Supporting Concurrent Ontology Development: Framework, Algorithms and Tool

Author1[a], Author2[b], Author3[b], Author4[a]

[a]*Institution1.*
Phone:
[b]*Institution2.*
Phone:

Abstract

We propose a novel approach to facilitate the concurrent development of ontologies by different groups of experts. Our approach adapts Concurrent Versioning, a successful paradigm in software development, to allow several developers to make changes concurrently to an ontology. Conflict detection and resolution are based on novel techniques that take into account the structure and semantics of the ontology versions to be reconciled by using precisely-defined notions of structural and semantic differences between ontologies and by extending state-of-the-art ontology debugging and repair techniques. We also present ContentCVS, a system that implements our approach, and a preliminary empirical evaluation which suggests that our approach is both computationally feasible and useful in practice.

Key words: Ontologies, Knowledge Representation, Knowledge Engineering, OWL, Semantic Web

1. Introduction

The Web Ontology Language (OWL), and its revision OWL 2, are well-known languages for ontology modeling in the Semantic Web [8, 26].

OWL ontologies are already being used in domains as diverse as bio-medicine, geology, agriculture, and defence. In particular, OWL is extensively used in the

Email addresses: Mail1 (Author1), Mail2 (Author2), Mail3 (Author3), Mail4 (Author4)

clinical sciences; prominent examples of OWL ontologies are the National Cancer Institute (NCI) Thesaurus [19], the Systematised Nomenclature of Medicine and Clinical Terms (SNOMED CT) [52], the Gene Ontology (GO) [1], the Foundational Model of Anatomy (FMA) [40], and GALEN [45].

These ontologies are large and complex; for example, SNOMED CT currently describes more than 300,000 concepts whereas NCI and FMA describe around 70,000 concepts. Furthermore, these ontologies are in continuous evolution [20]. For example, the developers of NCI perform approximately 900 changes to the ontology each month.

Most realistic ontologies, including the ones just mentioned, are being developed by several groups, which can be geographically distributed and may contribute in different ways and to different extents. For example, the NCI ontology is being developed by 20 full time editors and one curator. Each editor works on a separate copy of the ontology and, at the end of a two week editing cycle, the curator uses a workflow management tool to review and approve the changes made by each editor [9]. Similarly, the SNOMED CT development team consists of a main team and four geographically distributed groups that are in charge of the development of different parts of the ontology; every two weeks, the parts developed by the different groups are integrated and the possible conflicts are reconciled by the main team.

Therefore, designing and maintaining such large ontologies is a highly complex process, which involves many complicated tasks:

1. to agree on a unified conceptual design and common modeling guidelines;
2. to assign different responsibilities and tasks to each group of developers;
3. to track and manage the frequent changes to the ontology made by different developers from distributed locations;
4. to compare different versions of the ontology lexically (e.g., names of the introduced ontology entities), structurally (e.g., shape of the axioms), and semantically (e.g., logical consequences);
5. to detect and reconcile conflicting views of the domain by merging different ontology versions; and
6. to minimise the introduction of errors (e.g., to ensure that the ontology does not have unintended logical consequences).

In recent years, there has been a growing interest in the development of techniques and tools to support ontology developers in performing these tasks (see for example [4, 53, 57, 11, 21, 51, 39, 13, 46, 49, 18]), and we refer the reader to our Related Work section for a detailed discussion.

In this paper, we present a novel approach to facilitate the *concurrent development* of ontologies. Our approach adapts Concurrent Versioning —a successful paradigm in software development— to allow several developers to make changes concurrently and remotely to the same ontology, track changes, and manage ontology versions. Version comparison, conflict detection and conflict resolution are based on a novel approach that takes into account the structure and semantics of the versions to be reconciled. First, we propose precise notions of structural and semantic difference between ontology versions to facilitate the detection of potential conflicts and errors; second, we propose various improvements to state-of-the-art ontology debugging and repair techniques [34, 47, 35, 33, 25, 23] in order to fix the identified errors in a concurrent setting. We then present ContentCVS, a system that implements our approach, and a preliminary empirical evaluation which suggests that our approach is both computationally feasible and useful in practice.

Our contribution hence focuses on facilitating tracking and management of concurrent changes, version comparison, conflict identification, (semi)automatic conflict resolution, and version merging (tasks 3-6 above). We do not address in this work other important aspects of collaborative ontology development, such as conceptual design, distribution of responsibilities and tasks among developers, specification of modeling guidelines, and so on. Furthermore, concerning conflict detection and resolution, we see our techniques as *complementary* to other conflict resolution techniques that are more focused on collaborative and social aspects [17, 14, 13, 11] (e.g., on facilitating the discussion between groups of developers, or on achieving consensus). For example, discussion threads and annotations could be used in ContentCVS to inform other users of the reasons for certain changes. Finally, we do not consider the automatic resolution of conflicts that are merely *lexical* (e.g., two different developers adding the same concept independently, but using different names), which is an important and difficult issue especially in the development of inter-organisational ontologies [21]. Although such conflicts would be detected by ContentCVS and reported to the relevant groups of users, their automatic resolution would require, for example, the application of ontology matching techniques [3] or the use of a reference thesaurus [32].

We believe that a complete, multi-user platform for ontology development should provide *both* collaborative and social features, as well as those presented here and implemented in ContentCVS. The design and implementation of such an integrated platform is beyond the scope of this paper. We are planning, however, to progressively incorporate collaborative features into ContentCVS.

This paper extends the results of preliminary workshop publications on concurrent ontology development [28, 29], as well as a prior work on conflict resolution in the context of ontology mapping [30].

2. Preliminaries

The formal underpinning of OWL DL and OWL 2 is provided by description logics (DLs) [2]—knowledge representation formalisms with well-understood formal properties. In this section, we very briefly summarise the basics of DLs, and refer the interested reader to [2, 8, 26] for further information.

DLs allow ontology developers to describe a domain of interest in terms of *individuals*, *atomic concepts* (usually called *classes* in OWL), and *roles* (also called *properties*). DLs also allow for *concept descriptions* (i.e., complex concepts) to be composed of atomic concepts and roles by providing a set of *concept constructors*. The DLs underlying OWL provide for intersection ($\sqcap$), union ($\sqcup$) and complement ($\neg$), as well as enumerated classes (called *oneOf* in OWL) and restricted forms of existential ($\exists$), universal ($\forall$) and cardinality restrictions ($\geq, \leq, =$) involving an atomic role R or its inverse R^{-}. A DL ontology $\mathcal{O}$ consists of a set of axioms. In the DLs underlying OWL it is possible to assert that a concept (or concept description) C is subsumed by (is a sub-concept of) D (written $C \sqsubseteq D$), or is exactly equivalent to D (written $C \equiv D$). It is also possible to assert subsumption of and equivalence between roles as well as to establish special constraints on roles (e.g., that a role should be interpreted as a transitive or as a functional relation).

A (fragment of) a DL ontology about arthritis that we will use as a running example is given in Table 1, where RA stands for 'Rheumatoid Arthritis' and RF for 'Rheumatoid Factor'. For example, axiom α_2 states that every systemic disease is a disease that affects the whole body.

DLs are equipped with a formal semantics, which enables the development of reasoning algorithms for answering complex queries about the domain. DLs, in fact, can be seen as decidable subsets of first-order logic, with individuals being equivalent to constants, concepts to unary predicates and roles to binary predicates. As in the case of a first order knowledge base, an interpretation I is a model of an ontology $\mathcal{O}$ (written $I \models \mathcal{O}$) if I satisfies all the axioms in $\mathcal{O}$; $\mathcal{O}$ entails an axiom α (respectively an ontology $\mathcal{O}'$), written $\mathcal{O} \models \alpha$ ($\mathcal{O} \models \mathcal{O}'$), if $I \models \alpha$ (respectively $I \models \mathcal{O}'$) for every model I of $\mathcal{O}$. Finally $\mathcal{O}$ and $\mathcal{O}'$ are logically equivalent (written $\mathcal{O} \equiv \mathcal{O}'$) if $\mathcal{O} \models \mathcal{O}'$ and $\mathcal{O}' \models \mathcal{O}$.

	Ontology $\mathcal{O}^0$
α_1	RA $\sqsubseteq$ Disease
α_2	Systemic_Disease $\sqsubseteq$ Disease $\sqcap \exists$affects.WholeBody
α_3	Disease $\sqcap \exists$affects.WholeBody $\sqsubseteq$ Systemic_Disease
α_4	Disease $\sqcap \exists$suffered_By.Child $\sqsubseteq$ Juvenile_Disease
α_5	Negative_RF $\sqcap$ Positive_RF $\sqsubseteq \bot$
α_6	AbnormalRA $\sqsubseteq$ RA $\sqcap \forall$hasRF.Negative_RF
α_7	MultiJoint_Disease $\sqsubseteq$ Disease $\sqcap \geqslant 5$ affects.Joint

Table 1: A fragment of an ontology about arthritis

3. Practical challenges

In this section, we introduce some of the challenges that ontology engineers face when developing ontologies concurrently. To this end, we consider as an example the development of an ontology about arthritis. This example is intended only for illustration purposes and hence it is rather simplistic. The types of conflicting ontology changes illustrated in this section, however, are indeed realistic as we will see later on in Section 7.1, where we analyse a sequence of versions of a medical ontology used in a real scenario.

Suppose that two developers, John and Anna, are in charge of extending a version $\mathcal{O}^0$ of the ontology in Table 1 by describing types of systemic arthritis and juvenile arthritis, respectively. To this end, both John and Anna define a kind of arthritis called JRA (Juvenile Rheumatoid Arthritis), which is both systemic and juvenile. Hence, even if largely distinct, the domains described by John and Anna overlap, which may lead to conflicts. For simplicity, in what follows we only consider John and Anna's descriptions of JRA.

Suppose that John and Anna construct their respective versions $\mathcal{O}^1$ and $\mathcal{O}^2$ by adding to $\mathcal{O}^0$ the axioms $(\Delta\mathcal{O})^1$ and $(\Delta\mathcal{O})^2$ from Table 2 (i.e., $\mathcal{O}^1 = \mathcal{O}^0 \cup (\Delta\mathcal{O})^1$ and $\mathcal{O}^2 = \mathcal{O}^0 \cup (\Delta\mathcal{O})^2$). Some of the axioms added by John and Anna are the same (e.g., β_1), or present only minor (and semantically irrelevant) differences (e.g., β_2 and β_2'); however, other axioms are clearly different (e.g., γ_3 and δ_3).

To compare John and Anna's conceptualisations of the domain, the upper part of Table 3 presents some axioms and their entailment status w.r.t. $\mathcal{O}^1$ and $\mathcal{O}^2$. The table shows that John and Anna agree on some points; e.g., both think that Polyarticular JRA is a kind of disease (entailment σ_3) and neither claimed that every JRA is also a Systemic JRA (non-entailment σ_4). However, John's and Anna's views also present significant differences; e.g., John defined the notion of

	Ontology $(\Delta\mathcal{O})^1$**:**		**Ontology** $(\Delta\mathcal{O})^2$**:**
β_1	RA $\sqsubseteq$ $\exists$hasRF.$\top$		RA $\sqsubseteq$ $\exists$hasRF.$\top$
β_2	JRA $\sqsubseteq$ $\exists$treatment.(Steroid $\sqcup$ DMAR)	β_2'	JRA $\sqsubseteq$ $\exists$treatment.(DMAR $\sqcup$ Steroid)
γ_1	JRA $\sqsubseteq$ RA $\sqcap$ Systemic_Disease	δ_1	JRA $\sqsubseteq$ RA $\sqcap$ $\exists$suffered_By.Child
γ_2	RA $\sqcap$ Systemic_Disease $\sqsubseteq$ JRA	δ_2	JRA $\sqcap$ $\exists$affects.WholeBody $\sqsubseteq$ SystemicJRA
γ_3	Poly_JRA $\sqsubseteq$ JRA $\sqcap$ MultiJoint_Disease	δ_3	Poly_JRA $\sqsubseteq$ JRA $\sqcap = 3$ affects.Joint
γ_4	Poly_JRA $\sqsubseteq$ AbnormalRA	δ_4	Poly_JRA $\sqsubseteq$ $\forall$hasRF.Positive_RF
γ_5	Oly_JRA $\sqsubseteq$ JRA $\sqcap$ $\neg$Poly_JRA	δ_5	SystemicJRA $\sqsubseteq$ JRA $\sqcap$ $\exists$hasSymptom.Fever

