The Knowledge Engineering Review, Vol. 30:1, 45–75. © Cambridge University Press, 2013
doi:10.1017/S0269888913000349
First published online 28 August 2013

Ontology evolution: a process-centric survey

FOUAD ZABLITH[1], GRIGORIS ANTONIOU[2], MATHIEU d'AQUIN[3], GIORGOS FLOURIS[2], HARIDIMOS KONDYLAKIS[2], ENRICO MOTTA[3], DIMITRIS PLEXOUSAKIS[2] and MARTA SABOU[4]

[1]*Olayan School of Business, American University of Beirut, PO Box 11-0236, Riad El Solh, 1107 2020, Beirut, Lebanon; e-mail: fouad.zablith@aub.edu.lb;*
[2]*Institute of Computer Science, FORTH, PO Box 1385, GR 71110, Heraklion, Greece; e-mail: antoniou@ics.forth.gr, fgeo@ics.forth.gr, kondylak@ics.forth.gr, dp@ics.forth.gr;*
[3]*Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom; e-mail: m.daquin@open.ac.uk, e.motta@open.ac.uk;*
[4]*Department of New Media Technology, MODUL University Vienna, Am Kahlenberg 1, 1190 Vienna, Austria; e-mail: marta.sabou@modul.ac.at*

Abstract

Ontology evolution aims at maintaining an ontology up to date with respect to changes in the domain that it models or novel requirements of information systems that it enables. The recent industrial adoption of Semantic Web techniques, which rely on ontologies, has led to the increased importance of the ontology evolution research. Typical approaches to ontology evolution are designed as multiple-stage processes combining techniques from a variety of fields (e.g., natural language processing and reasoning). However, the few existing surveys on this topic lack an in-depth analysis of the various stages of the ontology evolution process. This survey extends the literature by adopting a process-centric view of ontology evolution. Accordingly, we first provide an overall process model synthesized from an overview of the existing models in the literature. Then we survey the major approaches to each of the steps in this process and conclude on future challenges for techniques aiming to solve that particular stage.

1 Introduction

Ontologies are formal artifacts that are designed to represent the knowledge related to a specific or generic domain in terms of the relevant concepts, relationships between these concepts and the instances of these concepts. The ontology engineering research community has been focusing for many years on supporting the development of ontologies, through tools, techniques and methods for knowledge acquisition, knowledge elicitation, knowledge representation, ontology validation, ontology-based reasoning and others (Studer *et al.*, 1998; Gomez-Perez *et al.*, 2003). Partly, as a result of this extensive research, ontologies have recently gained more attention as a formal basis for the Semantic Web (Antoniou & Harmelen, 2004), including the development of standard ontology representation languages (notably, OWL[1]) and the availability of sufficiently mature tools to manipulate these ontologies, such as parser libraries (Bechhofer *et al.*, 2003), reasoners (Sirin *et al.*, 2007) and ontology editors (e.g., Protégé[2], the NeOn Toolkit[3]). There are now

[1] http://www.w3.org/TR/owl-ref/
[2] http://protege.stanford.edu/
[3] http://www.neon-toolkit.org/

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

thousands of ontologies available and directly exploitable on the Web (d'Aquin *et al.*, 2007), as witnessed also by the W3C Linked Open Data Initiative[4].

This new Semantic Web environment, where ontologies are distributed and used in more and more mainstream applications, implies a new focus for research in ontology engineering: supporting the complete lifecycle of ontologies, beyond the initial steps of acquisition, development and deployment. Ontologies should be maintained and evolved according to changes in the domains they represent or the requirements from the applications they support. These changes should be integrated into the ontology in a way that allows maintaining its structure, consistency and relevance, and should be managed to ensure continuity and traceability across different versions of the ontology. In other words, ontology evolution involves a number of steps, for which a variety of approaches, techniques and tools might be required.

While key stakeholders in the ontology evolution field have proposed various process models capturing the key steps of an evolution task, currently, there is no clear understanding of how these models relate to each other. Even more, since methods and tools for these key steps are often drawn from a variety of disjoint research fields (e.g., natural language processing, inconsistency reasoning), no single survey provides an in-depth overview of the state of the art relevant for individual steps.

An early survey on ontology evolution was published in 2004 and consisted of a project-centric overview of the field (Haase & Sure, 2004). This project deliverable focused primarily on tools that offer ontology evolution support. It also discusses the various stages of the evolution process as proposed by Stojanovic (2004), however, it does not provide a comprehensive overview of works for each stage. Four years later, Flouris *et al.* (2008) published a comprehensive survey focusing on the broad topic of *ontology change*. The primary goal of this survey is to identify research areas dealing with ontology change aspects, to define their boundaries and to provide terminological clarifications. As such, ontology evolution is considered as one of the 11 ontology change tasks. In addition, although Leenheer and Mens (2008) focus on describing approaches to collaborative ontology evolution, they also provide a brief overview of the main steps involved in single-user ontology evolution. They follow a process model inspired by software engineering and aim to define each of these steps rather than to perform a thorough overview of existing approaches in each area. In addition to software engineering, the authors explore approaches in other areas such as the argumentation field, and analyze how ontology evolution can benefit from these approaches. Their objective is to support inter-organizational ontologies, where the level of complexity of the domain and dynamics are more substantial than single-user ontologies. Finally, a survey related to database schema, XML-based and ontology evolution was released (Hartung *et al.*, 2011). In this work the authors identify a set of requirements for a successful evolution (e.g., backward compatibility, versioning and mapping support, etc.), and build a survey around these resolved themes. While this is a well-covered survey, its limitation with respect to our objective is in the fact that it revolves around three different fields (i.e., database, XML schemas and ontologies), to which not all functionalities and tasks can be interchangeably applied.

The goal of this article is to fill in the current gap in the literature by providing a complete and detailed overview of the current research activities in ontology evolution. Our contributions are twofold. First, we present and discuss the various process models that were proposed for the ontology evolution tasks and derive an overarching ontology evolution process that captures the consensus of individual models. Second, based on the tasks of this newly derived process model we perform an in-depth overview of approaches that support each task, thus providing a unique overview over several research fields.

The article is structured as follows. We define ontology evolution, present existing process models and derive the overarching ontology evolution cycle in Section 2. In subsequent sections, we describe in detail the existing and ongoing work to tackle each individual task, namely,

[4] http://linkeddata.org/

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

detecting the need for evolution (Section 3), suggesting ontology changes (Section 4), validating ontology changes (Section 5), assessing the impact of evolution (Section 6) and managing ontology changes (Section 7). In each section, we describe the general, common approaches reported in the literature and detail the specific realizations of each approach in different works. In the final section of the article, we comment on a general view on the current state of the ontology evolution area, on the need for more mature and integrated implementations of techniques and tools for ontology evolution, and on the impact such tools could have on the realization of the Semantic Web.

2 Ontology evolution: definition and process model

2.1 Definition

Ontology evolution has been defined in various ways. Haase and Stojanovic (2005) see ontology evolution as the process to 'adapt and change the ontology in a timely and consistent manner'. Flouris *et al.* (2008) define ontology evolution as a process aiming to 'respond to a change in the domain or its conceptualization' by implementing a set of change operators over a source ontology. The recently compiled NeOn Glossary of ontology engineering tasks states that ontology evolution is 'the activity of facilitating the modification of an ontology by preserving its consistency'[5].

A common characteristic of the above definitions is that they have a strict view on ontology evolution focusing only on updating the ontology based on the required changes and therefore they see ontology versioning, the process of managing different ontology versions, as a separate activity. We argue that ontology versioning is intrinsically linked to the ontology evolution task and therefore should always be considered when discussing ontology evolution. As a result, in this paper, we adopt a broader view of ontology evolution encompassing both the changes made to an ontology as well as its versioning.

2.2 Ontology evolution stages

Ontology evolution is not an atomic, well-defined and self-contained notion. Supporting the evolution of an ontology implies the completion of a number of different tasks, for which different approaches can be envisaged. For this reason, several attempts at structuring and conceptualizing the general process of ontology evolution can be found in the literature. While these attempts are often motivated by the need to build a software framework for supporting ontology evolution, they are also useful for practitioners who need to have a complete picture of the tasks involved in ontology evolution. Despite these compelling reasons for a unified process model, the community has often created classifications that separate and isolate ontology evolution tasks (Flouris *et al.*, 2008). For example, the need for changing the ontology can be identified both by analyzing user activity (the main focus of Klein & Noy, 2003; Stojanovic, 2004; Vrandecic *et al.*, 2005; Noy *et al.*, 2006), or external domain data (approach taken by Zablith, 2009), and ontology learning tools (Cimiano & Volker, 2005; Maynard *et al.*, 2009). Similar to the management side of the evolution, the community has created separate threads under the umbrella of ontology versioning and consistency checking, without connecting them back to the ontology evolution process. We therefore identify the need for a framework, which we refer to as *ontology evolution cycle*, connecting current views of the ontology evolution process models. We continue with a discussion of current ontology evolution process models and conclude with deriving a unified ontology evolution cycle.

In her thesis, Stojanovic (2004) proposed a framework for ontology evolution. The framework is a six-phase cyclic process, starting with the *change capturing* phase where changes to be applied to the ontology are identified. Three types of changes are identified based on *usage-driven* change

[5] http://mayor2.dia.fi.upm.es/oeg-upm/files/pdf/NeOnGlossary.pdf

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

discovery (i.e., derived from user behavior), *data-driven* discovery (i.e., derived from changes to the ontology instances) and *structure-driven* change discovery where changes are derived from the analysis on the structure of the ontology. Hence, this evolution framework treats the ontology as a closed entity by initiating the evolution from the analysis performed on the ontology itself, without opening it to external domain data such as relevant text corpora. Change capturing is followed by the *representation phase* where the changes are represented following a specific model that the author calls the 'evolution ontology'. The third phase is the *semantics of change* phase, during which syntactic and semantic inconsistencies that could arise as a result of the changes are addressed (Tamma & Bench-Capon, 2001). A syntactic inconsistency covers cases, such as violating constraints or using entities and concepts that have not been defined in the ontology. A semantic inconsistency occurs when an entity's meaning changes during the evolution process (Tamma & Bench-Capon, 2001). The fourth phase is the *implementation of change* phase coupled with user interaction for approving or cancelling changes. *Change propagation* is the fifth phase, allowing the update of outdated instances as well as recursively reflecting changes in referenced ontologies in the case of interconnected ontologies. The final phase is the *validation phase*, which checks that the performed changes led to a valid (or desirable) result, and allows the user to undo such changes if this is not the case.

Klein and Noy (2003) present a framework to support users when an ontology evolves from one version to another. Their framework is component based, and targets the following ontology evolution tasks: *data transformation*, where data in the old ontology version are transformed into a format compatible with the new ontology version; *ontology update*, where changes are propagated to the ontology under evolution; *consistent reasoning* to keep the ontology under evolution consistent; and, finally, *verification and approval*, where ontology developers perform final checks. The focus in this approach is mainly on the versioning side of the ontology, as an effect of the evolution. Hence, unlike the previous framework, this work does not deal with a change identification step, but mainly on making sure that the ontology consistently evolves from one version to another.

Noy *et al*. (2006) describe a framework for ontology evolution in collaborative environments. This framework is scenario based and consists of various Protégé plugins. It includes the following tasks: *examining changes* between ontology versions, presented, for example, in the form of a table; *accepting and rejecting changes*, helpful in curated ontology evolution, where changes are approved or rejected with the change action recorded; and *providing auditing information*, where authors' information (e.g., time of change, number of concepts changed) are compiled. Changes are recorded following the Change and Annotation Ontology (discussed by Klein & Noy, 2003 as well). The framework serves as a means to manage collaborative changes to be performed on an ontology, where the changes are proposed by the ontology curators.

Evolva is an ontology evolution tool built on a component-based framework, which aims to evolve ontologies from existing domain data that are external (Zablith, 2009), unlike focusing purely on changes derived from within the ontology as targeted by Stojanovic (2004). Such data can be found in text documents, folksonomies, RSS feeds or a list of terms. Each source requires a different method of content extraction handled by the *information discovery* component. The *data validation* component identifies new terms that are relevant to the ontology. It also checks the quality of content and filters out noise generated from the information discovery component. The validated information is passed to the *ontology changes* component in which lexical databases and online ontologies provide background knowledge for automating and evaluating the integration of new information into the ontology through its relation discovery and validation processes. As the evolution could generate conflicts and problems, such issues are handled at the level of the *evolution validation* component, by reusing existing solutions for consistency and duplication checks. Finally, the validated ontology is passed to the *evolution management* component where the user has control over the evolution, and changes are recorded and propagated to dependent ontologies. Part of the framework is implemented as a plugin for the NeOn Toolkit, and reuses some of the functionalities provided by the existing Toolkit plugins. The Evolva framework differs

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

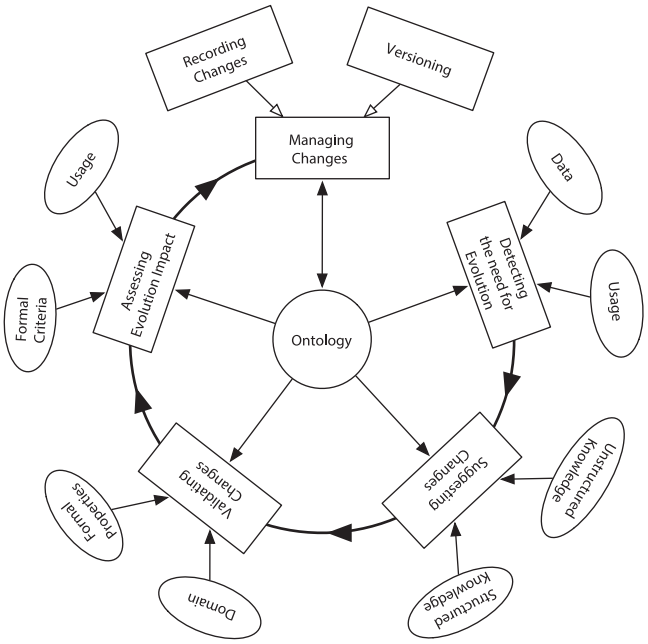


Figure 1 Ontology evolution cycle

from the previously described frameworks by opening the evolution of ontologies to external domain data, which serve as a starting point to initiate the evolution.

