Ontology Change

2.1 *What is an Ontology?*

The term ontology has come to refer to a wide range of formal representations, including taxonomies, hierarchical terminology vocabularies or detailed logical theories describing a domain (Noy & Klein 2004). For this reason, a precise definition of the term is rather difficult and different definitions have appeared in the literature (see, for example, (Gruber 1993a), (Guarino 1998)). One commonly used definition is based on the original use of the term in philosophy, where an ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented (Gruber 1993b); therefore, an ontology in the AI context is a structure that specifies a conceptualization, or, more accurately, a *specification of a shared conceptualization of a domain* (Gruber 1993a).

A more formal, algebraic, approach, identifies an ontology as a pair $\langle S, A\rangle$, where S is the *vocabulary* (or *signature*) of the ontology (being modeled by some mathematical structure, such as a poset, a lattice or an unstructured set) and A is *the set of ontological axioms*, which specify the intended interpretation of the vocabulary in a given domain of discourse (Kalfoglou & Schorlemmer 2003). A similar definition is given in (De Bruijn *et al.* 2004), where the signature S is broken down in three (not necessarily disjoint) sets, the set of concepts (C), the set of relations (R) and the set of instances (I); thus, an ontology is defined as a 4-tuple $\langle C, R, I, A\rangle$. Here, we will use the simpler $\langle S, A\rangle$ definition, i.e., we will use S to denote the signature of the ontology and A to denote its axiomatic part.

2.2 *Ontology Change*

As already mentioned, ontology change refers to the generic process of changing an ontology in response to a certain need. Several reasons for changing an ontology have been identified in the literature. An ontology, just like any structure holding information regarding a domain of interest, may need to change simply because the domain of interest has changed (Stojanovic *et al.* 2003); but even if we assume a static world (domain), which is a rather unrealistic assumption for most applications, we may need to change the perspective under which the domain is viewed (Noy & Klein 2004), or we may discover a design flaw in the original conceptualization of the domain (Plessers & de Troyer 2005); we may also wish to incorporate additional functionality, according to a change in users' needs (Haase & Stojanovic 2005). Similarly, a contradiction (usually an inconsistency or incoherency, see (Flouris *et al.* 2006a)) may be spotted, in which case we may want to take some action against the contradiction.

Furthermore, new information, which was previously unknown, classified or otherwise unavailable may become available or different features of the domain may become known and/or important (Heflin *et al.* 1999). Moreover, ontology development is becoming more and more a collaborative and parallelized process, whose subproducts (parts of the ontology) need to be combined to produce the final ontology (Klein & Noy 2003), (Noy *et al.* 2006); this process would require changes in each subontology to reach a consistent and valid final state; but even then, the so-called final state is rarely final, as ontology development is usually an ongoing process (Heflin *et al.* 1999).

There are also reasons related to the distributed nature of the Semantic Web: ontologies are usually depending on other ontologies, over which the knowledge engineer may have no control; if the remote ontology is changed for any of the above reasons, the dependent ontology might also need to be modified to reflect possible changes in terminology or representation (Heflin *et al.* 1999). In other cases, a certain agent, service or application may need to use an ontology whose terminology or representation is different from the one it can understand (Euzenat *et al.* 2004); in such cases, some kind of translation (change) needs to be performed in the imported ontology to be of use. Last but not least, we may need to combine information from two or more ontologies in order to produce a more appropriate one for a certain application (Pinto *et al.* 1999).

The problem of ontology change is far from trivial. Several philosophical issues related to the general problem of adaptation of knowledge to new information have been identified in the research area of *belief change*, also known as *belief revision* (Alchourron *et al.* 1985), (Gärdenfors 1992a), (Gärdenfors 1992b), (Hansson 1994), (Katsuno & Mendelzon 1990); most of them are also applicable to knowledge represented in ontologies (Flouris & Plexousakis 2005), (Flouris & Plexousakis 2006). The large size of modern day ontologies complicates this problem even further (McGuiness *et al.* 2000). But it's not just that: the Semantic Web is characterized by decentralization, heterogeneity and lack of central control or authority. This is both a blessing and a curse; these features have greatly contributed to the success of the WWW (and constitute key features of the Semantic Web) but they have also introduced several new, challenging and interesting problems, which don't exist in traditional AI.

As far as ontology change is concerned, one such problem is the lack of control over who uses a certain ontology once it has been published. Subtle changes in an ontology may have unforeseeable effects in dependent applications, services, data and ontologies (Stojanovic *et al.* 2002); ontology designers cannot know who uses which part of their ontology and for what purpose, so they cannot predict the effects that a given change on their ontology would have upon dependent elements. The same holds in the opposite direction: if an ontology is depending on other ontologies, there is no way for the ontology designer to control when and how these ontologies will change. These facts raise the need to support and maintain different interoperable versions of the same ontology (Heflin *et al.* 1999), (Huang & Stuckenschmidt 2005), (Klein *et al.* 2002), a problem greatly interwoven with ontology change (Klein & Fensel 2001). On the other hand, heterogeneity leads to the absence of a standard terminology for any given domain which may cause problems when an agent, service or application uses information from two different ontologies (Euzenat *et al.* 2004). As ontologies often cover overlapping domains from different viewpoints and with different terminology, some kind of translation may be necessary in many practical applications.

All these arguments indicate the importance of the problem of ontology change and motivate us to use the term in order to cover all aspects of ontology dynamics, as well as the problems that are indirectly related to the change operation such as the maintenance of different versions of an ontology or the translation of ontological information in a common terminology. More specifically, we will use the term ontology change to refer to *the problem of deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements*.

Notice that the decision on the modifications to perform may be made automatically, semi-automatically or manually; the implementation of the chosen modifications may (but need not) involve keeping a copy of the original ontology (versioning). The need to change the ontology may take several different forms, including, but not limited to, the discovery of new information (which could be some instance data, another ontology, a new observation or other), a change in the focus or the viewpoint of the conceptualization, information received by some external source, a change in the domain (i.e., a dynamic change in the modeled world), the discovery of a problematic pattern in the modeling process, communication needs between heterogeneous sources of information, the fusion of information from different ontologies and so on.

2.3 Ontology Change Subfields: A Short Discussion

Our definition of ontology change covers several related research fields which are studied separately in the literature. These fields are greatly interlinked and several papers and systems deal with more than one of these problems (see table 2 in the appendix). In other cases, the same term is used in different papers to describe different research areas. This situation can easily lead to misunderstandings, confusion and unnecessary waste of effort, especially for a newcomer. In the remainder of this paper we will attempt to precisely define the boundaries of each ontology change subarea and uncover their relations and differences. This attempt will hopefully draw a fine line between these areas, allowing the clarification of the meaning of each term and making the differences and similarities between them explicit. The provided definitions will not be arbitrary, but will be based on the most common uses of each term in the literature and on similar previous attempts, like (Kalfoglou & Schorlemmer 2003), (Pinto *et al.* 1999).

In particular, we will identify and study 10 subfields of ontology change, namely *ontology mapping, morphism, matching, articulation, translation, evolution, debugging, versioning, integration* and *merging*; in addition, we will clarify the meaning of the term *ontology alignment*, which is closely related to ontology matching. Each of these areas deals with a certain facet of the problem of change from a different view or perspective, covering different application needs, change scenarios or "needs for change". In this subsection, we provide a very short description of each of these fields; for more details, the reader is referred to the following sections, where the properties of each field are discussed in detail.

The first five fields in the above list (ontology mapping, morphism, matching, articulation and translation), as well as ontology alignment, are studied in section 3 and deal with heterogeneity resolution, i.e., how to resolve differences in terminology, language or syntax between ontologies. Usually, this problem is solved by providing a set of "translation rules" that identify similar ontology elements. The distinguishing difference between these fields is the methodology followed and the expected type of output (translation rules). These fields may look unrelated to ontology change, as there is no obvious "change" performed in the involved ontologies; translation rules do not seem to constitute change themselves. However, heterogeneity resolution falls under the definition of ontology change, in the wide sense of the term that we use in this paper.

Indeed, consider two agents with heterogeneous ontologies that need to communicate and a set of translation rules that allows this communication. In this particular example, the driving force (need) behind the process is the need for communication. The translation rules produced do not directly modify any ontology; however, they allow each agent to change the other agent's ontology locally to fit his own terminology, language and syntax. So the change in this case is made on-the-fly by each agent during each message exchange and it is trivial, given the translation rules. In this sense, heterogeneity resolution can be considered a type of ontology change that provides us with a method to change an ontology (but does not perform the change directly).

Furthermore, it is important to note that heterogeneity resolution constitutes a prerequisite for any type of successful ontology change, as it makes no sense to try to change an ontology in response to new information unless both the ontology and the new information are formulated using the same terminology, language and syntax. So, it makes practical sense to study these fields along with the problem of ontology change; this is also apparent in the relevant literature (see table 2 in the appendix), where many research efforts, systems or algorithms that deal with some specific aspect (subfield) of ontology change also deal with the problem of heterogeneity resolution (e.g., (De Bruijn *et al.* 2004), (Chalupsky 2000), (McGuiness *et al.* 2000), (Noy & Musen 1999a), (Noy & Musen 1999b), (Noy & Musen 2000)).

Table 1 Summary of the various subfields of ontology change

Ontology Mapping	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	A mapping between the ontologies' vocabularies
	Properties:	The output identifies related vocabulary entities
Ontology Morphism	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	Mappings between the ontologies' vocabularies and axioms
	Properties:	The output identifies related vocabulary entities and axioms
Ontology Matching (its output is called Ontology Alignment)	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	A relation between the ontologies' vocabularies
	Properties:	The output identifies related vocabulary entities
Ontology Articulation	**Purpose:**	Heterogeneity resolution, interoperability of ontologies
	Input:	Two (heterogeneous) ontologies
	Output:	An intermediate ontology and mappings between the vocabularies of the intermediate ontology and each source
	Properties:	The output is equivalent to a relation and identifies related vocabulary entities (like ontology matching)
Ontology Translation (first reading)	**Purpose:**	Translation to a different ontology representation language
	Input:	An ontology and a target ontology representation language
	Output:	An ontology expressed in the target language
	Properties:	Should produce an equivalent ontology, if possible
Ontology Translation (second reading)	**Purpose:**	Implementation of a vocabulary mapping
	Input:	An ontology and a mapping
	Output:	An ontology
	Properties:	Implements a vocabulary change to the source ontology as specified by the input mapping
Ontology Evolution	**Purpose:**	Respond to a change in the domain or its conceptualization
	Input:	An ontology and a (set of) change operation(s)
	Output:	An ontology
	Properties:	Implements a (set of) change(s) to the source ontology
Ontology Debugging (is split into Ontology Diagnosis and Ontology Repair)	**Purpose:**	Restore an ontology's consistency or coherency
	Input:	An inconsistent/incoherent ontology
	Output:	A consistent/coherent ontology
	Properties:	Renders an ontology consistent/coherent
Ontology Versioning	**Purpose:**	Transparent access to different versions of an ontology
	Input:	Different versions of an ontology
	Output:	A versioning system
	Properties:	Uses version ids to identify versions; provides transparent access to the correct version; determines compatibility
Ontology Integration	**Purpose:**	Fuse knowledge from ontologies covering similar domains
	Input:	Two ontologies (covering similar domains)
	Output:	An ontology
	Properties:	Fuses knowledge to cover a broader domain
Ontology Merging	**Purpose:**	Fuse knowledge from ontologies covering identical domains
	Input:	Two ontologies (covering identical domains)
	Output:	An ontology
	Properties:	Fuses knowledge to describe the domain more accurately

The problems of ontology evolution and ontology debugging are very similar in nature and are studied in section 4. Ontology evolution deals with the problem of incorporating new information in an existing ontology, so it deals with the changes themselves. As the new information may contradict the existing one, part of the problem is how to guarantee that the change will not cause any contradictions in the resulting ontology. This is closely related to the problem of ontology debugging, whose purpose is to restore the consistency (or coherency) of an inconsistent (or incoherent) ontology.

Ontology versioning manages different versions of a changing ontology, trying to minimize any adverse effects that a change could have upon related (dependent) ontologies, agents, applications or other elements. This is done by providing transparent access to either the current or some older version of the ontology, depending on the accessing element. This ability allows the accessing (dependent) elements, to upgrade to the new version at their own pace (if at all), which is considered a very useful feature, given the distributed and decentralized nature of the Semantic Web (Heflin *et al.* 1999), (Heflin & Pan 2004). Unfortunately, the goals of ontology evolution and versioning are often confused in the literature, so we chose to study them together (in section 4).

Ontology integration and merging both deal with the fusion of knowledge from two or more source ontologies. There is a subtle difference between them, related to the domain covered by the source ontologies (Pinto *et al.* 1999). In section 5, we provide further details on these two areas and clarify their differences.

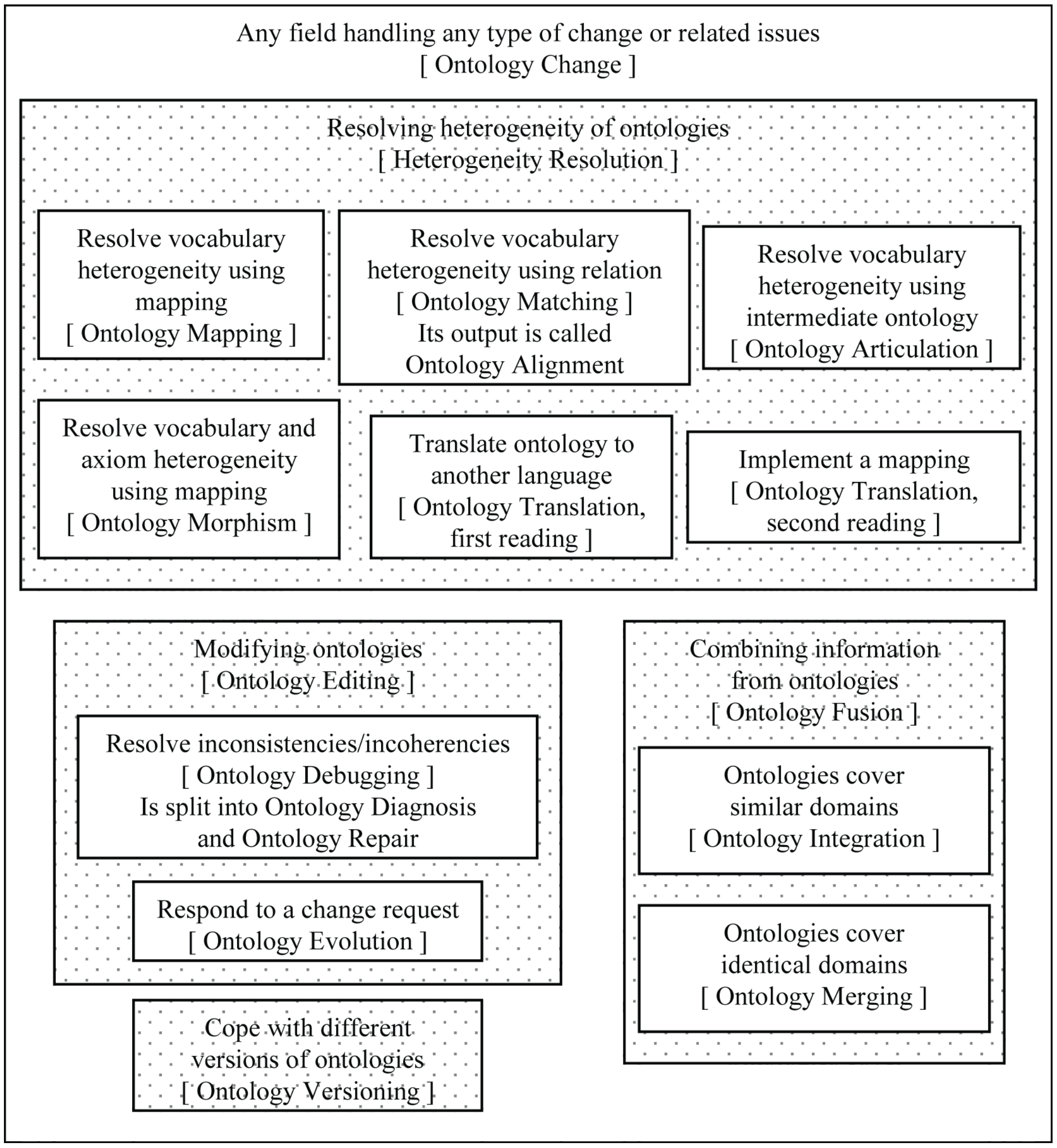
Any field handling any type of change or related issues
[Ontology Change]
Resolving heterogeneity of ontologies
[Heterogeneity Resolution]
Resolve vocabulary heterogeneity using mapping
[Ontology Mapping]
Resolve vocabulary heterogeneity using relation
[Ontology Matching]
Its output is called Ontology Alignment
Resolve vocabulary heterogeneity using intermediate ontology
[Ontology Articulation]
Resolve vocabulary and axiom heterogeneity using mapping
[Ontology Morphism]
Translate ontology to another language
[Ontology Translation, first reading]
Implement a mapping
[Ontology Translation, second reading]
Modifying ontologies
[Ontology Editing]
Resolve inconsistencies/incoherencies
[Ontology Debugging]
Is split into Ontology Diagnosis and Ontology Repair
Respond to a change request
[Ontology Evolution]
Cope with different versions of ontologies
[Ontology Versioning]
Combining information from ontologies
[Ontology Fusion]
Ontologies cover similar domains
[Ontology Integration]
Ontologies cover identical domains
[Ontology Merging]

Figure 1 A classification of ontology change subfields

In figure 1 the various fields related to ontology change are classified according to the need for change that motivates them; we can identify four groups, one related to heterogeneity resolution, one related to the modification of ontologies, one related to the combination of information from different ontologies and one related to versioning. Each group contains one or more of the ontology change subfields that are studied in this paper. Similarly, table 1 provides another compact description of the studied ontology change subfields and will be referenced throughout this work: in table 1, we summarize each subfield separately by describing the need for change that motivates each field (purpose), the expected input of an algorithm that deals with the problem (input), its expected output (output), as well as certain comments on its desired properties (properties).