Table 2: Versions $\mathcal{O}^1 = \mathcal{O}^0 \cup (\Delta\mathcal{O})^1$ and $\mathcal{O}^2 = \mathcal{O}^0 \cup (\Delta\mathcal{O})^2$ of an ontology $\mathcal{O}^0$

$\boldsymbol{\sigma}$	**Axiom:**	$\mathcal{O}^1 \models^? \sigma$	**follows from:**	$\mathcal{O}^2 \models^? \sigma$	**follows from:**
σ_1	Oly_JRA $\sqsubseteq$ Systemic_Disease	Yes	γ_1, γ_5	No	—
σ_2	JRA $\sqsubseteq$ Juvenile_Disease	No	—	Yes	$\alpha_1, \alpha_4, \delta_1$
σ_3	Poly_JRA $\sqsubseteq$ Disease	Yes	$\alpha_1, \gamma_1, \gamma_3$	Yes	$\alpha_1, \delta_1, \delta_3$
σ_4	JRA $\sqsubseteq$ SystemicJRA	No	—	No	—

$\boldsymbol{\sigma}$	**Axiom:**	$\mathcal{O}^3 \models^? \sigma$	$\mathcal{O}^1 \models^? \sigma$	$\mathcal{O}^2 \models^? \sigma$	**follows from:**	**Desirable?**
σ_4	JRA $\sqsubseteq$ SystemicJRA	Yes	No	No	$\gamma_1, \alpha_2, \delta_2$	No
σ_5	Poly_JRA $\sqsubseteq \bot$	Yes	No	No	$\gamma_4, \beta_1, \delta_4, \alpha_5, \alpha_6, \alpha_7, \gamma_3, \delta_3$	No
σ_6	Oly_JRA $\sqsubseteq$ Juvenile_Disease	Yes	No	No	$\gamma_5, \alpha_1, \alpha_4, \delta_1$	Yes

Table 3: Example Subsumption Relations in $\mathcal{O}^1$, $\mathcal{O}^2$, and $\mathcal{O}^3 = \mathcal{O}^1 \cup \mathcal{O}^2$

Olyarticular JRA , whereas Anna did not, and Anna's conceptualisation implies that JRA is a juvenile disease (entailment σ_2), whereas John's does not.

John and Anna's changes could be reconciled by building the union $\mathcal{O}^3 = \mathcal{O}^1 \cup \mathcal{O}^2$ of their ontologies. Due to complex interactions between $\mathcal{O}^1$ and $\mathcal{O}^2$, however, $\mathcal{O}^3$ entails new consequences which did not follow from either $\mathcal{O}^1$ or $\mathcal{O}^2$ alone; some of these are shown in the lower part of Table 3, together with an indication as to whether the consequence is desirable. Although some of these new consequences may be desirable (e.g., σ_6) others are clearly undesirable, and indicate modelling errors in the merged ontology (e.g., σ_4 and σ_5).

This example illustrates some of the challenges of concurrent ontology development. The development of an ontology may be the responsibility of several developers, each of whom typically makes small but relatively frequent modifications to the ontology. In this setting, developers need to regularly merge

and reconcile their modifications to ensure that the ontology captures a consistent unified view of the domain. The changes performed by different users may, however, interact and conflict in complex ways. Development tools should therefore provide means for: *(i)* keeping track of ontology versions and changes and reverting, if necessary, to a previously agreed upon version, *(ii)* comparing potentially conflicting versions and identifying conflicting parts, *(iii)* identifying errors in the reconciled ontology constructed from the conflicting versions, and *(iv)* suggesting possible ways to repair the identified errors with a minimal impact on the ontology.

To address *(i)*, we propose to adapt the Concurrent Versioning paradigm to ontology development as described in Section 4. To address *(ii)* we propose a notion of *conflict* between ontology versions and provide means for identifying conflicting parts based on it, as described in Section 5. To address *(iii)* we propose in Section 6 a framework for comparing the entailments that hold in the compared versions and in the reconciled ontology, based on the notion of a *deductive difference* [36] and also describe techniques for helping users decide which of the reported entailments are intended. Finally, to address *(iv)*, we propose in Section 6 several improvements to existing techniques for ontology debugging and repair and adapt them to our new setting.

In Sections 4, 5, and 6, we describe both our general approach and algorithmic techniques as well as their implementation in our tool ContentCVS,[1] a Protégé 4 plugin freely available for download. [2]

4. CVS-based concurrent development

In software engineering, a successful paradigm for collaboration in large projects has been the Concurrent Versioning Paradigm. A Concurrent Versioning System (CVS) uses a client-server architecture: a CVS server stores the current version of a project and its change history; CVS clients connect to the server to *check out* a copy of the project, allowing developers to work on their own 'local' copy, and then later to *commit* their changes to the server. This allows several developers to make changes concurrently to a project. To keep the system in a consistent state, the server only accepts changes to the latest version of any given project file. Developers should hence use the CVS client to regularly commit their changes and *update* their local copy with changes made by others. Manual intervention is only needed when a *conflict* arises between a committed

[1] A **C**oncurrent **ONT**ology **EN**gineering **T**ool.

[2] URL removed for review

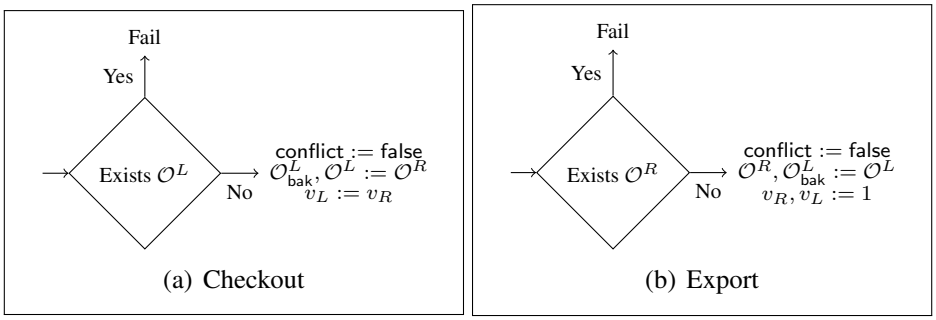


Figure 1: Semantics of the checkout and export operations in ContentCVS

version in the server and a yet-uncommitted local version. Conflicts are reported whenever the two compared versions of a file are not *equivalent* according to a given notion of equivalence between versions of a file.

Our tool ContentCVS closely follows the CVS paradigm. The most recent version $\mathcal{O}^R$ of the ontology, which represents the developers' shared understanding of the domain, is kept in a server's shared repository. Each developer with access to the repository maintains a local copy $\mathcal{O}^L$ of the ontology, which can be modified at will. This local copy can be either in conflict with $\mathcal{O}^R$ ($\mathsf{conflict} = \mathsf{true}$) or not in conflict ($\mathsf{conflict} = \mathsf{false}$). Furthermore, the system maintains version numbers v_R and v_L for $\mathcal{O}^R$ and $\mathcal{O}^L$ respectively as well a local 'backup' copy $\mathcal{O}^L_{\mathsf{bak}}$ of the latest local version that was 'written' to the repository.

At any time, a developer can access the repository using one of the following basic operations: *export*, *check-out*, *update* and *commit*. These operations involve checking whether two ontology files $\mathcal{O}$ and $\mathcal{O}'$ are 'equivalent' under a specific notion of equivalence between ontology files which will be introduced in Section 5 (denoted $\mathcal{O} \sim \mathcal{O}'$).

The *checkout* operation (Figure 1(a)) allows a developer to acquire a local copy $\mathcal{O}^L$ of $\mathcal{O}^R$, provided that $\mathcal{O}^L$ does not already exist. The ontology resulting from a successful checkout is obviously in a non-conflicted state (i.e., $\mathsf{conflict} = \mathsf{false}$), and it inherits the version number v_R of $\mathcal{O}^R$.

The *export* operation (Figure 1(b)) allows a developer to create a new repository, provided that none already exists. The local ontology is then 'exported' to the repository and the version numbers of $\mathcal{O}^L$ and $\mathcal{O}^R$ are initialised.

The *update* operation (Figure 2) allows developers to keep their local copy $\mathcal{O}^L$ up-to-date by accessing the repository and incorporating the changes made by others. The update process starts by checking whether $\mathcal{O}^L$ has not changed since it was last updated; in case $\mathcal{O}^L$ has changed, it next checks whether the changes made by others are consistent with those made locally. In either case,

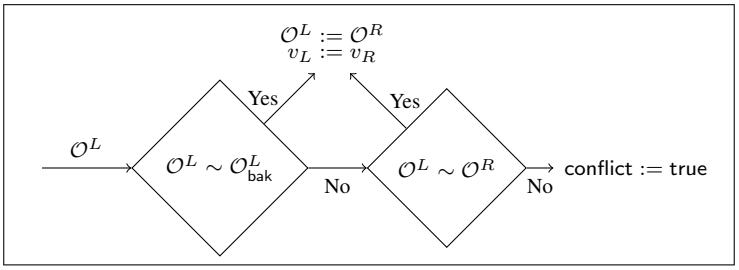


Figure 2: Semantics of the update operation in ContentCVS

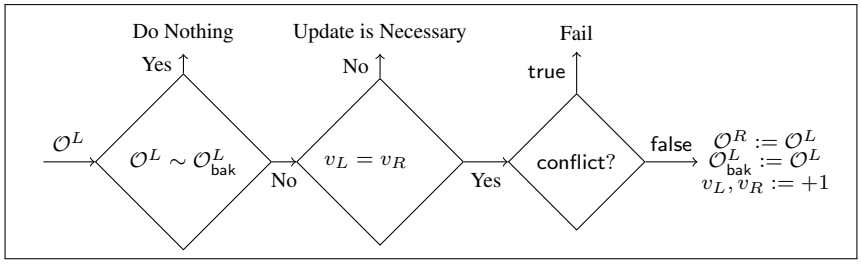


Figure 3: Semantics of the commit operation in ContentCVS

it is safe to replace $\mathcal{O}^L$ with the version $\mathcal{O}^R$ from the repository. Otherwise, a conflict is reported.

Finally, the *commit* operation (Figure 3), allows ontology developers to write (commit) their local changes to the repository. If $\mathcal{O}^L_{\mathsf{bak}} \backsim \mathcal{O}^L$ then there are no meaningful local changes and hence no action is required. Otherwise, the commit process only succeeds if $\mathcal{O}^L$ is up-to-date ($v_L = v_R$) and not in conflict ($\mathsf{conflict} = \mathsf{false}$). In case of success, the commit operation involves replacing $\mathcal{O}^R$ with $\mathcal{O}^L$ and incrementing the version number.

Consider our running example and suppose that Anna has already committed her changes, so $\mathcal{O}^R = \mathcal{O}^2$; meanwhile, John modifies his local copy, so $\mathcal{O}^L = \mathcal{O}^1$. If John then tries to commit his changes, the operation will fail because $v_L \neq v_R$ (the local copy is not up-to-date); if he tries to update his local copy, the operation will fail because there have been local changes and they are incompatible with those made by Anna, and $\mathcal{O}^L$ ends up in a conflicted state. Conflicts will need to be resolved before the commit operation can succeed.

5. Change and conflict detection

As mentioned in Section 4, change and conflict detection amounts to checking whether two compared versions of a file are not 'equivalent' according to a given notion of equivalence between versions of a file.

A typical CVS treats the files in a software project as 'ordinary' text files and hence checking equivalence amounts to determining whether the two versions are syntactically equal (i.e., they contain exactly the same characters in exactly the same order). This notion of equivalence is, however, too strict in the case of ontologies, since OWL files have very specific structure and semantics. For example, if two OWL files are identical except for the fact that two axioms appear in different order, the corresponding ontologies should be clearly treated as 'equivalent': an ontology contains a *set* of axioms and hence their order is irrelevant [8].

Another possibility is to use logical equivalence as defined in Section 2. This notion of equivalence is, however, too permissive: even if $\mathcal{O} \equiv \mathcal{O}'$—the strongest assumption from a semantic point of view—conflicts may still exist. This might result from the presence of incompatible annotations (statements that act as comments and do not carry logical meaning) [8], or a mismatch in modelling styles; for example, $\mathcal{O}$ may be written in a simple language such as the OWL 2 EL profile [8, 41] and contain $\alpha := (A \sqsubseteq B \sqcap C)$, while $\mathcal{O}'$ may contain $\beta := (\neg B \sqcup \neg C \sqsubseteq \neg A)$. Even though $\alpha \equiv \beta$, the explicit use of negation and disjunction means that $\mathcal{O}'$ is outside the EL profile.

