Versioning in Distributed Semantic Registries

Christoph Ludwig
ludwig@fh-worms.de
University of Applied Sciences Worms
67549 Worms, Germany

Marc Wilhelm Küster
kuester@fh-worms.de
University of Applied Sciences Worms
67549 Worms, Germany

Graham Moore
gra@networkedplanet.com
Networked Planet
Oxford, UK

ABSTRACT

Change is a constant (not only) in the Semantic Web. Both instance data and ontologies evolve. These changes are a piecemeal process, morphing from version to version both for conceptualizations / ontologies and for the related instance data / individuals. Individual versions capture valuable information about ontologies and instance data that were valid at a given point in time.

In this article we present a design for versioning instance data and outline its particular role in ontology evolution. We describe an implementation strategy for versioning instance data in the Semantic Web in the light of existing approaches working change logs and change definition languages. We show how this strategy is implemented our Topic Map based semantic database Isidorus and is used in pan-European eGovernment applications. We also show how the versioning strategy plays together with a new Atom-based European specification for change propagation of distributed metadata.

General Terms

Semantic Registries; Topic Maps; Versioning; Semantic Web; Ontology Evolution

Keywords

Semantic Registries; Topic Maps; Versioning; Semantic Web; Ontology Evolution

1. AN APOLOGIA OF CHANGE

In change is rest, Heraclitus famously said 2500 years ago ([18], fragment 83). The world is in constant flux, and, change is one of the constants of our existence.

The Semantic Web is not immune from change. Changes can be on the simplest of levels — a person can move on from one job position to another one and thus invalidate the corresponding resource description. A company can cease to offer a given service and provide other services instead. Changes can be more drastic: government agencies can merge. Corporations can go bankrupt. Laws can be repealed. Whole states can split up.

Changes also impact concepts. New types of concepts can spring into being —– say, mobile services, new financial products or entirely redesigned organizational hierarchies with new types of job descriptions. Other concepts can be fundamentally redefined. Family and marriage were clearly delineated notions as little as twenty five years ago. They are no longer, and, if so, their conceptualizations are very different now. The concept of "phone" meant something quite different even five years ago from what it implies today.

Change is also fixture in software development. No serious development project would be complete without change management both on the business and on the technical level. In fact, a trend in software development is to transcend classical centralized single-repository version control systems such as CVS [2] or Subversion [19] and replace them with highly distributed, change-oriented systems such as Darcs [22], Mercurial [17] or Git [5] to embrace the reality of multiple changes happening in parallel.

With change also the concept of versioning needs to evolve — what does it mean to have versions in a distributed environment where assertions about a specific resource are made by a large number of different players and how does this impact its versioning?

2. STATE OF THE ART

Ontology evolution is an important, though not extremely active research area in knowledge modeling and knowledge management. Seemingly minor modifications to an ontology can have ripple effects and – as discussed in the literature – care has to be taken to maintain the consistency of both the ontology itself and its instance data. For instance, [23] suggests strategies for updating other ontology elements according to user defined policies if, say, a superclass ceases to exist.

That a given fact was true, that a given conceptualization was widely shared at a given point in time is an important fact in itself. Ontology evolution passes through a number of such fix points over time. So versioning plays a central, even though often overlooked role in ontology evolution.

However, in this article we shall look not at ontology evolution per se, but at a more general problem: strategies for versioning of distributed semantic information. As ontologies also constitute semantic information, ontology versioning is just an (important) special case. In typical scenarios,

iiWAS2008, November 24–26, 2008, Linz, Austria

though, the update of an ontology is much rarer than updates of the instance data that is governed by the ontology.

We therefore shall concentrate ourselves in this article on the versioning of instance data; since ontologies can be maintained alongside their instance data in the same semantic databases, this is no loss of generality. After demonstrating in Sect. 3 some of the issues one faces when versioning semantic networks, in particular Topic Maps based ones, we shall outline in Sect. 4 the versioning strategy for instance data we use in our semantic database / Topic Map engine Isidorus and how it can be generalized to the Semantic Web at large. In addition, we sketch in Sect. 5 how our versioning strategy plays together with a new Atom-based European specification for change propagation for semantic information.