3 Heterogeneity Resolution

3.1 Discussion on Heterogeneity Resolution

Works related to these areas try to mitigate the problems caused by the heterogeneity of the Semantic Web. The general motivation for these research efforts is that different ontologies (and sources of information based upon different ontologies) generally use different terminology, different representation languages and different syntax to refer to the same or similar concepts. A nice list of scenarios where this heterogeneity may cause problems can be found in (Euzenat *et al.* 2004).

The solution that is generally proposed for the problem stated above consists of a set of translation rules of some kind that allow us to nullify the differences in terminology or syntax. To put it simply, the goal of the whole process is to make two ontologies refer to the same entities using the same name and to different entities using different names. For example, an ontology matching algorithm should be able to identify that two concepts (classes) RESEARCHER and RESEARCH_STAFF_MEMBER that appear in two different ontologies refer to the same real-world concept, i.e., the class of all researchers. It should also be able to differentiate between two different uses of the entity CHAIR, as it could refer to the class of chairs (as a furniture) in one ontology and to the people forming a Workshop's Chair in another.

Therefore, these areas basically deal with the same problem (i.e., how to deal with heterogeneous information); however, the output of the methods (i.e., the result that is produced towards the solution of the problem) is different in each field. Thus, the different research areas can be identified based on the type of translation rules that is produced as the output. Due to the close relationship between these topics, sometimes the terms ontology alignment (in (Euzenat *et al.* 2004)), ontology mapping (in (Kalfoglou & Schorlemmer 2003)), or ontology-schema matching (in (Shvaiko & Euzenat 2005)) are used to refer collectively to all these areas. In the following, we will provide a definition of these fields and outline their differences and similarities.

3.2 Ontology Mapping, Morphism, Matching, Alignment, Articulation and Translation

The term *ontology mapping refers to the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected* (Kalfoglou & Schorlemmer 2003). A similar (and equivalent) definition appears in (De Bruijn *et al.* 2004) where ontology mapping is defined as a (declarative) specification of the semantic overlap between two ontologies, which can be either one-way (injective) or two-way (bijective). The result of an ontology mapping algorithm is a function (morphism) between ontological signatures. We could differentiate between two different types of mappings, namely the *total ontology mappings* and the *partial ontology mappings*. A total ontology mapping maps the whole source ontology (say $O_1 = \langle S_1, A_1 \rangle$) to the target ontology O_2 while a partial ontology mapping maps a subontology of O_1 (say $O_1\prime = \langle S_1\prime, A_1\prime \rangle$, with $S_1\prime \subseteq S_1$ and $A_1\prime \subseteq A_1$) to O_2 (Kalfoglou & Schorlemmer 2003).

This definition restricts the mappings to ontological signatures. A more ambitious and interesting approach would be to create mappings that deal with both the signatures and the axioms of the ontologies. The term *ontology morphism* refers to that approach, i.e., *the development of a collection of functions (morphisms) that relate both ontological signatures and axioms* (Kalfoglou & Schorlemmer 2003). Notice that ontology morphism, unlike the other fields discussed in this section, is not restricted to the ontology vocabulary only, but covers the ontological axioms as well.

In both ontology mapping and ontology morphism, we try to relate the two ontologies via functions. Alternatively, the two ontologies could be related in a more general fashion, namely by means of a relation.

The task of finding relationships between (vocabulary) entities belonging to two different ontologies is called ontology matching and the output of that task is called ontology alignment (Shvaiko *et al.* 2006). Equivalently, we could say that ontology matching is the task of creating links between two ontologies (Choi *et al.* 2006). Another similar (but less specific) definition appears in (De Bruijn *et al.* 2004), where ontology matching is defined as the process of discovering similarities between two source ontologies.

The output of ontology matching (i.e., ontology alignment) is a binary relationship between the vocabularies of the two ontologies. This approach is more liberal, allowing greater flexibility, so it is more commonly used in practice. In ontology matching, the sources become consistent with each other but are kept separate, so ontology matching is usually performed when dealing with complementary domains. Notice that, in several cases, e.g., (Aumueller *et al.* 2005), (David *et al.* 2006), (Giunchiglia *et al.* 2006), (Hu & Qu 2006), (Shvaiko & Euzenat 2005), (Zhdanova & Shvaiko 2006), the term *ontology (schema) matching* is used to refer to ontology alignment and the two terms are used interchangeably (see also (Madhavan *et al.* 2005)).

A binary relationship could be decomposed into a pair of total functions from a common intermediate source; therefore, the alignment of two ontologies could be described by means of a pair of ontology mappings from a common intermediate ontology. We use the term *ontology articulation to refer to the process of determining the intermediate ontology and the two mappings to the initial ontologies* (Kalfoglou & Schorlemmer 2003).

Finally, the term *ontology translation* is used in the literature to describe two different things. Under one understanding (Kalfoglou & Schorlemmer 2003), ontology translation refers to *the process of changing the formal representation of the ontology from one language to another* (say from OWL (Dean *et al.* 2004) to RDF (Miller *et al.* 2006)). This changes the syntactic form of the axioms, but not the vocabulary or the semantics of the ontology. Under the second understanding (Kalfoglou & Schorlemmer 2003), ontology translation refers to *a translation of the vocabulary*, in a manner similar to that of ontology mapping. The difference between ontology mapping and ontology translation is that the former specifies the function that relates the two ontologies' vocabularies, while the latter applies this function to actually implement the mapping.

3.3 Matchers and Heterogeneity Resolution

In the previous subsection it was made clear that the aforementioned research areas try (with a few exceptions) to relate the vocabularies of two ontologies. This problem is far from trivial; an implementation of a matching-mapping process may use multiple match algorithms or matchers. This allows the selection of the matchers depending on the application domain and schema types.

Given that the use of multiple matchers may be required, two subproblems emerge. The first is the design of individual matchers, each of which computes a mapping (or relationship) based on a single matching criterion. The second is the combination of individual matchers, either by using multiple matching criteria within a hybrid matcher (e.g., name and type equality) or by combining multiple match results produced by different match algorithms in parallel or sequentially. The matching approaches can also be discriminated on the basis of the kind of relations that they produce. Some consider symmetric (equivalence) relations, while others additionally use asymmetric relations such as subsumption or implication. A detailed classification appears in figure 2 (Manakanatas & Plexousakis 2006), which is an extended and improved version of a figure appearing in (Rahm & Bernstein 2001). For individual matchers, the following largely-orthogonal criteria are considered for classification (a more detailed description of these methods can be found in (Euzenat *et al.* 2004)):

- *Instance vs. schema:* Matching approaches can consider instance data i.e., data content (as in the system GLUE (Doan *et al.* 2000), (Doan *et al.* 2002)), or only schema-level information (as in Cupid (Madhavan *et al.* 2001), SF (Melnik *et al.* 2002), COMA++ (Aumueller *et al.* 2005) and its predecessor, COMA (Do & Rahm 2002)).
- *Element vs. structure matching* (or *extensional vs intensional* respectively)*:* Match can be performed for individual ontology elements, such as attributes (as in GLUE (Doan *et al.* 2002) and COMA,

COMA++ (Aumueller *et al.* 2005), (Do & Rahm 2002)), or for combinations of elements, such as complex schema structures (as in the systems SF (Melnik *et al.* 2002), Protoplasm (Bernstein *et al.* 2004) and S-MATCH (Giunchiglia *et al.* 2004)).

- *Language vs. constraint:* A matcher can use a linguistic approach (e.g., based on names and textual descriptions of ontology elements, like in (Manakanatas & Plexousakis 2006), (Ferrara 2004)), or a constraint-based approach (e.g., based on keys and relationships, like in S-MATCH (Giunchiglia *et al.* 2004) and GLUE (Doan *et al.* 2002)).
- *Matching cardinality:* Each element of the resulting matching may match one or more elements of one schema to one or more elements of the other, yielding four cases: 1:1, 1:n, n:1, n:m. In addition, there may be different matching cardinalities at the instance level. This criterion determines whether we have a matching or a mapping algorithm.
- *Auxiliary information:* Most matchers not only rely on the input ontologies, but also on auxiliary information, such as dictionaries (e.g., in (Manakanatas & Plexousakis 2006), (Ferrara 2004)), global schemas (e.g., in Clio (IBM corporation)), previous matching decisions (e.g., in COMA, COMA++ (Aumueller *et al.* 2005), (Do & Rahm 2002) and in (Manakanatas & Plexousakis 2006)), or user input (e.g., in Clio (IBM corporation) and Cupid (Madhavan *et al.* 2001)).

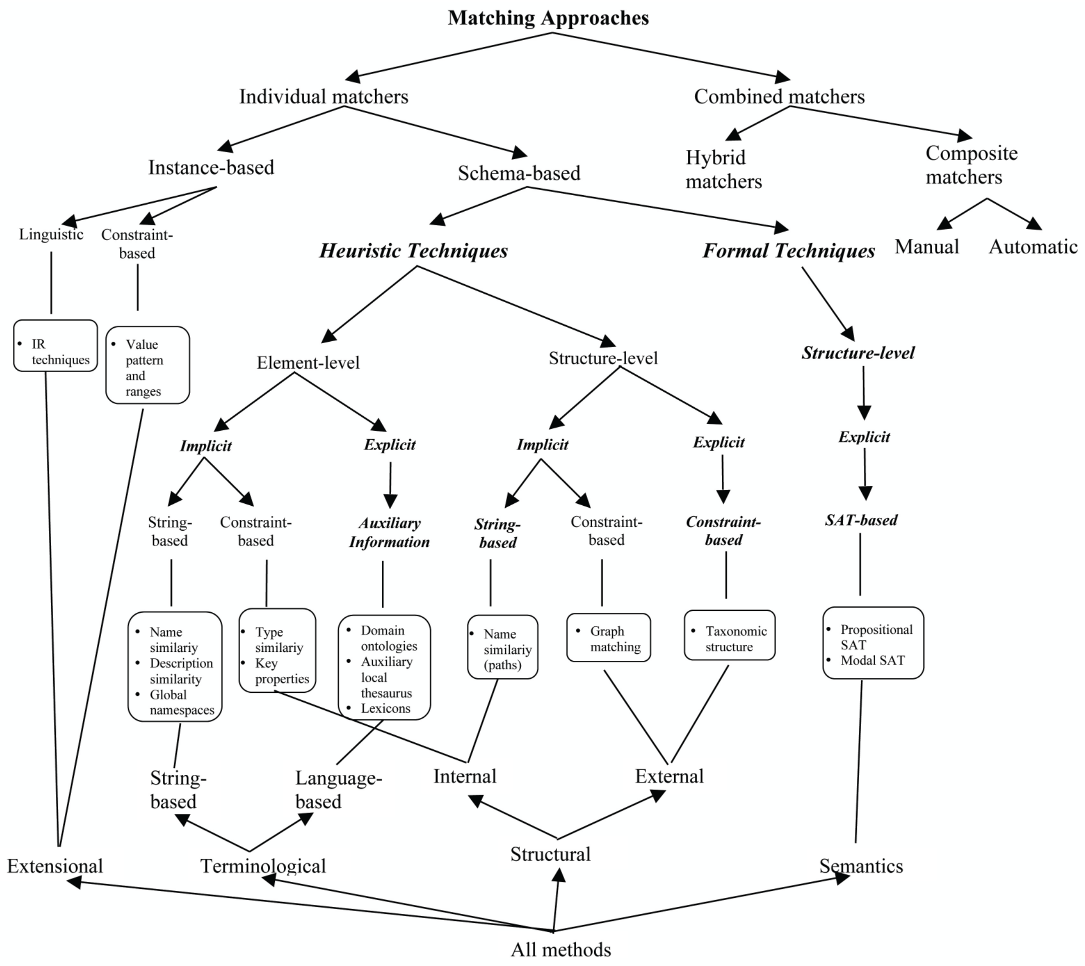

Figure 2 Classification of matching approaches

3.4 State of the Art in Heterogeneity Resolution

In most cases, the process of heterogeneity resolution determines a similarity value in the interval [0,1] for every possible matching between two ontological elements. PROMPT (Noy & Musen 2000), (Noy & Musen 1999a) (originally called SMART (Noy & Musen 1999b)) and Chimaera (McGuiness *et al.* 2000) are two interesting systems dealing with heterogeneity. Another important system is Clio (IBM corporation): Clio consists of a set of Schema Readers, which read a schema and translate it to an internal representation, a Correspondence Engine, which is used to identify matching parts of ontologies, and a Mapping Generator, which generates view definitions to map data from the source ontology into data in the target ontology. The Correspondence Engine uses n:m element level matchings (i.e., relationships) that have been gained by knowledge or provided by the user via a graphical interface.

On the contrary, Cupid (Madhavan *et al.* 2001) is a hybrid schema matcher that combines a name matcher and a structure-based matcher. This tool finds the element matchings of a schema, using the similarity of their names and types at the leaf level. The system SF (Melnik *et al.* 2002) uses certain filters that allow the user to choose the best matchings from a set of possible ones returned from the structure-based matcher; this system uses no external dictionary. A general-purpose approach to the problem of translation is described in (Chalupsky 2000). In (Castano *et al.* 2006b), an ontology matching algorithm (namely H-MATCH (Castano *et al.* 2006a)) is used as an aid to the ontology evolution process.

In (Calvanese *et al.* 2002), an interesting formal framework for defining mappings is proposed; however, this work is placed in the context of ontology integration and the focus is on being able to answer queries over heterogeneous ontologies using the defined mappings. (Mocan *et al.* 2006) use the notion of perspective to define another formal framework for mapping creation; this model is applied in order to visualize the results of the mapping process. Similarly, (Kondylakis *et al.* 2006) present a language used to describe ontology alignments, which is adequate to capture the most common occurrences of heterogeneity encountered in a broad sample of cases, yet simple enough to be used by domain experts.

In (Manakanatas & Plexousakis 2006) the term ontology matching is used to refer to an ontology mapping algorithm based on the linguistic properties of terms, using a thesaurus based on WordNet (Miller 1995); a similar approach appears in (Ferrara 2004). In (Stoilos *et al.* 2005), a certain string metric is proposed to evaluate name similarities of elements in different ontologies, upon which a linguistic matcher could be based. Some thoughts on the issue of heterogeneity in the context of the SHOE language can be found in (Heflin *et al.* 1999), (Heflin & Hendler 2000). A machine learning approach for ontology matching, which is based on shared individuals (instance-based matcher), appears in (Palmisano *et al.* 2006). An interesting method of improving the results of a matching process, which exploits user validation combined with machine learning techniques, can be found in (Ehrig *et al.* 2005). In (Mitra *et al.* 2005), a probabilistic technique is used towards this aim; in that work, the final similarity evaluation of an ontology mapping algorithm is affected by the similarity probabilities of each entity's neighborhood, improving the initial mapping result. Another method based on probabilistic analysis, which takes into account uncertainty issues in the mapped ontologies can be found in (Pan *et al.* 2005).