Therefore, the notion of a conflict should be based on a notion of ontology equivalence 'in-between' syntactical equality and logical equivalence. We propose to borrow the notion of *structural equivalence* from the OWL 2 specification [42]. Intuitively, this notion of equivalence is based solely on comparing structures by using the definition of the modeling constructs available in OWL and OWL 2; for example, several modeling constructs are defined as sets of objects (e.g., ontologies are defined as sets of axioms, conjunction of concepts as a set of conjuncts, and so on); hence changes in the order in which these set elements appear in the ontology file should be seen as irrelevant.

In DL syntax, structural equivalence can be formalised as given next. For the sake of simplicity, our definition here comprises only the description logic $\mathcal{SROIQ}$, which provides the logical underpinning for OWL 2. This definition can be easily extended to cover also datatypes and extra-logical statements, such as annotations. We refer the reader to [42] for a complete characterisation of structural equivalence.

Definition 1. *The structural equivalence relation $\backsim$ over a set of concepts* $\mathbf{Con}$ *is defined by induction. First, $C \backsim C$ for each $C \in \mathbf{Con}$. For the induction step, we have:*

- *$C \backsim D$ implies $(\neg C) \backsim (\neg D)$;*

- $C \backsim D$ *implies* $\Diamond R.C \backsim \Diamond R.D$ *for* $\Diamond \in \{\exists, \forall, \geq n, \leq n, = n\}$*; and*
- $C_1 \backsim C_2$ *and* $D_1 \backsim D_2$ *implies* $(C_1 \bowtie D_1) \backsim (C_2 \bowtie D_2) \backsim (D_2 \bowtie C_2)$*, for* $\bowtie \in \{\sqcap, \sqcup\}$.

The relation $\backsim$ *is extended to axioms over a set of concepts* **Con** *and roles* **Rol** *as follows:* $\alpha \backsim \alpha$ *for each concept or role axiom* α *and, if* $C_1 \backsim C_2$ *and* $D_1 \backsim D_2$*, then* $(C_1 \sqsubseteq D_1) \backsim (C_2 \sqsubseteq D_2)$*, for* $C_i, D_i \in$ **Con**. *Finally* $\backsim$ *is extended to ontologies as follows:* $\mathcal{O} \backsim \mathcal{O}'$ *if, for every* $\alpha \in \mathcal{O}$ *(respectively* $\alpha \in \mathcal{O}'$*) there is a* $\beta \in \mathcal{O}'$ *(respectively* $\beta \in \mathcal{O}$*) such that* $\alpha \backsim \beta$.

For example, the axioms β_2 and β_2' in Table 2 are structurally equivalent because they only differ in the order of the elements in a disjunction. If $\mathcal{O} \backsim \mathcal{O}'$ we can safely assume that they are compatible and thus not in conflict.

The use of the notion of structural equivalence for detecting conflicts between ontology versions presents a number of compelling advantages:

- It is a notion 'in-between' syntactical equality and logical equivalence: on the one hand irrelevant syntactic differences are ruled out as conflicts based solely on the structure of the OWL language; on the other hand, structurally equivalent ontologies are also logically equivalent.
- It preserves species and profiles [41] of OWL and OWL 2 respectively; for example if $\mathcal{O}$ and $\mathcal{O}'$ are structurally equivalent and $\mathcal{O}$ is in OWL Lite (respectively in any of the profiles of OWL 2), then $\mathcal{O}'$ is also in OWL Lite (respectively in the same OWL 2 profile as $\mathcal{O}$).
- It takes into account extra-logical components, such as annotations.
- It is an agreed-upon notion, defined after extensive discussions within the W3C OWL Working Group during the standardisation of OWL 2. Furthermore, it is not exclusive to OWL 2: it can be directly applied to OWL DL, and a similar notion could be devised for most other ontology languages.
- It is supported by mainstream ontology development APIs, such as the OWL API [22].

The identification of the conflicting parts in $\mathcal{O}$ and $\mathcal{O}'$ using the notion of structural equivalence can be performed by computing their *structural difference*.

Definition 2. *The structural difference between* $\mathcal{O}_1$ *and* $\mathcal{O}_2$ *is the set* Λ_s *of axioms* $\alpha \in \mathcal{O}_i$ *for which there is no* $\beta \in \mathcal{O}_j$ *s.t.* $\alpha \backsim \beta$ *with* $i, j \in \{1, 2\}$ *and* $i \neq j$.

ContentCVS reuses the functionality available in the OWL API for deciding structural equivalence and implements a straightforward algorithm for computing structural differences that follows directly from Definition 2.

6. Conflict resolution

Conflict resolution is the process of constructing a reconciled ontology from two conflicting ontology versions. In a CVS, the conflict resolution functionality is provided by the CVS client. Our approach is based on the principle that a CVS client should allow users to resolve conflicts at two different levels:

- *Structural*, where only the structure of the compared ontology versions is taken into account to build the reconciled ontology (see Section 6.1).
- *Structural and semantic*, where both the structure and the logical consequences of the compared ontology versions as well as of the reconciled ontology are taken into consideration (see Sections 6.1—6.4).

In the former case, the overhead involved in using a reasoner and examining its output is avoided; however, the reconciled ontology may contain errors (e.g., undesired logical consequences), which would remain undetected.

In the latter case, errors in the reconciliation process can be detected, with the assistance of a reasoner, by computing the logical consequences of the reconciled ontology and comparing them to those of the relevant ontology versions. Errors in the reconciled ontology could manifest themselves, however, not only as unsatisfiable concepts or unintended (or missing) subsumptions between atomic concepts, but also as unintended (or missing) entailments involving complex concepts. We propose to use the notion of *deductive difference* for error detection (see Section 6.2), which ensures that errors associated with complex entailments are also detected. However, considering complex entailments obviously comes at a price, both in terms of computational cost and of complication of the GUI. Thus, a CVS client should allow users to customise the types of relevant entailments for error detection and guide them in the selection process (see Section 6.2). Finally, error repair is a complicated process for which tool support should be provided, and our approach involves a number of techniques to achieve this goal (see Sections 6.3 and 6.4).

Our approach is summarised in Table 4. The steps marked with a tickmark (✓) are those that require human intervention. We next describe in detail each of the steps in Table 4.

6.1. Selection of axioms using structural difference

Conflict resolution in text files is usually performed by first identifying and displaying the conflicting sections in the two files (e.g., a line, or a paragraph) and then manually selecting the desired content. Analogously, our proposal for structural conflict resolution involves computing and displaying the structural

Input: $\mathcal{O}^L, \mathcal{O}^R$: ontologies with $\mathcal{O}^L \not\sim \mathcal{O}^R$, $\mathsf{conflict} = \mathsf{true}$ and structural difference Λ_s
Output: $\mathcal{O}^L$: ontology; $\mathsf{conflict}$: boolean value;

1: (✓) Select $\mathcal{S} \subseteq \Lambda_s$
2: $\mathcal{O}^L_{\mathsf{temp}} := (\mathcal{O}^L \setminus \Lambda_s) \cup \mathcal{S}$
3: (✓) **if** $\mathcal{O}^L_{\mathsf{temp}}$ is satisfactory **return** $\mathcal{O}^L := \mathcal{O}^L_{\mathsf{temp}}, \mathsf{conflict} := \mathsf{false}$
4: (✓) Select approximation function $\mathsf{diff}_{\approx}$
5: Compute $\mathsf{diff}_{\approx}(\mathcal{O}^L_{\mathsf{temp}}, \mathcal{O}^L)$, $\mathsf{diff}_{\approx}(\mathcal{O}^L_{\mathsf{temp}}, \mathcal{O}^R)$, $\mathsf{diff}_{\approx}(\mathcal{O}^L, \mathcal{O}^L_{\mathsf{temp}})$ and $\mathsf{diff}_{\approx}(\mathcal{O}^R, \mathcal{O}^L_{\mathsf{temp}})$
6: (✓) Select $\Im^+ \subseteq \mathsf{diff}_{\approx}(\mathcal{O}^L_{\mathsf{temp}}, \mathcal{O}^L) \cup \mathsf{diff}_{\approx}(\mathcal{O}^L_{\mathsf{temp}}, \mathcal{O}^R)$
7: (✓) Select $\Im^- \subseteq \mathsf{diff}_{\approx}(\mathcal{O}^L, \mathcal{O}^L_{\mathsf{temp}}) \cup \mathsf{diff}_{\approx}(\mathcal{O}^R, \mathcal{O}^L_{\mathsf{temp}})$
8: Compute minimal plans $\mathbb{P}$ for $\mathcal{O}^L_{\mathsf{temp}}$ given $\Im^+$, $\Im^-$, $\mathcal{O}^+ := \Lambda_s \setminus \mathcal{S}$, and $\mathcal{O}^- := \mathcal{S}$
9: (✓) **if** no satisfactory plan in $\mathbb{P}$, **return** $\mathcal{O}^L, \mathsf{conflict} := \mathsf{true}$
10: (✓) Select $\mathcal{P} = \langle \mathcal{P}^+, \mathcal{P}^- \rangle \in \mathbb{P}$
11: **return** $\mathcal{O}^L := (\mathcal{O}^L_{\mathsf{temp}} \cup \mathcal{P}^+) \setminus \mathcal{P}^-, \mathsf{conflict} := \mathsf{false}$

Table 4: Conflict Resolution Method.

difference Λ_s (i.e., those axioms for which a structurally equivalent axiom does not occur in both ontologies) and then manually selecting which of these axioms should be included in a (provisional) version $\mathcal{O}^L_{\mathsf{temp}}$ of $\mathcal{O}^L$ (Step 1). The ontology $\mathcal{O}^L_{\mathsf{temp}}$ is obtained from the non-conflicting part of $\mathcal{O}^L$ plus the selected axioms $\mathcal{S}$ from the conflicting part (Step 2).

After constructing $\mathcal{O}^L_{\mathsf{temp}}$, the user may declare the conflict resolved (Step 3), in which case conflict resolution remains a purely syntactic process, as in the case of text files. Otherwise, ontology developers can use a reasoner to examine the semantic consequences of their choices and make sure that $\mathcal{O}^L_{\mathsf{temp}}$ meets their requirements (typically, includes as much information as possible without leading to inconsistencies or other undesired entailments).

ContentCVS implements a simple GUI to facilitate the selection of axioms from the structural difference, which is shown in Figure 4 for our running example. The left-hand-side (respectively the right-hand-side) of the figure shows the axioms in $\Lambda_s \cap \mathcal{O}^L$ (respectively in $\Lambda_s \cap \mathcal{O}^R$).

To facilitate the comparison, axioms are sorted and aligned according to the entities they define. Axioms not aligned with others are shown last in a distinguished position. The selected axioms are indicated in the GUI using a highlighted tickmark (✓). Furthermore, ContentCVS provides additional functionality for determining the origin of each axiom in the structural difference. In particular, ContentCVS, indicates whether an axiom appears in the difference as a result of an addition or a deletion by comparing $\mathcal{O}^L$ and $\mathcal{O}^R$ to the local 'backup' copy $\mathcal{O}^L_{\mathsf{bak}}$ of the latest local version that was 'written' to the repository. For example, the axiom $(\mathsf{Poly\_JRA} \sqsubseteq \mathsf{AbnormalRA})$ on the left-

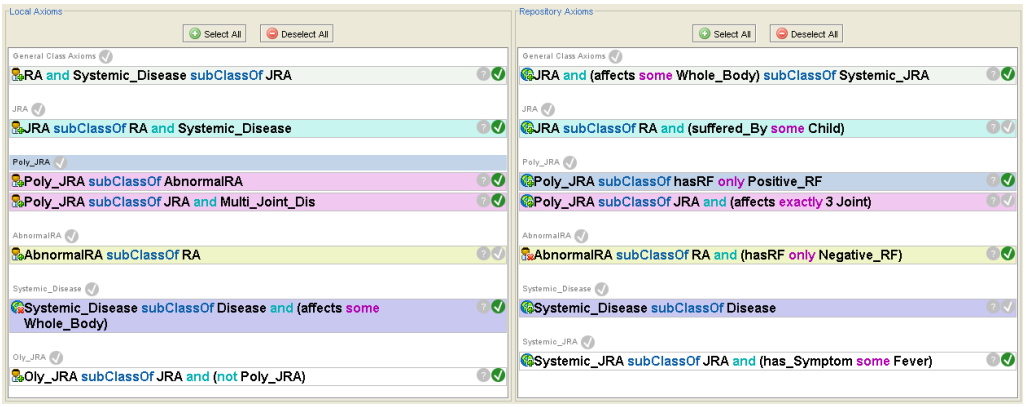


Figure 4: GUI for Structural Differences in ContentCVS

hand-side of Figure 4 was added to $\mathcal{O}^L_{\mathsf{bak}}$ in the local ontology (indicated by an icon representing a user with a '+'), whereas the axiom $(\mathsf{Systemic\_Disease} \sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{affects.WholeBody})$ was deleted from $\mathcal{O}^L_{\mathsf{bak}}$ in the repository (indicated by an icon representing the Globe with a cross '$\times$').