Similar to the work of Noy *et al.* (2006), DILIGENT deals with the collaborative aspect in evolving ontologies (Vrandecic *et al.*, 2005). It is a decentralized user-centric methodology proposing an ontology engineering process targeting 'user-driven' ontology evolution, rather than its initial design. At a glance, the process starts by having a *core ontology collaboratively built* by users. After the building step, the ontology will be locally adapted without changing the core ontology. A board of users will then *analyze the local changes*, in order to come up with the changes that need to be incorporated in the shared ontology. The requests of changes are supported by arguments using an argumentation framework in order to come up with a balanced decision reflecting all the evolution requests. The *changes are revised* by the board of knowledge experts in order to maintain compatibility between different versions. The evolution of the ontology is a result of the experts' decision. Finally, the shared evolved ontology is locally adapted at the different involved locations.

Commonalities can be easily detected in the descriptions of the different frameworks above. Figure 1 depicts the general ontology evolution cycle we rely on in this article and which intends to abstract from the specificity of each framework. This cycle is made of five main steps or tasks (represented in rectangles, where 'Recording Changes' and 'Versioning' are sub-tasks of the 'Managing Changes' step, and the subsumption is depicted by the empty arrow heads), with each step potentially relying on different inputs and background information (represented in the ovals). The ontology in the center serves both as input to all the tasks, and receives input from the 'Managing Changes' tasks, hence the double-sided arrow. The cycle is repeated after applying the changes and passing again through the ontology. We hereby define these steps as well as their relation to other frameworks.

Detecting the Need for Evolution initiates the ontology evolution process by detecting a need for change. Such a need can be derived from user behavior (i.e., the usage of a system that relies on the ontology) or data sources both internal or external to the ontology. This stage corresponds to the *change capturing* step in Stojanovic's process model, to the *information discovery* task of Evolva and to the *local changes* step of DILIGENT, but it is not present in the other approaches that focus on the versioning aspect of evolution, namely Klein and Noy (2003) and Noy *et al.* (2006).

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 1 Relations between tasks of the ontology evolution cycle and components of existing ontology evolution frameworks

Referenced work	Detecting Need for Evolution	Suggesting Changes	Validating Changes	Assessing Impact	Managing Changes
KAON (Stojanovic, 2004)	Change capturing	Representation	Semantics of change/validation	Propagation	Implementation of changes
Klein and Noy (Klein & Noy, 2003)		Data transformation	Consistency verification/approval		Update
Protégé (Noy *et al.*, 2006)			Examining changes	Auditing	Accept/reject recording
Evolva (Zablith, 2009)	Information discovery	Relation discovery	Validation		Management
DILIGENT (Vrandecic *et al.*, 2005)	Local changes		Revision		Local adaptation

Suggesting Changes represents and suggests changes to be applied to the ontology. Some approaches handle this task by applying patterns to text corpora representing the domain of the ontology (unstructured sources), while others rely on structured data sources, such as online ontologies, to suggest the appropriate changes. This stage corresponds to the *representation* phase of Stojanovic, to the *data transformation* step of Klein and Noy and the *relation discovery* task of Evolva.

Validating Changes filters out those changes that should not be added to the ontology as they could lead to an incoherent or inconsistent ontology, or an ontology that does not satisfy domain or application-specific constraints. A similar stage is present in all the frameworks that we have previously described. Current approaches handle both a formal validation of changes making sure the ontology is logically consistent as per specified constraints, and a domain validation of changes focusing on the domain relevance of the changes.

Assessing Impact measures the impact on external artifacts that are dependent on the ontology (i.e., other ontologies, application) or criteria such as costs and benefits of the proposed changes. Currently, impact is based on the cost involved in adding a suggested change to the ontology, or the effect of such a change at the application level, for example, to the ability to answer specific queries. Only two frameworks have similar steps, namely Stojanovic's *propagation* stage and Noy *et al.*'s *auditing* step.

Managing Changes applies and records changes and keeps track of the various versions of the ontology. This is a continuous task, active through the entire ontology cycle, and is divided into two sub-tasks: recording the changes realized on the ontology, and keeping track of the different versions of the ontology. Indeed, all existing frameworks acknowledge the need for such a task.

Table 1 gives an overview of the way the steps in the evolution cycle described here relate to the components of the frameworks detailed above. While commonalities exist with other frameworks, one particular advantage of our proposed framework is that it offers a more granular process to evolve ontologies, a bridge among functionalities that are often considered separately within the community. For example, bringing new entities from external data sources has been confined to the area of ontology learning, where the management side of changes is not of core interest. In the next sections, we describe each of the five tasks of our cycle in more detail and survey current approaches, techniques and tools that support them.

3 Detecting the need for evolution

An ontology is a 'specification of a shared conceptualization of a domain' (Gruber, 1993) and therefore needs to change (i.e., to evolve) whenever changes occur in the underlying domain or in its conceptualization. For example, the UN's Food and Agriculture Organization maintains a

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

large agricultural thesauri, AGROVOC, used for indexing internal data with terms from domains as varied as biology, geography or chemistry. Changes to these domains (e.g., discovery of new plant species, new political borders, meteorological phenomena) are continuously added to AGROVOC by experts working in the respective fields. Another heavily explored domain evolution is applied in life sciences, where existing work explore the evolution of the ontology and their corresponding domain mappings (Hartung *et al.*, 2012).

In recent years, ontologies have become key components of information systems where they are often used to index large document corpora or collections of facts and directly support user interaction with the system through functionalities such as browsing and querying. These ontologies must evolve to reflect the content of the indexed document set and therefore they often change when new documents are added or old ones are removed. They also need to evolve to match the activity of the system's users. For example, the AKTRO ontology is used to structure the knowledge bases of several applications that can be searched by users. Alani *et al.* (2006) log the queries to these applications and use them to determine which ontology concepts and relations are never accessed and therefore are good candidates for elimination from the ontology.

The goal of the 'Detecting the Need for Evolution' stage is to detect whether new concepts and relations should be added to the ontology, or whether some ontology elements can be deleted. Besides domain experts pro-actively identifying the need for an ontology to change, programmatic methods used to identify potential changes make use either of relevant data collections (Section 3.1) or application usage patterns (Section 3.2). An evolution activity initiated by changes detected in the related domain or application usage patterns is also referred to as *bottom-up* ontology evolution (Stojanovic *et al.*, 2002), as opposed to a *top-down* approach where changes would be dictated by managers or experts. Another work investigates a pattern-based ontology changes detection by relying on graph analysis (Javed *et al.*, 2011).

Example: Consider an ontology about the academic domain used to index a research lab's documents, and the available job vacancies in particular. The addition of a new document mentioning the availability of a job vacancy for a 'research assistant', a concept not available in the corresponding domain ontology, should lead to the need for extending this ontology with the appropriate concept to represent a research assistant.

3.1 Detecting the need for evolution from data

The need for evolution can be initiated from the analysis of various types of data. While some approaches limit the data analysis to information available within the ontology, for example, the work of Stojanovic (2004), other tools identify ontology changes by analyzing external data sources, including unstructured sources, for example, text documents (Velardi *et al.*, 2001; Cimiano & Volker, 2005; Bloehdorn *et al.*, 2006; Novacek *et al.*, 2007; Ottens *et al.*, 2007; Maynard *et al.*, 2009) and metadata (Maynard *et al.*, 2007), or structured data, such as databases (Haase & Sure, 2004).

Stojanovic (2004) defines *data-driven* ontology evolution as the process of discovering ontology changes based on the analysis of the ontology instances, for example, by relying on data mining techniques. Another type of change detection defined by Stojanovic is *structure driven*, where the evolution is initiated based on the analysis performed on the ontology structure using a set of heuristics. For example, 'if all subconcepts have the same property, the property may be moved to the parent concept', or 'a concept with a single subconcept should be merged with its subconcept' (Stojanovic, 2004).

Another type of data source for detecting the need for evolution are domain data, external to the ontology under evolution. Such more 'traditional' forms of storing information about the domain often contain valuable knowledge that should be encapsulated in the ontology itself. Bloehdorn *et al.* (2006) based their work on the six-phase ontology evolution process proposed by Stojanovic (2004). They identified that valuable information reside in databases and documents, but require better structuring and easy accessibility through the use of ontologies. Unike Stojanovic (2004),

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

who considers data-driven ontology evolution as the evolution triggered from the ontology instances, they consider *data-driven changes* as changes happening in external data sources, such as the addition and deletion of documents in a corpus, or changes occurring in databases (Haase & Sure, 2004; Bloehdorn *et al.*, 2006). Other tools that initiate ontology changes from text documents include the ontology learning tools Text2Onto (Cimiano & Volker, 2005) and Semantic Pattern Recognition and Annotation Tool (SPRAT) (Maynard *et al.*, 2009). Moreover, Evolva detects the need for evolution by identifying terms from various types of data sources including RSS feeds, text corpus or a list of raw terms (Zablith, 2009). In addition, Dino is a framework for integrating ontologies which are learned from text (Novacek *et al.*, 2007; Laera *et al.*, 2008), and Dynamo is a multi-agent system-based approach that falls in this category of tools as well (Ottens *et al.*, 2009). We discuss in more details the processes involved within these approaches in the next sections.

3.2 Detecting the need for evolution from usage

In addition to using data analysis as a starting point for detecting the need for evolution, some approaches rely on the study of usage patterns to which the ontology is subject to. For example, Alani *et al.* (2006) propose that, based on what parts of the ontology are mostly used by applications, the ontology can shrink to better fit its purpose in the environment. In addition to application usage, user behavior is studied to detect the need for evolution, which is called *usage-driven* ontology evolution (Stojanovic, 2004). Bloehdorn *et al.* (2006) rely on a usage-log, which is a record kept of the interaction between the user and the ontology (e.g., user behavior and contextual search history), to analyze and detect the need for evolution. Such a log file can store information about what has been queried, which elements in the ontology have been viewed by the user, etc., and used to derive usage preferences and suggest changes to the ontology. This source of change is useful to keep the ontology adapted to the user needs, while removing the parts that become unused in the environment. This would indirectly help in the maintenance cost, and increase the efficiency of processing the ontology to perform the required tasks based on the assumption that smaller ontologies are easier to manage and explore. Another work identifies the need for an agile approach to maintain ontologies and adapt them based on the application and user requirements (Luczak-Rosch, 2009). Luczak-Rosch (2009) proposes a methodology and framework for ontology maintenance, which takes into account two types of feedback: dynamic application feedback and user feedback. This will help adapt the ontology to the information relevant to the scenario in which it is used. Javed *et al.* (2011) employ pattern detection applied on the ontology's change log analysis. This will help derive ontology changes from analyzing the historical ontology changes, and identifying frequent change sequences. Another approach proposes the use of adaptive ontologies that evolve depending on a user context and evolution of the domains on the Web (Pruski *et al.*, 2011). The objective is to enhance information retrieval from the Web relevant to the user needs.

Table 2 provides a summary of the various methods involved in detecting the need for evolution from data and usage. We can conclude that the majority of evolution detection methods make use of data sources (as opposed to usage information) and that, within data-centric methods, those that exploit text corpora are the most widespread. One possible explanation of this phenomenon lies in the availability of several off-the-shelf corpus analysis methods from the areas of Natural Language Processing (NLP) in general, and ontology learning in particular.

4 Suggesting ontology changes

Once it has been detected that the ontology needs to change (e.g., adding the *ResearchAssistant* concept to the ontology) it is important to understand what are the concrete ontology change operations needed to evolve the ontology (e.g., add *ResearchAssistant* as a subconcept of the *AcademicStaff* concept that already exists in the base ontology). We call this task 'Suggesting Ontology Change'. Various approaches that address this stage derive changes by limiting

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 2 Approaches used for detecting the need for evolution

Referenced work(s)	Data	Usage
Stojanovic (2004)	Ontology instances and structure	User behavior
Evolva (Zablith, 2009)	RSS feeds, terms list	
Evolva (Zablith, 2009); Text2Onto (Cimiano & Volker, 2005); SPRAT (Maynard *et al.*, 2009); Dino (Novacek *et al.*, 2007); Dynamo (Ottens *et al.*, 2007); Velardi (Velardi *et al.*, 2001; Bloehdorn *et al.*, 2006)	Text corpus	
Maynard *et al.* (2007)	Metadata	
Bloehdorn *et al.* (2006); Haase (Haase & Sure, 2004)	Databases	
Alani *et al.* (2006)		Application usage
Javed *et al.* (2011)		Ontology change log
Luczak-Rosch (2009)		Application and user feedback
Bloehdorn *et al.* (2006)		Usage-log analysis (e.g., log of queries and search history)
Pruski *et al.* (2011)		User context and domain information

their focus on the content available in unstructured documents (e.g., text documents). Other works broaden their scope to rely on external structured knowledge sources (e.g., lexical databases or online ontologies) to support ontology change suggestions. In this section, we review both sets of approaches.