2.1 Change logs

As soon as we attach the notion of versions to a knowledge base, it is natural to ask for the changes that occurred between its versions i and j. Being able to produce all changes is even more important in a distributed setting where ontologies and instance data are often replicated. As [10] points out, there are basically two options when propagating changes of a knowledge base in such an environment: One may either compare ad hoc the full local replica to its authoritative source or one keeps an explicit log of all change operations applied to the knowledge bases and merges these change operations as needed. Since comparing knowledge bases is computationally expensive and may incur a significant communication overhead, the latter option is much more reasonable.

The naïve approach represents change operations as a textual diff of the serializations of both versions in some suitable serialization format. This becomes quickly unmanageable, though, as it depends on access to the original serialization and is very fragile under semantics-preserving modifications of the serialization, cf. [1].

The literature (e. g. [9, 16, 1]) therefore concentrates on non-text based approaches for representing differences between knowledge base versions. [24] gives algorithms for RDF-based knowledge bases to decide the equivalence of change logs and to reduce sequences of elementary change operations. This allows to combine several individual changes into one change set if one wishes, say, to keep only monthly diffs for changes that date far back.

Other authors address the visualization of changes to a knowledge base [1, 16], so users can actually review change logs.

Neither the visualization nor the reduction of change logs is in the scope of this article. The Atom-feeds proposed in Sect. 5 do indeed constitute a change log, but this article's focus is rather on maintaining the versions implicitly generated by each change and on propagating them in a distributed setting.

2.2 The Change Definition Language (CDL)

[20] point out that it necessary to distinguish between a simple log of individual changes in an ontology — the version log — and the sequence of interpretations an ontology passes through — the evolution log. The paper develops a formal model for version logs alongside a language to express the changes in interpretation, the *Change Definition Language (CDL)*. Unlike the version log the evolution log also captures changes that are detected as indirect repercussions of direct change requests.

The versioning mechanism we demonstrate conforms to the formal model for version logs, treating versions as a total ordering between concept versions of the corresponding concepts in the registry space. We basically agree with the approach of treating a version as a tuple containing concept name, the defining axioms and a start and end-time. We shall however extend it to cover a wider spectrum of concept characteristics.

For the evolution log [20] defines the concept of *snapshots*, which can be either the global state of the set of all concepts at a particular point t or the state of an individual concept at t. The CDL uses both significations, whereas we shall propose to separate these two interpretations, reserving the term "snapshot" for the former.

2.3 The RDF Update Language

Modifying information in a knowledge base typically requires either the use of a specific interactive application or programming against a database specific API. If there were a widely accepted declarative update language, similar to the standardized exchange formats both for Topic Maps and RDF, then updates could be easily shared between replications using different implementations and even ontology changes causing bulk updates could be realized without re-building the database from scratch. The change logs mentioned in the previous subsections could then be implemented in a vendor neutral format in terms of this update language. [11] proposes such an update language for RDF based databases.

This article does not define an update language but it does outline in Sect. 5 an Atom-based vendor neutral protocol for propagating changes that also ensures the consistency of the database. Whereas [11] addresses in the first place bulk updates that are applied locally, our protocol supports heavily distributed scenarios where clients typically update their replication of a database in regular intervals over the Internet.

3. VERSIONING SEMANTIC NETWORKS: PROBLEMS

This work arose from two of the authors' involvement in the project team of the CEN/ISSS Workshop on Discovery of and Access to eGovernment Resources (eGov-Share [3]). The information this workshop is concerned with is foremost intended for human consumption rather than for automatic inferencing, whence the project team concentrated first on a Topic Maps [7] based datamodel and distribution protocol. In order to facilitate a wide adoption of the eventual specification — a so-called CEN Workshop Agreement (CWA) —, mappings to W3C's Resource Description Framework (RDF [8]) are planned as well, but here we concentrate on the Topic Maps based setting.

3.1 The Topic Maps Data Model

In the context of Topic Maps, the term *subject* has an intentionally broad notion: "A subject can be anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever." [6, Sect. 5.3.1]. A *topic* is a subject's unique representation within a topic map.

The Topic Maps Data Model [6] formalizes this representation and that of the topics' attributes (*characteristics*) and relations (*associations*) as an entity-relationship model.

3.1.1 Topic Map Items

Characteristics can be either *occurrences*, *topic names*, or *variant names*. Whereas topic names are basically human readable labels attached to a topic, occurrence items add information about the subject to a topic. Associations represent n-ary ($n \geq 1$) relations between subjects where each subject participates in the relation in a certain *role*. With the sole exception of the topic map item representing the topic map itself, all topic map items are typable where the types — classes in RDFS and OWL terminology — are again subjects and therefore represented by topics.