6.2. *Deductive differences*

In contrast to text files, the selected parts from $\mathcal{O}^L$ and $\mathcal{O}^R$ can interact in unexpected ways, which may lead to errors that should be repaired. To help users detect such errors, we propose to compare the entailments that hold in $\mathcal{O}^L_{\mathsf{temp}}$ with those that hold in $\mathcal{O}^L$ and $\mathcal{O}^R$ by using the notion of *deductive difference* [36].

Definition 3. *The deductive difference* $\mathsf{diff}(\mathcal{O}, \mathcal{O}')$ *between* $\mathcal{O}$ *and* $\mathcal{O}'$ *expressed in a DL* $\mathcal{DL}$ *is the set of* $\mathcal{DL}$*-axioms* α *s.t.* $\mathcal{O} \not\models \alpha$ *and* $\mathcal{O}' \models \alpha$.

Intuitively, this difference is the set of *all* (possibly complex) entailments that hold in one ontology but not in the other. In our running example, for the selection in Figure 4, there are entailments that *(i)* hold in $\mathcal{O}^L_{\mathsf{temp}}$ and not in $\mathcal{O}^L$, such as $\tau_1 := (\mathsf{SystemicJRA} \sqsubseteq \exists \mathsf{has\_Symptom.Fever})$; *(ii)* hold in $\mathcal{O}^L_{\mathsf{temp}}$ but not in either $\mathcal{O}^L$ or $\mathcal{O}^R$, such as σ_5 from Table 3; *(iii)* hold in $\mathcal{O}^L$ but not in $\mathcal{O}^L_{\mathsf{temp}}$, such as $\tau_2 := (\mathsf{RA} \sqcap \exists \mathsf{affects.WholeBody} \sqsubseteq \mathsf{JRA})$; and finally *(iv)* hold in $\mathcal{O}^R$ but not in $\mathcal{O}^L_{\mathsf{temp}}$, such as σ_2 in Table 3.

Therefore, we argue that the relevant deductive differences between $\mathcal{O}^L_{\mathsf{temp}}$, $\mathcal{O}^L$ and $\mathcal{O}^R$ capture all potential errors that may have been introduced in the reconciliation process. However, the notion of deductive difference has several drawbacks in practice. First, checking whether $\mathsf{diff}(\mathcal{O}, \mathcal{O}') = \emptyset$ is undecidable in expressive DLs, such as $\mathcal{SROIQ}$ (OWL 2) and $\mathcal{SHOIQ}$ (OWL DL) [36].

Second, the number of entailments in the difference can be huge (even infinite), and so likely to overwhelm users. These practical drawbacks motivate the need for *approximations*— subsets of the deductive difference (see Step 4 in Table 4).

Definition 4. *A function* $\mathsf{diff}_{\approx}(\mathcal{O}, \mathcal{O}')$ *is an approximation for* $\mathsf{diff}(\mathcal{O}, \mathcal{O}')$ *if for each pair* $\mathcal{O}, \mathcal{O}'$ *of* $\mathcal{DL}$*-ontologies,* $\mathsf{diff}_{\approx}(\mathcal{O}, \mathcal{O}') \subseteq \mathsf{diff}(\mathcal{O}, \mathcal{O}')$.

A useful approximation should be easy to compute, yet still provide meaningful information to the user. One possibility is to define an approximation by considering only entailments of a certain form. Our tool ContentCVS allows users to customise approximations by selecting among the following kinds of entailments, where A, B are atomic concepts (including $\top$, $\bot$) and R, S atomic roles or inverses of atomic roles: (i) $A \sqsubseteq B$, (ii) $A \sqsubseteq \neg B$, (iii) $A \sqsubseteq \exists R.B$, (iv) $A \sqsubseteq \forall R.B$, and (v) $R \sqsubseteq S$. The smallest implemented approximation considers only axioms of the form (i), which amounts to comparing the classification hierarchy of both ontologies, while the largest considers all types (i)—(v). Clearly, the larger the class of entailments presented to the user, the more errors could be detected. The corresponding differences, however, are harder to compute, harder to present to the user, and may be harder for the user to understand.

Although these approximations can all be algorithmically computed, only the entailments of the form (i) and (v) are typically provided by reasoners as standard outputs of classification. Computing deductive differences based on entailments (ii)-(iv) is potentially very expensive since it may involve performing a large number of additional entailment tests. To reduce the number of tests, ContentCVS uses the notion of a *locality-based module* [7, 6]. Locality-based modules enjoy several useful properties: first, they can be computed efficiently; second, if an ontology $\mathcal{O}$ entails an axiom of the form $A \sqsubseteq C$, for A atomic and C a possibly complex concept, then the locality-based module $\mathcal{O}_A$ for A in $\mathcal{O}$ also entails the axiom. Finally, locality-based modules are typically very small compared to the size of the original ontology.

To check for entailments of the form (ii)-(iv), ContentCVS first extracts the locality-based module for A and looks for potential entailments only within the module. For example, in the case of (iii) ContentCVS would only test entailments where both R and B are in the vocabulary of the module $\mathcal{O}_A$, which significantly reduces the search space. Furthermore, the actual relevant entailments can be checked w.r.t. the (small) module, and not with respect to the (potentially large) original ontology. Our experiments in Section 7 suggest that the use of locality-based modules makes the computation of approximations to the deductive difference based on all types (i)-(v) of entailments computationally feasible.

6.3. Selection of entailments

While some entailments in the computed differences are intended, others reveal errors in $\mathcal{O}^L_{\mathsf{temp}}$, as illustrated by the following example.

Example 1. *In our example (Table 3), the entailment* $\mathsf{JRA} \sqsubseteq \mathsf{Juvenile\_Disease}$ *(σ_2) is intended. In contrast,* $\mathsf{Poly\_JRA} \sqsubseteq \bot$ *(σ_5) reveals an inconsistency in $\mathcal{O}^L_{\mathsf{temp}}$, and hence an obvious error.*

Steps 6 and 7 thus involve selecting entailments that: *(i)* are intended and should follow from $\mathcal{O}^L_{\mathsf{temp}}$ (written $\Im^+$ in Table 4), and *(ii)* are unintended and should not follow from $\mathcal{O}^L_{\mathsf{temp}}$ (written $\Im^-$).

The development of techniques to help users understand the relevant deductive differences and subsequently select the sets of intended and unintended entailments is especially challenging. First, a tool should explain, on the one hand, why the new entailments that hold in $\mathcal{O}^L_{\mathsf{temp}}$ do not hold in $\mathcal{O}^L$ and $\mathcal{O}^R$ alone and, on the other hand, why the lost entailments that hold in $\mathcal{O}^L$ and $\mathcal{O}^R$ do not hold in $\mathcal{O}^L_{\mathsf{temp}}$. The notion of a *justification* has proved very useful in ontology debugging [47, 35]:

Definition 5. *Let $\mathcal{O} \models \alpha$. A justification for α in $\mathcal{O}$ is an ontology $\mathcal{O}' \subseteq \mathcal{O}$ satisfying the following properties: (i) $\mathcal{O}' \models \alpha$, and (ii) there is no $\mathcal{O}'' \subset \mathcal{O}'$ s.t. $\mathcal{O}'' \models \alpha$. We denote by* $\mathsf{Just}(\alpha, \mathcal{O})$ *the set of all justifications for α in $\mathcal{O}$.*

In order to explain an entailment, ContentCVS presents all its justifications. Computing all justifications is expensive, so ContentCVS uses the optimisations from [33, 54], which have proved effective in practice. In particular, as described in [54], our algorithm for extracting all the justifications for an entailment of the form $A \sqsubseteq C$, for A an atomic concept, is based on extracting first the locality-based module for A in the ontology and then compute the justifications w.r.t. the module instead of w.r.t. the whole ontology.

Even with explanations provided, the potentially large number of relevant entailments may overwhelm users. These entailments should therefore be presented in a way that makes them easier to understand and manage. To this end, ContentCVS extends known ontology debugging techniques by identifying dependencies between entailments. As an illustration, consider $\sigma_2 := (\mathsf{JRA} \sqsubseteq \mathsf{Juvenile\_Disease})$ from Table 3 and $\tau_4 := (\mathsf{SystemicJRA} \sqsubseteq \mathsf{Juvenile\_Disease})$ which hold in $\mathcal{O}^L_{\mathsf{temp}}$ but not in $\mathcal{O}^L$. The entailment τ_4 *depends* on σ_2 since whenever σ_2 is invalidated by removing axioms from $\mathcal{O}^L_{\mathsf{temp}}$, then τ_4 is also invalidated. Similarly, the entailment $\tau_5 := (\mathsf{Oly\_JRA} \sqsubseteq \exists \mathsf{affects.WholeBody})$ depends on the entailment $\tau_6 := (\mathsf{JRA} \sqsubseteq \exists \mathsf{affects.WholeBody})$: adding any set

of axioms from $\mathcal{O}^L$ or $\mathcal{O}^R$ that causes τ_6 to hold in $\mathcal{O}^L_{\mathsf{temp}}$ would also cause τ_5 to hold. We formalise these intuitions in our setting as follows:

Definition 6. *Let $\mathcal{O} \models \alpha, \beta$. The axiom β depends on α w.r.t. $\mathcal{O}$, written $\alpha \triangleright \beta$ iff for each $\mathcal{J}_\beta \in \mathsf{Just}(\beta, \mathcal{O})$ there is $\mathcal{J}_\alpha \in \mathsf{Just}(\alpha, \mathcal{O})$ such that $\mathcal{J}_\alpha \subseteq \mathcal{J}_\beta$. We say that α is a $\triangleright$-minimum (respectively $\triangleright$-maximum) if there is no β s.t. α depends on β (respectively β depends on α).*

The relation $\triangleright$ is consistent with our intuitions as shown in the following proposition, which follows directly from Definitions 5 and 6:

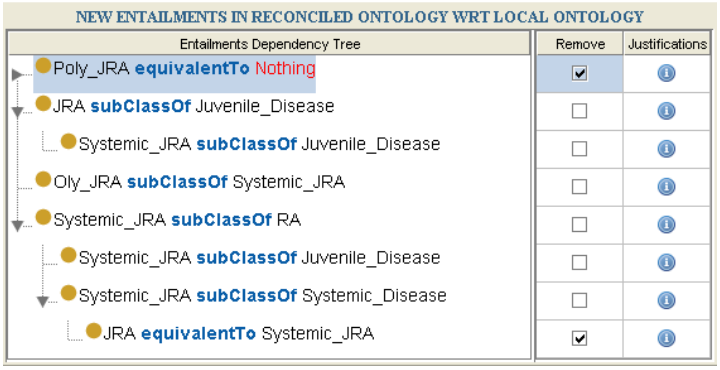
Proposition 1. *Let $\mathcal{O} \models \alpha, \beta$, $\mathcal{O}' \subset \mathcal{O}$ and $\alpha \triangleright \beta$. Then: 1) $\mathcal{O}' \not\models \alpha$ implies $\mathcal{O}' \not\models \beta$, and 2) $\mathcal{O}' \models \beta$ implies $\mathcal{O}' \models \alpha$.*

Proof. First we prove Condition 1. Let $\alpha \rhd \beta$ and $\mathcal{O}' \not\models \alpha$. Since $\mathcal{O} \models \alpha$, for each $\mathcal{J}_\alpha \in \mathsf{Just}(\alpha, \mathcal{O})$ there is an axiom $\gamma \in \mathcal{J}_\alpha$ s.t. $\gamma \notin \mathcal{O}'$. Otherwise, $\mathcal{O}'$ would imply α. Let $\mathcal{J}_\beta \in \mathsf{Just}(\beta, \mathcal{O})$. By definition of $\rhd$, there must be a $\mathcal{J}_\alpha \in \mathsf{Just}(\alpha, \mathcal{O})$ s.t. $\mathcal{J}_\alpha \subseteq \mathcal{J}_\beta$. Since $\mathcal{O}'$ does not include one axiom from each $\mathcal{J}_\alpha$, then it also does not include one axiom from each $\mathcal{J}_\beta$ and therefore $\mathcal{O}' \not\models \beta$, as required. We now prove Condition 2. Let $\alpha \rhd \beta$ and $\mathcal{O}' \models \beta$. Since $\mathcal{O} \models \beta$, there exists a justification $\mathcal{J}_\beta \in \mathsf{Just}(\beta, \mathcal{O})$ s.t. $\mathcal{J}_\beta \subseteq \mathcal{O}'$. By definition of $\rhd$ there must exist a $\mathcal{J}_\alpha \in \mathsf{Just}(\alpha, \mathcal{O})$ s.t. $\mathcal{J}_\alpha \subseteq \mathcal{J}_\beta$. Therefore $\mathcal{J}_\alpha \subseteq \mathcal{O}'$ and $\mathcal{O}' \models \alpha$, as required. □