Protoplasm (Bernstein *et al.* 2004) is an interesting system that offers a new architecture for schema matching; it includes a special internal representation, called the Schema Matching Model Graph, as well as an interface which handles the mapping algorithms and a smart technique for combining their execution. COMA++ (Aumueller *et al.* 2005) offers a large library of matchers and supports several ways of combining their results. These matchers support finding structure-based matchings as well as element-level matchings.

Most of the aforementioned approaches are intensional (schema-based) and symmetric; none is both asymmetric and extensional (element-based). A symmetric and extensional approach is GLUE (Doan *et al.* 2000), (Doan *et al.* 2002), which uses Bayesian learners in order to classify instances of the one ontology into the other and vice-versa; this allows the estimation of the joint probability distribution and the prediction of concept similarities. To the best of our knowledge, there is only one intensional method considering asymmetric relations, namely S-MATCH (Giunchiglia *et al.* 2004); S-MATCH identifies equivalence, more general, less general, mismatch and overlapping relations between concepts and uses a lot of single matchers: 13 linguistic-based and 3 logic-based ones.

Currently, there are more than 50 systems available dealing with ontology matching (Shvaiko *et al.* 2006) and a lot of research effort is being put in the area. The interested reader is referred to (De Bruijn *et al.* 2004), (Choi *et al.* 2006), (Euzenat *et al.* 2004), (Euzenat & Shvaiko, 2007), (Kalfoglou & Schorlemmer 2003), where a more extensive list of systems and works related to these research areas can be found. Evaluations of such works appear in (Avesani *et al.* 2005), (Do *et al.* 2002), (Svab *et al.* 2007), (Yatskevich 2003).

Unfortunately, heterogeneity resolution in ontologies is a time consuming process and relies heavily on human intervention. Several works envisage the process becoming fully automatic so that it becomes more practical (Kalfoglou & Schorlemmer 2003). In this direction, advances in the field of natural language processing will probably help researchers improve their understanding on the processes behind automatic heterogeneity resolution (Kalfoglou & Schorlemmer 2003). We argue however that the problem of heterogeneity resolution cannot be dealt with using solely fully automatic algorithms; the most realistic path is the development of semi-automatic solutions that would reduce human intervention.

4 Ontology Evolution, Debugging and Versioning

4.1 Disambiguating the Terms

In some works, ontology versioning is considered a stronger variant of ontology evolution (Haase & Sure 2004). Under that viewpoint, ontology evolution is concerned with the ability to change the ontology without losing data or negating the validity of the ontology, whereas ontology versioning should additionally allow access to different variants of the ontology. Thus, while ontology evolution is concerned with the validity of the newest version, ontology versioning additionally deals with the validity, interoperability and management of all previous versions, including the current (newest) one.

This viewpoint is influenced by related research on relational and object-oriented database schema evolution and versioning (Banerjee *et al.* 1987), (Franconi *et al.* 2000), (Kim & Chou 1988), (Peters & Ozsu 1997). A recent survey on this issue, studying the differences and similarities of the two contexts can be found in (Noy & Klein 2004), where ontologies are compared with database schemas outlining their differences and these differences' impact with respect to (ontology and database schema) evolution and versioning. In the same paper it is argued that, under the understanding of (Haase & Sure 2004), ontology evolution and versioning become indistinguishable; due to the distributed and decentralized nature of the Semantic Web, multiple versions of ontologies are bound to exist and must be supported. Furthermore, ontologies and dependent elements are likely to be owned by different parties or spread across different servers; as a result, some parties may be unprepared to change and others may even be opposed to it (Heflin *et al.* 1999). All these facts force us to maintain and support different versions of ontologies, making ontology evolution (under this understanding) useless in practice.

From a similar standpoint, the evolution of an ontology represents the evolution of our understanding of the domain. Thus, a newer version of an ontology does not invalidate the old one, but replaces it in terms of usability (only); as a result, the different versions of an evolving ontology should coexist. This is reminiscent of the viewpoint employed in temporal databases (Tansel *et al.* 1993). The adoption of this viewpoint makes ontology versioning the only valid method of changing an ontology, whereas evolution is an internal part of versioning, dealing only with the determination of the next version.

We believe that the issue of modifying the ontology (ontology evolution) should be clearly separated from the issue of maintaining the interoperability of the different versions of the ontology that occur because of these modifications (ontology versioning). This distinction is not always clear in the literature, because the extensive web of interrelationships that is usually formed around an ontology forces us to consider the issue of propagating the changes to dependent elements as an indispensable part of the changes themselves (Maedche *et al.* 2003). This tight coupling has caused ontology evolution algorithms (and systems) to deal with versioning issues as well.

For example, according to (Plessers *et al.* 2005), the purpose of ontology evolution is to define methods (algorithms) to cope with ontology changes and techniques to maintain consistency of depending artifacts. Similarly, in (Stojanovic *et al.* 2002), ontology evolution is defined as the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, as well as the consistent management and propagation of these changes to dependent elements. Some would consider that these definitions include both ontology evolution and versioning.

What is important here is to clarify the details of the propagation process and whether the original version of the ontology will still be available after the change has been completed (and propagated). If, after informing dependent ontologies and other elements of the changes that were just performed in the local ontology (propagation of changes), the original version ceases to be available, then the role of propagation is to let these elements know that future interaction might be problematic unless the dependent elements synchronize themselves with the change. In this case, the process of propagation can be rightfully considered a part of the change itself, i.e., a part of ontology evolution.

On the contrary, if the propagation has a purely informative character and both the new and the old versions are available, then the dependent elements can synchronize with the changes at their own pace (if at all); to support this desirable feature (Heflin *et al.* 1999), the local ontology needs to employ an ontology versioning algorithm which will be responsible for managing and providing proper access to the old and the new version. This algorithm cannot be considered part of the ontology evolution process.

Despite the obvious relationships between ontology evolution and versioning algorithms, caused by ontology interoperability, we firmly believe that they are independent research areas facing different research challenges; thus, our definitions will strictly separate the two areas (see also table 1). More specifically, we define *ontology evolution to refer to the process of modifying an ontology in response to a certain change in the domain or its conceptualization* (Flouris & Plexousakis 2005). On the other hand, *ontology versioning refers to the ability to handle an evolving ontology by creating and managing different variants (versions) of it* (Klein & Fensel 2001). In other words, ontology evolution is restricted to the process of modifying an ontology while maintaining its validity, whereas ontology versioning deals with the process of managing different versions of an evolving ontology, maintaining interoperability between versions and providing transparent access to each version as required by the accessing element (data, service, application or other ontology). For a graphical representation of the role of the two fields in ontology change, see figures 3, 5 in the following subsections.

Another field that is closely related to ontology evolution is ontology debugging. In ontology debugging, the reason for change is the realization that an ontology contains some kind of logical contradiction, usually in the form of an inconsistency or an incoherency (Flouris *et al.* 2006a). Such logical contradictions are generally undesirable, so ontology debugging attempts to “correct” or “repair” such problems. Thus, we define ontology debugging as *the process of identifying and removing undesirable logical contradictions (inconsistencies/incoherencies) from an ontology*.

The main difference between ontology debugging and ontology evolution is that ontology evolution attempts to restore logical contradictions (usually inconsistencies) that are caused by the introduction of some new information, whereas in ontology debugging there is no new information, but the contradiction is in the ontology to begin with. In fact, the identification and resolution of any inconsistencies that may arise as a result of a change is one of the most important tasks to be performed during ontology evolution (Plessers & de Troyer 2006). On the other hand, the problem of ontology evolution can be reduced to the problem of ontology debugging (Haase *et al.* 2005). We argue that ideas employed in each field could also be applied in the other without too much trouble, and cross-fertilization would give rise to significant progress in both fields.

Ontology debugging is often associated with *ontology diagnosis* and *ontology repair*, where *ontology diagnosis refers to the identification of the possible causes of the logical contradiction(s)* and *ontology repair refers to the determination of the best way to resolve the logical contradiction(s)* (Haase *et al.* 2005), (Schlobach & Cornet 2003). Under this understanding, ontology debugging consists of these two processes, namely ontology diagnosis (which identifies the causes of contradictions – note that there may be more than one alternative causes), and ontology repair (which uses this information to select one of the alternatives for removing the contradiction). According to (Schlobach & Cornet 2003), the selection involved in ontology repair requires some understanding of the meaning of the underlying terms, so a human agent should be involved in the process.

4.2 Ontology Evolution

4.2.1 General Discussion on Ontology Evolution

Ontology evolution could be considered as the purest type of ontology change, in the sense that it deals with the changes themselves (table 1); it is an important problem, as the effectiveness of an ontology based application heavily depends on the quality of the conceptualization of the domain by the underlying ontology (Stojanovic *et al.* 2003), which is directly affected by the ability of an evolution algorithm to properly adapt the ontology both to changes in the domain (as ontologies often model dynamic environments (Stojanovic *et al.* 2002)) and to changes in the domain's conceptualization (as no conceptualization can ever be perfect). Figure 3 is a graphical depiction of the role of ontology evolution.

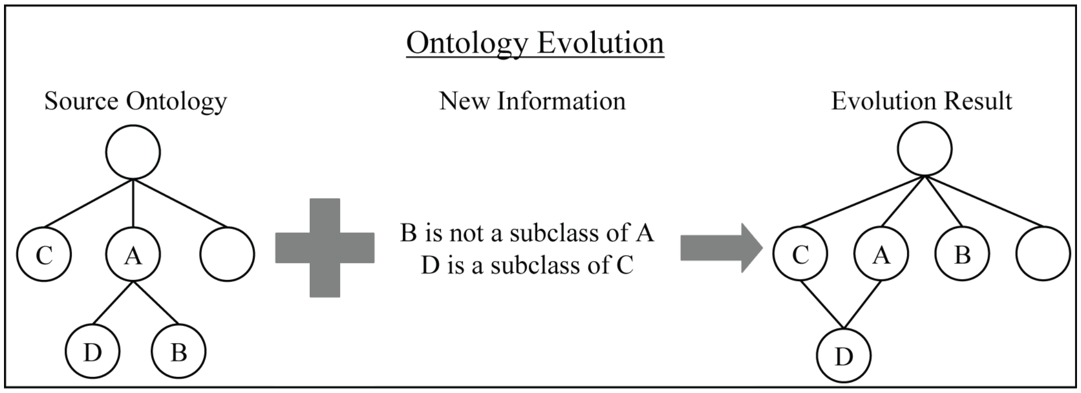

Figure 3 Ontology evolution

As already stated, an ontology is, according to (Gruber 1993a), a specification of a shared conceptualization of a domain. Thus, a change may be caused by either a change in the domain, a change in the conceptualization or a change in the specification (Klein & Fensel 2001). The third type of change (change in the specification) refers to a change in the way the conceptualization is formally recorded, i.e., a change in the representation language. This type of change is dealt with in the field of ontology translation, studied in the previous section (see also table 1). Thus, our definition of ontology evolution covers the first two types of change only (changes in the domain and changes in the conceptualization).

Both types of changes are not rare. The conceptualization of the domain may change for several reasons, including a new observation or measurement, a change in the viewpoint or usage of the ontology, newly-gained access to information that was previously unknown, classified or otherwise unavailable and so on. The domain itself may also change, as the real world itself is generally not static but evolves over time. More examples of reasons initiating changes can be found in (Klein & Fensel 2001), (Noy & Klein 2004), (Stojanovic & Motik 2002).

4.2.2 Ontology Evolution Phases

In order to tame the complexity of the problem, six phases of ontology evolution have been identified in (Stojanovic *et al.* 2002), occurring in a cyclic loop. Initially, we have the *change capturing* phase, where the changes to be performed are determined; these changes are formally represented during the *change representation* phase. The third phase is the *semantics of change* phase, in which the effects of the change(s) to the ontology itself are determined; during this phase, possible problems that might be caused to the ontology by these changes are also identified and resolved. The *change implementation* phase follows, where the changes are physically applied to the ontology, the ontology engineer is informed of the changes and the performed changes are logged. These changes need to be propagated to dependent elements; this is the role of the *change propagation* phase. Finally, the *change validation* phase allows the ontology engineer to review the changes and possibly undo them, if desired. This phase may uncover further problems with the ontology, thus initiating new changes that need to be performed to improve the conceptualization; in this case, we need to start over by applying the change capturing phase of a new evolution process, closing the cyclic loop. A similar approach which identifies five phases can be found in (Plessers & de Troyer 2005).

We will try to illustrate the role of the six phases using an example. The process is initiated by the change capturing phase, where the need for a change is identified. Three types of change capturing are described in (Haase & Sure 2004), (Stojanovic & Motik 2002), namely structure-driven, usage-driven and data-driven; a fourth one (discovery-driven) is used in (Castano *et al.* 2006b). In our example, suppose that, for some reason, we decide to remove concept C from our ontology. This decision is taken during the change capturing phase and the six-phase process of ontology evolution is initiated.

During the change representation phase we determine and formally represent the change(s) that must be performed in order to remove C. There are two major types of changes, namely elementary and composite changes (Stojanovic *et al.* 2002), (Stojanovic & Motik 2002) also called atomic and complex in (Stuckenschmidt & Klein 2003). Elementary changes represent simple, fine-grained changes; composite changes represent more coarse-grained changes and can be replaced by a series of elementary changes. Even though possible, it is not generally appropriate to use a series of elementary changes to replace a composite one, as this might cause undesirable side-effects (Stojanovic *et al.* 2002). The proper level of granularity should be identified at each case. Examples of elementary changes are the addition and deletion of elements (concepts, properties etc) from the ontology. In our particular example, a simple Remove_Concept operation (which is an elementary operation) should be enough to perform the required change. This is not always the case though. For example, we might have wished to move the concept C to some other point in the concept hierarchy. This is represented by a composite change (or by a series of elementary changes). There is no general consensus in the literature on the type and number of composite changes that are necessary. In (Stojanovic *et al.* 2002), 12 different composite changes are identified; in (Noy & Klein 2004), 22 such operations are listed; in (Stuckenschmidt & Klein 2003) however, the authors mention that they have identified 120 different interesting composite operations and that the list is still growing! In fact, the number of definable composite operations can only be limited by setting a granularity threshold on the operations considered; if we allow unlimited granularity, we will be able to define more and more operations of coarser and coarser granularity, limited only by our imagination (Klein & Noy 2003). Thus, creating a complete list of composite operations is not possible, but, fortunately, it is not necessary either, since a composite operation can always be defined as a series of elementary ones (Klein & Noy 2003).

Once the required change is identified (i.e., Remove_Concept for the concept C in our example), we can proceed to the next step of the evolution process, which is the semantics of change phase. At this point, we should identify any problems that will be caused when the chosen action is actually implemented, thus guaranteeing the validity of the ontology at the end of the process. In our example, we need (among other things) to determine what to do with the instances of the concept C; for example, we could delete them or re-classify them to one of the superconcepts of C. In (Stojanovic *et al.* 2002), the authors suggest that the final decision should be made indirectly by the ontology engineer, through the selection of certain pre-determined evolution strategies, which indicate the appropriate action in such cases. Other (manual or semi-automatic) approaches are also possible (see (Haase & Sure 2004)). This phase is probably the most crucial of ontology evolution, because during that phase the direct and indirect changes caused by a given change request (i.e., the effects and side-effects of the change) are determined.

During the implementation phase, the changes identified in the two previous phases are actually implemented in the ontology, using an appropriate tool, like, for example, the KAON API (Stojanovic *et al.* 2002). Such a tool should have transactional properties, based on the ACID model, i.e., guaranteeing Atomicity, Consistency, Isolation and Durability of changes (Haase & Sure 2004). It should also present the changes to the ontology engineer for final verification and keep a log of the implemented changes (Haase & Sure 2004).

The change propagation phase should ensure that all induced changes will be propagated to the interested parties. In (Maedche *et al.* 2003), two different methods to address the problem are compared, namely push-based and pull-based approaches. Under a push-based approach, the changes are propagated to the dependent ontologies as they happen; in a pull-based approach, the propagation is initiated only after the explicit request of each of the dependent ontologies. In both (Maedche *et al.* 2003) and (Stojanovic *et al.* 2002) the authors choose to use the former approach (push-based). However, in the Semantic Web context, this may not always be possible, as the dependent elements may be unknown. Alternatively, one could avoid this step altogether, by using an ontology versioning algorithm (Klein *et al.* 2002), allowing the interested parties to work with the original version of the ontology and update to the newer version at their own pace, if at all. This alternative is considered more realistic for practical purposes, given the decentralized and distributed nature of the Semantic Web (Heflin *et al.* 1999).