A noteworthy property of Topic Maps is that all statements about subjects (i.e., characteristics and associations as well as their roles) may be qualified by *scopes*, i. e. sets of topics that specify in which context the respective statement should be considered valid.

3.1.2 Identifiers and Merging

Each topic map item can be assigned an arbitrary number of *item identifiers* that can be used to refer to the respective item. In addition, one may also assign an arbitrary number of *subject indicators* and *subject locators* to topics. The differences are subtle but important: Whereas a topic's item identifier refers to the topic itself, a subject indicator identifies the subject represented by the topic (i. e. they are resources on the web). A subject locator, if present, can be used to actually access the subject. While subject locators may change in the course of time – an organization's home page may be relocated from one website to another, causing a change of the home page's URL – subject indicators are presumed to be persistent. Unfortunately, the TMDM itself blurs the line between item and subject identifiers in its topic equality rules and lacks any statement regarding the item identifiers' stability, which causes issues in a distributed setting where topic maps are expected to change.

The Topic Maps standard supports the aggregation of topic maps that, say, are maintained by different organizations. Even if the topic maps contain statements about common subjects – in which case there will be a corresponding topic in each of these topic maps – the aggregated topic map must represent the subject by a unique topic. Therefore, TMDM defines an algorithm for merging topic map items; this algorithm has to be applied whenever two items are equal according to the rules set forth by TMDM.

However, two topic items are considered equal if one's subject identifier appears among the other's item identifiers. Given the uncertainty of the item identifiers' stability, this can lead to unwarranted behavior if we consider not only static topic maps but account for changes in the course of time.

In consequence, item identifiers must be just as stable as subject identifiers, even though TMDM does not address this issue. Furthermore, authors and maintainers of topic maps should ensure that subject identifiers and item identifiers never coincide by accident, e. g. by a policy that requires the identifiers to be chosen from disjunct namespaces, and thus avoid this particular aspect of TMDM's equality rule for topic items.

3.2 Subject Centric Computing and Versions

One of the tenets of Topic Maps (and, under different labels, also RDF) is the subject centric computing. Users are typically interested in information on subjects rather than how this information was produced. Therefore, subject centric computing puts the focus on identifying subjects and providing semantic data about them rather than on the processes necessary to retrieve or compute this data.

However, we claim that a concept for identifying and maintaining different versions of this semantic data is indispensable for subject centric computing. That is because information on subjects changes, but the information history is valuable information as well. For example, the name of a person may change due to a marriage, but existing documents will still refer to the previous name that therefore has to be accessible as an superseded version of the person's name.

The need for a versioning concept is even more evident if the information on subjects is maintained by several parties that share their information. If no version is attached to the data exchanged then it becomes impossible to tell who has the more recent information and references that are established in the context of the currently available information will become stale or inconsistent if they cannot point to specific information versions.

Versioning cannot resolve another issue that arises in an distributed setting and that neither the Topic Maps nor the RDF / OWL specifications address: A consumer down the line may receive information on a subject through several paths. Currently, there is no generally agreed on way to tell who "owns" the subject (or specific properties of it), i. e., who is the authoritative source for information on this subject. This may affect either concepts that are part of an ontology (and therefore should be grounded in community agreement) or instance data that may be reasonably controlled by one party (cf. [13]).

The syndication protocol described in Sect. 5 therefore provides source locator prefixes that are matched against the identifiers of information sets to establish which data the server is claiming responsibility for. This scheme does not protect against a malicious server that assumes authority over namespaces it does not have, though, but it works in webs of trust.

4. IMPLEMENTATION STRATEGY

4.1 Isidorus

Isidorus[1] is a new Open Source semantic database implemented in Common Lisp. It was started as a replacement for an earlier Python-based engine that failed to meet the performance matrix to serve as one of the two reference implementations for the eGov-Share [3] specifications. Isidorus' datamodel consciously mirrors the TMDM structure, using Elephant [4] as a transparent high-performance persistence solution.

This proved to be a good design decision, surprisingly not only for Topic Map-oriented representations, but also for RDFS and OWL. However, TMDM has no idea of concept versioning, while one of the key goals of the eGov-Share specifications is to syndicate changes across heterogeneous semantic networks (cf. below, sec. 5).