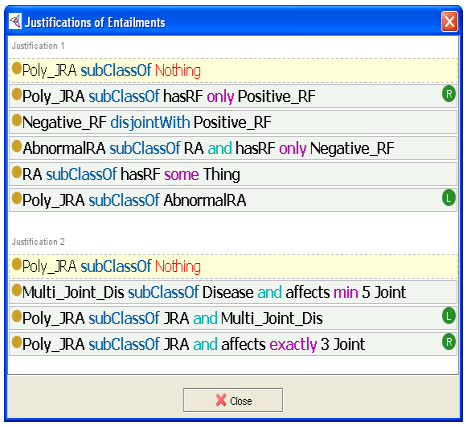
Figure 5(a) shows the ContentCVS GUI for selecting $\Im^-$. A similar interface is used to select $\Im^+$. The left-hand-side of the figure displays the ($\triangleright$)-dependency tree, which can be expanded and contracted in the usual way; on the right-hand side, the user can select an entailment to remove and show its justification(s). The justifications for the entailment highlighted in Figure 5(a) are shown in Figure 5(b). The operation and GUI provided by Protégé 4 [24] was extended in order to indicate which axioms in these justifications were selected in Step 2 of Table 4 from $\mathcal{O}^L$ and $\mathcal{O}^R$, marking them with 'L' and 'R' respectively. The unmarked axioms occur in both ontologies.

6.4. Plan generation, selection and execution

Changing the set of entailments can only be achieved by modifying the ontology itself. In general, there may be zero or more possible choices of sets of axioms to add and/or remove in order to satisfy a given set of requirements. We call each of these possible choices a *repair plan* (or *plan*, for short).



(a) Selection of Entailments



(b) Justifications Poly_JRA $\sqsubseteq \bot$

Figure 5: GUI for Selection of Entailments in ContentCVS

Definition 7. *Let $\mathcal{O}$, $\mathfrak{Y}^+$, $\mathfrak{Y}^-$, $\mathcal{O}^+$ and $\mathcal{O}^-$ be finite sets of axioms s.t. $\mathcal{O}^- \subseteq \mathcal{O}$, $\mathcal{O}^+ \cap \mathcal{O} = \emptyset$, $\mathcal{O} \models \mathfrak{Y}^-$, $\mathcal{O} \cup \mathcal{O}^+ \models \mathfrak{Y}^+$, and $\mathcal{O} \not\models \alpha$ for each $\alpha \in \mathfrak{Y}^+$. A repair plan for $\mathcal{O}$ given $\mathcal{O}^+$, $\mathcal{O}^-$ $\mathfrak{Y}^+$ and $\mathfrak{Y}^-$ is a pair $\mathcal{P} = \langle \mathcal{P}^+, \mathcal{P}^- \rangle$ such that $\mathcal{P}^+ \subseteq \mathcal{O}^+$, $\mathcal{P}^- \subseteq \mathcal{O}^-$ and the following conditions hold:*

1. $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^- \models \alpha$ *for each* $\alpha \in \mathfrak{Y}^+$, *and*
2. $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^- \not\models \beta$ *for each* $\beta \in \mathfrak{Y}^-$.

$\mathcal{P}$ is minimal if there is no $\mathcal{P}_1$ s.t. $\mathcal{P}_1^+ \subset \mathcal{P}^+$ and $\mathcal{P}_1^- \subset \mathcal{P}^-$.

Definition 7 extends the notion of a plan proposed in the ontology repair literature (e.g., see [34]). In particular, the goal of a plan in [34] is always to remove a set of axioms so that certain entailments do not hold anymore; hence, a plan is always guaranteed to exist. In contrast, a plan as in Definition 7 also involves

adding axioms so that certain entailments hold; therefore, possibly conflicting sets of constraints need to be satisfied. Furthermore, existing repair techniques (e.g. [34]) are restricted to obvious inconsistencies (i.e., unsatisfiable concepts), whereas our techniques apply to errors caused by arbitrary entailments.

Step 8 from Table 4 involves the computation of all minimal plans (denoted $\mathbb{P}$). The ontology $\mathcal{O}$ to be 'repaired' is $\mathcal{O}^L_{\mathsf{temp}}$ from Step 3. The intended and unintended entailments ($\mathfrak{S}^+$ and $\mathfrak{S}^-$) are those selected in Steps 6 and 7. We assume that a plan can add any subset of the axioms in Λ_s, which were not selected in Step 2 (i.e., $\mathcal{O}^+ = \Lambda_s \setminus \mathcal{S}$), and it can remove any subset of the selected axioms (i.e., $\mathcal{O}^- = \mathcal{S}$). Hence, we assume that a plan should not remove axioms outside Λ_s, since they are common to both versions of the ontology.

Example 2. *Given* $\mathfrak{S}^+ = \{\sigma_2\}$ *and* $\mathfrak{S}^- = \{\sigma_5\}$ *from Example 1, four minimal plans can be identified:* $\mathcal{P}_1 = \langle\{\delta_1\}, \{\delta_3, \delta_4\}\rangle$*;* $\mathcal{P}_2 = \langle\{\delta_1\}, \{\delta_3, \gamma_4\}\rangle$*;* $\mathcal{P}_3 = \langle\{\delta_1\}, \{\gamma_3, \gamma_4\}\rangle$*;* $\mathcal{P}_4 = \langle\{\delta_1\}, \{\gamma_3, \delta_4\}\rangle$.

In Step 9 users can select from $\mathbb{P}$ a plan to be applied. If no plan matches their intentions, the conflict resolution process ends as it started; that is, by returning the old version of $\mathcal{O}^L$ in a conflicting state (Step 9). In contrast, if a plan $\mathcal{P}$ is selected, then $\mathcal{P}$ is applied by returning the ontology $(\mathcal{O}^L_{\mathsf{temp}} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$ in a non-conflicting state (Steps 10–11), which is then ready to be committed.

Definition 7 suggests a simple procedure for computing all plans: for each possible $\mathcal{P}^+ \subseteq \mathcal{O}^+$ and $\mathcal{P}^- \subseteq \mathcal{O}^-$, use a reasoner to check if $\langle\mathcal{P}^+, \mathcal{P}^-\rangle$ satisfies Conditions 1 and 2 from Definition 7. ContentCVS, however, implements an optimised version of Algorithm 1, which uses inverted indexes, reduces the number of combinations and avoids potentially expensive entailment checks by reusing the justifications already computed when obtaining the dependency relation ($\triangleright$) from Definition 6. The correctness of the algorithm is a direct consequence of the following: *(i)* in order for an entailment $\alpha \in \mathfrak{S}^+$ to hold after the execution of a plan $\langle\mathcal{P}^+, \mathcal{P}^-\rangle$, $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$ must contain at least one justification for α in $\mathcal{O} \cup \mathcal{O}^+$ (Lines 6–10); *(ii)* in order for an entailment $\beta \in \mathfrak{S}^-$ not to hold after the execution of a plan $\langle\mathcal{P}^+, \mathcal{P}^-\rangle$, it is sufficient to show that no justification for β in $\mathcal{O} \cup \mathcal{O}^+$ is contained in $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$ (Lines 11–15). The set of all minimal plans can be straightforwardly computed once all the plans have been obtained.

Proposition 2. *Algorithm 1 returns all plans for the given input.*

Proof. For any $\mathcal{P} \in \mathbf{P}$, with $\mathcal{P} = \langle\mathcal{P}^+, \mathcal{P}^-\rangle$, we show that $\mathcal{P}$ is a plan for $\mathcal{O}$ given $\mathcal{O}^+$, $\mathcal{O}^-$ $\mathfrak{S}^+$ and $\mathfrak{S}^-$, as per Definition 7. The facts that $\mathcal{P}^+ \subseteq \mathcal{O}^+$ and

Algorithm 1 Computing All Plans Using Justifications

Procedure all_Plans($\mathcal{O}, \mathcal{O}^+, \mathcal{O}^-, \Im^+, \Im^-$)
Input: $\mathcal{O}$, $\mathcal{O}^+$, $\mathcal{O}^-$, $\Im^+$, $\Im^-$ as in Definition 7, $\mathsf{Just}(\gamma, \mathcal{O} \cup \mathcal{O}^+)$ for each $\gamma \in \Im^+ \cup \Im^-$
Output: $\mathbf{P}$: set of all plans

1: $\mathbf{P} := \emptyset$
2: **for each** $\mathcal{P}^+ \subseteq \mathcal{O}^+$ and **each** $\mathcal{P}^- \subseteq \mathcal{O}^-$ **do**
3: validPlan := true
4: **for each** $\alpha \in \Im^+$ **do**
5: foundJust := false
6: **for each** $\mathcal{J}_\alpha \in \mathsf{Just}(\alpha, \mathcal{O} \cup \mathcal{O}^+)$ **do**
7: **if** $\mathcal{J}_\alpha \subseteq (\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$ foundJust := true
8: **end for**
9: **if** foundJust = false **then** validPlan := false
10: **end for**
11: **for each** $\beta \in \Im^-$ **do**
12: **for each** $\mathcal{J}_\beta \in \mathsf{Just}(\beta, \mathcal{O} \cup \mathcal{O}^+)$ **do**
13: **if** $\mathcal{J}_\beta \subseteq (\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$ **then** validPlan := false
14: **end for**
15: **end for**
16: **if** validPlan = true **then** $\mathbf{P} := \mathbf{P} \cup \{\langle \mathcal{P}^+, \mathcal{P}^- \rangle\}$
17: **end for**
18: **return** $\mathbf{P}$

$\mathcal{P}^- \subseteq \mathcal{O}^-$ are clear from Line 2. For each $\alpha \in \Im^+$, Line 7 implies that there is a $\mathcal{J}_\alpha \in \mathsf{Just}(\alpha, \mathcal{O} \cup \mathcal{O}^+)$ s.t. $\mathcal{J}_\alpha \in (\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$, and the definition of justification immediately implies that $\mathcal{P}$ satisfies Condition 1 from Definition 7. For each $\beta \in \Im^-$, Lines 14 and 15 imply that $\mathcal{P}^-$ removes at least one axiom from each justification for β in $\mathcal{O} \cup \mathcal{P}^+$; hence $\mathcal{P}$ also satisfies Condition 2.