4.1 Suggesting changes by relying on unstructured knowledge

Text2Onto derives ontology changes through processing text documents and extracting ontological entities (Cimiano & Volker, 2005). It is designed to overcome the limitations of other ontology learning tools which (i) are domain dependent, (ii) lack user interaction during the ontology learning process and (iii) execute the ontology learning process from scratch whenever a change occurs in the text corpus. Text2Onto uses a Probabilistic Ontology Model, coupled with data-driven change discovery that enables specific changes detection from new text documents, without having to process all the corpus when new documents are added. In addition to the extraction of concepts and instances (i.e., corresponding to the 'Detecting the Need for Evolution' task), Text2Onto includes lexico-syntactic pattern-based algorithms to extract various types of relations, including 'Instance-of', 'Subclass-of', 'Part-of' and other general relations. When these relations are discovered between a concept that already exists in the ontology and a newly derived concept, they represent concrete ontology change suggestions.

Similar to Text2Onto, SPRAT suggests ontology changes from text documents (Maynard *et al.*, 2009). It combines existing ontology-based information extraction (OBIE) techniques, named entity recognition and relation extraction from text. It provides additional patterns to refine the process of entity identification and relations between them, and to transform them into ontological entities. SPRAT relies on lexico-syntactic patterns applied to text documents to identify terms and their corresponding relations.

Furthermore, Bloehdorn *et al.* (2006) propose an architecture applied in a digital library domain or other electronic repositories. The authors specify that ontology learning algorithms, such as the ones provided by Text2Onto, can be used to extract document contents, which can be used to evolve the ontology based on the information in the corpus.

Another tool developed to detect changes from text documents and merging ontologies is Dino (Novacek *et al.*, 2007; Laera *et al.*, 2008). Dino is a framework for integrating ontologies, and

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

includes a semi-automatic integration of learned ontologies with a master ontology built by ontology designers. It relies on the use of ontology alignment, coupled with agent-negotiation techniques, to generate and select mappings between learned ontologies from text and the base ontology. In more detail, Text2Onto is used to extract information from documents in the Dino framework. The learning algorithms of Text2Onto are customized through a user interface, and the confidence values of extracted terms are fed to an ontology alignment/negotiator wrapper (Novacek *et al.*, 2007). The learned ontology representing new concepts, and the master ontology collaboratively developed by the knowledge experts are aligned, through a set of mappings between the classes, entities and relations of the two ontologies created using the alignment wrapper. Agreement of the semantics used is reached through negotiation using the negotiation wrapper. An axiom ontology is created containing the merged statements between the learned and the master ontology.

Dynamo also falls in the category of exploiting external data sources for building ontologies (Ottens *et al.*, 2009). It consists of a multi-agent system for dynamic ontology construction from domain-specific sets of text documents. Dynamo relies on an adaptive multi-agent system architecture, within a framework where the ontology designer interacts with the system during the process of building the ontology. The system considers the extracted entities from text sources as separate agents, which are related to other entities (agents) through a certain relationship. In other words, an ontology is treated as a multi-agent system.

4.2 Suggesting changes by relying on structured knowledge

Structured knowledge represents and defines conceptual information entities, connected through explicit relations. Such representation allows reusing knowledge entities with less effort compared to the unstructured knowledge sources discussed previously. In this section we focus on approaches that suggest ontology changes during ontology evolution by making use of two types of structured sources: lexical databases and online ontologies.

4.2.1 Lexical databases

WordNet (Fellbaum, 1998) is one of the major lexical databases providing a wealth of entities interconnected with taxonomical links represented in the form of hyponyms and hypernyms, in addition to other types of relations including meronymy and holonymy links. WordNet is used to support various tasks including word sense disambiguation (Li *et al.*, 1995; Ide & Veronis, 1998; Banerjee & Pedersen, 2002), information retrieval (Li *et al.*, 1995) and question answering (Pasca & Harabagiu, 2001; Clark *et al.*, 2008).

Maedche *et al.* (2002) propose the use of WordNet to improve semantic bridging and similarity computation during ontology evolution. Ontology learning tools such as SPRAT (Maynard *et al.*, 2009) and Text2Onto (Cimiano & Volker, 2005) use pattern-based relation extraction techniques over unstructured data sources to suggest changes. These tools make use of WordNet to improve their pattern detection algorithms, for example, by extracting pattern examples from WordNet's relations.

As part of ontology evolution, WordNet is used to discover taxonomical relations between newly discovered concepts and existing concepts in the ontology (Zablith *et al.*, 2008). The authors devise a technique based on the WordNet Java library[6], which identifies the appropriate relation between two terms, along with the relation path. Since WordNet can be stored locally, extracting information from it is faster than using remote structured sources, such as online ontologies. However, Zablith *et al.* (2008) found that WordNet lacks the richness of named relations and the steadily increasing diversity provided by the online ontologies. As a result, they later used online ontologies as an alternative to WordNet.

[6] `http://sourceforge.net/projects/jwordnet/`

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 3 Approaches to suggesting ontology changes from external data sources

Referenced work(s)	Unstructured sources	Structured sources
Dino (Novacek *et al*., 2007)	Text corpus	
Dynamo (Ottens *et al*., 2009)	Text corpus	
Evolva (Zablith *et al*., 2008)		Online ontologies/lexical databases
Text2Onto (Cimiano & Volker, 2005)	Text corpus	Lexical databases
SPRAT (Maynard *et al*., 2009)	Text corpus	Lexical databases

4.2.2 Online ontologies

Online ontologies form a ready-to-reuse body of knowledge and have been used to perform a variety of tasks, including ontology matching (Sabou *et al*., 2008) and development (Alani, 2006), question answering (Lopez *et al*., 2009), folksonomy enrichment (Angeletou *et al*., 2008) and word sense disambiguation (Gracia *et al*., 2009). Besides their advantages discussed before, their uptake is also due to their increased availability and the presence of tools, such as Watson (d'Aquin *et al*., 2007), Swoogle (Ding *et al*., 2005) and Sindice (Oren *et al*., 2008), for discovering and consuming them. The use of online ontologies has been pioneered in the area of ontology building (Alani, 2006) and lead to the identification of a few challenges when using this paradigm: (1) Semantic Web tools are not mature enough yet—although the situation has changed dramatically since 2006; (2) not all ontologies created by individuals are made available online; (3) large ontologies can provide big segments, resulting with a big messy ontology that is hard to clean; (4) the quality of the online ontologies affects the overall quality of the resulting ontology; also, a segment of a good ontology does not necessarily preserve the quality of the source ontology.

Zablith *et al*. (2008) use online ontologies as background knowledge for integrating newly discovered concepts in the ontology under evolution. In their Evolva evolution framework, new concepts are discovered from external data sources, including concepts from text corpora or RSS feeds. These concepts trigger the need for evolution, and are integrated by relying on background knowledge provided mainly by online ontologies (Zablith *et al*., 2008). Online ontologies enable a mechanism for checking how new concepts connect with existing knowledge in the ontology. Unlike reusing ontology segments as described above by Alani (2006), this work limits the reuse of ontologies to the level of statements, that is, ontologies are not processed as one block of statements. Statements are easier to evaluate by users, and offer a more granular control over what to add or ignore during ontology evolution. The process of identifying the possible relations between concepts relies on the Scarlet relation discovery engine[7]. Scarlet (Sabou *et al*., 2008) uses the Watson Semantic Web gateway (d'Aquin *et al*., 2007) to automatically select and explore online ontologies to discover relations between two given concepts. For example, when relating two concepts labeled *ResearchAssistant* and *AcademicStaff*, Scarlet (1) identifies (at run-time) online ontologies that can provide information about how these two concepts inter-relate and then (2) combines this information to infer their relation. Besides subsumption relations, Scarlet is also able to identify disjoint relations (e.g., *ResearchAssistant* is disjoint from *Professor*) and named relations. By reusing online ontologies, new changes proposed to the ontology are ready to be applied without further transformation, as they are already represented in an ontology compatible format.

Table 3 summarizes the approaches discussed in this section and the types of sources they use to suggest ontology changes. Unstructured, textual sources are the most frequent sources for suggesting ontology changes. Structured sources, such as the WordNet lexical database are used to improve the pattern-based relation extraction mechanisms that typically work over textual data (SPRAT, Text2Onto) as well as to derive potential changes from its structure. Finally, the use of online ontologies is a recent, promising trend to ontology change suggestion.

[7] `http://scarlet.open.ac.uk/`

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 4 Domain-based validation approaches

Referenced work(s)	Purpose	Level of change	Technique used	Domain resources
Text2Onto (Cimiano & Volker, 2005); SPRAT (Maynard *et al.*, 2009)	Relevance	Concepts/terms	Statistical measures (TF-IDF)	Text corpus
Dino (Novacek *et al.*, 2007)	Relevance	Statement	String similarity	User-maintained list of words
Evolva (Zablith *et al.*, 2010)	Relevance	Statement	Pattern based	Ontology to evolve/ online ontologies
d'Aquin (d'Aquin, 2009)	Correctness	Statement	Formal measures	Online ontologies
Sabou (Sabou *et al.*, 2009)	Correctness	Statement	Length/relatedness/ popularity measures	Online ontologies

5 Validating ontology changes

Not all the changes resulting from the 'Suggesting Ontology Changes' phase should be incorporated into the evolving ontology. Indeed, some of these changes could lead to an incoherent or inconsistent ontology, or an ontology that does not satisfy domain-specific or application-specific constraints. The role of the 'Validating Ontology Changes' stage is to filter out those changes that should not be added to the ontology. Typically, changes are validated at two different levels. *Domain-based validation* (Section 5.1) relies on domain data to evaluate whether the proposed change aligns with the content of the ontology, that is, to check whether it is within the domain of the ontology. *Formal properties-based validation* (Section 5.2) uses formal techniques to ensure that the proposed change does not invalidate the specified constraints, such as consistency or coherence.

5.1 Domain-based validation of suggested changes

Domain-based ontology changes validation uses existing domain data to evaluate suggested changes before being applied to the ontology. Approaches in this area differ in terms of (i) *their purpose*, which can be either domain relevance or correctness; (ii) the granularity of the change (*change level*) as some assess only the relevance of the newly added concepts/instances, while others validate a logical statement that corresponds to the proposed change; (iii) the *techniques* they use; and (iv) the type of *domain data resources* employed. We use these criteria to structure the discussion of the approaches and to summarize them in Table 4.

Approaches that detect and suggest changes by analyzing text corpora such as Text2Onto (Cimiano & Volker, 2005) and SPRAT (Maynard *et al.*, 2009), often include statistical techniques to assess the domain relevance of a term that they suggest to add to the ontology. Their purpose, therefore, is to assess relevance in terms of how representative a given term is for a text corpus. These approaches focus on validating individual terms rather than entire statements. The technique employed both by Text2Onto and SPRAT is TF-IDF, an information retrieval measure that quantifies how representative a term is for a given text corpus. Due to relying on statistical measures, these methods usually require a large corpora size to perform well. Additionally, these assessments mainly focus on term relevance with respect to the external corpus and are agnostic to the ontology that is being evolved. However, these approaches are highly useful when large text repositories are available to evolve the ontology.

In Dino (Novacek *et al.*, 2007), the proposed ontology changes are sorted according to a relevance measure and only those above a certain threshold are shown to the users of the tool. In this case, unlike the previous approaches who mainly focus on term-based relevance, the authors propose applying relevance at the level of triples (i.e., in the form of subject, predicate, object). In this case the relevance measure relies on a string similarity between the entities of the triple, and a set of wanted or unwanted words specified by users. At this level, it is expected that users manually create a list of words that reflect domain relevance. Similar to the previous approaches, this work does not take the

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

ontology into consideration to check the relevance. In other words, this approach is mainly based on matching the labels in the proposed change triple with the user-defined set, without performing structural content analysis that the triple would bring to the ontology.

Zablith *et al.* (2010) also highlight the need for relevance checking in their line of work. In this case the authors propose an approach to change validation that takes into account the ontology to be evolved when computing the relevance of a change triple, that is, it aims to make sure that the proposed change is relevant to the ontology in question. To determine this relevance, their technique compares the evolving ontology to the ontological context of the logical statement that represents the proposed change. This ontological context is extracted from online ontologies where the statement appears through a recursive technique that selects the entities linked to the *subject* and *object* of the statement. The comparison between the base ontology and the statement's context uses a pattern-based approach that considers the structure of both data structures. For example, one of the five patterns that the technique relies on is applied when the statement is introducing to the ontology a new concept and this new concept has siblings named identically both in the base ontology and in the statement's context. For example, adding the statement $\langle$ *Tutorial, subClass, Event* $\rangle$ derived from the International Semantic Web Conference ontology context[8], to the Semantic Web for Research Communities ontology[9], is ranked as highly relevant because *Tutorial* has the siblings *Workshop* and *Conference* in both data structures. The study and evaluation of this work shows the superiority of the use of relevance patterns, compared to baseline techniques, such as randomly generated statements or context overlap-based techniques (Zablith *et al.*, 2010). While this work focuses mainly on statement relevance checking, it does not answer the question whether the statement in focus is correct or not.

Additional work has been done on checking the correctness of ontology statements that could be used to filter out invalid relations that should not be added to the ontology in the first place. One approach measures the level of agreement and disagreement within online ontologies on how to represent specific statements, by relying on a formal framework using the semantics within the ontology (d'Aquin, 2009). Even though formal measures are applied at this level, we classify this work as a domain-based validation rather than a formal properties one, because it employs online ontologies as domain knowledge to perform the evaluation. Another work checks the correctness of statements on the Semantic Web, allowing the prediction of how two concepts should be correctly linked based both on the length of the relation path connecting the two concepts in online ontologies and the statement's popularity in online ontologies (Sabou *et al.*, 2009).

Table 4 summarizes the approaches for domain-based validation of changes. We observe a wide variety in terms of the techniques and the domain resources used: (i) approaches stemming from NLP-based research use statistical methods over text corpora to compute term relevance; (ii) Dino moves the focus to statement (rather than term) level relevance although the technique, in essence, measures string similarity between the triple's concepts and a user-specified list of words; (iii) Evolva introduces a technique that takes into account the labels and structure of the evolving ontology, and relies on online ontologies to provide an ontological context for the validated statement. We also discussed methods that measure statement correctness, and could act as a first filter for validating changes.