[1] `http://www.isidor.us`

4.2 Abandoned Designs

Coming to grips with semantic versioning from very practical rather than theoretic point of view proved to be challenging. We tried out and quickly abandoned several design choices, amongst others:

- create new topics and associations for each new version and link the versions through specialized associations. While the most obvious mapping of concept changes, it would have created synchronization issues for the cross-references between concept versions with our underlying persistence solution [4]
- control versioning information through suitable conventions for the naming of identifiers. This we found to be too fragile.

4.3 Implementation

The chosen design realizes a version log in which instances of the `VersionInfo`-class are associated with all topic map constructs, cf. Fig. 1.

`VersionInfo` objects (VIOs) record start and end dates of a specific version of a topic map construct. Every change operation — creation, update or deletion — thus results in a new such object being associated to the changed construct. However, most changes have indirect repercussions since larger constructs such as topics are build through the attribution of many topic map constructs such as identifiers, names and occurrences. Changes to these constructs thus implicitly cause the encompassing construct to evolve without a new VIO being associated with that construct itself.

This particular design strategy works because of TMDM's strict equivalence rules for constructs. Topic characteristics and associations — data and object properties in OWL terminology — are immutable (other than for the addition of new identifiers) once created. Any change to their contents thus automatically results in ending the lifespan of one object and the instantiation of a new one. Furthermore, topics themselves persist once created even if they might loose all their properties.

Accessing a construct for time t automatically takes versioning into account and only returns it if it is or has been valid at a given time and with the identifiers it had at that point. This way all query operations automatically are version-aware.

4.4 Versioning as an Extension of the TMDM Datamodel

The versioning mechanism organically extends TMDM, though not without requiring one important addition to its present state. The relationship between a construct and its VIOs can be conceptualized as a set of associations between the reification of the construct and its versions, reified themselves as topics of a specific type. However, this would entail that all constructs could be reified in the first place, which works for characteristics and associations, but is in the current TMDM inexplicably not the case for identifiers.

4.5 A Model for Versioning RDF

This design applies equally well to instance data in RDF. Assuming a set of RDF triples s_t that are conceived as single logical unit (often an instance of an RDFS or OWL class or corresponding data or object properties), s_t can be reified using a sequence of normal triple reifications ([25], 3.3.1). The reification of s_t can then be linked to VIOs through a sequence of normal RDF triples / assertions — at least conceptually, as internally, many semantic databases provide a richer information model that can allow for direct links between the statements and their version objects (for a warning of the naïve use of triple reification cf. also [12]).

Defining an evolution log for RDF representations (cf. 5) would benefit from explicit equivalence rules for (reifications of) triples and sets of triples in RDF on the model of those in TMDM. Only then we can sensibly determine if any operation really updated a triple or rather deleted it and added a new, unrelated one. Furthermore, it profits from persistent resource identifiers. Even if a resource r ceases to exist, both old assertions must retain their anchor point and new assertions — say, "r was withdrawn at time t" — must continue to be possible.

Query operations in, say, SPARQL [21] must be version-aware and return valid hits against a given snapshot at time t. The way to specify t is implementation dependent.

4.5.1 Isidorus' Perspective

Isidorus works at the level of topics, characteristics and associations. These map neatly to OWL individuals, data properties and object properties, respectively. Versioning can thus work along the same lines for both TMDM-compliant and OWL/RDFS-compliant instance data [**TODO:** really true?] (though, in the latter case it cannot at present actually validate constraints). Reification of the construct is implicit through links between VIOs and the corresponding constructs, but could easily be explicated through triple reification for serialization.

5. PROPAGATING CHANGES

The upcoming European *Protocol for the Syndication of Semantic Descriptions* [14, 15] specifies a mechanism for the exchange of semantic descriptions. The protocol defines how a RESTful web service can publish an evolution log through a series of Atom-based syndication feeds that describe snapshots and evolutions to a collection of semantic descriptions. It also defines how a client should process the feeds provided by the service such that a local store is in sync. A client can synchronize with more than one server to act as an aggregator for semantic descriptions.

The CWA defines several layers of syndication feeds that must be provided by a conforming application:

- a "collections feed" that lists all collections that a registry hosts
- a feed for each collection which gives links to a snapshot feed and the evolution log
- one feed each for a list of snapshots[2] and for the evolution log

The specification also defines algorithms for the provision and processing of the different feeds on both server and client.

[2]A snapshot covers always the complete collection.