Let us assume that there exists a plan $\langle \mathcal{P}^+, \mathcal{P}^- \rangle \notin \mathbf{P}$ for $\mathcal{O}$ given $\mathcal{O}^+$, $\mathcal{O}^-$ $\Im^+$ and $\Im^-$. This means that either there is an $\alpha \in \Im^+$ s.t. no justification for α in $\mathcal{O} \cup \mathcal{O}^+$ is contained in $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$, or there is a $\beta \in \Im^-$ and a justification $\mathcal{J}_\beta$ for β in $\mathcal{O} \cup \mathcal{O}^+$ s.t. $\mathcal{J}_\beta \subseteq (\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$. In the first case, there can be no justification for α in $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$, because $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^- \subseteq \mathcal{O} \cup \mathcal{O}^+$, so $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^- \not\models \alpha$, contradicting our assumption that $\langle \mathcal{P}^+, \mathcal{P}^- \rangle$ is a plan. In the second case, $\mathcal{J}_\beta \in (\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^-$, which implies $(\mathcal{O} \cup \mathcal{P}^+) \setminus \mathcal{P}^- \models \beta$, also contradicting our assumption. □

Figure 6 illustrates the ContentCVS GUI for visualising plans. It shows two of the minimal plans for our running example. ContentCVS provides additional functionality to help the user select the most suitable minimal plan (see Step 10). In particular, it identifies the axioms in the structural difference of $\mathcal{O}^L$ and $\mathcal{O}^R$ that are shared by all minimal plans (axioms marked with a 'P' in Figure 6), and

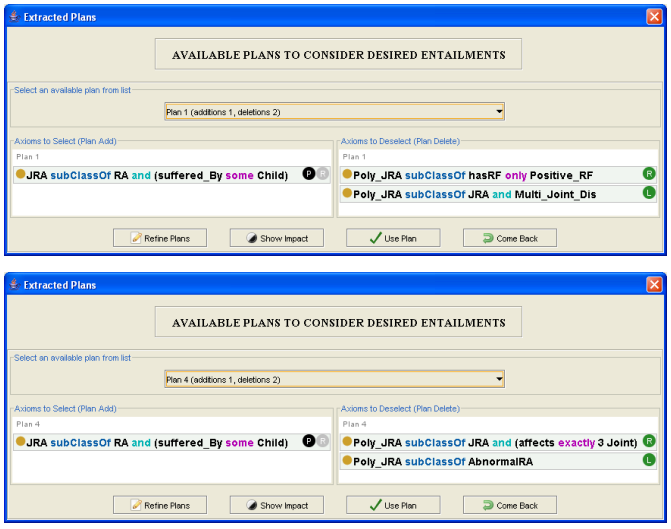


Figure 6: GUI for Plan Selection in ContentCVS

presents the remaining (i.e., non-shared) axioms in a separate frame, allowing the user to select which ones among them a plan must either add or delete. The tool then filters the set of minimal plans according to the user's selections. For example, if the axiom $\mathsf{Poly\_JRA} \sqsubseteq \mathsf{Abnormal\_JRA}$ in Figure 6 is selected, the tool would discard any plan that does not involve adding or deleting this axiom.

7. Evaluation

Our evaluation tries to answer the following important questions:

1. Do conflicts of the kind described in Section 3 occur in practice?
2. Are the implemented algorithms computationally feasible in practice?
3. Do our techniques provide useful assistance to ontology engineers?
4. Is ContentCVS easy and intuitive to use?

To address the first question we analyse in Section 7.1 a sequence of versions of a realistic ontology and the respective change logs of each version. To address the second question, we describe in Section 7.2 a set of synthetic experiments using the same sequence of versions. Finally, to address the last two questions, we have conducted a pilot user study, which we describe in Section 7.3.

Change	$\mathcal{O}_2$	$\mathcal{O}_3$	$\mathcal{O}_4$	$\mathcal{O}_5$	$\mathcal{O}_6$	$\mathcal{O}_7$	$\mathcal{O}_8$	$\mathcal{O}_9$	$\mathcal{O}_{10}$
Concepts added	8	3	10	38	24	1	27	11	23
Concepts deleted	2	0	1	1	0	0	1	3	1
Roles added	0	0	3	4	3	0	6	1	0
Roles deleted	0	0	1	0	0	0	0	1	0
Concept axioms added	31	6	18	53	51	5	47	26	52
Concept axioms deleted	8	0	10	9	0	3	5	14	17
Role axioms added	0	0	3	6	6	0	7	2	0
Role axioms deleted	0	0	3	0	0	0	2	0	0
Annotations added	10	5	32	19	30	2	43	26	38
Annotations deleted	2	2	9	2	0	0	2	6	6

Table 5: Summary of change logs.

	Excerpt from version $\mathcal{O}_4$
α_1	$\mathsf{Document\_entity} \sqcap \mathsf{Domain\_entity} \sqsubseteq \bot$
α_2	$\mathsf{Document\_content\_spec} \sqsubseteq \mathsf{Document\_entity}$
α_3	$\mathsf{Indicant} \sqsubseteq \mathsf{Domain\_entity}$
α_4	$\mathsf{Present\_absent\_indicant} \sqsubseteq \mathsf{Indicant}$
α_5	$\mathsf{Section\_content\_spec} \sqsubseteq \mathsf{Document\_content\_spec}$
α_6	$\exists \mathsf{includes\_sub\_doc}.\top \sqsubseteq \mathsf{Document\_content\_spec}$
α_7	$\mathsf{has\_sub\_doc} \sqsubseteq \mathsf{includes\_sub\_doc}$

Table 6: A relevant fragment of version $\mathcal{O}_4$

7.1. Analysis of real changes

In this section, we study a sequence of 10 versions of a medical ontology developed at the University of Manchester and used in the context of the Clinergy project.[3] The sizes of the different versions vary from 71 concepts, 13 roles and 195 axioms in the first version to 207 concepts, 38 roles and 620 axioms in the last version; all the versions are expressed in the description logic $\mathcal{SHIQ}(D)$. The ontology was developed during a short period of time: from July 21st 2008 until July 28th 2008. On average, the developers generated one or two versions of the ontology each day. This situation, in which versions are generated very frequently, is consistent with the scenario described in Section 3.

[3] `http://www.opengalen.org/sources/software.html`.

	Changes over Domain Entities $(\Delta\mathcal{O}_4)^1$
γ_1	Bruise_to_surface_structure $\sqsubseteq$ Trauma_to_surface_structure
γ_2	Trauma_to_surface_structure $\sqsubseteq$ Present_absent_indicant
γ_3	Trauma_to_surface_structure $\sqsubseteq$ $\forall$has_locus.Surface_Anatomical_structure
γ_4	Surface_Anatomical_structure $\sqsubseteq$ Anatomic_structure
γ_5	Anatomic_structure $\sqsubseteq$ Domain_entity
	Changes over Document Entities $(\Delta\mathcal{O}_4)^2$
δ_1	Bruise_to_surface_structure $\sqsubseteq$ Section_content_spec
δ_2	Surface_trauma_subsection_spec $\sqsubseteq$ Section_content_spec
δ_3	Bruise_to_surface_structure $\sqsubseteq$ $\exists$has_sub_doc.First_heart_sound_clin_holder
δ_4	Bruise_to_surface_structure $\sqsubseteq$ $\exists$has_sub_doc.Second_heart_sound_clin_holder
δ_5	Bruise_to_surface_structure $\sqsubseteq$ $\exists$has_sub_doc.Heart_murmur_clin_holder

Table 7: Excerpt of changes $(\Delta\mathcal{O}_4)^1$ and $(\Delta\mathcal{O}_4)^2$ performed over version $\mathcal{O}_4$

Table 5 summarises the content of the change logs of each version[4]. For example, the second column represents the changes performed from version $\mathcal{O}_1$ to version $\mathcal{O}_2$. The table clearly shows how the ontology grows as it is being developed: most of the changes involve addition of entities, axioms and annotations. Most of the added axioms are concept axioms (mostly inclusions, equivalence and disjointness axioms), which is a typical situation when modelling using OWL. Interestingly, there are also a significant number of deletions, which reflects the fact that ontology developers are revising their modelling choices and fixing errors. Extra-logical changes such as modifications in the annotations are also fairly common, which suggests that they should be taken into account when identifying potential conflicts.

In order to verify that conflicts of the kind described in Section 3 are likely to occur in practice, we have also performed a detailed analysis of the change logs. Our findings suggest that changes leading to an error, such as the unsatisfiability of a concept, may involve the simultaneous modification of different aspects of the domain. For example, consider the evolution from version $\mathcal{O}_4$ to version $\mathcal{O}_5$ and the two groups of changes $(\Delta\mathcal{O}_4)^1$ and $(\Delta\mathcal{O}_4)^2$ indicated in Table 7, which together with the fragment of $\mathcal{O}_4$ from Table 6 lead to the un-

[4]A document with an overview of the changes can be downloaded from: *URL removed for review*

satisfiability of the concept Bruise_to_surface_structure. The changes in $(\Delta\mathcal{O}_4)^1$ describe the concept Bruise_to_surface_structure as an anatomical structure (and hence as a 'domain entity'), whereas the changes in $(\Delta\mathcal{O}_4)^2$ describe it as a document concept (and hence as a 'document entity'); and concepts Domain_entity and Document_entity are disjoint according to $\mathcal{O}_4$ (see axiom α_1 from Table 6).

Thus, under the assumption that changes in $(\Delta\mathcal{O}_4)^1$ and $(\Delta\mathcal{O}_4)^2$ have been performed concurrently by different ontology engineers, the presence of incompatible changes leads precisely to the issues pointed out in Section 3. This assumption is reasonable, as different aspects of the domain are likely to be developed by different experts.

7.2. Performance evaluation

In our experiments we have simulated the evolution of an ontology by using the sequence of versions from Section 7.1. The experiments were performed on a laptop computer with a 1.82 GHz processor and 3Gb of RAM. The average classification time of an ontology in the sequence is approximately one second when using the Pellet reasoner [50].

For each pair $\mathcal{O}_i, \mathcal{O}_{i+1}$, $i \in \{1, \ldots, 9\}$ of consecutive versions, and both the smallest and largest approximations of the deductive difference implemented in ContentCVS, we have performed the experiment in Table 8. The Roman numbers in Table 8 refer to measurements that are stored during the experiment and presented in Table 9. These experiments follow our approach for conflict resolution in Table 4, with the assumption that version $\mathcal{O}_i$ is the local ontology, version $\mathcal{O}_{i+1}$ is the ontology in the repository, and the steps in Table 4 requiring manual intervention are performed randomly.

Table 9 summarises our results.[5] Most of the values in the table are either average or maximum values for the 200 iterations in the loop from Table 8. Average values are indicated with the tag 'avg' in the header, and maximum values with the tag 'max'. Several conclusions can be drawn from these experiments.

First, from a computational point of view, the main bottleneck is the computation of all the justifications for the entailments of interest. Once the justifications have been computed, the time needed for computing the plans is relatively low. In Table 9, we can see that the average time needed per justification can reach 5.9 seconds (see **V**, $\mathcal{O}_4\&\mathcal{O}_5$); if 300 justifications have to be computed in total, then the total time may reach 30 minutes. Hence, it is important to investigate optimisation techniques for computing all justifications; first steps in this direction have been taken in [54, 25, 23].

[5] URL removed for review

Input: $\mathcal{O}, \mathcal{O}'$: ontologies; approximation function $\mathsf{diff}_{\approx}$

Compute Λ_s and store its size (**I**) and computation time (**II**)
repeat
Randomly select $\mathcal{S} \subseteq \Lambda_s$, and compute $\mathcal{O}_{\mathsf{aux}} := \mathcal{O} \cup \mathcal{S}$
Compute $\mathsf{diff}_{\approx}(\mathcal{O}_{\mathsf{aux}}, \mathcal{O}) \cup \mathsf{diff}_{\approx}(\mathcal{O}_{\mathsf{aux}}, \mathcal{O}')$ and store its size (**III**)
Compute $\mathsf{diff}_{\approx}(\mathcal{O}, \mathcal{O}_{\mathsf{aux}})$ and store its size (**IV**)
Get all justifications for entailments in $\mathsf{diff}_{\approx}$; store avg. time per justification (**V**)
Compute $\triangleright$, and store the number of $\triangleright$-minimums (**VI**)
Randomly select $\mathfrak{S}^-$ from minimums of $\triangleright$ and $\mathfrak{S}^+$ from maximums of $\triangleright$
Compute $\mathbb{P}$ (min. plans); store number of plans (**VII**) and extraction time (**VIII**)
until 200 iterations have been performed

Table 8: Synthetic Experiments

			Smallest $\mathsf{diff}_{\approx}$ approximation						Largest $\mathsf{diff}_{\approx}$ approximation					
	I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**
$\mathcal{O}$ **&** $\mathcal{O}'$			avg	avg	avg	avg	avg/max	avg	avg	avg	avg	avg	avg/max	avg
$\mathcal{O}_1$&$\mathcal{O}_2$	50	0.03	15	6	0.1	15	1 / 1	1.5	111	17	2.0	33	495 / 5508	10.3
$\mathcal{O}_3$&$\mathcal{O}_4$	82	0.02	13	4	0.26	13	3 / 18	1.7	128	90	0.9	30	46 / 896	3.6
$\mathcal{O}_4$&$\mathcal{O}_5$	93	0.02	31	14	0.1	29	3 / 32	1.2	267	48	5.9	49	2.7 / 6	30
$\mathcal{O}_7$&$\mathcal{O}_8$	110	0.03	19	15	0.02	18	1 / 4	0.07	216	78	1.2	47	488 / 3888	4
$\mathcal{O}_8$&$\mathcal{O}_9$	79	0.02	15	6	0.06	14	1 / 2	0.3	251	14	3.7	46	101 / 720	21.5
$\mathcal{O}_9$&$\mathcal{O}_{10}$	117	0.01	24	8	1.5	24	7 / 50	15.6	208	154	5.3	31	35 / 225	22.7

Table 9: Summary of Results. Roman numbers refer to Table 8. Time given in seconds

Second, the amount of information presented to the user largely depends on the selected approximation for the deductive difference (see Section 6.2). In the case of the smallest approximation, the average number of axioms in the relevant differences (see **III** and **IV**) is in the range 4–31, and the average number of minimal plans (see **VII**) is in the range 1–50. In contrast, in the case of the largest approximation, these average numbers are in the ranges 14–267, and 6–5508 respectively. The amount of information the user would need to consider is thus much larger. Table 9 also shows that the use of the dependency relation $\triangleright$ can lead to a significant reduction in the amount of information that is initially presented to the user (**VI**). Note that for the largest approximation the number of minimums for $\triangleright$ is comparable to the size of the relevant deductive differences for the smallest approximation.

