Data & Knowledge Engineering 70 (2011) 146–164

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Supporting concurrent ontology development: Framework, algorithms and tool

E. Jiménez Ruiz [a,*], B. Cuenca Grau [b,1], I. Horrocks [b,1], R. Berlanga [a,2]

[a] Llenguatges i Sistemes Informatics, Universitat Jaume I, 12071, Castellón, Spain
[b] Computing Laboratory, University of Oxford, Wolfson Building, OX13QD, Oxford, UK

ARTICLE INFO

Article history:
Received 22 October 2009
Received in revised form 4 October 2010
Accepted 4 October 2010
Available online 13 October 2010

Keywords:
Ontologies
Knowledge representation
Knowledge engineering
OWL
Semantic Web

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.

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 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.

* Corresponding author. Tel.: +34 964 728367.
E-mail addresses: ejimenez@uji.es (E. Jiménez Ruiz), berg@comlab.ox.ac.uk (B.C. Grau), ian.horrocks@comlab.ox.ac.uk (I. Horrocks), berlanga@uji.es (R. Berlanga).
[1] Tel.: +44 1865 283529.
[2] Tel.: +34 964 728367.

0169-023X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2010.10.001

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 [4,11,13,18,21,39,46,49,51,53,57]), 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 [23,25,33–35,47] 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 [11,13,14,17] (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