5.2 Formal properties-based validation

Several recent works have acknowledged the need for imposing custom, application-specific or user-defined properties (in the form of integrity constraints) upon ontologies (Serfiotis *et al.*, 2005; Motik *et al.*, 2007; Lausen *et al.*, 2008; Tao *et al.*, 2010). In this case, formal validation is necessary to prevent cases where the application of a change upon the ontology causes it to violate the imposed integrity constraints. In addition, even if no integrity constraints are considered, Description Logic (DL)

[8] http://annotation.semanticweb.org/ontologies/iswc.owl

[9] http://kmi-web05.open.ac.uk:81/cache/6/98b/5ca1/94b45/7e29980b0f/dfc4e24088dffe851

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

(Baader *et al.*, 2002) or OWL[10] ontologies are usually required to be consistent and coherent (see Flouris *et al.*, 2006), which is another form of validity. Note that in RDF/S[11] ontologies, inconsistency or incoherence cannot occur.

In this subsection, we study works that prevent invalidities from occurring during changes, through a careful application of the changes guaranteeing that any invalidities that occur will be detected and resolved, either automatically, or with the help of the ontology engineer. For reasons of conciseness and uniformity, we do not consider works which fall into the closely related field of ontology debugging (Flouris *et al.*, 2008), that is, works dealing with (and resolving) invalidities without considering how these invalidities occurred.

The requirement of applying changes in a way that the result satisfies the imposed properties (integrity constraints, consistency, coherency) is called the Principle of Validity. Moreover, validation during changes often requires that the changes performed to guarantee validity are 'minimal' (per the Principle of Minimal Change; Alchourron *et al.*, 1985), in the sense of having minimal effects on the ontology. Note that, even though various works have tried to quantify the 'impact' of a change, or to define what 'minimality' is, in various contexts (e.g., Alchourron *et al.*, 1985; Gärdenfors, 1992; Flouris *et al.*, 2006; Konstantinidis *et al.*, 2008a, 2008b; Ribeiro *et al.*, 2009; Flouris *et al.*, 2013), this notion is, in principle, application dependent. Finally, in most cases, we also want the original change to be actually applied to the ontology, that is, we do not want the process of resolving the invalidity to 'undo' one of the changes that the original evolution operation caused; this is called the Principle of Success.

Originally, validation was performed manually by the editor/curator using ontology editors (e.g., Protégé: Noy *et al.*, 2000, 2006; OilEd: Bechhofer *et al.*, 2001) and reasoners used to pinpoint invalidities (for a related evaluation and a list of related editor features, see Stojanovic & Motik, 2002). Since then, more specialized tools appeared, which can identify the changes to be performed to guarantee validity, possibly with some user interaction. User interaction may be direct, through an intuitive interface (e.g., Lam *et al.*, 2005), or indirect through parameters, like evolution strategies (used by Stojanovic *et al.*, 2002). Examples of such tools are KAON (Gabel *et al.*, 2004), OntoStudio (formerly OntoEdit; Sure *et al.*, 2003) and ReTax++ (Lam *et al.*, 2005).

A formal method for applying changes in the presence of integrity constraints appears in Konstantinidis *et al.* (2008a, 2008b) and Flouris *et al.* (2013). This work considers explicitly the three principles described above (validity, minimal change and success) and automatically determines the actions to be taken to resolve invalidities created by the update. A declarative approach for data updating in RDF/S ontologies, using the RUL language, appears in Magiridou *et al.* (2005); this work considers a number of constraints on the resulting RDF/S ontology, and guarantees that the result will satisfy them.

In EvoPat (Riess *et al.*, 2010), the validation is performed using SPARQL[12] queries to determine invalidities (called 'bad smells' by Riess *et al.*, 2010); a 'bad smell' is associated with one or more SPARQL Update[13] statements that resolve it. A similar approach (defining inconsistency patterns and resolving them using change patterns) appears in Djedidi and Aufaure (2009, 2010). Moguillansky *et al.* (2008) present an approach for describing the process of detecting and resolving inconsistencies and incoherencies during evolution using ideas from argumentation frameworks. Updating for DL ontologies is addressed by Liu *et al.* (2006) and Roger *et al.* (2002); these works focus on validating and guaranteeing the consistency and coherency of the result. Haase *et al.* (2005) use ontology debugging techniques to guarantee the validity of the evolution result.

Another family of approaches uses belief revision (Gärdenfors, 1992) techniques and ideas to validate the consistency of the ontology and guarantee minimal changes. For example,

[10] http://www.w3.org/TR/owl-features/

[11] http://www.w3.org/RDF/, http://www.w3.org/TR/rdf-schema/

[12] http://www.w3.org/TR/rdf-sparql-query/

[13] http://www.w3.org/Submission/SPARQL-Update/

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Lee and Meyer (2004), deal with ontologies represented in the *ALU* fragment of DLs; OWL ontologies are handled by Halaschek-Wiener and Katz (2006) and Ribeiro and Wassermann (2007) deal in general with knowledge representation formalisms that do not support negation (making it applicable to RDF/S ontologies, as well as ontologies represented using certain DL fragments).

Gutierrez *et al*. (2006) study the problem of 'erasing' in RDF/S ontologies. Erasing consists of removing triples from an RDF/S ontology to reflect the fact that a given relationship is no longer true in the domain represented by the ontology. Even though integrity constraints are not considered, and incoherence or inconsistency cannot occur in RDF/S ontologies, the problem is far from trivial, because the removed triple may reappear in the result through RDFS entailment (thereby violating the Principle of Success). The approach of Gutierrez *et al*. (2006) addresses this problem using a technique inspired by belief revision.

The most successful paradigm for formalizing the principles of success, validity and minimal change in the context of belief revision is the so-called AGM postulates (Alchourron *et al*., 1985). A series of works studied the feasibility and consequences of applying these postulates in the ontological context (Flouris *et al*., 2004, 2005, 2006; Flouris, 2006a, 2006b; Flouris & Plexousakis, 2006). These works showed that the AGM postulates cannot be applied in several ontology representation formalisms, because, in general, DLs are not closed with respect to updates, in the sense that the set of models corresponding to the 'correct' update (as specified by the AGM postulates, or any other belief revision paradigm) may not be expressible in the given DL. This motivated the development of approximation techniques, that is, approaches resulting to a DL ontology whose set of models is as close as possible to the desired one (Giacomo *et al*., 2007; Wang *et al*., 2010). Giacomo *et al*. (2009) propose two approaches: the first extends DL-Lite$_F$ to a specially designed DL that happens to be closed with respect to data updates (so data updates can be normally supported using belief revision techniques) and the second uses an approximation technique as above. Ribeiro and Wassermann (2006) and Ribeiro *et al*. (2009) present the belief revision notion of relevance (which was proposed in the belief revision literature as another formalization of the Principle of Minimal Change; Hansson, 1991), as an alternative to some of the AGM postulates for the ontological context. Note that this class of works considers only consistency.

The maxi-adjustment algorithm (Benferhat *et al*., 2004) is an approach for repairing inconsistencies in stratified propositional KBs in a minimal manner; the works by Qi *et al*. (2006a, 2006b) were based on this approach to develop evolution algorithms that guarantee the validity of the result in the context of stratified ontologies. Note, however, that this line of work assumes that ontologies are expressed using disjunctive DLs (Meyer *et al*., 2005). The approach by Qi and Du (2009) proposes three different revision operators that guarantee the consistency of a DL ontology after an update, putting special emphasis on the result being syntax independent. Validation of changes may also be done at the level of the ontology metadata; this is done by Maynard *et al*. (2007), where the effects of the ontological changes on the ontology metadata (and vice versa) are studied, in order to validate that the data is consistent with respect to the associated metadata, and vice versa.

Table 5 summarizes the works presented in this subsection. The second column describes the ontology representation language that is supported by each work, the third column the type of properties that are considered (e.g., custom validity rules, consistency, coherency) and the fourth column describes how the problems are resolved when the validation check fails (e.g., using user input, an automated process, or some process inspired by belief revision or other approaches). Note that the various works (in the first column) have been grouped in categories, depending on the content of the other three columns (e.g., Protégé and OilEd appear together, because they share the same properties as related to the other columns).

6 Assessing the impact of evolution

Following a change, an important task is to assess the impact of the evolution that resulted from this change. While the previous phase, 'Validating Ontology Changes' focuses on how changes will impact on the ontology itself, this phase measures the impact on external artifacts that are

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 5 Formal properties-based validation approaches

Referenced work(s)	Supported language	Properties considered	Resolution method
Protégé (Noy *et al.*, 2000, 2006), OilEd (Bechhofer *et al.*, 2001)	OWL	Custom	Manual/editors
KAON (Gabel *et al.*, 2004); OntoStudio (Sure *et al.*, 2003); ReTax++ (Lam *et al.*, 2005)	OWL	Coherence consistency	Semi-automatic
EvoPat (Riess *et al.*, 2010; Konstantinidis *et al.*, 2008a, 2008b; Flouris *et al.*, 2013; Djedidi & Aufaure, 2009, 2010)	RDF/S	Custom	Automatic
RUL (Magiridou *et al.*, 2005)	RDF/S (data only)	Custom	Automatic
Moguillansky *et al.* (2008), Liu *et al.* (2006), Roger *et al.* (2002), Haase *et al.* (2005)	DL	Coherence consistency	Automatic
Lee and Meyer (2004)	*ALU* DL	Consistency	Belief revision
Halaschek-Wiener and Katz (2006)	OWL	Consistency	Belief revision
Ribeiro and Wassermann (2007)	Languages with no negation	Principle of success (deletions)	Belief revision
Gutierrez *et al.* (2006)	RDF/S	Principle of success (deletions)	Belief revision
Flouris *et al.* (2004, 2005, 2006), Flouris (2006a, 2006b), Flouris and Plexousakis (2006), Ribeiro and Wassermann (2006), Ribeiro *et al.* (2009)	General	Consistency	Belief revision
Giacomo *et al.* (2007), Wang *et al.* (2010), Giacomo *et al.* (2009)	DL	Consistency	Approximate
Qi *et al.* (2006a, 2006b)	Disjunctive DL (Stratified)	Consistency	Maxi-adjustment
Qi and Du (2009)	DL	Consistency	Belief revision
Maynard *et al.* (2007)	Metadata	Custom	Manual

DL = Description Logic

dependent on the ontology (i.e., other ontologies, application) or criteria such as costs and benefits of performing a given (set of) change(s). Accordingly, the impact on application and usage determines whether the evolution would have an effect on the operations of the entities that depend on the ontology (Section 6.1); the formal criteria give a quantifiable measure of the impact of a change by using formal properties as the basis of the approach (Section 6.2).

6.1 Application and usage

Former research on assessing the impact of ontology evolution mostly focused on the possible inconsistencies inside the ontology. However, since ontologies are widely used in several application scenarios, the consequences of ontology evolution with respect to the dependent artifacts should be carefully examined as well. The need for such mechanisms was identified in several works that will be presented below. For example, Klein and Fensel (2001) differentiate among invalidation of *data instances*, *dependent ontologies* and *applications*. We adopt this classification as a backbone for structuring this section.

Qin and Atluri (2009) deal with instance invalidation, where the authors distinguish between structural and semantic validity of data instances and propose approaches to ensure them. To achieve that, they propose semantic views as a subset of the ontology and demonstrate that the

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

semantic view, rather than the entire ontology, is responsible for the validity of a data instance. Those views are then used to detect instance invalidation.

A generic framework allowing the systematic study of ontology evolution and instance data sources, as well as the evolution of ontology-related mappings is proposed by Hartung *et al.* (2008). The framework supports the computation of several measures to describe individual ontology versions and mappings, as well as their evolution. Then, it is used to evaluate the evolution of the most popular life science ontologies and to determine the impact of the evolution on the dependent mappings and instances.

Ontologies are often interconnected in intricate ontology networks established through reusing ontology elements (e.g., one ontology extends a concept defined in another ontology), alignments established between ontologies or versioning relations. If one of the ontologies in the network evolves, the impact on the other members of the network (i.e., *dependent ontologies*) should be assessed. For example, an approach that determines whether the changes in one ontology affect the reasoning inside other mapped ontologies is presented by Klein and Stuckenschmidt (2003): the authors developed a change detection and analysis method that predicts the effect of changes on the concept hierarchy and allows ontologies to evolve without unpredictable effects on other ontologies. Thor *et al.* (2009) propose a generic approach to annotate generated ontology mappings independently from the matching approach used. Then, as the ontology evolves, stability measures are calculated over the annotations to identify the quality of the mappings and the impact of ontology evolution on them.

Regarding the impact of ontology evolution on the *dependent applications*, MORE (Huang & Stuckenschmidt, 2005) is an early attempt that evaluates the consequences of ontology changes for compatibility with applications that rely on it. The authors show that temporal logic can provide a solid semantic foundation and serve as an extended query language to detect the ontology change and its consequences. Their approach can answer queries about the facts that were true in previous versions that no longer hold, as well as to determine the ontology version that can answer specific queries.

The floating version model (Xuan *et al.*, 2006) is another approach to restrict the evolution scenarios for maintaining compatibility. The authors propose the *principle of ontological continuity*, according to which the evolution of an ontology should not falsify axioms that were previously true, so only the addition of new information is permitted; this limits the impact of ontology evolution on existing applications.

Wang *et al.* (2008) propose approaches to maintain the consistency and to keep the continuousness of the dependent applications during ontology evolution. A virtual-space framework is put forward and most of the changes are to be made there. The impact of two specific change operations (namely, the *change property range* and the *split property*) on the dependent application is evaluated and resolution strategies are proposed.