Overall, we believe that this experiment demonstrates that the algorithms implemented in ContentCVS exhibit reasonable performance, and that our approach is computationally feasible. The use of larger approximations of deduc-

tive difference may, however, require improved techniques for computing justifications. The use of larger approximations may also risk overwhelming the user with information, although presentation techniques such as the dependency one implemented in ContentCVS can help to ameliorate this problem.

7.3. User study

We have conducted a pilot user study to evaluate the usability of the GUI implemented in ContentCVS, as well as to provide empirical evidence of the adequacy of our approach in practice. The details of the conducted study, including the questionnaire and the test ontologies, are available online.[6]

7.3.1. Design of the study

The user study consists of three main parts, each of which involves the completion of a number of tasks, as we describe next.

Part 1: Local evolution of an ontology

The first part simulates a conventional ontology repair scenario where a (single) developer performs a number of changes to his/her ontology $\mathcal{O}_0$ and, as a result, creates a new version $\mathcal{O}_1$ of the ontology in which errors may have been introduced. The main goal is to evaluate the repair techniques implemented in ContentCVS, in particular the identification of errors using deductive differences and error repair via the generation and selection of suitable plans.

The test ontology used in this part of the study describes the domain of academic publications and bibliographic references, which the participants in the study are expected to be relatively familiar with. The changes to the original version $\mathcal{O}_0$ involve the definition of three new concepts and the deletion of a property domain restriction. Users were first asked to use a reasoner to classify $\mathcal{O}_1$, examine the resulting entailments and try to understand the given justifications. Next, users were asked to identify and repair two kinds of errors, namely the occurrence of unintended entailments in $\mathcal{O}_1$ that did not occur in $\mathcal{O}_0$, and the lost of intended entailments that held in $\mathcal{O}_0$, but not in $\mathcal{O}_1$. Finally, users were asked to repeat this process by taking into account not only simple subsumptions, but also entailments of the form $A \sqsubseteq \neg B$, $A \sqsubseteq \exists R.B$ and $A \sqsubseteq \forall R.B$.

Part 2: Reconciliation of two independently-developed ontology versions

This part of the study simulates the scenario where a (single) developer working with a local copy $\mathcal{O}^L$ of an ontology performs a CVS-update and needs to reconcile the local version $\mathcal{O}^L$ with the version $\mathcal{O}^R$ in the repository. The main

[6]URL removed for review

goal is twofold; first, to evaluate the functionality in ContentCVS for directly comparing ontology versions, both from a syntactic and from a semantic point of view; second, to evaluate the means provided by ContentCVS for building an error-free reconciled version ready to be committed to the CVS repository.

Users were asked to reconcile two versions of an ontology describing types of Juvenile Ideopathic Arthritis. To this end, they first examined the structural difference between both versions and selected the axioms to be included in a temporary version $\mathcal{O}^L_{\mathsf{temp}}$ of the reconciled ontology. Next, users classified $\mathcal{O}^L_{\mathsf{temp}}$ and identified errors in the form of missing intended entailments or new unintended ones. As in Part 1, users were then asked to repeat this latter step by considering additional types of entailments and to use the proposed dependency relation between entailments to group them. Finally users were asked to repair the identified errors by selecting a suitable plan. Note that the test ontology versions used in this part of the study closely reproduces our running example, and the tasks involved follow the steps in Table 4 from Section 6.

Part 3: Concurrent development of an ontology

The final part of the study simulates the scenario where a number of users are developing an ontology concurrently using ContentCVS. As in Part 1, we used the familiar domain of publications and bibliographic references.

Each test involved three or four participants in the study. To produce a controlled experiment, each participant was asked to extend an initial version of the ontology by performing a number of changes specified a-priori. The first participant was in charge of performing changes concerning different types of academic staff members; the second one made changes concerning events such as conferences; the third one made changes concerning academic organisations; finally, the fourth one was asked to describe different kinds of resources and publications. Each participant was asked to perform a CVS-commit either upon completion of all the changes, or when explicitly indicated in their task sheets. In order to provide a more realistic environment, the exact point in time in which users attempt to commit their changes was not a-priori fixed. If the commit failed, the participant was asked to perform an update and reconcile the changes using their ContentCVS client. Once the participants had agreed upon a reconciled version of the ontology, they were asked to discuss it among themselves and with the coordinator of the study.

7.3.2. Results and discussion

In total, eight people participated in Parts 1 and 2 of the study. In the case of Part 3, we conducted three tests each of which involved either three or four

participants. The participants of the study are academic researchers, most of them working in fields other than the Semantic Web. For example, some of the participants work in a bio-genomics group, others in a robotics and cognitive sciences group, and so on. Most users evaluated their experience in knowledge representation as 'intermediate', in first order logic as either 'intermediate' or 'low' and in description logics and OWL also as either 'intermediate' or 'low'. All participants except for one had tried Protégé before and half of them had used a reasoner before when developing an ontology. However, none of the participants was familiar with justification-based explanations. The results can be summarised as follows:

- *Part 1*: Most users were able to understand the justifications provided by Protégé, although most of them found it 'hard' or 'very hard' to resolve potential errors manually. All the participants could identify both new unintended entailments and lost intended entailments when using ContentCVS and described the functionality provided by our tool for identifying these entailments as either 'good' or 'very good'. Most participants were satisfied with the smallest approximation of the deductive difference implemented in ContentCVS, and complained about excessive amounts of displayed information when using the largest implemented approximation instead. None of them considered that ContentCVS should aim at implementing richer approximations. Concerning the generation of plans, most users declared this functionality as either 'useful' or 'very useful' and found the capabilities of ContentCVS to recommend plans also useful.
- *Part 2*: Most users considered either 'useful' or 'very useful' the computation of structural differences between ontology versions. However, many users found it difficult to detect potential errors simply by examining the structural difference. As in Part 1, users liked the functionality in ContentCVS for detecting potential errors using approximations of the deductive difference. Interestingly, by using the largest approximation implemented in ContentCVS, users were able to detect errors other than unsatisfiable concepts and atomic subsumptions, which they found useful. All users considered that the use of a large approximation leads to an excessive amount of displayed information; however, all of them also found the presentation technique based on the dependency relation ($\triangleright$) very useful in alleviating this problem, but complained about the response time. Finally, most users were either 'very satisfied' or 'satisfied' with the reconciled ontology obtained after the execution of the selected repair plan.
- *Part 3*: Most participants had used a CVS system before for managing text

files and described the CVS functionality implemented in ContentCVS as either 'very useful' or 'useful'. Many participants emphasised the importance of some previous training for taking full advantage of the CVS functionality in ContentCVS. As in parts 1 and 2, the use of a combination of structural and deductive differences for detecting errors plus the computation of plans for repairing them was evaluated very positively. Concerning the ontology finally obtained, the participants were able to obtain an error-free ontology and were satisfied with the result. Only in one case the final discussion revealed an error in the final ontology; however the participants acknowledged that this error was not due to a deficiency of the tool.

Finally, all users evaluated the tool very positively. Most of them evaluated the GUI as 'good' and the ontology development workflow implemented in the tool as either 'very good' or 'good'. Therefore, we consider the feedback very positive in general. The main points of criticism were the following:

- Excessive amounts of information displayed when using 'large' approximations of the deductive difference. Even if the identification of dependencies between entailments helped in alleviating this problem, we consider it important to investigate new ways of organising a potentially overwhelming number of entailments.
- Slow response of the tool when computing all justifications of certain entailments and/or computing large approximations of the deductive difference. For large-scale ontology development, the further optimisation of our algorithms will be necessary. To this end, we consider especially promising the use of *incremental reasoning* techniques (see for example first results in [5]), which aim at avoiding unnecessary re-computations after performing a (small) number of changes to the ontology.

Addressing these deficiencies will be part of our future work.

8. Related work

In recent years, there have been several proposals for improving the support for collaboration in ontology engineering tools.

Collaborative Protégé [17, 14, 57] allows developers to hold discussions, chat, and annotate changes. Ontology versions are compared using the PromptDiff algorithm [15, 16], which creates a 'structural diff' between them using a number of heuristics. Changes can be annotated as instances of an ontology [14].

Collaborative Protégé has been tested in different scenarios obtaining good results [48]; however users still asked for more sophisticated communication techniques, as well as for a mechanism to prevent undesired logical consequences.

The authors of [44] have proposed a framework to manage and propagate changes in collaborative workflows in which OWL 2 changes are formalised using an ontology [43]. The authors of [46] have presented an ontology change management system in which the change history is stored on a server and the system can identify differences in the change sets from different clients.

DOGMA-MESS [12, 11] is a methodology with tool support for community-grounded ontology engineering. The methodology emphasises the importance of developing common conceptual models, especially when the process of collaborative ontology engineering crosses the boundaries of a single organisation. Ontology developers extend a consensual upper ontology following some predefined restrictions of the extension of upper knowledge (reuse policies [10]). If a reuse policy is violated a conflict arises. [31] and [27] present similar frameworks where reuse policies are based on the locality property of ontologies [7].

Several methodologies and tools focus on the definition of formal or semi-formal argumentation models to achieve a consensus over changes. For example, HCOME [38] and DILIGENT [56, 55, 51] are methodologies which follow a formal argumentation model. DILIGENT exploits an argumentation ontology to describe and store (as instances) the different discussion threads. Thus, the revision of past decisions and conclusions can be easily retrieved and reviewed, unlike traditional communications means like e-mail or chat. Cicero [13] implements and argumentation model based on the DILIGENT methodology, whereas [37] presents a system based on HCOME.

We believe that the functionality and guidelines described in [17, 14, 46, 11, 51, 37] and our techniques naturally complement each other. For example, discussion threads and annotations as well as formal argumentation models could be used in ContentCVS to assist users in selecting intended and unintended consequences (i.e. in Steps 9 and 10 of Table 4), and for recording the rationale behind their selections. Similarly, the guidelines provided by the DOGMA-MESS methodology could be integrated in our framework to help developers from different organisations to reach consensus.

The authors of [49] propose a 'locking' mechanism that allows a user, for example, to establish a lock over a concept, meaning that other users are not allowed to make changes that 'affect' that concept until the lock has been released. Although errors can still occur, the idea is that these locks would mitigate them. The precise guarantees provided by these locks are, however, not clear. Conflicts

are still likely to arise, and the approach in [49] does not provide any means for detecting and resolving them if they do.

The OWLDiff tool [7] provides a GUI for computing the deductive differences between pairs of OWL 2 EL ontologies. These differences are shown as a set of highlighted concepts, which are the ones whose meaning differs between the two ontologies. However, the tool does not provide means for explaining these differences to the user, for defining approximations, for building a reconciled ontology or for resolving conflicts.

Finally, the techniques we propose for conflict resolution extend those used for debugging and repairing inconsistencies based on justifications (e.g., [34, 47, 35, 33, 25, 23]). In Section 6 we have already pointed out the specific improvements w.r.t. existing techniques implemented in ContentCVS.

9. Conclusion

We have proposed a novel approach for facilitating concurrent ontology development, described a tool that implements it and presented a preliminary evaluation of the tool. The main contributions of our research can be summarised as follows:

- We have adapted the Concurrent Versioning paradigm to ontology engineering, which allows developers to make changes concurrently and remotely to the same ontology, track changes, and manage versions.
- We have proposed notions of equivalence and difference between ontology versions.
- We have proposed a collection of techniques for resolving conflicts between ontology versions both at the structural and at the semantic level.
- We have adapted state-of-the art ontology debugging and repair techniques to our setting and proposed several improvements.
- We have developed and evaluated a prototypical tool and obtained promising preliminary results as well as encouraging feedback from users.