Finally, the change validation phase should allow the ontology engineer to review and possibly undo the changes performed, or initiate a new sequence of changes to further improve the conceptualization of the domain as represented by the ontology.

The chosen model for ontology evolution allows us to ignore heterogeneity issues. Obviously, any ontology evolution method will collapse in the face of heterogeneous information, unless coupled with an algorithm like those discussed in the previous section. However, heterogeneity is not an issue under the proposed model, because ontology evolution algorithms assume human participation in the process: during the change representation phase, the ontology engineer specifies the changes to be performed, so it can be reasonably assumed that these changes will be represented in a suitable language and terminology.

4.2.3 State of the Art in Ontology Evolution

The current state of the art in ontology evolution, as well as a list of existing tools that aid the process can be found in (Haase & Sure 2004). Some of these tools are simple ontology editors, like Protégé (Noy *et al.* 2000), which is one of the most popular tools for ontology design and creation but is often also used for ontology evolution and management. Lately, the functionality of Protégé has been enhanced (Noy *et al.* 2006) so as to provide several interesting features useful for both ontology design and evolution. Another ontology editor often used for ontology evolution is OilEd (Bechhofer *et al.* 2001). Unlike Protégé, OilEd is rather restrictive in the sense that it disallows any change that would cause some contradiction (e.g., inconsistency, incoherency) in the ontology, rather than taking action against such a contradiction; in addition, it supports less update operations than Protégé. For a list of desired editor features (with respect to ontology evolution) and an evaluation of some existing editors in this context see (Stojanovic & Motik 2002).

Apart from Protégé and OilEd, more specialized tools are also available and provide similar editing features to the user. In some such tools, like KAON (Gabel *et al.* 2004) and OntoStudio (formerly OntoEdit (Sure *et al.* 2003)), the user can define some kind of pre-defined evolution strategies (Stojanovic *et al.* 2002) that control how indirect changes (side-effects) will be determined, thus allowing the tool to calculate side-effects and perform some of the required changes automatically; in this respect, OntoStudio provides more options for parameterization, but uses a more restricted model and supports less operations than KAON. Other tools allow collaborative edits, i.e., several users can work simultaneously on the same ontology (Duineveld *et al.* 2000), while others support transactional changes (Haase & Sure 2004). In other works, features related to ontology versioning, undo/redo operations and other helpful utilities are supported (Duineveld *et al.* 2000). Some of these tools provide intuitive graphical interfaces that help the visualization of the process (Lam *et al.* 2005). For a list of desirable features for ontology evolution tools refer to (Noy *et al.* 2006), (Stojanovic & Motik 2002); for a detailed account of related systems refer to (Duineveld *et al.* 2000), (Haase & Sure 2004).

The above class of tools focuses on providing an intuitive environment allowing the ontology engineer to perform the changes in an efficient manner, but provide little or no support for the determination of the changes' side-effects, which need to be determined by the ontology engineer as well (either directly or indirectly). An essential and complementary research path attempts to determine (in an automatic or semi-automatic way) what the side-effects of each change should be.

Some initial ideas on this issue can be found in (Stojanovic *et al.* 2002) where evolution strategies are used for this purpose (see also (Stojanovic & Motik 2002)); this approach is applied in KAON and OntoStudio. A 5-step workflow which should be followed by an ontology evolution algorithm to determine the side-effects of a change was proposed in (Konstantinidis *et al.* 2007). The authors show that a number of existing ontology evolution algorithms follow this pattern and provide a general formal framework that captures this 5-step pattern. In addition, they claim that their framework can be applied for several different formalisms and a particular instantiation of it for RDF is presented and evaluated. The proposed five-step workflow roughly corresponds to the phases of change representation and semantics of change (phases 2 and 3 respectively), as defined in (Stojanovic *et al.* 2002).

A more direct approach to the problem can be found in (Magiridou *et al.* 2005), where a declarative language for changing (evolving) the data portion of an RDF ontology is introduced; each possible change in this language has a well-defined set of implied side-effects, which are applied automatically by the system. A semantics for updating Description Logic (DL) systems (Baader *et al.* 2002) can be found in (Roger *et al.* 2002); this is relevant to ontology evolution, since the family of DLs is one of the most popular formalisms for ontology representation (Baader *et al.* 2003). A similar, interesting semantics for DL updating and some related feasibility results appear in (Liu *et al.* 2006). In (Haase *et al.* 2005) the problem of ontology evolution is addressed by reducing it to the problem of ontology debugging.

An alternative, novel approach that is constantly gaining ground in the area, uses belief change (Gärdenfors 1992a) techniques to handle ontology evolution. In such works, popular belief change techniques, normally inapplicable for ontology representation languages, are being modified so as to be able to deal with ontologies and ontology evolution. This methodology is particularly applicable to the most important (and difficult) phase of ontology evolution, namely the semantics of change phase (Qi *et al.* 2006b) and constitutes another approach towards the automatic determination of the changes' side-effects.

Arguments in favor of this research path and some results related to its feasibility for a particular class of belief change theories, based on the AGM postulates (Alchourron *et al.* 1985), have appeared in a series of papers (Flouris 2006), (Flouris *et al.* 2006a), (Flouris & Plexousakis 2006), (Flouris *et al.* 2004), (Flouris *et al.* 2005), (Flouris *et al.* 2006b). A similar, but less mature, proposal appears in (Kang & Lau 2004).

A formalization of updates in RDF, which is inspired by belief change ideas, appears in (Gutierrez *et al.* 2006); this formalization allows automatically determining what the result of an update upon an RDF ontology should be. In (Lee & Meyer 2004), a preference ordering (inspired by similar orderings in belief change) is used to determine how new information should be added to ontologies represented using a particular DL (namely, *ALU*).

In (Ribeiro & Wassermann 2007) the belief change idea of kernel operators (Hansson 1994) is used, by adapting it for knowledge representation formalisms that do not allow axiom negation (such as the formalisms used in ontologies); this allows kernel operators to be used for ontology evolution. Other works (Halaschek-Wiener & Katz 2006) combine ideas from ontology debugging (in particular, the approach of (Kaluyanpur 2006)) and kernel operators, in order to propose an ontology evolution operator for OWL ontologies.

Another interesting application of belief change results for the purposes of ontology evolution was spawned by the work of (Meyer *et al.* 2005), where the authors attempt to recast the maxi-adjustment algorithm (Benferhat *et al.* 2004), originally introduced for propositional knowledge integration, in the DL context. The maxi-adjustment algorithm allows the elimination of any inconsistencies that could arise in a stratified propositional KB after its expansion with a new proposition. In (Meyer *et al.* 2005) this idea was applied for stratified DL ontologies, leading to an ontology debugging algorithm. Subsequently, in (Qi *et al.* 2006a), it was adapted and applied to the ontology evolution context, leading to an evolution algorithm for stratified DL ontologies. In (Qi *et al.* 2006b) the theoretical backbone of this approach was presented, leading to a set of postulates, inspired by the belief change postulates of (Katsuno & Mendelzon 1990). Unfortunately, this entire line of work requires the introduction of disjunctive DLs, a certain DL extension for which axiom disjunction is allowed (Meyer *et al.* 2005); this constitutes a departure from the classical DL model, limiting the applicability of the approach.

An interesting variation of the problem of ontology evolution appears in (Foo 1995), (Wassermann 1998), (Wassermann & Fermé 1999). In these papers, evolution is addressed from a different standpoint, in which the evolving objects (and therefore the main objects of study) are the concepts; this viewpoint is quite different from the standard one in which the evolving object is the ontology as a whole. The term ontology revision is often used to refer to this research path (Foo 1995). Another variation, used for multimedia ontology evolution, appears in (Castano *et al.* 2006b), (Castano *et al.* 2007), where ontology matching approaches are used to assist the ontology engineer in defining new concepts to be added to the ontology. Ontology evolution does not normally deal with the evolution of an ontology's metadata; however, this is the case in (Maynard *et al.* 2007), which addresses the problem of identifying the effects of ontological changes to the ontology's metadata and vice-versa, constituting another variation of ontology evolution.

4.3 Ontology Debugging

As already mentioned, the purpose of ontology debugging is the identification and resolution of logical contradictions (see figure 4). Logical contradictions can occur due to several reasons or actions, such as modeling errors, ontology merging or integration, migration from other formalisms and ontology evolution (Haase & Qi 2007).

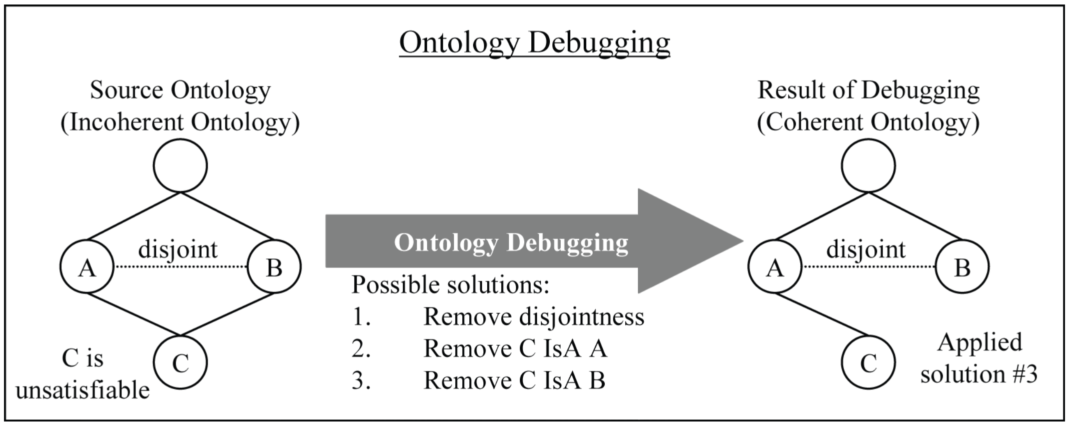

Figure 4 Ontology debugging

Standard reasoners are of little help for the task of ontology debugging, because, even though they can identify the existence of a contradiction, they provide little support for resolving and eliminating it (Meyer *et al.* 2006). The resolution of contradictions is a challenging task for ontology modelers (Lam *et al.* 2006) which cannot be undertaken manually, especially when expressive ontological languages are used. Therefore, a more powerful approach is required in order to identify the part(s) of the ontology that led to the contradiction (Meyer *et al.* 2005).

In principle, there are two types of logical contradictions that can occur in ontologies: *inconsistency* and *incoherency*. Often, the term inconsistency is used to refer to both types of contradictions; here we follow the terminology of (Flouris *et al.* 2006a), where an inconsistent ontology was defined as one which has no model, and an incoherent ontology was defined as one which contains an unsatisfiable concept (i.e., a concept which can have no instances).

Under the above definition, inconsistency corresponds to the standard meaning of the term in logical formalisms and indicates an ontology with trivial consequences (to be exact, anything follows from an inconsistent ontology); as a result, an inconsistency renders the ontology unusable, so ontology debugging techniques should be applied to avoid this trivialization. In the ontological context, incoherency is also important because unsatisfiable concepts often indicate a conceptualization problem in the ontology and should thus be avoided. In fact, most of the approaches in ontology debugging deal with incoherency rather than inconsistency (Meyer *et al.* 2006).

In (Haase *et al.* 2005) four alternative methods for dealing with inconsistency are discussed. In particular, inconsistencies can be avoided (using ontology evolution techniques), corrected (using ontology debugging techniques), ignored (using smart reasoning techniques that will allow meaningful reasoning in the presence of inconsistencies) or addressed indirectly (using versioning techniques that prevent incompatibilities between interrelated ontologies); of course, the latter technique is applicable only when the inconsistency is caused by the interaction of ontologies, i.e., due to some ontology using a version of another ontology with which it is not compatible. In this subsection, we consider the second case (correcting inconsistencies using ontology debugging), whereas ontology evolution and versioning are discussed in other subsections of section 4; approaches that allow reasoning in the presence of inconsistencies are outside the scope of this paper and will not be further discussed.

An interesting overview of the ontology debugging field can be found in (Haase & Qi 2007); in that paper, various different approaches that are directly or indirectly related to debugging are classified and critically evaluated along various criteria. One of the conclusions of (Haase & Qi 2007) is that, non-surprisingly, none of the surveyed approaches is universally applicable for all application scenarios, but different approaches are good for different purposes.

Most of the works related to the field of ontology debugging actually deal with the problem of diagnosis, i.e., the identification of the causes of the contradiction, rather than the resolution of the contradiction. This is because of the general agreement that the best thing an automated system can do is to propose the alternative ways to repair an ontology, but it's up to a human expert to select the appropriate way to resolve the contradiction (Schlobach & Cornet 2003). Thus, research efforts have focused on the problem of diagnosis, leaving the problem of selecting the best way to resolve a contradiction (i.e., ontology repair) to human experts.

Many approaches to ontology debugging use some tableau-based algorithm to pinpoint the cause of an incoherence. One of the most influential approaches to ontology debugging was given in (Schlobach & Cornet 2003), where a tableau-based algorithm for handling incoherencies in DLs (in particular, for the DL *ALC*) was presented. Some useful definitions were also provided there. In (Kalyanpur 2006), (Kalyanpur *et al.* 2006) the tableau-based algorithm of (Schlobach & Cornet 2003) was extended to be able to indicate specific parts of axioms that are responsible for incoherencies in OWL ontologies. A different tableau-based algorithm for debugging OWL ontologies is presented in (Plessers & de Troyer 2006).

In (Meyer *et al.* 2006), another tableau-based approach to handle incoherencies for ontologies represented using a particular DL (namely, *ALC*) is proposed, and some preliminary experimental results on its use are reported. Inspired by this work, (Lam *et al.* 2006) propose a refinement of the above technique which allows a more fine-grained approach to the problem by tracing the parts of the axioms that are responsible for an incoherency; this gives us the ability to restore coherency by eliminating parts of axioms, rather than entire axioms.

A heuristic approach for debugging OWL DL ontologies appears in (Wang *et al.* 2005); this approach is based on five steps that are sequentially executed following the identification of an incoherency by a standard OWL reasoner and allow the pinpointing of the source of the incoherence. In (Schlobach 2005), a framework for debugging incoherent terminologies is proposed, which is based on traditional model-based diagnosis; a major advantage of this framework is that it can be proved to work with a variety of formalisms, including very expressive ones, unlike most approaches which are tailored for particular (and usually simple) DLs. The approach that appears in (Friedrich & Shchekotykhin 2005) is based on similar foundations and enjoys the same general applicability.

All the above works deal with incoherency debugging. In contrast, (Qi & Pan 2007) deals with inconsistency debugging; their approach consists in developing an algorithm that allows reasoning in the presence of inconsistencies and applying it for debugging inconsistent stratified DL ontologies. Similarly, (Meyer *et al.* 2005), despite its title, does not deal with ontology integration, but with ontology debugging; in particular, the authors use the conjunctive maxi-adjustment algorithm of (Benferhat *et al.* 2004) in order to provide a refined algorithm for debugging stratified DL ontologies. This algorithm also deals with the resolution of inconsistency, rather than incoherency, and is applicable for disjunctive DLs only.

4.4 Ontology Versioning

Following a change upon an ontology, ontology versioning algorithms come into play (see table 1 and figure 5). Ontology versioning typically involves the storage of both the old and the new version of the ontology and takes into account identification issues (i.e., how to identify the different versions of the ontology), the relation between different versions (i.e., a tree, or more generally, a directed acyclic graph, of versions resulting from the various ontology modifications) as well as some compatibility information (i.e., information regarding the compatibility of any pair of versions of the ontology). It has been argued that ontology versioning, and, in particular, compatibility determination, cannot be performed automatically (Heflin & Pan 2004), so ontology versioning tools should aim at assisting an ontology engineer specifying the related information.