Recent works try to assess the impact of ontology evolution on the dependent applications based on the end-user's incoming queries. In early works, Liang *et al.* (2006a, 2006b) keep track of the changes while updating the ontology and use that information to validate and repair queries of the dependent applications. A more formal approach in the same spirit is presented by Kondylakis and Plexousakis (2011a). A high-level language of changes is employed to capture ontology changes, and efficient algorithms are described to recognize the input queries affected (Kondylakis & Plexousakis, 2012). Besides the identification of the changes that invalidate the query, query rewriting techniques are used to repair the queries and/or produce best overapproximations (Kondylakis & Plexousakis, 2011b). In this context, other approaches apply *Stream Reasoning* techniques to reason over rapidly changing ontologies (Della Valle *et al.*, 2008; Ren & Pan, 2011). In this case, the reasoning occurs taking into account the changes that are continuously applied to an ontology, rather than considering the ontology at a static point in time. This would enable, for example, the ability to answer queries in real-time changing environments (Barbieri *et al.*, 2009). At this level, the evolution impact is directly reflected by the query applied.

Finally, a more liberal schema evolution approach that could be used in ontology evolution is presented by Papastefanatos *et al.* (2009, 2010). The authors discuss the problem of performing impact prediction for changes that occur at the schema level. In this approach, schema, queries

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 6 Evolution impact approaches

Referenced work(s)	Application or usage resources	Formal criteria employed
Hartung *et al.* (2008), Qin and Atluri (2009)	Data instances	
Hartung *et al.* (2008), Klein and Stuckenschmidt (2003)	Dependent ontologies	
Hartung *et al.* (2008), Thor *et al.* (2009)	Dependent mappings	
MORE (Huang & Stuckenschmidt, 2005; Wang *et al.*, 2008), floating version model (Xuan *et al.*, 2006)	Dependent applications	
Liang *et al.* (2006a, 2006b), Papastefanatos *et al.* (2009, 2010); Exelixis (Kondylakis & Plexousakis, 2011a, 2011b, 2012)	Dependent queries	
Pammer *et al.* (2010, 2009)		Assertional effects
Palmisano *et al.* (2008)		Cost of change measure
Haase and Stojanovic (2005)		Notion of minimal impact

and views are represented as directed graphs. Those graphs enable the user to create hypothetical evolution events and examine their impact over a graph. They also allow the definition of rules for regulating the evolution impact on the system and to automate its adaptation to evolution events.

6.2 Formal criteria

In addition to application and usage-based impact assessment, formal methods have also been employed to assess the impact of a change on an ontology under evolution. We discuss in this section the impact in terms of (i) assertional effects, (ii) cost of performing a change and (iii) notion of minimal impact.

Assertional effects measure what is gained or lost after performing an ontology change (Pammer *et al.*, 2009). This work is meant to aid the user to have a quick overview of a change impact, in order to make a decision about whether the change should be applied or not, while preserving conceptual consistency. The work formally describes the assertional effects, and an implementation is supplied as a support for the users during ontology development (Pammer *et al.*, 2010).

Another approach proposes the evaluation of changes in ontology evolution using an impact function, which computes the cost involved in performing the change (Palmisano *et al.*, 2008). This cost is aimed for agents using and changing the ontology, to make a better decision whether to apply the change or not. The authors propose an approach to compute such costs without the use of reasoning, but by identifying the parts of the ontology that will be affected as a result of the change. The impact takes into consideration the number of axioms involved in the change, and the expressivity of the ontology parts.

Haase and Stojanovic (2005) present the notion of *minimal impact*, a concept dependent on user requirements. The idea is based on selecting and implementing the minimum number of ontology changes, which result in a 'maximal consistent subontology'. The authors define the concept of maximal consistent subontology, as the part of the ontology to which you cannot add any axiom, without loosing its consistency. Table 6 presents a summary of works involved in measuring and evaluating impact resulting from evolving ontologies, highlighting the popularity of approaches that focus on measuring impact on dependent artifacts.

7 Managing changes

Managing changes involves keeping track of the performed changes (through recording or *a posteriori* detecting them), as well as keeping track of the various versions that the ontology went

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

through in its lifecycle (i.e., after each evolution). This would allow, for example, to restore a previous version of the ontology when needed, or trace back the history of ontological entities or facts, or help in scenarios where the ontology is built collaboratively. Note that the detection of different ontology versions (and the changes between them) is especially important in cases where the user has no control over the evolved ontology (e.g., when he/she reuses an ontology developed by an independent organization); in such a scenario, it is important for the user to be able to identify new versions and/or evolution mappings between different versions. We discuss in this section the works involved in recording ontology changes (Section 7.1), as well as ontology versioning (Section 7.2).

7.1 Recording changes

One of the crucial tasks related to ontology evolution is the management of the differences (deltas) of subsequent versions of ontologies. This proved to play a crucial role in various tasks, such as the synchronization of autonomously developed versions (Cloran & Irwin, 2005) or the visualization and understanding of the evolution history of an ontology (Noy *et al.*, 2006). Deltas are also useful to reduce communication or storage overhead, because they are usually more compact in size than entire versions (so they can be communicated and stored more efficiently) (Papavassiliou *et al.*, 2009, 2013).

Changes can be *recorded* as they happen in a manual or automatic manner, or they can be *a posteriori detected* using some change detection tool. Both applications require a *change language* used to represent such changes; a change language is essentially a set of different changes, along with their semantics, which the delta management system understands and records (or detects).

The recording process itself, which keeps track of changes as they happen, is trivial given a change language, so in this subsection we focus on two different aspects of change management: the various *change languages* that have been defined in the literature, and the *change detection tools* that have been proposed.

7.1.1 Change languages

Unfortunately, there is no standard, agreed-upon list of changes (change language) that are necessary for any given context or application. In effect, each change recording tool reports its own, different set of changes, which is of different nature and granularity (see, e.g., Oliver *et al.*, 1999; Noy & Musen, 2002; Stojanovic *et al.*, 2002; Stuckenschmidt & Klein, 2003; Klein, 2004; Noy & Klein, 2004; Stojanovic, 2004; Rogozan & Paquette, 2005; Palma *et al.*, 2007; Plessers *et al.*, 2007; Javed *et al.*, 2009; Papavassiliou *et al.*, 2009, 2013; Hartung *et al.*, 2012 for some different proposals of change languages). An interesting feature found in certain works (e.g., Auer & Herre, 2006; Plessers *et al.*, 2007; Djedidi & Aufaure, 2010) is the ability to define custom, user-defined changes using some syntax for determining the changes' semantics and intuition. Such *dynamic* change languages are useful because they can be adapted to different needs and applications.

An important aspect related to the usefulness of a change language is its granularity. A language of changes can be *low level*, consisting of simple add/remove operations, or *high level*, which describes more complex updates, and essentially groups several low-level changes into high-level ones in order to report more intuitive changes. Low-level languages are simpler to record or detect, but high-level languages produce more concise deltas, which are more easily understandable by humans and capture more closely the intuitions and intentions of the ontology editor (Hartung *et al.*, 2012; Papavassiliou *et al.*, 2013). On the other hand, low-level changes are necessary to capture fine-grained modifications to the ontology. The concept of low-level and high-level changes has been discussed under different names in various works in the literature (e.g., elementary/composite: Stojanovic & Motik, 2002; Stojanovic *et al.*, 2002 and atomic/complex: Stuckenschmidt & Klein, 2003).

Another important aspect is whether the language supports terminological changes, such as renaming or merging (Oliver *et al.*, 1999). Such changes occur often in practice (Papavassiliou *et al.*, 2013),

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

but are difficult to detect or record, because they can be easily confused with structural changes (e.g., a class renaming is implemented as an addition and deletion, and it is not always easy to discriminate whether an addition–deletion pair is a real renaming or not).

In many works, change languages are represented using an *ontology of changes* (Klein *et al.*, 2002; Klein & Noy, 2003; Plessers & De Troyer, 2005; Noy *et al.*, 2006). Instantiations of such an ontology describe the changes (delta) that have occurred between versions. This representation of a delta is useful because it allows the manipulation and communication of deltas using popular Semantic Web technologies. Alternative ways that can be used to represent changes, plus possible interactions between such representations, can be found in the work of Klein and Noy (2003).

As already mentioned, given a change language, change recording can be easily performed. Sometimes manual recording is used, but such a method is often incomplete or erroneous, even for ontologies that are centrally managed and edited (Stojanovic *et al.*, 2002; Papavassiliou *et al.*, 2013); for example, Papavassiliou *et al.* (2013) identified changes in a centrally curated ontology that were not properly recorded, despite the curators' best (manual) efforts. Automatic recording tools can help in this respect, but their use is hindered in applications where the changes are not centrally managed (e.g., in distributed environments).

7.1.2 Change detection tools

To address the problem of identifying the changes between two (subsequent) versions of an ontology when direct recording is not possible, change detection tools can be used. Such tools can identify the changes that happened between versions *after* the change has occurred, that is, using as input only the two ontology versions. Change detection tools are based on some language of changes, and can be categorized depending on whether the corresponding language is low level or high level.

Low-level change detection tools (e.g., Volkel *et al.*, 2005; Zeginis *et al.*, 2007, 2011; Konev *et al.*, 2008; Kontchakov *et al.*, 2008; Franconi *et al.*, 2010) usually report simple add/delete operations; despite the simplicity of the underlying change language, such tools differ in their semantics and properties, as well as in the supported ontology representation language. Given that low-level changes are not concise or intuitive enough to guarantee human readability, many such works focus on formally studying the change detection process and guaranteeing useful formal properties for the produced deltas (Zeginis *et al.*, 2007, 2011; Franconi *et al.*, 2010).

On the other hand, high-level change detection tools usually focus on the definition of a change language that is intuitive and concise enough to capture the editor's intuition. As a result, less focus is placed on the formal properties of the language, or the detection algorithm (Rogozan & Paquette, 2005; Palma *et al.*, 2007). This causes the semantics of the various supported change operations to be presented informally, often resulting in unclear definitions; furthermore, there is usually no formal machinery to guarantee any properties regarding the detection algorithm.

Klein (2004), Noy and Musen (2004, 2002) and Noy *et al.* (2004) all present a high-level change detection approach, which is implemented in PromptDiff, an extension of Protégé, and employs heuristic-based matchers to detect the changes between versions. As a result, the detection process involves an uncertainty aspect, and has been measured to have a recall of 96% and a precision of 93%. A similar approach appears in the context of OntoView (Klein *et al.*, 2002).

Plessers *et al.* (2007) propose detecting changes using temporal queries over a version log that is maintained during updates. The most important feature of this approach is that it can use a dynamic, user-defined change language. Thus, the user can define custom changes, through the Change Definition Language, and these changes can be subsequently detected using the approach. The downside is that the detection process requires a version log to be maintained, so it essentially requires recording information on the changes as they happen; in addition, terminological changes are not supported.

Papavassiliou *et al.* (2009, 2013) propose a formal framework for defining high-level change operations and define a set of requirements for such operators, such as conciseness, intuitiveness, consistent application and detection semantics, reversibility and others. Then, a particular language of

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 7 Change languages and change detection approaches

Referenced work(s)	Formalism considered	Change language characteristics	Change detection algorithm characteristics
Javed *et al*., (2009), Klein and Noy (2003), Noy and Klein (2004), Noy *et al*. (2006), Oliver *et al*. (1999), Stojanovic (2004), Stojanovic *et al*. (2002), Stojanovic and Motik (2002), Stuckenschmidt and Klein (2003)	Generic	High level	No Algorithm
PromptDiff (Klein, 2004; Noy *et al*., 2004; Noy & Musen, 2004, 2002); OntoView (Klein *et al*., 2002)	OWL	High level	Based on heuristics
Plessers and De Troyer (2005), Plessers *et al*. (2007)	Generic	High level, dynamic, no terminological	Requires version log
Hartung *et al*. (2012), Palma *et al*. (2007), Rogozan and Paquette (2005)	Generic	High level	High level
Auer and Herre (2006)	RDF	High level, dynamic	No algorithm
Djedidi and Aufaure (2010)	Generic	High level, dynamic	No algorithm
Papavassiliou *et al*. (2009, 2013)	RDF/S	High level, formal	Deterministic, formal
Zeginis *et al*. (2007, 2011)	RDF/S	Low level, formal	Deterministic, formal, low level
Franconi *et al*. (2010)	Generic	Low level, formal	Deterministic, formal, low level
Konev *et al*. (2008), Kontchakov *et al*. (2008)	DL	Low level, formal	Deterministic, formal, low level
Volkel *et al*. (2005)	RDF	Low level	Low level

DL = Description Logic

changes and the corresponding detection algorithm are proposed, and the authors show that their language (and the corresponding detection algorithm) satisfies the proposed requirements. Apart from the detection semantics, emphasis is also put in the ability to traverse the history of versions in both ways by applying the detected changes or their inverses.

A similar approach is COnto-Diff (Hartung *et al*., 2012), which detects high-level changes according to a language of changes defined by Hartung *et al*. (2012). The detection process uses a rule-based approach, coupled with a mapping between the elements (concepts, properties) of the two ontology versions. The application of the detected changes (and their inverses) is also considered in this work.

Table 7 summarizes the works presented in this subsection. For each work, we describe the considered formalism (second column), as well as the characteristics of the change language and the detection algorithm (if any), in the third and fourth column, respectively. Note that the various works (in the first column) have been grouped in categories, depending on the content of the other three columns (e.g., PromptDiff and OntoView appear together, because they share the same properties as related to the other columns).