In future work, we plan to improve our tool in a number of ways. First, we are working on improving the system's performance and in particular the computation of justifications. As pointed out in Section 7, the use of incremental reasoning techniques is particularly interesting in this regard. Second, we are enhancing the tool with new features. In particular, we plan to support richer

[7] OWLDiff: `http://sourceforge.net/projects/owldiff`

approximations and the use of *unit tests*—files containing a set of unintended entailments that can be used to detect modelling errors. Furthermore, we aim to integrate in our tool some of the functionality provided by state-of-the-art frameworks, such as Collaborative Protégé, for holding discussions and annotating changes. Another interesting direction for future research would be to provide means for assigning responsibilities and duties to different ontology developers and support for automatically checking whether changes made by developers are consistent with their duties. Finally, the integration of a reference thesaurus [32] within the proposed framework might help reducing the lexical conflicts caused by the use of different labels to refer to the same entity.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, G. Sherlock, Gene Ontology: tool for the unification of biology, Nature Genetics 25 (1) (2000) 25–29.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[3] N. Choi, I.-Y. Song, H. Han, A survey on ontology mapping, SIGMOD Rec. 35 (3) (2006) 34–41.

[4] Ó. Corcho, M. Fernández-López, A. Gómez-Pérez, Methodologies, tools and languages for building ontologies: Where is their meeting point?, Data Knowl. Eng. 46 (1) (2003) 41–64.

[5] B. Cuenca Grau, C. Halaschek-Wiener, Y. Kazakov, History matters: Incremental ontology reasoning using modules, in: Proc. of the 6th International Semantic Web Conference (ISWC), vol. 4825 of LNCS, Springer, 2007, pp. 183–196.

[6] B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler, Just the right amount: extracting modules from ontologies, in: Proc. of the 16th International Conference on World Wide Web (WWW), 2007, pp. 717–726.

[7] B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler, Modular reuse of ontologies: Theory and practice, Journal of Artificial Intelligence Research (JAIR) 31 (2008) 273–318.

[8] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics 6 (4) (2008) 309–322.

[9] S. de Coronado, L. W. Wright, G. Fragoso, M. W. Haber, E. A. Hahn-Dantona, F. W. Hartel, S. L. Quan, T. Safran, N. Thomas, L. Whiteman, The NCI Thesaurus quality assurance life cycle, Journal of Biomedical Informatics 42 (3) (2009) 530 – 539.

[10] P. De Leenheer, A. de Moor, R. Meersman, Context dependency management in ontology engineering: A formal approach, J. Data Semantics 8 (2007) 26–56.

[11] P. De Leenheer, C. Debruyne, DOGMA-MESS: A tool for fact-oriented collaborative ontology evolution, in: On the Move to Meaningful Internet Systems: OTM 2008 Workshops, vol. 5333 of LNCS, Springer, 2008, pp. 797–806.

[12] A. de Moor, P. De Leenheer, R. Meersman, DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering, in: 14th International Conference on

Conceptual Structures, ICCS, vol. 4068 of Lecture Notes in Computer Science, Springer, 2006, pp. 189–202.

[13] K. Dellschaft, H. Engelbrecht, J. M. Barreto, S. Rutenbeck, S. Staab, Cicero: Tracking design rationale in collaborative ontology engineering, in: The Semantic Web: Research and Applications, 5th European Semantic Web Conference (ESWC), LNCS, Springer, 2008, pp. 782–786.

[14] N. Fridman Noy, A. Chugh, W. Liu, M. A. Musen, A framework for ontology evolution in collaborative environments, in: Proc. of the 5th International Semantic Web Conference (ISWC), vol. 4273 of LNCS, Springer, 2006, pp. 544–558.

[15] N. Fridman Noy, S. Kunnatur, M. Klein, M. Musen, Tracking changes during ontology evolution, in: Proc. of the Third International Semantic Web Conference (ISWC), vol. 3298 of LNCS, Springer, 2004, pp. 259–273.

[16] N. Fridman Noy, M. A. Musen, Ontology versioning in an ontology management framework, IEEE Intelligent Systems 19 (4) (2004) 6–13.

[17] N. Fridman Noy, T. Tudorache, S. de Coronado, M. A. Musen, Developing biomedical ontologies collaboratively, in: Proc. of AMIA Symposium, 2008.

[18] A. Gómez-Pérez, M. C. Suárez-Figueroa, Scenarios for building ontology networks within the NeOn methodology, in: Proc. of the fifth international conference on Knowledge capture (K-CAP), ACM, New York, NY, USA, 2009, pp. 183–184.

[19] F. W. Hartel, S. de Coronado, R. Dionne, G. Fragoso, J. Golbeck, Modeling a description logic vocabulary for cancer research, Journal of Biomedical Informatics 38 (2) (2005) 114–129.

[20] M. Hartung, T. Kirsten, E. Rahm, Analyzing the evolution of life science ontologies and mappings., in: Proc. of the 5th international workshop on Data Integration in the Life Sciences (DILS), vol. 5109 of LNCS, Springer, 2008, pp. 11–27.

[21] M. Hepp, P. De Leenheer, A. de Moor, Y. Sure (eds.), Ontology Management, Semantic Web, Semantic Web Services, and Business Applications, vol. 7 of Semantic Web And Beyond Computing for Human Experience, Springer, 2008.

[22] M. Horridge, S. Bechhofer, The OWL API: A Java API for working with OWL 2 ontologies, in: OWLED, vol. 529 of CEUR Workshop Proceedings, CEUR-WS.org, 2009.

[23] M. Horridge, B. Parsia, From justifications towards proofs for ontology engineering, in: Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, KR, AAAI Press, 2010.

[24] M. Horridge, B. Parsia, U. Sattler, Explanation of OWL entailments in Protege 4, in: International Semantic Web Conference (Posters & Demos), vol. 401 of CEUR Workshop Proceedings, CEUR-WS.org, 2008.

[25] M. Horridge, B. Parsia, U. Sattler, Laconic and precise justifications in OWL, in: The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC, vol. 5318 of Lecture Notes in Computer Science, Springer, 2008, pp. 323–338.

[26] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen, From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language, Journal of Web Semantics 1 (1) (2003) 7–26.

[27] L. Iannone, I. Palmisano, A. L. Rector, R. Stevens, Assessing the safety of knowledge patterns in owl ontologies, in: The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC, vol. 6088 of Lecture Notes in Computer Science, Springer, 2010, pp. 137–151.

[28] E. Jiménez-Ruiz, B. Cuenca Grau, I. Horrocks, R. Berlanga, Building ontologies collabo-

ratively using ContentCVS, in: Proc. of the International Workshop on Description Logics (DL), vol. 477 of CEUR Workshop Proceedings, 2009.

[29] E. Jiménez-Ruiz, B. Cuenca Grau, I. Horrocks, R. Berlanga, ContentCVS: A CVS-based Collaborative ONTology ENgineering Tool (demo), in: Proceedings of the 2nd International Workshop on Semantic Web Applications and Tools for Life Sciences (SWAT4LS 2009), Amsterdam, The Netherlands, vol. 559 of CEUR Workshop Proceedings, 2009.

[30] E. Jiménez-Ruiz, B. Cuenca Grau, I. Horrocks, R. Berlanga, Ontology integration using mappings: Towards getting the right logical consequences, in: Proc. of the European Semantic Web Conference (ESWC), Springer LNCS, 2009, pp. 173–187.

[31] E. Jiménez-Ruiz, B. Cuenca Grau, U. Sattler, T. Schneider, R. Berlanga, Safe and economic re-use of ontologies: A logic-based methodology and tool support, in: The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC, vol. 5021 of Lecture Notes in Computer Science, Springer, 2008, pp. 185–199.

[32] A. Jimeno-Yepes, E. Jiménez-Ruiz, R. Berlanga, D. Rebholz-Schuhmann, Reuse of terminological resources for efficient ontological engineering in life sciences, BMC Bioinformatics 10 (Suppl 10).

[33] A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in: Proc. of the 6th International Semantic Web Conference (ISWC), vol. 4825 of LNCS, Springer, 2007, pp. 267–280.

[34] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca Grau, Repairing unsatisfiable concepts in OWL ontologies, in: Proc. of the 2nd European Semantic Web Conference (ESWC), vol. 4011 of LNCS, Springer, 2006, pp. 170–184.

[35] A. Kalyanpur, B. Parsia, E. Sirin, J. A. Hendler, Debugging unsatisfiable classes in OWL ontologies, Journal of Web Semantics 3 (4) (2005) 268–293.

[36] B. Konev, D. Walther, F. Wolter, The logical difference problem for description logic terminologies, in: Proc. of the 4th International Joint Conference on Automated Reasoning (IJCAR), vol. 5195 of LNCS, Springer, 2008, pp. 259–274.

[37] K. Kotis, On supporting HCOME-3O ontology argumentation using semantic wiki technology, in: On the Move to Meaningful Internet Systems: OTM 2008 Workshops, 2008, pp. 193–199.

[38] K. Kotis, G. A. Vouros, Human-centered ontology engineering: The HCOME methodology, Knowl. Inf. Syst. 10 (1) (2006) 109–131.

[39] A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz, An infrastructure for searching, reusing and evolving distributed ontologies, in: Proc. of the International World Wide Web Conference (WWW), 2003, pp. 439–448.

[40] J. L. V. Mejino Jr., C. Rosse, Symbolic modeling of structural relationships in the Foundational Model of Anatomy, in: Proc. of First International Workshop on Formal Biomedical Knowledge Representation (KR-MED), 2004, pp. 48–62.

[41] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, OWL 2 Web Ontology Language: Profiles, W3C Recommendation (2009).

[42] B. Motik, P. Patel-Schneider, B. Parsia, OWL 2 web ontology language structural specification and functional-style syntax, W3C Recommendation (2009).

[43] R. Palma, P. Haase, O. Corcho, A. Gómez-Pérez, Change representation for OWL 2 ontologies, in: Proceedings of the Sixth OWLED Workshop on OWL: Experiences and Directions, 2009.

[44] R. Palma, P. Haase, Q. Ji, D1.3.2. Change management to support collaborative workflows,

NeOn Deliverable available at: http://www.neon-project.org/ (December, 2008).
[45] A. L. Rector, J. Rogers, Ontological and practical issues in using a description logic to represent medical concept systems: Experience from GALEN, in: Proc. of Reasoning Web, 2006, pp. 197–231.
[46] T. Redmond, M. Smith, N. Drummond, T. Tudorache., Managing change: An ontology version control system, in: Proc. of OWL: Experiences and Directions, OWLED, 2008.
[47] S. Schlobach, Z. Huang, R. Cornet, F. van Harmelen, Debugging incoherent terminologies, Journal of Automated Reasoning 39 (3) (2007) 317–349.
[48] D. Schober, J. Malone, R. Stevens, Observations in collaborative ontology editing using Collaborative Protégé, in: Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2009), 2009.
[49] J. Seidenberg, A. L. Rector, A methodology for asynchronous multi-user editing of semantic web ontologies, in: Proc. of the 4th International Conference on Knowledge Capture (K-CAP), ACM, 2007, pp. 127–134.
[50] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, J. Web Sem. 5 (2) (2007) 51–53.
[51] H. Sofia Pinto, C. Tempich, S. Staab, Ontology engineering and evolution in a distributed world using DILIGENT, in: Handbook on Ontologies, Springer, 2009, pp. 153–176.
[52] K. Spackman, SNOMED RT and SNOMED CT. Promise of an international clinical ontology, M.D. Computing 17.
[53] H. Stuckenschmidt, M. Klein, Reasoning and change management in modular ontologies, Data Knowl. Eng. 63 (2) (2007) 200–223.
[54] B. Suntisrivaraporn, G. Qi, Q. Ji, P. Haase, A modularization-based approach to finding all justifications for OWL DL entailments, in: Proc. of the 3rd Asian Semantic Web Conference (ASWC), vol. 5367 of LNCS, Springer, 2008, pp. 1–15.
[55] C. Tempich, E. Simperl, M. Luczak, R. Studer, H. S. Pinto, Argumentation-based ontology engineering, IEEE Intelligent Systems 22 (6) (2007) 52–59.
[56] C. Tempich, H. Sofia Pinto, Y. Sure, S. Staab, An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT), in: The Semantic Web: Research and Applications, Second European Semantic Web Conference (ESWC), 2005, pp. 241–256.
[57] T. Tudorache, N. Fridman Noy, S. W. Tu, M. A. Musen, Supporting collaborative ontology development in Protégé, in: International Semantic Web Conference (ISWC), LNCS, Springer, 2008, pp. 17–32.