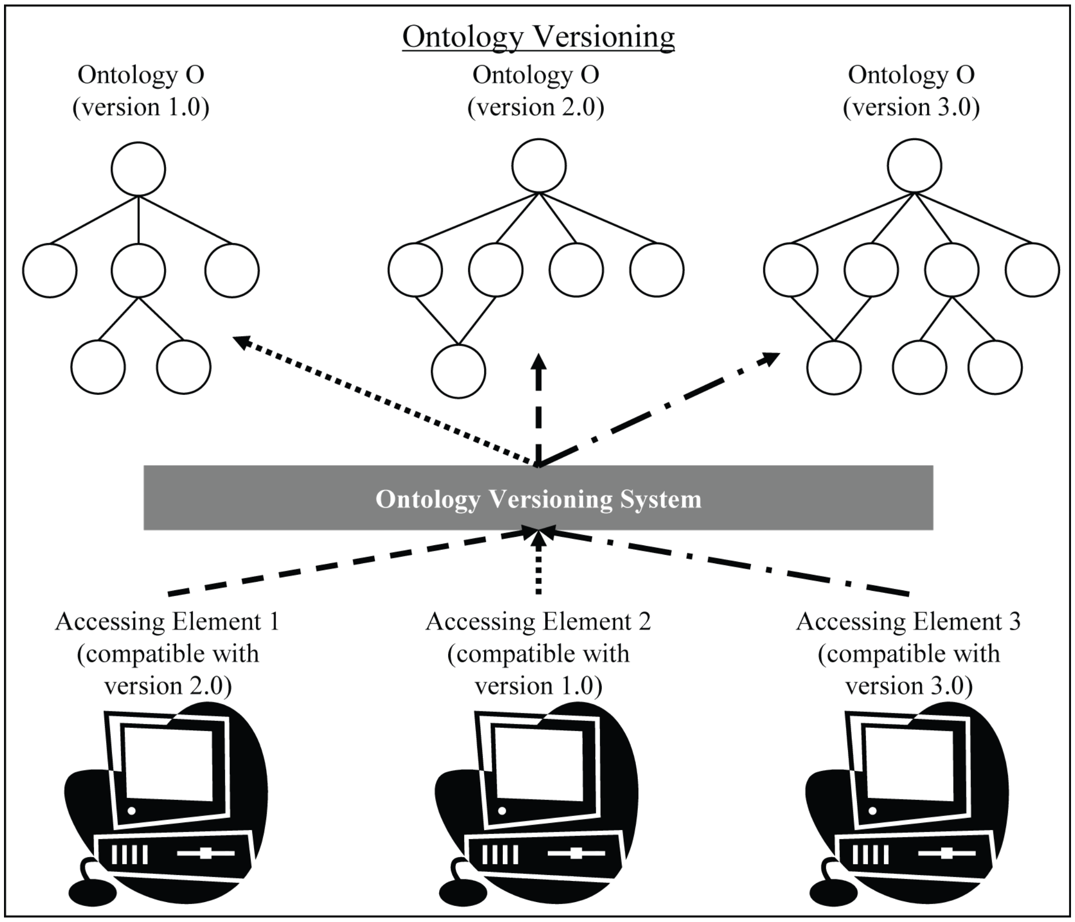
Ontology Versioning
Ontology O (version 1.0)
Ontology O (version 2.0)
Ontology O (version 3.0)
Ontology Versioning System
Accessing Element 1 (compatible with version 2.0)
Accessing Element 2 (compatible with version 1.0)
Accessing Element 3 (compatible with version 3.0)

Figure 5 Ontology versioning

Several non-trivial problems are associated with ontology versioning. For example, any ontology versioning algorithm should be based on some type of identification mechanism to differentiate between various versions of an ontology. This task is not as easy as it may seem; for example, it is not clear when two ontologies constitute different versions. Should any change in the file that stores the ontology specification constitute the creation of a new version? When a concept specification changes, but the new specification is semantically equivalent to the original one, should this constitute a new version? More generally, when the ontology changes syntactically, but not semantically, should this constitute a new version? These and similar problems are dealt with in (Heflin *et al.* 1999), (Klein *et al.* 2002).

Another desirable property of an ontology versioning system is the ability to allow transparent access to different versions of the ontology, by automatically relating versions with data sources, applications and other dependent elements (Klein & Fensel 2001). Other issues involved is the so-called "packaging of changes" (Klein *et al.* 2002) as well as the different types of compatibility and how these are identified (Klein & Fensel 2001).

Another related problem is the introduction of a certain version relation between ontological elements (such as classes) appearing in different versions of the ontology and the properties that such a relation should have. This relation is called a change specification in (Klein & Fensel 2001) and its role is to make the relationship between different versions of ontological elements explicit. Using this relation, one can identify the changes that any given element went through between different versions; in addition, a version relation should include certain meta-data regarding these changes (Klein *et al.* 2002). In (Plessers & de Troyer 2005) this relation is stored using a version log which is actually a specially designed ontology storing the different versions of each element, as well as the relation between them and some related meta-data. Similar considerations led to the definition of migration specifications (Zhang *et al.* 2003), which associate concepts between different versions of an ontology after a change has been performed. A similar approach is presented in (Noy *et al.* 2006), where the CHAO ontology is introduced in order to provide a framework for explicitly storing the changes performed to the ontology.

As an aid to the task of ontology versioning, certain tools have been developed which automatically identify the differences between ontology versions; unfortunately, most such tools provide information at the level of elementary changes only (Klein & Noy 2003). For example, PROMPTDIFF (Noy & Musen 2002), (Noy & Musen 2004), (Noy *et al.* 2004) uses certain heuristics to compare different versions of ontologies and outline their differences, by producing a structural diff between them; OntoView (Klein *et al.* 2002) contains a tool similar to PROMPTDIFF, whose output is a certain ontology of changes, i.e., a specially designed ontology that stores the identified types of changes, thus providing a kind of mapping between versions. In (Zeginis *et al.* 2007) four different ways ("delta functions") to compare RDF ontologies are provided; these functions are based on the ontologies' semantics and some of them take into account the inference mechanism of RDF. In the same paper, these delta functions are compared along several dimensions, like correctness, size etc. The results of comparison tools can be greatly improved if combined with an explicit description of (some of) the changes that the ontology went through, e.g., using the CHAO ontology (Noy *et al.* 2006) which was developed as an add-on for PROMPTDIFF.

A survey on the different ways that can be used to represent a set of changes, as well as the relations and possible interactions between such representations can be found in (Klein & Noy 2003). In the same paper, a standard ontology of changes is proposed, containing both basic (i.e., elementary) and complex (i.e., composite) operations. A similar ontology of changes is proposed in (Plessers & de Troyer 2005), where the changes are identified through a version log stored in this ontology of changes.

SemVersion (Völkel *et al.* 2005) is an RDF-based ontology versioning system that separates the management aspects of the problem from the versioning core functions. An interesting proposal for a theoretical foundation for ontology versioning appears in (Heflin & Pan 2004). A method to identify compatibility between ontology versions is presented in (Heflin *et al.* 1999), (Heflin & Hendler 2000) where the SHOE language (Luke *et al.* 1997) is used to make backward compatibility between versions explicit in a machine-readable format, allowing a computer agent to determine compatibility between versions. This is an indirect approach to the problem of ontology versioning, as it allows the computer agent to determine autonomously which version to use, as opposed to (Klein & Fensel 2001), (Klein *et al.* 2002), where a more direct and centralized path is taken.

In (Huang & Stuckenschmidt 2005), a temporal logic approach is used to allow access and reasoning in different versions of an ontology. Ideas from temporal databases are also employed in (Plessers *et al.* 2005), where a methodology of capturing the evolution of ontology instances with the purpose of performing efficient ontology versioning is described. A temporal DL, coupled with a temporal query language, is presented in (Keberle *et al.* 2007); the authors provide the means to record the evolution of each ontological element through time as well as to determine the status of each element at any given time point (using their temporal DL and query language). This way, one provides an indirect way to support versioning (called ontology evolution analysis in (Keberle *et al.* 2007)), as we are able to determine the status of the ontology at any given time in the past.

5 Ontology Integration and Merging

5.1 Definitions and Discussion on Ontology Integration and Merging

In short, both ontology integration and ontology merging refer to the creation of a new ontology based on the information found in two or more source ontologies. As we will see however, the two terms refer to slightly different research areas. Unfortunately, the exact meaning of each term is not always clear in the literature, as they are often used interchangeably. This situation has led to a certain amount of confusion regarding the exact boundaries of each field.

In this work, we will define these terms along the lines of (Pinto *et al.* 1999), which was an attempt to disambiguate between different uses of the term "ontology integration". Three different uses of the term were identified in that paper. The first refers to *the composition (via reuse) of ontologies covering loosely related (i.e., similar) domains (subjects)*; this is mainly used when building a new ontology that covers all these subjects. The term *ontology integration* has been reserved for this process. The second use of the word integration refers to *the combination of ontologies covering highly overlapping or identical domains*; this process is used to fuse ontologies that contain information about the same subject into one large (and hopefully more accurate) ontology. The term *ontology merging* was attached to this interpretation. Finally, the third use of the term refers to *the development of an application that uses one or more ontologies*; the more appropriate term *ontology use* was reserved for this process. In this work, we will be interested only in the first two research areas, namely ontology integration and merging, because ontology use could not be considered a type of ontology change.

There are certain subtle differences between the processes of ontology integration and merging (see table 1). Ontology integration is mainly applied when the main concern is the reuse of other ontologies. The domain of discourse of the new ontology is usually more general than the domain of any of the sources, and integration often places the different source ontologies in different modules that comprise the resulting ontology; these modules are only loosely interconnected in the final result. Using ontology integration, the process of ontology development can become more efficient, because previously developed and tested ontologies can be reused as building blocks for the creation of the new (and more general) ontology in a modular manner. For a graphical depiction of the process of ontology integration, see figure 6.

On the other hand, in ontology merging, the focus is on creating an ontology that combines information on a given topic from different sources. In this case, the information from the source ontologies is greatly intermingled, so it's difficult to identify the part(s) of the final ontology that resulted from each source ontology. This process is useful when each source ontology models the domain under question only partially, or under a particular viewpoint, and we would like to combine the information into a larger, more accurate and complete ontology. For a graphical depiction of the process of ontology merging, see figure 7. A more extensive discussion on the differences between integration and merging can be found in (Pinto *et al.* 1999).

5.2 Alternative Understandings of the Terms

It is a common practice in the literature to consider heterogeneity resolution to be an internal part of ontology merging or integration (De Bruijn *et al.* 2004), (Choi *et al.* 2006), (Heflin *et al.* 1999), (Pinto *et al.* 1999). This is a reasonable choice, because in most cases the fused ontologies come from different sources, so they are generally heterogeneous in terms of vocabulary, syntax, representation etc. Therefore, the task of relating (matching) the heterogeneous elements of the source ontologies constitutes the first step (and a major part) of the task of ontology merging or integration (Choi *et al.* 2006). This is mostly true in merging, where the domain of discourse is (almost) identical.

Unfortunately however, this has led to even more confusion on the exact meaning of the terms, as ontology merging (or integration) are often identified with the process of resolving heterogeneity issues and several works consider ontology merging (or integration) and matching to be variations of the same problem. In (Noy & Musen 2000), (Noy & Musen 1999a), for example, ontology merging is defined as the process of creating a new, coherent ontology that includes information from two or more source

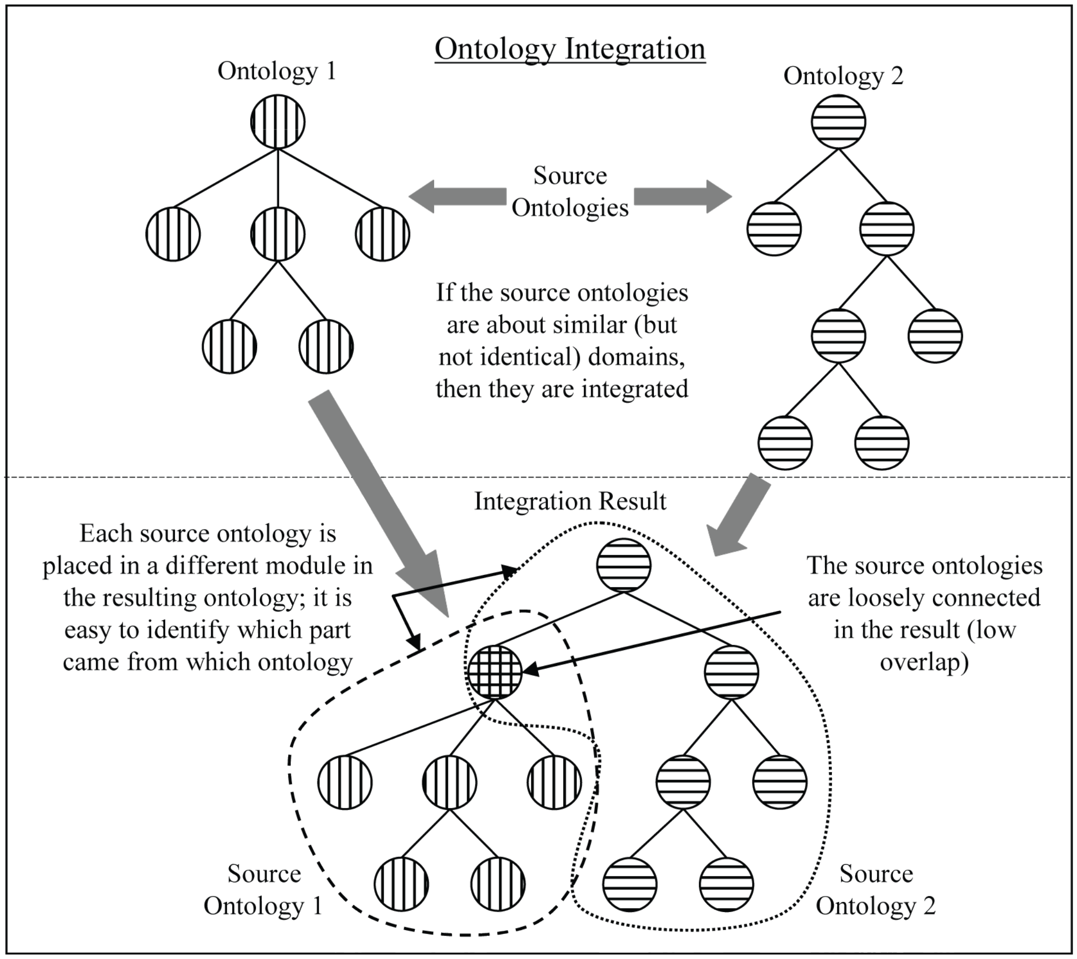
Ontology Integration
Ontology 1
Ontology 2
Source Ontologies
If the source ontologies are about similar (but not identical) domains, then they are integrated
Integration Result
Each source ontology is placed in a different module in the resulting ontology; it is easy to identify which part came from which ontology
The source ontologies are loosely connected in the result (low overlap)
Source Ontology 1
Source Ontology 2

Figure 6 Ontology integration

ontologies. In these papers, ontology merging is implicitly assumed to include the process of resolving any possible heterogeneities between the merged ontologies, even though this is not apparent in their definition. In addition, under their understanding, the difference between ontology merging and matching is that ontology merging results in the creation of a new ontology, whereas in ontology matching the merged ontologies persist, with links established between them.

A similar use of the term can be found in (McGuiness *et al.* 2000), whereas, in (Heflin *et al.* 1999), the same research area is described using the term "ontology integration". According to (Lee & Meyer 2004), ontology merging amounts to making sure that different agents use the same terms in identical ways (in a manner similar to ontology matching). In (Kalfoglou & Schorlemmer 2003) ontology integration is defined along our lines, i.e., as the process of combining ontologies to build new ones, but whose respective vocabularies are usually not interpreted in the same domain of discourse.