As can be seen also in Table 7, most existing approaches employ high-level change languages, acknowledging the fact that these languages are most useful from a user perspective, because they result in more concise and intuitive deltas. In this respect, the most challenging issue is the identification of a 'standard' high-level language that would be suitable for each of the major representation formalisms (e.g., RDF/S, OWL, etc.), as well as the formal definition of its semantics and corresponding detection algorithm. As already mentioned above, the proposed high-level languages are different in structure, and most of them are not coupled with formal semantics and/or are not associated with a deterministic detection algorithm, which causes various problems when it comes to automatically manipulating deltas.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

7.2 Ontology versioning approaches

Ontology versioning refers to the ability to handle an evolving ontology by creating and managing its different versions (Klein & Fensel, 2001). Given that ontologies are often interlinked or reused, versioning of evolving ontologies is necessary to guarantee smooth interoperation. Furthermore, ontologies are being used by various agents, applications or other elements, and a change could potentially cause problems in such accessing elements; in such a case, versioning can allow the agent or application to switch to the older version until such problems are fixed.

To achieve a smooth evolution we need not only to store the different versions of an evolving ontology, but also to manage these versions, with the aim of minimizing any adverse effects that a change in a given ontology could have upon related (dependent) ontologies, agents, applications or other elements. This can be done by relating versions with accessing elements (i.e., ontologies, applications or agents) and transparently providing access to either the current or some older version of the ontology, depending on the needs of the accessing element (Klein & Fensel, 2001). This ability allows the dependent elements to upgrade to the new version at their own pace, if at all, which is considered a very useful feature (Heflin *et al.*, 1999; Heflin & Pan, 2004). In fact, it has been argued that ontology versioning should be an indispensable part of ontology management tools such as Protégé (Noy & Musen, 2004).

Several issues are associated with versioning. One such issue is related to identification, namely, how to identify and label new versions. This issue is not as trivial as it may seem: for example, it is not clear whether a subtle syntactic change should result in the creation of a new version or the overwriting of the existing one (Heflin *et al.*, 1999; Klein *et al.*, 2002).

Another issue is related to the identification and recording of the relationship between different versions. The term 'relationship' in this context could refer to the identification of which version emerged from which, and how. As such, it could involve information regarding compatibility between the versions (Klein & Fensel, 2001), fine-grained information regarding the relationships between ontological elements in the two versions (Klein & Fensel, 2001), the delta that led from one ontology to the other (Klein *et al.*, 2002; Klein, 2004) and other metadata regarding the evolution and the versions themselves (Klein *et al.*, 2002). The above information would form a tree of versions, which shows which version evolved out of which and contains some relevant metadata such as those described above.

Maintaining the compatibility information between versions is very important to correctly relate versions with accessing elements without causing problems in the functioning of such elements. Klein and Fensel (2001) define and study different types of compatibility. It has been argued that compatibility determination cannot be performed fully automatically (Heflin & Pan, 2004); to address this problem, Heflin and Hendler (2000) and Heflin *et al.* (1999) proposed to make backwards compatibility between versions explicit in a machine readable format using the SHOE language (Luke *et al.*, 1997). This allows a computer agent to determine compatibility between versions, and to choose automatically which version to use; this approach is in contrast with work by Klein and Fensel (2001) and Klein *et al.* (2002), where a centralized approach is adopted.

Usually, versioning metadata refer to the two versions as a whole; it has been argued that such metadata could also be defined at the level of ontological elements (Klein & Fensel, 2001). This results in a more fine-grained specification of the relationship between the versions, which allows the explication of the relation between ontological elements and the identification of the evolution history of each element independently.

Another work related to ontology versioning is performed by Redmond *et al.* (2008), who present a system for managing changes in a multi-editor environment, providing metadata about different revisions. Kirsten *et al.* (2009) design a database to store the changes on biomedical ontologies supporting different kinds of change analysis. Allocca *et al.* (2009) present an approach for automatically detecting version relations between different ontologies, using heuristics based on the ontologies' URIs and identifiers.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 8 Ontology versioning

Referenced work(s)	Problems addressed (related to versioning)	Metadata considered
Allocca *et al.* (2009)	Automatic detection of version relations	Ontologies' URIs and identifiers
Klein and Fensel (2001)	Transparent version management Types of compatibility Centralized compatibility determination	Compatibility between versions Fine-grained (ontology elements)
Klein *et al.* (2002)	Identification Centralized compatibility determination	Evolution information (e.g., delta) Relationship between versions
Heflin and Hendler (2000), Heflin *et al.* (1999)	Identification Explicit compatibility determination	Relationship between versions
Klein (2004), Redmond *et al.* (2008), Kirsten *et al.* (2009)	Version management	Evolution information (e.g., delta)
Grandi (2009), Huang and Stuckenschmidt (2005), Keberle *et al.* (2007), Plessers *et al.* (2005)	Temporal reasoning to perform versioning	Temporal information on versions
Obst and Chan (2005)	Generic versioning system	Relationship between versions
Heflin and Pan (2004)	Theoretical aspects	Relationship between versions

Another line of work in versioning is inspired by temporal reasoning (Huang & Stuckenschmidt, 2005; Plessers *et al.*, 2005; Keberle *et al.*, 2007; Grandi, 2009). Obst and Chan (2005) discuss the architecture and requirements of a system to handle ontology management, by presenting initial ideas regarding the need for a generic versioning system that supports users while evolving ontologies. Theoretical aspects of the problem are studied by Heflin and Pan (2004).

Table 8 summarizes the works presented in this subsection. The second column of the table shows the main versioning-related problems that each work addresses, and the third column shows the version-related metadata considered. The various works (in the first column) have been grouped in categories, depending on the content of the other columns.

Most of the current works related to ontology versioning require human input, or user-provided metadata. The development of an automated versioning system that would automatically detect versions, and, most importantly, compatibility information and/or other metadata between versions, would be a great step forward in the field. Such an automated system would successfully address the problem of ontology interoperability in dynamic settings, which is an important problem being faced in many practical applications involving ontologies.

8 Conclusion

Ontology evolution is one of the core requirements for keeping ontologies usable within the environments they are being applied. There exists a substantial amount of work in the area, but there is significant fragmentation as specific works are classified in quite distinct subfields. In this paper we present an ontology evolution cycle, in which we identify the tasks needed for evolving ontologies at the appropriate level of granularity, with the aim to bridge the existing works across different groups of the community. We use the evolution cycle to guide our survey of the relevant literature involved in each of the tasks in the cycle.

We map in Table 9 the referenced works identified in this paper to the tasks of the ontology evolution cycle. The aim of this table is to give a degree of guidance for fellow researchers in ontology evolution, to identify the degree of research undergone in specific areas compared to

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Table 9 Relations between tasks of the ontology evolution cycle and components of existing ontology evolution framework

Cycle step	References
Detecting the Need for Evolution – Data	Velardi *et al.* (2001), Stojanovic (2004), Cimiano and Volker (2005), Bloehdorn *et al.* (2006), Novacek *et al.* (2007), Ottens *et al.* (2007), Maynard *et al.* (2009), Zablith (2009)
Detecting the Need for Evolution – Usage	Stojanovic (2004), Alani *et al.* (2006), Bloehdorn *et al.* (2006), Luczak-Rosch (2009), Javed *et al.* (2011), Pruski *et al.* (2011)
Suggesting Changes – Unstructured Knowledge	Cimiano and Volker (2005), Novacek *et al.* (2007), Maynard *et al.* (2009), Ottens *et al.* (2009)
Suggesting Changes – Structured Knowledge	Stojanovic (2004), Cimiano and Volker (2005), Zablith *et al.* (2008), Maynard *et al.* (2009)
Validating Changes – Domain	Cimiano and Volker (2005), Novacek *et al.* (2007), d'Aquin (2009), Maynard *et al.* (2009), Sabou *et al.* (2009), Zablith *et al.* (2010)
Validating Changes – Formal Properties	Bechhofer *et al.* (2001), Sure *et al.* (2003), Gabel *et al.* (2004), Lee and Meyer (2004), Haase *et al.* (2005), Lam *et al.* (2005), Magiridou *et al.* (2005), Flouris (2006a), Flouris and Plexousakis (2006), Gutierrez *et al.* (2006), Halaschek-Wiener and Katz (2006), Liu *et al.* (2006), Noy *et al.* (2006), Qi *et al.* (2006a), Ribeiro and Wassermann (2006, 2007), Giacomo *et al.* (2007, 2009), Maynard *et al.* (2007), Konstantinidis *et al.* (2008a), Moguillansky *et al.* (2008), Djedidi and Aufaure (2009), Qi and Du (2009), Riess *et al.* (2010), Wang *et al.* (2010)
Assessing Evolution Impact – Formal Criteria	Haase and Stojanovic (2005), Palmisano *et al.* (2008), Pammer *et al.* (2010, 2009)
Assessing Evolution Impact – Usage	Klein and Stuckenschmidt (2003), Huang and Stuckenschmidt (2005), Xuan *et al.* (2006), Liang *et al.* (2006a), Hartung *et al.* (2008), Wang *et al.* (2008), Papastefanatos *et al.* (2009), Qin and Atluri (2009), Thor *et al.* (2009), Kondylakis and Plexousakis (2011b)
Managing Changes – Recording Changes	Klein *et al.* (2002), Klein and Noy (2003), Stuckenschmidt and Klein (2003), Klein (2004), Noy and Klein (2004), Noy *et al.* (2004), Stojanovic (2004), Plessers and De Troyer (2005), Rogozan and Paquette (2005), Volkel *et al.* (2005), Auer and Herre (2006), Noy *et al.* (2006), Palma *et al.* (2007), Zeginis *et al.* (2007), Konev *et al.* (2008), Kontchakov *et al.* (2008), Javed *et al.* (2009), Papavassiliou *et al.* (2009), Djedidi and Aufaure (2010), Franconi *et al.* (2010)
Managing Changes – Versioning	Heflin and Hendler (2000), Klein and Fensel (2001), Klein *et al.* (2002), Heflin and Pan (2004), Klein (2004), Huang and Stuckenschmidt (2005), Obst and Chan (2005), Plessers *et al.* (2005), Keberle *et al.* (2007), Redmond *et al.* (2008), Allocca *et al.* (2009), Grandi (2009)

others. We observe that some tasks have been more researched than others. This is particularly true for works which are not limited to the area of ontology evolution. For example, the work on change validation based on formal properties is well explored in the area of consistency checking and management, which could be directly exploited in the area of ontology evolution. However, there are other areas that can directly benefit from further research; for example, in exploiting domain information for change validation, or expanding the work on usage-driven ontology evolution.

In order to achieve a better platform for evolving ontologies, we identify the need to work toward integrating solutions and tools for ontology evolution. Ultimately, such integration should be the ground for providing users with a seamless experience in addressing all issues of the evolution process, ranging from triggering the evolution to handling inconsistencies and change management. In addition to solving the partial requirements for evolving ontologies, the community needs to think about how the conducted work can connect to and reuse other approaches that are working on similar tasks.

We believe that the integration of approaches and standards in ontology evolution will have a positive impact on maintaining the backbones of Semantic Web systems. Having in place such

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

integrated approaches would encourage people to represent their information in ontologies, by decreasing the maintenance and evolution costs needed to keep the ontologies up-to-date. This will have an impact on the overall realization of the vision of the Semantic Web, which is pushing towards 'moving from documents, to data and information' (Shadbolt *et al.*, 2006).

We foresee that one way to move toward converging and integrating the different approaches in the ontology evolution community is by encouraging researchers to engage and share their work not only in workshops and conferences, but also through Web portals (e.g., the Ontology Dynamics portal[14]). Such portals can serve as a reference or social network where anyone in the community can easily exchange ideas. One thing that the community lacks is access to gold standards and scenario-centric data. We believe that communicating use cases where ontology evolution is needed, will offer the means for researchers in the community to work on common problems, and hence have access to shared domain data, ontologies and gold standards that will push the research toward better and more effective results.

Acknowledgments

We would like to thank Nathalie Aussenac-Gilles, Philippe Laublet and Jeff Z. Pan for their invaluable time and feedback for producing earlier versions of this paper. This work was partially supported by the PlanetData NoE (FP7:ICT-2009.3.4, 257641).

References

Alani, H. 2006. Position paper: ontology construction from online ontologies. In *Proceedings of the 15th International Conference on World Wide Web (WWW-06)*, Les Carr, David De Roure, Arun Iyengar, Carole A. Goble & Michael Dahlin (eds). ACM, Edinburgh, Scotland, 491–495.

Alani, H., Harris, S. & O'Neil, B. 2006. Winnowing ontologies based on application use. In *Proceedings of 3rd European Semantic Web Conference (ESWC-06)*, Budva, Montenegro.

Alchourron, C., Gärdenfors, P. & Makinson, D. 1985. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic* **50**, 510–530.

Allocca, C., d'Aquin, M. & Motta, E. 2009. Detecting different versions of ontologies in large ontology repositories. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09)*, Chantilly, VA, USA.

Angeletou, S., Sabou, M. & Motta, E. 2008. Semantically enriching folksonomies with FLOR. In *Proceedings of the 1st International Workshop on Collective Semantics: Collective Intelligence and the Semantic Web (CISWeb-08)*, Tenerife, Canary Islands, Spain.

Antoniou, G. & Harmelen, F. V. 2004. *A Semantic Web Primer*. The MIT Press.

Auer, S. & Herre, H. 2006. A versioning and evolution framework for RDF knowledge bases. In *Perspectives of Systems Informatics, 6th International Andrei Ershov Memorial Conference (PSI-06), Revised Papers*, Akademgorodok, Novosibirsk, Russia.

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

Banerjee, S. & Pedersen, T. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the 3rd International Conference on Computational Linguistics and Intelligent Text Processing*, Alexander F. Gelbukh (ed.). Mexico, 136–145.

Barbieri, D., Braga, D., Ceri, S., Della Valle, E. & Grossniklaus, M. 2009. C-SPARQL: SPARQL for continuous querying. In *Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, 1061–1062.