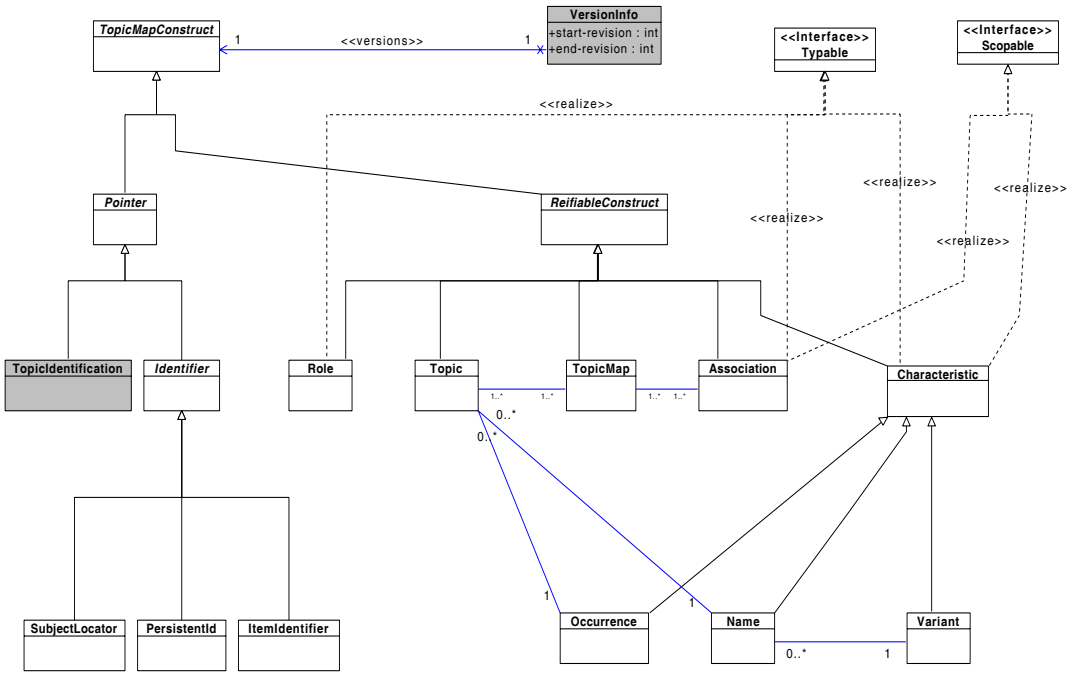

Figure 1: Versioning in the Isidorus datamodel (non-TMDM classes shaded)

5.1 Fragments

Here we shall concentrate ourselves on the design of the evolution log dubbed the *fragments feed*. The fragments feed is centered around logical bundles of individual changes including changes that result as repercussions of other modifications. The log itself is structured as a sequence of Atom entries:

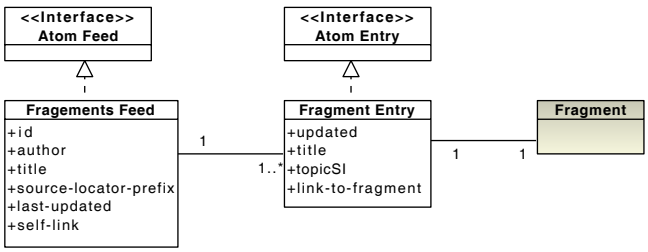

Figure 2: Fragments feed

A fragments feed has the normal Atom-metadata such as feed id, author and title. It also identifies a prefix — the `ServerSrcLocatorPrefix` — for all properties that it is responsible for. For a distributed semantic network many partners can publish properties for a given topic / a given object. A server can only update such properties that it itself is responsible for.

Most importantly, the fragments feed includes entries for every logical change bundle: the date it was released (which does not necessarily have to coincide with the time-stamp of any one individual change), a human-readable title — often an indication of the topic(s) affected —, one or more machine-processable identifiers (`topicSI`) that point to the affected metadata sets in the fragment. Finally, an entry links to a file with the actual serialization in either XTM or RDM/XML.

Fragments describe the new state of given instance data after all changes have been applied. If existing, a client first deletes all corresponding old instance data from the server in question that the feed is responsible for and replaces it with the new data, provided that the new state is consistent with all ontological constraints.