However, it is important to note that simply resolving the heterogeneity issues between two ontologies is not sufficient for successful integration (or merging); recall that different ontologies may encode different viewpoints regarding the real world, thus several conceptual differences are bound to exist, even if the same terminology is used. This is reminiscent of how beliefs held by different people are often different (and in some cases contradictory), even if a common terminology is agreed upon. Similarly, modeling conventions and choices may be different; one example of a modeling choice that often depends on personal taste or convention is whether to model a certain distinction between similar elements by introducing separate classes or by introducing a qualifying attribute relation in one class (Chalupsky 2000); an example of this particularly difficult situation appears in figure 8. Such modeling

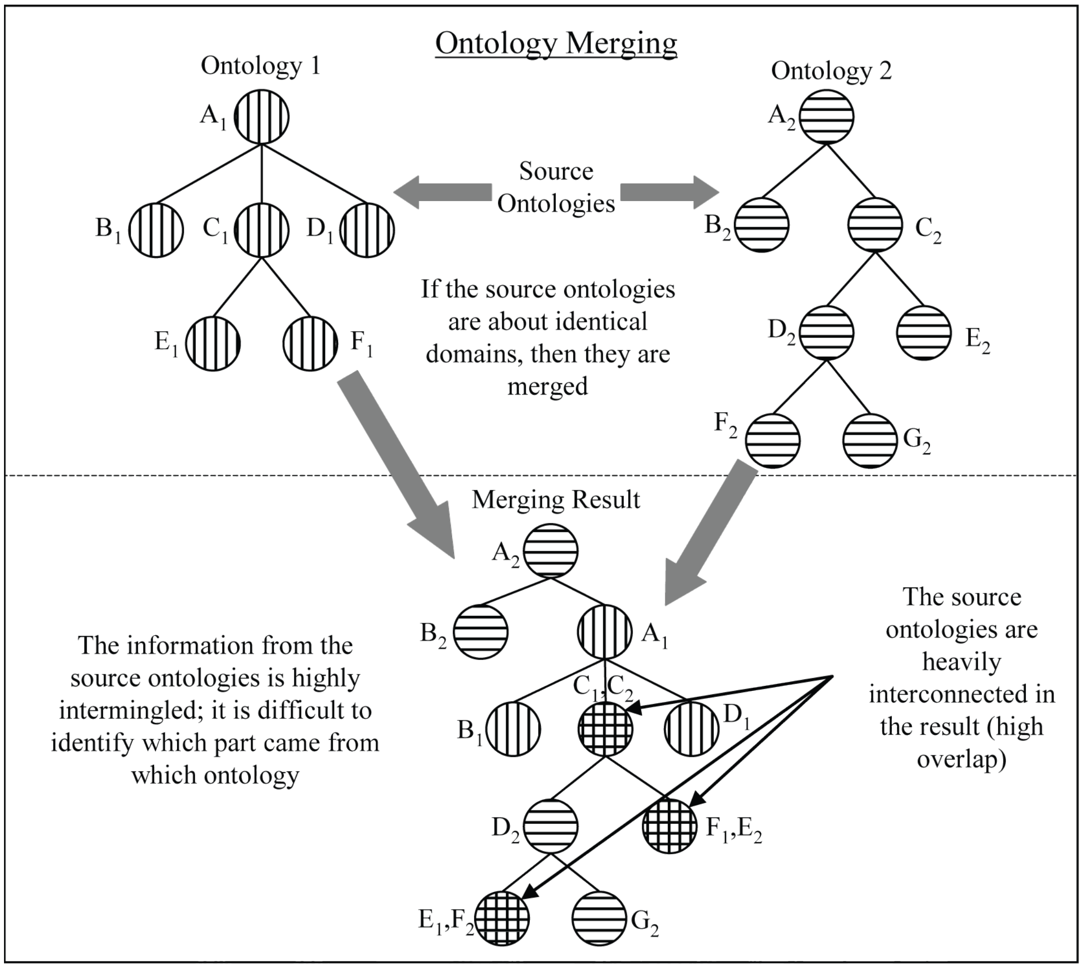
Ontology Merging
Ontology 1
A_1
B_1
C_1
D_1
E_1
F_1
Source Ontologies
Ontology 2
A_2
B_2
C_2
D_2
E_2
F_2
G_2
If the source ontologies are about identical domains, then they are merged
Merging Result
A_2
B_2
A_1
B_1
C_1,C_2
D_1
D_2
F_1,E_2
E_1,F_2
G_2
The information from the source ontologies is highly intermingled; it is difficult to identify which part came from which ontology
The source ontologies are heavily interconnected in the result (high overlap)

Figure 7 Ontology merging

differences need to be taken into account when selecting what to keep from each ontology during the integration or merging process. Reckless inclusion of ontology elements from the source ontologies (even when homogeneous) is likely to lead to a problematic, invalid, contradictory, incoherent or inconsistent ontology. For this reason we argue that ontology merging (and integration) are not simply a variation of ontology matching.

A slightly different use of the term "ontology integration" appears in (Calvanese *et al.* 2002), where the term is used to refer to the process of combining a number of local ontologies in order to build a global one, with the purpose of being able to answer queries over the local ontologies, using the unified terminology imposed by the global ontology. Again, the focus is on heterogeneity resolution and a formal framework for defining and exploiting mappings with respect to query answering is proposed.

In (De Bruijn *et al.* 2004), ontology merging is defined as the process of unifying two or more ontologies; this definition poses no restrictions as to the domain of the unified ontologies, so it includes both ontology integration and merging. In the same paper, two approaches to ontology merging are identified: the union approach, where the merged ontology is the union of all entities in both source ontologies, taking into account heterogeneity issues, and the intersection approach, where only overlapping parts of the source ontologies are included in the result.

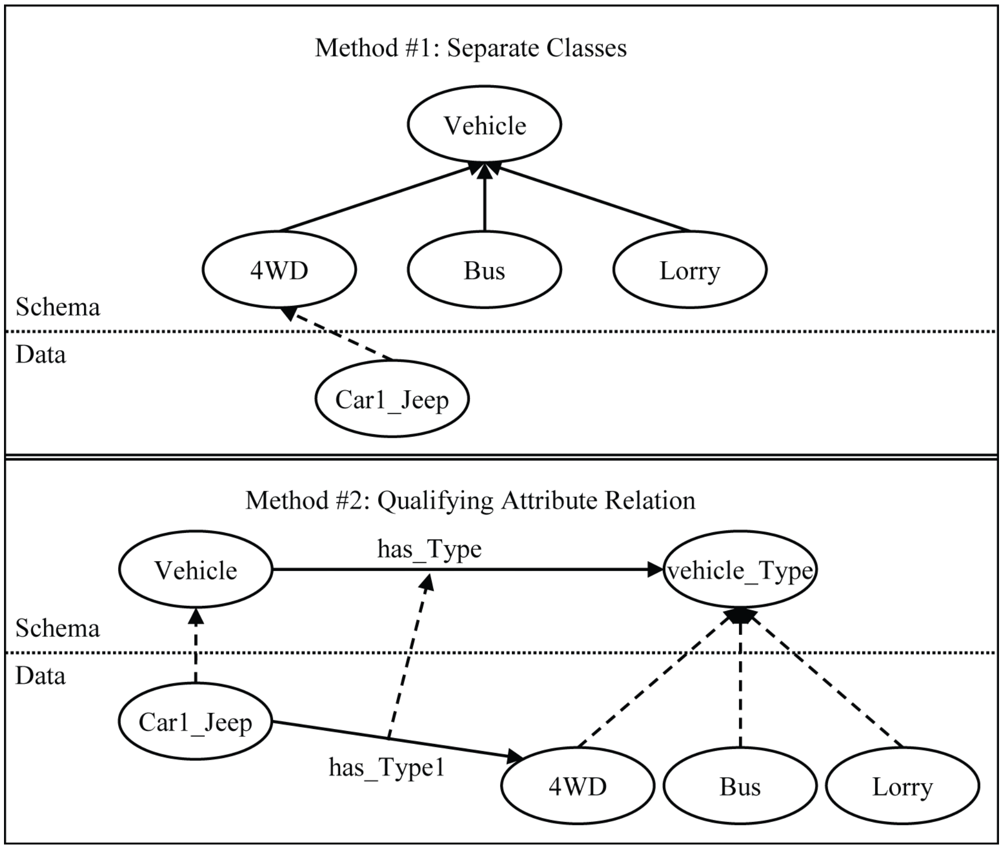

Figure 8 A modeling convention that needs to be taken into account in ontology integration and merging

5.3 State of the Art in Ontology Integration and Merging

According to (Chalupsky 2000), the process of merging can be broken down in five steps. During the first step, we identify the semantic overlap between the source ontologies; during the second, we devise ways (transformations) that will bring the sources into mutual agreement in terms of terminology, representation etc. In the third step, we apply these transformations, so we can now take the union of the sources, which is the fourth step of the process. The final step consists of evaluating the resulting ontology for consistency, uniformity, redundancy, quality of conceptualization etc; this evaluation might force us to repeat some or all of the above steps. We believe that the same generic steps are necessary for integration as well, but they should be adapted to the different properties of the process. In (Chalupsky 2000), it is claimed that OntoMorph (a translation tool described in the same paper) is a necessary part of any merging algorithm, as it facilitates the design and implementation of the transformations used in the merging process (steps 2, 3 respectively).

An extensive list of works related to ontology merging can be found in (De Bruijn *et al.* 2004), (Choi *et al.* 2006), (Pinto *et al.* 1999). The main tools used for ontology merging are PROMPT (Noy & Musen 2003), (Noy & Musen 2000) and Chimaera (McGuiness *et al.* 2000). These tools use a semi-automatic approach focused on suggesting how elements from the source ontologies should be merged in the resulting ontology. The final choice relies on the ontology engineer.

Some ideas on ontology merging (called integration there) in the context of the SHOE language can be found in (Heflin *et al.* 1999); however, this work is focused on the part of merging that deals with heterogeneity resolution. The FCA-MERGE algorithm (Stumme & Maedche 2002) performs ontology integration in a very efficient way, but is based on certain strong assumptions. In (Noy & Musen 1999a) some interesting connections of object-oriented programming with the problem of ontology merging are uncovered. An interesting theoretical approach to ontology integration (also applicable to ontology merging) appears in (Calvanese *et al.* 2002), which focuses on the formal definition of mappings between the resulting and the source ontologies and how these mappings can be exploited for query answering; another theoretical approach to ontology merging can be found in (Bench-Capon & Malcolm 1999).

It should be emphasized that, to the best of our knowledge, all currently available tools use manual or semi-automatic methods to perform merging (or integration), guiding the user through the different steps of the process via some intuitive interface. The automatic processes behind ontology merging and integration are not yet well-understood: in (Pinto *et al.* 1999), it is claimed that "ontology merging is currently more of an art". Ontology integration (and merging) can benefit from advances in the related field of database schema integration (and merging); an indicative list of relevant systems (performing database schema integration) includes Quete (Kondylakis, 2006), BACIIS (Ben-Miled *et al.* 2005), TAMBIS (Stevens *et al.* 2000), ONTOFUSION (Perez-Ray *et al.* 2006), MECOTA (Goasdoui *et al.* 2000) and SEMEDA (Kohler *et al.* 2003). Further details on these systems are omitted, as they deal with databases rather than ontologies, so they are outside the scope of this paper.

Even though the issue of evaluating ontology integration and merging techniques is still open (Stumme & Maedche 2002), certain comparison attempts have been made. In (Lambrix & Edberg 2003), the authors perform a comparison between PROMPT and Chimaera in the context of bioinformatics. In (Noy & Musen 2000), the same two tools are compared with the generic Protégé (Noy *et al.* 2000), whereas (McGuiness *et al.* 2000) compares, in the context of merging, the efficiency of a simple plain-text editor, the Ontolingua editor and the specialized tool Chimaera (described in the same paper). These comparisons are made from a certain standpoint; a general, objective comparison is difficult, as it is not clear how the utility of such tools could be measured (McGuiness *et al.* 2000).

6 Conclusion

We performed a literature review covering all the diverse types of ontology change, focusing on classification and breadth of coverage rather than on depth of analysis. Our aim was to propose a terminology in an area that is plagued by the presence of underspecified and confusing terms. A comprehensive summary of our results can be found in table 1. The proposed terminology was not introduced in an arbitrary manner, but was based on similar previous attempts (like, for example, (Kalfoglou & Schorlemmer 2003), (Pinto *et al.* 1999)) and on the most common uses of the terms in the literature. In addition to the proposed terminology, we discussed alternative definitions and uses of the terms that have appeared in the literature. This would minimize the amount of effort required for a reader in order to follow a work that uses an interpretation different from the one proposed here and provides a more complete view of the area of ontology change.

In this paper, we used the term "ontology change" in a very general sense, so as to engulf any approach that is related to the dynamics of ontologies, including works that are indirectly related to the problem, such as those dealing with heterogeneity resolution or the management of different versions of an evolving ontology. We identified 10 research subfields which can fit under the general definition of ontology change, namely ontology mapping, morphism, matching, articulation, translation, evolution, debugging, versioning, integration and merging. In short, ontology mapping, morphism, matching, articulation and translation address the problem of heterogeneity resolution, ontology evolution deals with the incorporation of new knowledge in an ontology, ontology debugging seeks techniques that would remove contradictions from an ontology, ontology versioning addresses the management of different versions of an evolving ontology and ontology integration and merging deal with the fusion of knowledge from different ontologies into a single one.

Many of the above fields are closely related; as a result, many works and approaches deal with more than one of them. For example, several works addressing the problem of ontology merging or integration also deal with heterogeneity issues, and many ontology evolution algorithms address versioning or debugging issues (see also table 2). This is the primary reason for the confusion that often exists as to the meaning of the various terms discussed here; the clarification of this confusion constitutes the main motivation behind this survey.

The second purpose of this survey was an overview of the ontology change literature, focusing on breadth of coverage rather than depth of analysis. In this respect, we briefly described several works related to ontology change and classified them according to the problem(s) that they address (see table 2 for a summary). Our analysis indicated the main trends in each subfield of ontology change and showed the main strengths and weaknesses of the dominating research paradigms.

More specifically, as far as heterogeneity resolution is concerned, it was made clear that the problem is very difficult and multi-faceted. Therefore, despite the persistent efforts of many capable researchers, the results are still not entirely satisfactory; many people believe that the process cannot be fully automated and that the more realistic path is to resort to semi-automatic methods that would significantly reduce (but not eliminate) human intervention. At the moment, there are several available systems that perform mapping, matching or some other flavor of heterogeneity resolution, and many complementary approaches to address the problem have been proposed (see figure 2 for a summary).

In ontology evolution, two major research paths can be identified. The first focuses on aiding the user performing changes through some intuitive interface providing useful editing features; such approaches resemble an ontology editor, even though they often provide many more features than a simple ontology editor would. The second research path focuses on the development of automated methods to determine the effects and side-effects of any given update request; this approach often borrows ideas from the related field of belief change. The first class of tools is more mature at the moment, but the second approach is more promising and more interesting from a research point of view; for this reason, it is gaining increasing attention during the last few years. The two research paths are complementary, as results from the second could be applied to the first in order to further improve the quality of the front-end editing tools; similarly, automated approaches are of little use unless coupled with tools that address the practical issues related to evolution, like support for multi-user environments, transactional issues, change propagation etc.

Ontology debugging is a recent field; (Schlobach & Cornet 2003) was practically the first approach dealing with the problem. This field has strong connections to the ontology evolution field (especially the second research path described above), so many ontology evolution approaches could be adapted to solve problems related to ontology debugging and vice-versa. At the moment, mainly the problem of incoherence is addressed in ontology debugging, but the field is in need of approaches that could handle both inconsistency and incoherency.

Often, ontology versioning approaches do not deal with the problem of versioning itself (recording and storing versions) but with peripheral problems, like the identification and recording of the changes between versions or the determination of compatibility between versions. This is normal, given that these problems constitute subproblems of ontology versioning and are more interesting, from a research point of view, than the (purely technical) problem of deciding how to record and store versions. There is a number of interesting results in this respect, but most problems related to versioning remain open.

Most approaches to ontology merging and integration deal with the part of merging and integration that is related to heterogeneity resolution. Even though this is an important aspect of the problem, one should not underestimate the difficulties related to the merging (or integration) of homogeneous ontologies; given that different ontologies encode different viewpoints, the reckless addition of facts from one ontology to the other would, most likely, lead to an inconsistent, incoherent or otherwise invalid ontology. We believe that this aspect of merging and integration would be an interesting field for future research.

In general, ontology change is a complex and multi-faceted problem, subparts of which are being addressed by different disciplines. We hope that our work will prove helpful towards the clarification of the terminology, as well as the boundaries and relations between these disciplines. We aim this work to serve as a starting point for researchers interested in any of the many facets of the problem.

Acknowledgements

This paper is a heavily revised and extended version of (Flouris *et al.* 2006c). The authors would like to thank Panos Constantopoulos, Vassilis Christophides and Nicolas Spyratos for helpful comments on an earlier draft of this work, and the anonymous reviewers for their helpful comments. This work was partly carried out during the first author's tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

Appendix

Table 2 contains a comprehensive classification of the works referenced in this paper. Using this table, the reader can get a quick glimpse of the papers related to any specific subfield of ontology change, as well as to determine the relevance of any given work to any particular discipline; for a more accurate account of the contribution of each paper to the respective discipline, please refer to the appropriate section of this survey. Notice that referenced works that are not directly related to ontology change have not been included in the table.