Bechhofer, S., Horrocks, I., Goble, C. & Stevens, R. 2001. OilEd: a reasonable ontology editor for the semantic web. In *Proceedings of the 24th German/9th Austrian Conference on Artificial Intelligence (KI-01)*, Vienna, Austria.

Bechhofer, S., Volz, R. & Lord, P. 2003. Cooking the semantic web with the OWL API. In *Proceedings of the 2nd International Semantic Web Conference (ISWC-03)*, Sanibel Island, Florida, USA, 659–675.

Benferhat, S., Kaci, S., Le Berre, D. & Williams, M. 2004. Weakening conflicting information for iterated revision and knowledge integration. *Artificial Intelligence* **153**, 339–371.

[14] http://www.ontologydynamics.org/

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Bloehdorn, S., Haase, P., Sure, Y. & Volker, J. 2006. Ontology evolution. In *Semantic Web Technologies—Trends and Research in Ontology-Based Systems*, John Davies, Rudi Studer & Paul Warren (eds). John Wiley & Sons, 51–70.

Cimiano, P. & Volker, J. 2005. Text2Onto: a framework for ontology learning and data-driven change discovery. In *Natural Language Processing and Information Systems*, Alicante, Spain, 227–238.

Clark, P., Fellbaum, C. & Hobbs, J. 2008. Using and extending WordNet to support question-answering. In *Proceedings of the 4th Global WordNet Conference (GWC-08)*, Szeged, Hungary, 111–119.

Cloran, R. & Irwin, B. 2005. Transmitting RDF graph deltas for a cheaper semantic web. In *Proceedings of the 8th Annual Southern African Telecommunication Networks and Applications Conference (SATNAC-05)*, Drakensberg, South Africa.

d'Aquin, M. 2009. Formally measuring agreement and disagreement in ontologies. In *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP-09)*, Redondo Beach, California, USA, 145–152.

d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M. & Motta, E. 2007. Characterizing knowledge on the semantic web with Watson. In *Proceedings of the 5th International EON Workshop, Colocated with the International Semantic Web Conference (ISWC-07)*, Busan, Korea.

d'Aquin, M., Baldassarre, C., Gridinoc, L., Sabou, M., Angeletou, S. & Motta, E. 2007. Watson: supporting next generation semantic web applications. In *Proceedings of the WWW/Internet Conference*, Vila Real, Portugal.

Della Valle, E., Ceri, S., Barbieri, D., Braga, D. & Campi, A. 2008. A first step towards stream reasoning. In *Proceedings of the Future Internet Symposium (FIS-08)*, Vienna, Austria, 72–81.

Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y. & Kolari, P. 2005. Finding and ranking knowledge on the semantic web. In *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, Galway, Ireland.

Djedidi, R. & Aufaure, M. 2009. Change management patterns (CMP) for ontology evolution process. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09)*, Chantilly, VA, USA.

Djedidi, R. & Aufaure, M. 2010. ONTO-EVOAL an ontology evolution approach guided by pattern modeling and quality evaluation. In *Proceedings of the 6th International Symposium on Foundations of Information and Knowledge Systems (FoIKS-10)*, Sofia, Bulgaria, 286–305.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Flouris, G. 2006a. *Doctoral Dissertation*. PhD thesis, Department of Computer Science, University of Crete.

Flouris, G. 2006b. On belief change in ontology evolution. *AI Communications Journal*, **19**(4), 395–397.

Flouris, G., Huang, Z., Pan, J., Plexousakis, D. & Wache, H. 2006. Inconsistencies, negations and changes in ontologies. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, Boston, Massachusetts, 1295–1300.

Flouris, G., Konstantinidis, G., Antoniou, G. & Christophides, V. 2013. Formal foundations for RDF/S KB evolution. *International Journal on Knowledge and Information Systems (KAIS-13)* **35**(1), 153–191.

Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D. & Antoniou, G. 2008. Ontology change: classification and survey. *Knowledge Engineering Review* **26**(2), 117–152.

Flouris, G. & Plexousakis, D. 2006. Bridging ontology evolution and belief change. In *Advances in Artificial Intelligence*, Hobart, Australia, 486–489.

Flouris, G., Plexousakis, D. & Antoniou, G. 2004. Generalizing the AGM postulates: preliminary results and applications. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR-04)*, Whistler, Canada, 171–179.

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

Flouris, G., Plexousakis, D. & Antoniou, G. 2006. On generalizing the AGM postulates. In *Proceedings of the 3rd European Starting AI Researcher Symposium (STAIRS-06)*, Riva del Garda, Italy, 132–143.

Franconi, E., Meyer, T. & Varzinczak, I. 2010. Semantic diff as the basis for knowledge base versioning. In *Proceedings of the 13th International Workshop on Non-Monotonic Reasoning (NMR-10)*, Toronto, Canada.

Gabel, T., Sure, Y. & Voelker, J. 2004. D3.1.1.a: KAON – ontology management infrastructure. SEKT informal deliverable, Institute AIFB, University of Karlsruhe.

Gärdenfors, P. 1992. *Belief Revision: An Introduction*. Cambridge University Press, 1–20.

Giacomo, G. D., Lenzerini, M., Poggi, A. & Rosati, R. 2007. On the approximation of instance level update and erasure in Description Logics. In *Proceedings of the 22nd Conference of the American Association for Artificial Intelligence (AAAI-07)*, Vancouver, British Columbia, 403–408.

Giacomo, G. D., Lenzerini, M., Poggi, A. & Rosati, R. 2009. On instance-level update and erasure in Description Logic ontologies. *Journal of Logic and Computation* **19**(5), 745–770.

Gomez-Perez, A., Corcho, O. & Fernandez-Lopez, M. 2003. *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition*. Springer.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Gracia, J., d'Aquin, M. & Mena, E. 2009. Large scale integration of senses for the semantic web. In *Proceedings of the 18th International Conference on World Wide Web (WWW-09)*, ACM, 611–620.
Grandi, F. 2009. Multi-temporal RDF ontology versioning. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09)*, Chantilly, VA, USA.
Gruber, T. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition* **5**(2), 199–220.
Gutierrez, C., Hurtado, C. & Vaisman, A. 2006. The meaning of erasing in RDF under the Katsuno-Mendelzon approach. In *Proceedings of the 9th International Workshop on the Web and Databases (WebDB-06)*, Chicago, Illinois.
Haase, P. & Stojanovic, L. 2005. Consistent evolution of OWL ontologies. In *The Semantic Web: Research and Applications*, Heraklion, Crete, Greece, 182–197.
Haase, P. & Sure, Y. 2004. D3.1.1.b state of the art on ontology evolution. SEKT Deliverable, Institute AIFB, University of Karlsruhe.
Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. 2005. A framework for handling inconsistency in changing ontologies. In *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, Galway, Ireland, 353–367.
Halaschek-Wiener, C. & Katz, Y. 2006. Belief base revision for expressive Description Logics. In *Proceedings of OWL: Experiences and Directions 2006 (OWLED-06)*, Athens, GA, USA.
Hansson, S. O. 1991. Belief contraction without recovery. *Studia Logica* **50**(2), 251–260.
Hartung, M., Groß, A. & Rahm, E. 2012. COnto–Diff: generation of complex evolution mappings for life science ontologies. *Journal of Biomedical Informatics* **46**(1), 15–32.
Hartung, M., Kirsten, T. & Rahm, E. 2008. Analyzing the evolution of life science ontologies and mappings. In *Proceedings of the 5th International Workshop on Data Integration in the Life Sciences (DILS-08)*, Amos Bairoch, Sarah Cohen Boulakia & Christine Froidevaux (eds). Springer-Verlag, Evry, France, 11–27.
Hartung, M., Terwilliger, J. & Rahm, E. 2011. Recent advances in schema and ontology evolution. In *Schema Matching and Mapping*, Bellahsene, Z., Bonifati, A. & Rahm, E. (eds). Data-Centric Systems and Applications. Springer, 149–190.
Heflin, J., Hendler, J. & Luke, S. 1999. Coping with changing ontologies in a distributed environment. In *Proceedings of the Workshop on Ontology Management of the 16th National Conference on Artificial Intelligence (AAAI-99)*, Orlando, Florida.
Heflin, J. & Hendler, J. 2000. Dynamic ontologies on the web. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, Austin, Texas, 443–449.
Heflin, J. & Pan, Z. 2004. A model theoretic semantics for ontology versioning. In *Proceedings of the 3rd International Semantic Web Conference (ISWC-04)*, Hiroshima, Japan, 62–76.
Huang, Z. & Stuckenschmidt, H. 2005. Reasoning with multi-version ontologies: a temporal logic approach. In *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, Galway, Ireland, 398–412.
Ide, N. & Veronis, J. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics* **24**(1), 2–40.
Javed, M., Abgaz, Y. & Pahl, C. 2009. A pattern-based framework of change operators for ontology evolution. In *On the Move to Meaningful Internet Systems Workshop (OTM-09)*, Vilamoura, Portugal, 544–553.
Javed, M., Abgaz, Y. & Pahl, C. 2011. Graph-based discovery of ontology change patterns. In *Proceedings of the Joint Workshop on Knowledge Evolution and Ontology Dynamics (EvoDyn) at ISWC*, Bonn, Germany.
Keberle, N., Litvinenko, Y., Gordeyev, Y. & Ermolayev, V. 2007. Ontology evolution analysis with OWL-MeT. In *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, Innsbruck, Austria, 1–12.
Kirsten, T., Hartung, M., Gross, A. & Rahm, E. 2009. Efficient management of biomedical ontology versions. In *Proceedings of the 2009 On The Move Workshops (OTM-09)*, Vilamoura, Portugal, 574–583.
Klein, M. 2004. *Change Management for Distributed Ontologies*. PhD thesis, Vrije University.
Klein, M. & Fensel, D. 2001. Ontology versioning on the semantic web. In *Proceedings of the International Semantic Web Working Symposium (SWWS-01)*, California, USA, 75–91.
Klein, M., Fensel, D., Kiryakov, A. & Ognyanov, D. 2002. Ontology versioning and change detection on the web. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-02)*, Siguenza, Spain.
Klein, M. & Noy, N. 2003. A component-based framework for ontology evolution. In *Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems*, Acapulco, Mexico.
Klein, M. & Stuckenschmidt, H. 2003. Evolution management for interconnected ontologies. In *Proceedings of the Semantic Integration Workshop at the 2nd International Semantic Web Conference (ISWC-03)*, Florida, USA.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Kondylakis, H. & Plexousakis, D. 2011a. Exelixis: evolving ontology-based data integration system. In *ACM International Conference on Management of Data (SIGMOD-12)*, Scottsdale, AZ, USA, 1283–1286.

Kondylakis, H. & Plexousakis, D. 2011b. Ontology evolution in data integration: query rewriting to the rescue. In *International Conference on Conceptual Modelling (ER-11)*, Brussels, Belgium, 393–401.

Kondylakis, H. & Plexousakis, D. 2012. Ontology evolution: assisting query migration. In *International Conference on Conceptual Modelling (ER-12)*, Florence, Italy, 331–344.

Konev, B., Walther, D. & Wolter, F. 2008. The logical difference problem for Description Logic terminologies. In *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR-08)*, Armando, A., Baumgartner, P. & Dowek, G. (eds), LNAI **5195**, 259–274. Springer-Verlag.

Konstantinidis, G., Flouris, G., Antoniou, G. & Christophides, V. 2008a. A formal approach for RDF/S ontology evolution. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, Patras, Greece, 405–409.

Konstantinidis, G., Flouris, G., Antoniou, G. & Christophides, V. 2008b. On RDF/S ontology evolution. In *Post-Proceedings of the Joint ODBIS & SWDB Workshop on Semantic Web, Ontologies, Databases (SWDB-ODBIS-07)*, Vienna, Austria, 21–42.

Kontchakov, R., Wolter, F. & Zakharyaschev, M. 2008. Can you tell the difference between DL-Lite ontologies? In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-08)*, Lang, J. & Brewka, G. (eds). AAAI Press/MIT Press, 285–295.

Laera, L., Handschuh, S., Zemanek, J., Volkel, M., Bendaoud, R., Hacene, M., Toussaint, Y., Delecroix, B. & Napoli, A. 2008. *D2.3.8 v2 Report and Prototype of Dynamics in the Ontology Lifecycle*. Technical report.

Lam, S., Sleeman, D. & Vasconselos, W. 2005. ReTAX++: a tool for browsing and revising ontologies. In *Poster Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, Galway, Ireland.

Lausen, G., Meier, M. & Schmidt, M. 2008. SPARQLing constraints for RDF. In *Proceedings of 11th International Conference on Extending Database Technology (EDBT-08)*, Nantes, France, 499–509.

Leenheer, P. D. & Mens, T. 2008. Ontology evolution: state of the art and future directions. In *Ontology Management for the Semantic Web, Semantic Web Services, and Business Applications*, Hepp, M., Leenheer, P. D., de Moor, A. & Sure, Y. (eds). Springer, 131–176.

Lee, K. & Meyer, T. 2004. A classification of ontology modification. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI-04)*, 248–258.

Liang, Y., Alani, H. & Shadbolt, N. 2006a. Changing ontology breaks queries. In *Proceedings of the 5th International Semantic Web Conference (ISWC-06)*, Athens, GA, USA, 982–985.

Liang, Y., Alani, H. & Shadbolt, N. 2006b. *Enabling Active Ontology Change Management Within Semantic Web-Based Applications*. Mini-thesis, PhD upgrade report.

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

Li, X., Szpakowicz, S. & Matwin, S. 1995. A WordNet-based algorithm for word sense disambiguation. In *International Joint Conference on Artificial Intelligence (IJCAI-95)* **14**, Montreal, Canada, 1368–1374.