A fragment closes over all references in the evolved instance data. This is particularly important for XTM where topics reference each other through volatile ids which in turn map to persistent identifiers. A fragment therefore has to contain stubs for all referenced topics. For RDF referential completeness is not as central.

5.2 Snapshots

In addition to the fragments a server also must publish a feed in which all entries correspond to snapshots of the collection at a particular point in time. This allows clients who may not have had any exposure to the server's semantic descriptions to bootstrap themselves. Once so initialized, they can then apply the individual fragments.

6. CASE STUDY

To concretize the above elaborations let us look at a real, if slightly abstracted example. A German town *X* operates a web portal that lists (amongst others) the type of services it provides to its citizens and businesses. These can range from passport renewal and the registration of births to issuing building permits. The portal lists detailed contact information, opening hours etc. that it retrieves from a database with an interface that conforms to the aforementioned syndication protocol.

The portal, however, also includes services that are operated by companies on behalf of the town in areas such as waste management. The portal will point citizens to

company Y for all services related to waste management. However, the town does not by itself have the mandate to regulate Y's internal *modus operandi* and to specify many of the details that citizens need to know to interact with Y.

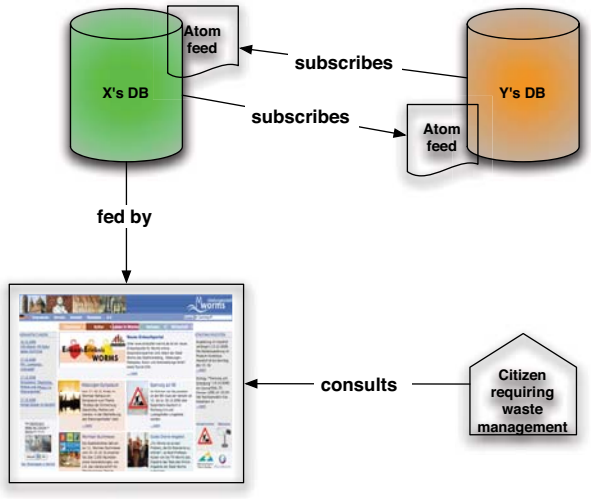

Figure 3: Waste Management

The town's database therefore defines a stub topic for the waste management service, links it to the corresponding descriptive metadata and publishes it in its Atom feed. The corresponding details are managed within a second database operated by Y. Y imports the topic and republishes a new version of it that is enhanced with contact details, opening hours etc. X reimports the enhanced version and merges it into its datastore and merges it with the existing description. Should the internal responsibilities in Y change, Y publishes a new version of the topic for X's consumption, cf. Fig. 4.

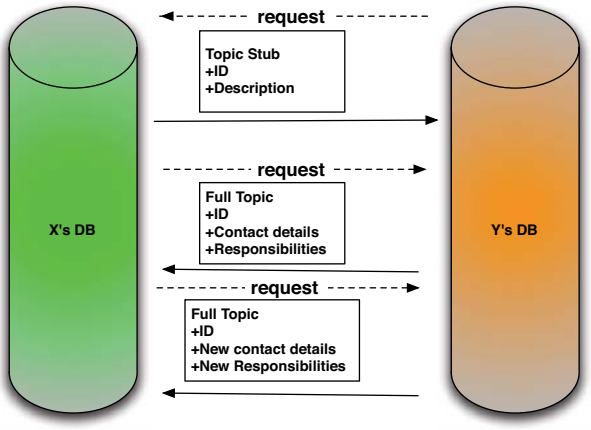

Figure 4: Messages exchanged

Even though Y's feed need only contain statements e. g. on contact details it itself is responsible for. X can happily maintain its own set of statements e. g. linking the topic to its internal taxonomy of services. Their different `ServerSrcLocatorPrefix`es keeps the players from applying changes to statements that did not originate from the server in question.

7. CONCLUSION

We demonstrated a simple and organic implementation approach for versioning in distributed semantic registries that fits particularly well the Topic Maps Data Model but can equally be applied to RDF. We found that we needed some augmentation in TMDM. However, these augmentations naturally fill a logical oversight in TMDM since they would anyway be required to make statements about identifiers within the TMDM.

Our approach makes it possible to reason about specific versions of a subject which, as we argued, is indispensable for subject centric computing. This way versions of instance data become addressable entities and hence instance data in their own right.

Finally, we outlined a syndication protocol that facilitates the propagation of changes in semantic databases. This protocol, that is part of an upcoming European specification, provides both a version log and snapshots for all collections hosted at a particular site.