Table 2: Referenced papers and related ontology change subfields

Referenced Paper	Mapping	Morphism	Matching	Articulation	Translation #1	Translation #2	Evolution	Debugging	Versioning	Integration	Merging
Aumueller *et al.* 2005			✓								
Avesani *et al.* 2005	✓		✓								
Bechhofer *et al.* 2001							✓				
Bench-Capon & Malcolm 1999											✓
Bernstein *et al.* 2004			✓								
De Bruijn *et al.* 2004	✓		✓							✓	✓
Calvanese *et al.* 2002	✓									✓	
Castano *et al.* 2007							✓				
Castano *et al.* 2006a			✓								
Castano *et al.* 2006b			✓				✓				
Chalupsky 2000						✓					✓
Choi *et al.* 2006	✓		✓							✓	✓
David *et al.* 2006			✓								
Do & Rahm 2002						✓					
Do *et al.* 2002	✓		✓								
Doan *et al.* 2000			✓								
Doan *et al.* 2002			✓								
Duineveld *et al.* 2000							✓				
Ehrig *et al.* 2005			✓								
Euzenat *et al.* 2004	✓		✓								
Euzenat & Shvaiko 2007	✓		✓	✓							
Ferrara 2004	✓		✓								
Flouris 2006							✓				
Flouris *et al.* 2006a							✓				
Flouris & Plexousakis 2005	✓	✓		✓	✓	✓	✓		✓	✓	✓
Flouris & Plexousakis 2006							✓				
Flouris *et al.* 2004							✓				
Flouris *et al.* 2005							✓				
Flouris *et al.* 2006b							✓				
Flouris *et al.* 2006c	✓	✓		✓	✓	✓	✓		✓	✓	✓
Friedrich & Shchekotykhin 2005								✓			
Foo 1995							✓				
Gabel *et al.* 2004							✓				
Giunchiglia *et al.* 2004			✓								
Giunchiglia *et al.* 2006			✓								

continued on next page

Table 2: Referenced papers and related ontology change subfields *(cont.)*

Referenced Paper	Mapping	Morphism	Matching	Articulation	Translation #1	Translation #2	Evolution	Debugging	Versioning	Integration	Merging
Gutierrez *et al.* 2006							✓				
Haase *et al.* 2005							✓	✓			
Haase & Qi 2007								✓			
Haase & Stojanovic 2005							✓				
Haase & Sure 2004							✓		✓		
Halaschek-Wiener & Katz 2006							✓				
Heflin & Hendler 2000	✓								✓		
Heflin *et al.* 1999	✓								✓		✓
Heflin & Pan 2004									✓		
Hu & Qu 2006			✓								
Huang & Stuckenschmidt 2005									✓		
(IBM Corporation)			✓								
Kalfoglou & Schorlemmer 2003	✓	✓	✓	✓	✓	✓				✓	
Kalyanpur 2006								✓			
Kalyanpur *et al.* 2006								✓			
Kang & Lau 2004							✓				
Keberle *et al.* 2007									✓		
Klein & Fensel 2001							✓		✓		
Klein *et al.* 2002									✓		
Klein & Noy 2003							✓		✓		
Kondylakis *et al.* 2006	✓		✓								
Konstantinidis *et al.* 2007							✓				
Lam *et al.* 2006								✓			
Lam *et al.* 2005							✓				
Lambrix & Edberg 2003											✓
Lee & Meyer 2004							✓				
Liu *et al.* 2006							✓				
Luke *et al.* 1997									✓		
Madhavan *et al.* 2005			✓								
Madhavan *et al.* 2001			✓								
Maedche *et al.* 2003							✓				
Manakanatas & Plexousakis 2006	✓		✓								
Magiridou *et al.* 2005							✓				
Maynard *et al.* 2007							✓				
McGuiness *et al.* 2000	✓		✓							✓	✓
Melnik *et al.* 2002			✓								
Meyer *et al.* 2005								✓			
Meyer *et al.* 2006								✓			
Mitra *et al.* 2005	✓		✓								
Mocan *et al.* 2006	✓										
Noy *et al.* 2006							✓		✓		
Noy *et al.* 2000							✓				

continued on next page

Table 2: Referenced papers and related ontology change subfields *(cont.)*

Referenced Paper	Mapping	Morphism	Matching	Articulation	Translation #1	Translation #2	Evolution	Debugging	Versioning	Integration	Merging
Noy & Klein 2004							✓				
Noy *et al.* 2004									✓		
Noy & Musen 1999a			✓								✓
Noy & Musen 1999b			✓								✓
Noy & Musen 2000			✓								✓
Noy & Musen 2002											✓
Noy & Musen 2003	✓		✓								✓
Noy & Musen 2004									✓		
Palmisano *et al.* 2006	✓		✓								
Pan *et al.* 2005	✓		✓								
Pinto *et al.* 1999										✓	✓
Plessers & de Troyer 2005							✓		✓		
Plessers & de Troyer 2006								✓			
Plessers *et al.* 2005							✓		✓		
Qi *et al.* 2006a							✓				
Qi *et al.* 2006b							✓				
Qi & Pan 2007								✓			
Rahm & Bernstein 2001			✓								
Ribeiro & Wassermann 2007							✓				
Roger *et al.* 2002							✓				
Schlobach 2005								✓			
Schlobach & Cornet 2003								✓			
Shvaiko & Euzenat 2005			✓								
Shvaiko *et al.* 2006			✓								
Stoilos *et al.* 2005	✓		✓								
Stojanovic *et al.* 2002							✓				
Stojanovic *et al.* 2003							✓				
Stojanovic & Motik 2002							✓				
Stuckenschmidt & Klein 2003							✓				
Stumme & Maedche 2002										✓	
Sure *et al.* 2003							✓				
Svab *et al.* 2007	✓		✓								
Völkel *et al.* 2005									✓		
Wang *et al.* 2005								✓			
Wassermann 1998							✓				
Wassermann & Fermé 1999							✓				
Yatskevich 2003	✓		✓								
Zeginis *et al.* 2007									✓		
Zhang *et al.* 2003									✓		
Zhdanova & Shvaiko 2006			✓								

References

Alchourron, C., Gärdenfors, P., & Makinson, D. 1985. "On the Logic of Theory Change: Partial Meet Contraction and Revision Functions" *Journal of Symbolic Logic*, **50**, pp. 510-530.

Aumueller D., Do, H.H., Massmann, S. & Rahm, E. 2005. “Schema and Ontology Matching with COMA++” *ACM SIGMOD International Conference on Management of Data*, pp. 906-908.

Avesani, P., Giunchiglia, F. & Yatskevich, M. 2005. “A Large Scale Taxonomy Mapping Evaluation” *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 67-81.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. eds 2002. *The Description Logic Handbook: Theory, Implementation and Applications* Cambridge University Press.

Baader, F., Horrocks, I. & Sattler, U. 2003. “Description Logics as Ontology Languages for the Semantic Web” in Hutter, D. & Stephan, W. eds *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag.

Banerjee, J., Kim, W., Kim, H. J. & Korth, H.F. 1987. “Semantics and Implementation of Schema Evolution in Object-Oriented Databases” *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pp. 311-322.

Bechhofer, S., Horrocks, I., Goble, C. & Stevens, R. 2001. “OilEd: A Reason-able Ontology Editor for the Semantic Web” *Proceedings of the 24th German / 9th Austrian Conference on Artificial Intelligence (KI-01).*

Ben-Miled, Z., Li, N. & Bukhres, O. 2005. “BACIIS: Biological and Chemical Information Integration Systems” *Journal of Database Management*, **16** (3), pp. 73-85.

Bench-Capon, T. & Malcolm, G. 1999. “Formalizing Ontologies and Their Relations” *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA-99)*, pp. 250-259.

Benferhat, S. Kaci, S., Le Berre, D. & Williams, M. 2004. “Weakening Conflicting Information for Iterated Revision and Knowledge Integration” *Artificial Intelligence*, **153**, pp. 339-371.

Berners-Lee, T., Hendler, J. & Lassila, O. 2001. “The Semantic Web” *Scientific American*, **284** (5), pp. 34-43.

Bernstein, P.A., Melnik, S., Petropoulos, M. & Quix, C. 2004. “Industrial-Strength Schema Matching” *SIGMOD Record*, **33**(4), pp. 38-43.

De Bruijn, J., Martin-Recuerda, F., Manov, D. & Ehrig, M. 2004. “D4.2.1: State of the Art Survey on Ontology Merging and Aligning” available on the Web (last visited November, 2007):
http://www.aifb.uni-karlsruhe.de/WBS/meh/publications/debruijn04state.pdf

Calvanese, D., De Giacomo, G. & Lenzerini, M. 2002. “A Framework for Ontology Integration” in Cruz, I., Decker, S., Euzenat, J. & McGuinness, D. eds *The Emerging Semantic Web. Selected Papers from the First Semantic Web Working Symposium*, IOS Press, pp. 201-214.

Castano, S., Espinosa, S., Ferrara, A., Karkaletsis, V., Kaya, A., Melzer, S., Moller, R., Montanelli, S. & Petasis, G. 2007. “Ontology Dynamics with Multimedia Information: The BOEMIE Evolution Methodology” *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 41-54.

Castano, S., Ferrara, A. & Montanelli, S. 2006a. “Matching Ontologies in Open Networked Systems: Techniques and Applications” *Journal on Data Semantics (JoDS)* Vol. **V**.

Castano, S., Ferrara, A. & Hess, G.N. 2006b. “Discovery-Driven Ontology Evolution” *Proceedings of the 3rd Workshop on Semantic Web Applications and Perspectives (SWAP-06).*

Chalupsky, H. 2000. “OntoMorph: A Translation System for Symbolic Knowledge” *Proceedings of the 7th International Conference on Knowledge Representation and Reasoning (KR-00).*

Choi, N., Il-Yeol, S. & Han, H. 2006. “A Survey on Ontology Mapping” *ACM SIGMOD Record*, **35** (3), pp. 34-41.

David, J., Guillet, F., Gras, R. & Briand, H. 2006. “Conceptual Hierarchies Matching: An Approach Based on Discovery of Implication Rules Between Concepts” *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06)*, pp. 357-361.

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuiness, D., Patel-Schneider, P. & Stein, L. A. 2004. “OWL Web Ontology Language Reference” W3C Recommendation, available on the Web (last visited November, 2007):
http://www.w3.org/TR/owl-ref/

Do, H.H. & Rahm, E. 2002. “COMA – A System for Flexible Combination of Schema Matching Approaches” *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB-02)*, pp. 610-621.

Do, H.H., Rahm, E. & Melnik, S. 2002. “Comparison of Schema Matching Evaluations” *Proceedings of the GI-Workshop of Web and Databases.*

Doan, A.H., Domingos, P. & Levy, A. 2000. “Learning Source Descriptions for Data Integration” *Proceedings of the 3rd International Workshop on Web and Databases (WebDB-00)*, pp. 81-92.

Doan, A.H., Madhavan, J., Domingos, P. & Halevy, A. 2002. “Learning to Map Between Ontologies on the Semantic Web” *Proceedings of the 11th International WWW Conference (WWW-02)*, pp. 662-673.

Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B. & Benjamins, V.R. 2000. “WonderTools? A Comparative Study of Ontological Engineering Tools” *International Journal of Human-Computer Studies*, **52**(6), pp. 1111-1133.

Ehrig, M., Staab, S. & Sure, Y. 2005. “Bootstrapping Ontology Alignment Methods with APFEL” *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 186-200.

Euzenat, J., Le Bach, T., Barrasa, J., Bouquet, P., de Bo, J., Dieng, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., Maynard, D., Napoli, A., Stamou, G., Stuckeschmidt, H., Shvaiko, P., Tessaris, S., van Acker, S. & Zaihrayeu, I. 2004. “D2.2.3: State of the Art on Ontology Alignment” available on the Web (last visited November, 2007):
http://www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf

Euzenat, J. & Shvaiko, P. 2007. *Ontology Matching*, Springer-Verlag.

Ferrara, A. 2004. "Methods and Techniques for Ontology Matching and Evolution in Open Distributed Systems" *Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAISE-04).*

Flouris, G. 2006. *On Belief Change and Ontology Evolution*, Doctoral Dissertation, Department of Computer Science, University of Crete.

Flouris, G., Huang, Z., Pan, J., Plexousakis, D. & Wache, H. 2006a. "Inconsistencies, Negations and Changes in Ontologies" *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06).*

Flouris, G. & Plexousakis, D. 2005. "Handling Ontology Change: Survey and Proposal for a Future Research Direction" Technical Report FORTH-ICS/TR-362.

Flouris, G. & Plexousakis, D. 2006. "Bridging Ontology Evolution and Belief Change" *Proceedings of the 4th Hellenic Conference on Artificial Intelligence (SETN-06).*

Flouris, G., Plexousakis, D. & Antoniou, G. 2004. "Generalizing the AGM Postulates: Preliminary Results and Applications" *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR-04)*, pp. 171-179.

Flouris, G., Plexousakis, D. & Antoniou, G. 2005. "On Applying the AGM Theory to DLs and OWL" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 216-231.

Flouris, G., Plexousakis, D. & Antoniou, G. 2006b. "Evolving Ontology Evolution" *Proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-06)*, Invited Talk.

Flouris, G., Plexousakis, D. & Antoniou, G. 2006c. "A Classification of Ontology Change" *Poster Proceedings of the 3rd Italian Semantic Web Workshop, Semantic Web Applications and Perspectives (SWAP-06).*

Foo, N. 1995. "Ontology Revision" *Proceedings of the 3rd International Conference on Conceptual Structures (ICCS-95)*, Lecture Notes in Artificial Intelligence (LNAI), Volume 954, Springer-Verlag, pp. 16-31.

Franconi, E., Grandi, F. & Mandreoli, F. 2000. "A Semantic Approach for Schema Evolution and Versioning in Object-Oriented Databases" *Proceedings of the 6th International Conference on Rules and Objects in Databases (DOOD-00).*

Friedrich, G. & Shchekotykhin, K. 2005. "A General Diagnosis Method for Ontologies" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 232246.

Gabel, T., Sure, Y. & Voelker, J. 2004. "D3.1.1.a: KAON – Ontology Management Infrastructure" SEKT informal deliverable, available on the Web (last visited November, 2007): http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/SEKT-D3.1.1.a.pdf

Gärdenfors, P. 1992a. "Belief Revision: An Introduction" in Gärdenfors, P. ed *Belief Revision*, pp. 1-20, Cambridge University Press.

Gärdenfors, P. 1992b. "The Dynamics of Belief Systems: Foundations Versus Coherence Theories" *Revue Internationale de Philosophie*, **44**, pp. 24-46.

Giunchiglia, F., Shvaiko, P. & Yatskevich, M. 2004. "S-match: an Algorithm and an Implementation of Semantic Matching" *Proceedings of the 1st European Semantic Web Symposium*, LNCS 3053, pp. 61-75.

Giunchiglia, F., Shvaiko, P. & Yatskevich, M. 2006. "Discovering Missing Background Knowledge in Ontology Matching" *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06)*, pp. 382-386.

Goasdoui, F., Lattes, V. & Rousset, M.C. 2000. "The Use of CARIN Language and Algorithms for Information Integration: the PICSEL Project" *International Journal of Cooperative Information Systems* **9** (4), pp. 383-401.

Gruber, T.R. 1993a. "A Translation Approach to Portable Ontology Specifications" *Knowledge Acquisition*, **5** (2), pp. 199-220.

Gruber, T.R. 1993b. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing" *Formal Ontology in Conceptual Analysis and Knowledge Representation*, also available as Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University.

Guarino, N. 1998. "Formal Ontology and Information Systems" *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems (FOIS-98)*, IOS Press, pp. 3-15.

Gutierrez, C., Hurtado, C. & Vaisman, A. 2006. "The Meaning of Erasing in RDF Under the Katsuno-Mendelzon Approach" *Proceedings of the 9th International Workshop on the Web and Databases (WebDB-06).*

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. 2005. "A Framework for Handling Inconsistency in Changing Ontologies" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 353-367.

Haase, P. & Qi, G. 2007. "An Analysis of Approaches to Resolving Inconsistencies in DL-based Ontologies" *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 97-109.

Haase, P. & Stojanovic, L. 2005. "Consistent Evolution of OWL Ontologies" *Proceedings of the 2nd European Semantic Web Conference (ESWC-05).*

Haase, P. & Sure, Y. 2004. "D3.1.1.b State of the Art on Ontology Evolution" available on the Web (last visited November, 2007):
http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/SEKT-D3.1.1.b.pdf

Halaschek-Wiener, C. & Katz, Y. 2006. "Belief Base Revision For Expressive Description Logics" *Proceedings of OWL: Experiences and Directions 2006 (OWLED-06).*

Heflin, J. & Hendler, J. 2000. "Dynamic Ontologies on the Web" *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pp. 443-449.