Lopez, V., Uren, V., Sabou, M. & Motta, E. 2009. Cross ontology query answering on the semantic web. In *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP-09)*. Redondo Beach, California, USA.

Luczak-Rosch, M. 2009. Towards agile ontology maintenance. In *Proceedings of the 8th International Semantic Web Conference (ISWC-09)*, vol. 5823, Springer, 965–972.

Luke, S., Spector, L., Rager, D. & Hendler, J. 1997. Ontology-based web agents. In *Proceedings of the 1st International Conference on Autonomous Agents*, Marina del Rey, CA, USA, 59–66.

Maedche, A., Motik, B., Stojanovic, L., Studer, R. & Volz, R. 2002. Managing multiple ontologies and ontology evolution in Ontologging. In *Proceedings of the Conference on Intelligent Information Processing, World Computer Congress*, Montreal, Canada.

Magiridou, M., Sahtouris, S., Christophides, V. & Koubarakis, M. 2005. RUL: a declarative update language for RDF. In *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, Galway, Ireland, 506–521.

Maynard, D., Funk, A. & Peters, W. 2009. SPRAT: a tool for automatic semantic pattern based ontology population. In *International Conference for Digital Libraries and the Semantic Web (ICSD-09)*, Trento, Italy.

Maynard, D., Peters, W., d'Aquin, M. & Sabou, M. 2007. Change management for metadata evolution. In *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, Innsbruck, Austria, 27–40.

Meyer, T., Lee, K. & Booth, R. 2005. Knowledge integration for Description Logics. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, USA, 645–650.

Moguillansky, M., Rotstein, N. & Falappa, M. 2008. A theoretical model to handle ontology debugging and change through argumentation. In *Proceedings of the 2nd International Workshop on Ontology Dynamics (IWOD-08)*, Karlsruhe, Germany.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Motik, B., Horrocks, I. & Sattler, U. 2007. Bridging the gap between OWL and relational databases. In *Proceedings of 17th International World Wide Web Conference (WWW-07)*, Banff, Canada, 807–816.
Novacek, V., Laera, L. & Handschuh, S. 2007. Semi-automatic integration of learned ontologies into a collaborative framework. In *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, Innsbruck, Austria.
Noy, N., Chugh, A., Liu, W. & Musen, M. 2006. A framework for ontology evolution in collaborative environments. In *Proceedings of the 5th International Semantic Web Conference (ISWC-06)*, Athens, GA, USA, 544–558.
Noy, N., Fergerson, R. & Musen, M. 2000. The knowledge model of Protégé-2000: combining interoperability and flexibility. In *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management: Methods, Models, and Tools (EKAW-00)*, Sophia Antipolis, France, 17–32.
Noy, N. & Klein, M. 2004. Ontology evolution: not the same as schema evolution. *Knowledge and Information Systems* **6**(4), 428–440.
Noy, N., Kunnatur, S., Klein, M. & Musen, M. 2004. Tracking changes during ontology evolution. In *Proceedings of the 3rd International Semantic Web Conference (ISWC-04)*, 259–273.
Noy, N. & Musen, M. 2002. PromptDiff: a fixed-point algorithm for comparing ontology versions. In *Proceedings of 18th National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada.
Noy, N. & Musen, M. 2004. Ontology versioning in an ontology management framework. *IEEE Intelligent Systems* **19**(4), 6–13.
Obst, D. & Chan, C. 2005. Towards a framework for ontology evolution. In *Electrical and Computer Engineering, 2005. Canadian Conference*, Saskatoon, Canada, 2191–2194.
Oliver, D., Shahar, Y., Shortliffe, E. & Musen, M. 1999. Representation of change in controlled medical terminologies. *Artificial Intelligence in Medicine* **15**(1), 53–76.
Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H. & Tummarello, G. 2008. Sindice.com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies (IJMSO-08)* **3**(1), 37–52.
Ottens, K., Gleizes, M. & Glize, P. 2007. A multi-agent system for building dynamic ontologies. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM, Honolulu, Hawaii.
Ottens, K., Hernandez, N., Gleizes, M. & Aussenac-Gilles, N. 2009. A multi-agent system for dynamic ontologies. *Journal of Logic and Computation* **19**(5), 831–858.
Palma, A., Haase, P., Wang, Y. & d'Aquin, M. 2007. *D1.3.1: Propagation Models and Strategies*. Technical report, NeOn Deliverable D1.3.1.
Palmisano, I., Tamma, V., Iannone, L., Payne, T. & Doran, P. 2008. Dynamic change evaluation for ontology evolution in the semantic web. In *Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 1, Sydney, Australia, 34–40.
Pammer, V., Ghidini, C., Rospocher, M., Serafini, L. & Lindstaedt, S. 2010. Automatic support for formative ontology evaluation. In *Poster Proceedings of the Conference on Knowledge Engineering and Knowledge Management by the Masses (EKAW-10)*. Lisbon, Portugal.
Pammer, V., Serafini, L. & Lindstaedt, M. 2009. Highlighting assertional effects of ontology editing activities in OWL. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09)*, Chantilly, VA, USA.
Papastefanatos, G., Vassiliadis, P., Simitsis, A. & Vassiliou, Y. 2009. Policy-regulated management of ETL evolution. *Journal of Data Semantics* **13**, 147–177.
Papastefanatos, G., Vassiliadis, P., Simitsis, A. & Vassiliou, Y. 2010. HECATAEUS: regulating schema evolution. In *Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE-10)*, Long Beach, CA, USA, 1181–1184.
Papavassiliou, V., Flouris, G., Fundulaki, I., Kotzinos, D. & Christophides, V. 2009. On detecting high-level changes in RDF/S KBs. In *Proceedings of the 8th International Semantic Web Conference (ISWC-09)*, Chantilly, VA, USA.
Papavassiliou, V., Flouris, G., Fundulaki, I., Kotzinos, D. & Christophides, V. 2013. High-level change detection in RDF(S) KBs. *Transactions on Database Systems* **38**(1), 1–42.
Pasca, M. & Harabagiu, S. 2001. The informative role of WordNet in open-domain question answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA, USA, 138–143.
Plessers, P. & de Troyer, O. 2005. Ontology change detection using a version log. In *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*.
Plessers, P., de Troyer, O. & Casteleyn, S. 2005. Event-based modeling of evolution for semantic-driven systems. In *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE-05)*, Porto, Portugal, 63–76.
Plessers, P., de Troyer, O. & Casteleyn, S. 2007. Understanding ontology evolution: A change detection approach. *Web Semantics* **5**(1), 39–49.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Pruski, C., Guelfi, N. & Reynaud, C. 2011. Adaptive ontology-based web information retrieval: the target framework. *International Journal of Web Portals* **3**(3), 41–58.

Qin, L. & Atluri, V. 2009. Evaluating the validity of data instances against ontology evolution over the semantic web. *Information and Software Technology* **51**(1), 83–97.

Qi, G. & Du, J. 2009. Model-based revision operators for terminologies in Description Logics. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, CA, USA, 891–897.

Qi, G., Liu, W. & Bell, D. 2006a. Knowledge base revision in Description Logics. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA-06)*, Liverpool, UK.

Qi, G., Liu, W. & Bell, D. 2006b. A revision-based approach for handling inconsistency in Description Logics. In *Proceedings of the 11th International Workshop on Non-Monotonic Reasoning (NMR-06)*, Lake District, UK.

Redmond, T., Smith, M., Drummond, N. & Tudorache, T. 2008. Managing change: an ontology version control system. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED-08)*, Karlsruhe, Germany.

Ren, Y. & Pan, J. 2011. Optimising ontology stream reasoning with truth maintenance system. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM-10)*, Toronto, Canada.

Ribeiro, M. & Wassermann, R. 2006. First steps towards revising ontologies. In *Proceedings of the 2nd Workshop on Ontologies and Their Applications (WONTO-06)*, Ribeirao Preto, Sao Paulo, Brazil.

Ribeiro, M. & Wassermann, R. 2007. Base revision in Description Logics—preliminary results. In *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, Innsbruck, Austria, 69–82.

Ribeiro, M., Wassermann, R., Antoniou, G., Flouris, G. & Pan, J. 2009. Belief contraction in web-ontology languages. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09), Short Paper*, Chantilly, VA, USA.

Riess, C., Heino, N., Tramp, S. & Auer, S. 2010. EvoPat—pattern-based evolution and refactoring of RDF knowledge bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC-10)*, Shanghai, China.

Roger, M., Simonet, A. & Simonet, M. 2002. Toward updates in Description Logics. In *Proceedings of the 9th International Workshop on Knowledge Representation Meets Databases (KRDB-02)*, Toulouse, France.

Rogozan, D. & Paquette, G. 2005. Managing ontology changes on the semantic web. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI-05)*, Compiegne, France, 430–433.

Sabou, M., d'Aquin, M. & Motta, E. 2008. Exploring the semantic web as background knowledge for ontology matching. *Journal on Data Semantics* **XI**, 156–190.

Sabou, M., Fernandez, M. & Motta, E. 2009. Evaluating semantic relations by exploring ontologies on the semantic web. In *Proceedings of the 14th International Conference on Applications of Natural Language to Information Systems (NLDB-09)*, Saarbrucken, Germany.

Serfiotis, G., Koffina, I., Christophides, V. & Tannen, V. 2005. Containment and minimization of RDF/S query patterns. In *Proceedings of the 4th International Semantic Web Conference (ISWC-05)*, Galway, Ireland, 607–623.

Shadbolt, N., Hall, W. & Berners-Lee, T. 2006. The semantic web revisited. *IEEE Intelligent Systems* **21**(3), 96–101.

Sirin, E., Parsia, B., Grau, B., Kalyanpur, A. & Katz, Y. 2007. Pellet: a practical OWL-DL reasoner. *Web Semantics* **5**(2), 51–53.

Stojanovic, L. 2004. *Methods and Tools for Ontology Evolution*. PhD thesis, FZI—Research Center for Information Technologies at the University of Karslruhe.

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

Stojanovic, L. & Motik, B. 2002. Ontology evolution within ontology editors. In *Proceedings of the OntoWeb-SIG3 Workshop*, Siguenza, Spain, 53–62.

Stuckenschmidt, H. & Klein, M. 2003. Integrity and change in modular ontologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico.

Studer, R., Benjamins, V. R. & Fensel, D. 1998. Knowledge engineering: principles and methods. *Data & Knowledge Engineering* **25**(1–2), 161–197.

Sure, Y., Angele, J. & Staab, S. 2003. OntoEdit: multifaceted inferencing for ontology engineering. *Journal on Data Semantics* **1**(1), 128–152.

Tamma, V. & Bench-Capon, T. 2001. A conceptual model to facilitate knowledge sharing in multi-agent systems. In *Proceedings of the Workshop on Ontologies in Agent Systems (OAS-01)*, Montreal, Canada, 69–76.

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press

Tao, J., Sirin, E., Bao, J. & McGuinness, D. 2010. Extending OWL with integrity constraints. In *Proceedings of the 23rd International Workshop on Description Logics (DL-10)*. CEUR-WS 573, Waterloo, Canada.

Thor, A., Hartung, M., Gross, A., Kirsten, T. & Rahm, E. 2009. An evolution based approach for assessing ontology mappings—a case study in the life sciences. In *Datenbanksysteme in Business, Technologie und Web (BTW)*, Munster, Germany, 277–286.

Velardi, P., Fabriani, P. & Missikoff, M. 2001. Using text processing techniques to automatically enrich a domain ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, Ogunquit, ME, USA, 270–284.

Volkel, M., Winkler, W., Sure, Y., Kruk, S. & Synak, M. 2005. SemVersion: a versioning system for RDF and ontologies. In *Proceedings of the 2nd European Semantic Web Conference (ESWC-05)*, Heraklion, Greece.

Vrandecic, D., Pinto, H. S., Sure, Y. & Tempich, C. 2005. The DILIGENT knowledge processes. *Journal of Knowledge Management* **9**(5), 85–96.

Wang, Y., Liu, X. & Ye, R. 2008. Ontology evolution issues in adaptable information management systems. In *Proceedings of the 2008 IEEE International Conference on e-Business Engineering (ICEBE-08)*. IEEE Computer Society, Xian, China, 753–758.

Wang, Z., Wang, K. & Topor, R. 2010. A new approach to knowledge base revision in DL-Lite. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, Georgia, USA.

Xuan, D., Bellatreche, L. & Pierra, G. 2006. A versioning management model for ontology-based data warehouses. In *Data Warehousing and Knowledge Discovery*, vol. 4081. Springer, 195–206.

Zablith, F. 2009. Evolva: a comprehensive approach to ontology evolution. In *Proceedings of the PhD Symposium of the 6th European Semantic Web Conference (ESWC-09)*, Heraklion, Greece, 944–948.

Zablith, F., d'Aquin, M., Sabou, M. & Motta, E. 2010. Using ontological contexts to assess the relevance of statements in ontology evolution. In *Proceedings of Knowledge Engineering and Knowledge Management by the Masses (EKAW-10)*, Springer-Verlag.

Zablith, F., Sabou, M., d'Aquin, M. & Motta, E. 2008. Using background knowledge for ontology evolution. In *Proceedings of the 2nd International Workshop on Ontology Dynamics (IWOD-08)*, Karlsruhe, Germany.

Zeginis, D., Tzitzikas, Y. & Christophides, V. 2007. On the foundations of computing deltas between RDF models. In *Proceedings of the 6th International Semantic Web Conference (ISWC-07)*, Busan, Korea.

Zeginis, D., Tzitzikas, Y. & Christophides, V. 2011. On computing deltas of RDF/S knowledge bases. *ACM Transactions on the Web* **5**(3).

https://doi.org/10.1017/S0269888913000349 Published online by Cambridge University Press