Heflin, J., Hendler, J. & Luke, S. 1999. "Coping with Changing Ontologies in a Distributed Environment" *Proceedings of the Workshop on Ontology Management of the 16th National Conference on Artificial Intelligence (AAAI-99)*, WS-99-13, AAAI Press, pp. 74-79.

Heflin, J. & Pan, Z. 2004. "A Model Theoretic Semantics for Ontology Versioning" *Proceedings of the 3rd International Semantic Web Conference (ISWC-04)*, LNCS 3298 Springer, pp. 62-76.

Hansson, S.O. 1994. "Kernel Contraction" *Journal of Symbolic Logic*, **59** (3), pp. 845859.

Hu, W. & Qu, Y. 2006. "Block Matching for Ontologies" *Proceedings of the 5th International Semantic Web Conference (ISWC-06)*, pp. 300-313.

Huang, Z. & Stuckenschmidt, H. 2005. "Reasoning with Multi-version Ontologies: A Temporal Logic Approach" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 398-412.

IBM Corporation. "Clio web page" available on the Web (last visited November, 2007): http://www.almaden.ibm.com/cs/projects/criollo/

Kalfoglou, Y. & Schorlemmer, M. 2003. "Ontology Mapping: the State of the Art" *Knowledge Engineering Review*, **18** (1), pp. 1-31.

Kalyanpur, A. 2006. *Debugging and Repair of OWL Ontologies* Doctoral Dissertation, University of Maryland, College Park.

Kalyanpur, A., Parsia, B., Sirin, E. & Cuenca-Grau, B. 2006. "Repairing Unsatisfiable Concepts in OWL Ontologies" *Proceedings of the 3rd European Semantic Web Conference (ESWC-06)*, pp. 170-184.

Kang, S.H. & Lau, S.K. 2004. "Ontology Revision Using the Concept of Belief Revision" *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES-04)*, part III, pp. 8-15.

Katsuno, H. & Mendelzon, A.O. 1990. "On the Difference Between Updating a Knowledge Base and Revising It" Technical Report on Knowledge Representation and Reasoning, University of Toronto, Canada, KRR-TR-90-6.

Keberle, N., Litvinenko, Y., Gordeyev, Y. & Ermolayev, V. 2007. "Ontology Evolution Analysis with OWL-MeT" *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 1-12.

Kim, W. & Chou, H.T. 1988. "Versions of Schema for Object-Oriented Databases" *Proceedings of the 14th International Conference on Very Large Data Bases (VLDB-88)*, pp. 148-159.

Klein, M. & Fensel, D. 2001. "Ontology Versioning on the Semantic Web" *Proceedings of the International Semantic Web Working Symposium (SWWS)*, pp. 75-91.

Klein, M., Fensel, D., Kiryakov, A. & Ognyanov, D. 2002. "Ontology Versioning and Change Detection on the Web" *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02)*.

Klein, M. & Noy, N. 2003. "A Component-Based Framework for Ontology Evolution" *Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems*, CEUR-WS, vol. 71.

Kohler, J., Philippi, S. & Lange, M. 2003. "SEMEDA: Ontology Based Semantic Integration of Biological Databases" *Bioinformatics* **19** (18), pp. 2420-2427.

Kondylakis, H. 2006. *QUETE: Query Processing in Distributed Database Systems*, Master of Science Thesis, Department of Computer Science, University of Crete.

Kondylakis, H., Doerr, M. & Plexousakis, D. 2006. "Mapping Language for Information Integration" Technical Report FORTH-ICS/TR-385.

Konstantinidis, G., Flouris, G., Antoniou, G. & Christophides, V. 2007. "Ontology Evolution: A Framework and its Application to RDF" *Proceedings of the Joint ODBIS & SWDB Workshop on Semantic Web, Ontologies, Databases (SWDB-ODBIS-07)*.

Lam, S.C., Pan, J., Sleeman, D. & Vasconcelos, W. 2006. "A Fine-Grained Approach to Resolving Unsatisfiable Ontologies" *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI-06)*.

Lam, S.C., Sleeman, D. & Vasconselos, W. 2005. "ReTAX++: A Tool for Browsing and Revising Ontologies" *Poster Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, PID-33.

Lambrix, P. & Edberg, A. 2003. "Evaluation of Ontology Merging Tools in Bioinformatics" *Proceedings of the 8th Pacific Symposium on Biocomputing*, pp. 589-600.

Lee, K. & Meyer, T. 2004. "A Classification of Ontology Modification" *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI-04)*, pp. 248-258.

Liu, H., Lutz, C., Milicic, M. & Wolter, F. 2006. "Updating Description Logic ABoxes" *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*.

Luke, S., Spector, L., Rager, D. & Hendler, J. 1997. "Ontology-based Web Agents" *Proceedings of the 1st International Conference on Autonomous Agents*, pp. 59-66.

Madhavan, J., Bernstein, P.A., Doan, A. & Halevy, A.Y. 2005. "Corpus-based Schema Matching" *Proceedings of the 21st International Conference on Data Engineering (ICDE-05)*, pp. 57-68.

Madhavan, J., Bernstein, P.A. & Rahm, E. 2001. "Generic Schema Matching with Cupid" *Proceedings of the 27th International Conference on Very Large Databases (VLDB-01)*, pp. 49-58.

Maedche, A., Motik, B., Stojanovic, L., Studer, R. & Volz, R. 2003. "An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies" *Proceedings of the 12th International World Wide Web Conference (WWW-03)*.

Manakanatas, D. & Plexousakis, D. 2006. "A Tool for Semi-Automated Semantic Schema Mapping: Design and Implementation" *Proceedings of the International Workshop on Data Integration and the Semantic Web (DISWeb-06), Proceedings of Workshops and Doctoral Consortium of the 18th Conference on Advanced Information Systems Engineering (CAISE-06)*, pp. 290-306.

Magiridou, M., Sahtouris, S., Christophides, V. & Koubarakis, M. 2005. "RUL: A Declarative Update Language for RDF" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 506-521.

Maynard, D., Peters, W., dAquin, M. & Sabou, M. 2007. "Change Management for Metadata Evolution" *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 27-40.

McGuiness, D., Fikes, R., Rice, J. & Wilder, S. 2000. "An Environment for Merging and Testing Large Ontologies" *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-00)*, also available as Technical Report KSL-00-16, Knowledge Systems Laboratory, Stanford University.

Melnik, S., Garcia-Molina, H. & Rahm, E. 2002. "Similarity Flooding: A Versatile Graph Matching Algorithm" *Proceedings of the 18th International Conference on Data Engineering (ICDE-02)*, pp. 117-128.

Meyer, T., Lee, K. & Booth, R. 2005. "Knowledge Integration for Description Logics" *Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning* and *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pp. 645-650.

Meyer, T., Lee, K., Booth, R. & Pan, J. 2006. "Finding Maximally Satisfiable Terminologies for the Description Logic ALC" *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pp. 269-274.

Miller, G.A. 1995. "WordNet: A Lexical Database for English" *Communications of the ACM (CACM)*, **38** (11), pp. 39-41.

Miller, E., Swick, R. & Brickley, D. 2006. "Resource Description Framework (RDF) / W3C Semantic Web Activity" available on the Web (last visited November, 2007):
http://www.w3.org/RDF/

Mitra, P., Noy, N. & Jaiswal, A.R. 2005. "OMEN: A Probabilistic Ontology Mapping Tool" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 537-547.

Mocan, A., Cimpian, E. & Kerrigan, M. 2006. "Formal Model for Ontology Mapping Creation" *Proceedings of the 5th International Semantic Web Conference (ISWC-06)*, pp. 459-472.

Noy, N., Chugh, A., Liu, W. & Musen, M. 2006. "A Framework for Ontology Evolution in Collaborative Environments" *Proceedings of the 5th International Semantic Web Conference (ISWC-06)*.

Noy, N., Fergerson, R. & Musen, M. 2000. "The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility" *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management: Methods, Models, and Tools (EKAW-00)*, pp. 17-32.

Noy, N. & Klein, M. 2004. "Ontology Evolution: Not the Same as Schema Evolution" *Knowledge and Information Systems*, **6** (4), pp. 428-440, also available as SMI technical report SMI-2002-0926.

Noy, N., Kunnatur, S., Klein, M. & Musen, M. 2004. "Tracking Changes During Ontology Evolution" *Proceedings of the 3rd International Semantic Web Conference (ISWC-04)*, Lecture Notes in Computer Science (LNCS), Volume 3298, pp. 259-273, Springer-Verlag.

Noy, N. & Musen, M. 1999a. "An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support" *Proceedings of the Workshop on Ontology Management at 16th National Conference on Artificial Intelligence (AAAI-99)*, also available as SMI technical report SMI-1999-0799.

Noy, N. & Musen, M. 1999b. "SMART: Automated Support for Ontology Merging and Alignment" *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management*, also available as SMI technical report SMI-1999-0813.

Noy, N. & Musen, M. 2000. "Algorithm and Tool for Automated Ontology Merging and Alignment" *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, also available as SMI technical report SMI-2000-0831.

Noy, N. & Musen, M. 2002. "PromptDiff: A Fixed-Point Algorithm for Comparing Ontology Versions" *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)* also available as SMI Technical Report SMI-2002-0927.

Noy, N. & Musen, M. 2003. "The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping" *International Journal of Human-Computer Studies* **59**(6), pp. 983-1024.

Noy, N. & Musen, M. 2004. "Ontology Versioning in an Ontology-Management Framework" *IEEE Intelligent Systems*, **19**(4), pp. 6-13, also available as SMI technical report SMI-2003-0961.

Palmisano, I., Iannone, L., Redavid, D. & Semeraro, G. 2006. *Proceedings of the 3rd Workshop on Semantic Web Applications and Perspectives (SWAP-06)*.

Pan, R., Ding, Z., Yu, Y. & Peng, Y. 2005. "A Bayesian Network Approach to Ontology Mapping" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 563-577.

Perez-Rey, D., Maojo, V., Garcia-Remesal, M., Alonso-Calvo, R., Billhardt, H., Martin-Sanchez, F. & Sousa, A. 2006. "ONTOFUSION: Ontology-based Integration of Genomic and Clinical Databases" *Computers in Biology and Medicine*, **36** (7-8), pp. 712-730.

Peters, R.J. & Ozsu, T. 1997. "An Axiomatic Model of Dynamic Schema Evolution in Objectbase Systems" *ACM Transactions on Database Systems*, **22** (1), pp. 75-114.

Pinto, H.S., Gomez-Perez, A. & Martins, J. P. 1999. "Some Issues on Ontology Integration" *Proceedings of the Workshop on Ontologies and Problem-Solving Methods (KRR5) at 16th International Joint Conference on Artificial Intelligence (IJCAI-99).*

Plessers, P. & de Troyer, O. 2005. "Ontology Change Detection Using a Version Log" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 578-592.

Plessers, P. & de Troyer, O. 2006. "Resolving Inconsistencies in Evolving Ontologies" *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, pp. 200-214.

Plessers, P., de Troyer, O. & Casteleyn, S. 2005. "Event-based Modeling of Evolution for Semantic-driven Systems" *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE-05)*, pp. 63 - 76.

Qi, G., Liu, W. & Bell, D.A. 2006a. "A Revision-Based Approach for Handling Inconsistency in Description Logics" *Proceedings of the 11th International Workshop on Non-Monotonic Reasoning (NMR-06).*

Qi, G., Liu, W. & Bell, D.A. 2006b. "Knowledge Base Revision in Description Logics" *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA-06).*

Qi, G. & Pan, J. 2007. "A Stratification-based Approach for Inconsistency Handling in Description Logics" *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 83-96.

Rahm, E. & Bernstein, P.A. 2001. "A Survey of Approaches to Automatic Schema Matching" *The VLDB Journal*, **10**(4) pp. 334-350.

Ribeiro, M.M. & Wassermann, R. 2007. "Base Revision in Description Logics - Preliminary Results" *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 69-82.

Roger, M., Simonet, A. & Simonet, M. 2002. "Toward Updates in Description Logics" *Proceedings of the 2002 International Workshop on Description Logics (DL-02)*, CEUR-WS Vol. 53, also *Proceedings of the 9th International Workshop on Knowledge Representation Meets Databases (KRDB-02).*

Schlobach, S. 2005. "Diagnosing Terminologies" *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-06).*

Schlobach, S. & Cornet, R. 2003. "Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies" *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03).*

Shvaiko, P. & Euzenat, J. 2005. "A Survey of Schema-Based Matching Approaches" *Journal of Data Semantics*, **IV**, pp. 146-171.

Shvaiko, P., Giunchiglia, F., Schorlemmer, M., McNeill, F., Bundy, A., Marchese, M., Yatskevich, M., Zaihrayeu, I., Ho, B., Lopez, V., Sabou, M., Abian, J., Siebes, R. & Kotoulas, S. 2006. "Dynamic Ontology Matching: A Survey" Technical Report University of Trento/DIT-06-046.

Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N. W., Goble, C.A. & Brass, A. 2000. "TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources" *Bioinformatics*, **16** (2), pp. 184-186.

Stoilos, G., Stamou, G. & Kollias, S. 2005. "A String Metric for Ontology Alignment" *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, pp. 624-637.

Stojanovic, L., Maedche, A., Motik, B. & Stojanovic, N. 2002. "User-driven Ontology Evolution Management" *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02)*, Lecture Notes in Computer Science (LNCS), Volume 2473, Springer-Verlag, pp. 285-300.

Stojanovic, L., Maedche, A., Stojanovic, N. & Studer, R. 2003. "Ontology Evolution as Reconfiguration-Design Problem Solving" *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP-03)*, pp. 162-171.

Stojanovic, L. & Motik, B. 2002. "Ontology Evolution Within Ontology Editors" *Proceedings of the OntoWeb-SIG3 Workshop*, pp. 53-62.

Stuckenschmidt, H. & Klein, M. 2003. "Integrity and Change in Modular Ontologies" *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03).*

Stumme, G. & Maedche, A. 2002. "Ontology Merging for Federated Ontologies on the Semantic Web" *Proceedings of the International Workshop on Foundations of Models for Information Integration*, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag.

Sure, Y., Angele, J. & Staab, S. 2003. "OntoEdit: Multifaceted Inferencing for Ontology Engineering" *Journal on Data Semantics*, **1**(1), pp. 128-152.

Svab, O., Svatek, V. & Stuckenschmidt, H. 2007. "A Study in Empirical and 'Casuistic' Analysis of Ontology Mapping Results" *Proceedings of the 4th European Semantic Web Conference (ESWC-07)*, pp. 655-669.

Tansel, A.U., Clifford, J., Gadia, S.K., Jajodia, S., Segev, A. & Snodgrass, R. T. 1993. *Temporal Databases: Theory, Design, and Implementation*, Benjamin/Cummings.

Tzitzikas, Y. & Kotzinos, D. 2007. "(Semantic Web) Evolution through Change Logs: Problems and Solutions" *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA-07).*

Völkel, M., Winkler, W., Sure, Y., Kruk, S.R. & Synak, M. 2005. "SemVersion: A Versioning System for RDF and Ontologies" *Proceedings of the 2nd European Semantic Web Conference (ESWC-05).*

Wang, H., Horridge, M., Rector, A., Drummond, N. & Seidenberg, J. 2005. "Debugging OWL-DL Ontologies: A Heuristic Approach" *Proceedings of the 4th International Semantic Web Conference (ISWC-05).*

Wassermann, R. 1998. "Revising Concepts" *Proceedings of the 5th Workshop on Logic, Language, Information and Communication (WoLLIC-98).*

Wassermann, R. & Fermé, E. 1999. "A Note on Prototype Revision" *Spinning Ideas* (Electronic Essays dedicated to Peter Gärdenfors on his 50th Birthday), also available as Research Report PP-1999-18 (ILLC Series).

Yatskevich, M. 2003. "Preliminary Evaluation of Schema Matching Systems" Technical Report DIT-03-028.

Zeginis, D., Tzitzikas, Y. & Christophides, V. 2007. "On the Foundations of Computing Deltas Between RDF Models" *Proceedings of the 6th International Semantic Web Conference (ISWC-07)*.

Zhang, Z., Zhang, L., Lin, C. X., Zhao, Y. & Yu, Y. 2003. "Data Migration for Ontology Evolution" *Poster Proceedings of the 2nd International Semantic Web Conference (ISWC-03)*.

Zhdanova, V.A. & Shvaiko, P. 2006. "Community-Driven Ontology Matching" *Proceedings of the 3rd European Semantic Web Conference (ESWC-06)*, pp. 34-49.