8. REFERENCES

[1] T. Berners-Lee and D. Connoly. Delta: An ontology for the distribution of differences between RDF graphs. `http://www.w3.org/DesignIssues/Diff` (2008-10-04).

[2] P. Cederquist et al. Version management with CVS. `http://ftp.gnu.org/non-gnu/cvs/source/stable/1.11.23/cederqvist-1.11.23.pdf` (2008-10-10).

[3] Discovery of and Access to eGovernment Resources. CEN/ISSS Workshop, 2008. `http://www.cen.eu/cenorm/sectors/sectors/isss/workshops/wsegovshare.asp` (2008-10-09).

[4] I. Eslick, R. L. Read, et al. Elephant: A persistent object metaprotocol and database for Common Lisp. Version 0.9, `http://common-lisp.net/project/elephant/` (2008-10-11).

[5] D. Greaves et al. Git user's manual. `http://www.kernel.org/pub/software/scm/git/docs/user-manual.html` (2008-10-10).

[6] I. JTC1/SC34/WG3. ISO/IEC 13250-2, Topic Maps Data Model. Updated Draft `http://www.isotopicmaps.org/sam/sam-model/2008-06-03/` (2008-10-09).

[7] I. JTC1/SC34/WG3. ISO/IEC 13250, Topic Maps, 2002. `http://www.isotopicmaps.org/` (2008-10-09).

[8] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and abstract syntax. W3C Recommendation, 2004. `http://www.w3.org/TR/rdf-concepts/` (2008-10-09).

[9] S. R. Kruk, M. Völkel, M. Synak, Y. Sure, and W. Winkler. SemVersion: A versioning system for RDF and ontologies. In *2nd European Semantic Web Conference, 2005*, Heraklion, Crete, Greece, June 2005.

[10] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302, 2003.

[11] M. Magiridou, S. Sahtouris, V. Christophides, and M. Koubarakis. RUL: A declarative update language for RDF. In Y. Yolanda Gil Enrico Motta Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *The Semantic Web – ISWC 2005*, volume 3729 of *LNCS*, pages 506–521. Springer, 2005.

[12] D. McDermott and D. Dou. Representing disjunction and quantifiers in RDF. Technical report, Yale Computer Science Department, 2002. `http://www.w3.org/TR/rdf-concepts/` (2008-10-09).

[13] R. Meersman. Community-grounded semantics, methodology and tools for enterprise interoperability, 2008. DEST 2008 Keynote presentation, `http://www.ieee-dest.curtin.edu.au/2008/slides/Robert.ppt` (2008-10-12).

[14] G. Moore and M. W. Küster. CWA Part 1b: protocol for the syndication of semantic descriptions. Draft CWA, 2008. `http://www.egovpt.org/fg/CWA_Part1b` (2008-10-09).

[15] G. Moore and M. W. Küster. Protocol for the syndication of semantic descriptions. In *Subject-centric Computing. Fourth International Conference on Topic Maps Research and Applications, TMRA 2008*, pages 223–232, Leipzig, 2008. Leipziger Beiträge zur Informatik: XII.

[16] N. F. Noy, S. Kunnatur, M. Klein, and M. A. Musen. Tracking changes during ontology evolution. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *The Semantic Web – ISWC 2004*, volume 3298 of *LNCS*, pages 259–273. Springer, 2004.

[17] B. O'Sullivan. Distributed revision control with Mercurial. `http://hgbook.red-bean.com/hgbook.pdf` (2008-10-10).

[18] G. Patrick, editor. *The Fragments of the Work of Heraclitus of Ephesus on Nature*. N. Murray, Baltimore, 1889.

[19] M. Pilato, B. Collins-Sussman, and B. Fitzpatrick. *Version Control With Subversion*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2004.

[20] P. Plessers, O. D. Troyer, and S. Casteleyn. Understanding ontology evolution: A change detection approach. *Web Semant.*, 5(1):39–49, 2007.

[21] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 01 2008. `http://www.w3.org/TR/rdf-concepts/` (2008-10-09).

[22] D. Roundy. Darcs: distributed version management in haskell. In *Haskell '05: Proceedings of the 2005 ACM SIGPLAN workshop on Haskell*, pages 1–4, New York, NY, USA, 2005. ACM.

[23] L. Stojanovic, A. Maedche, N. Stojanovic, and R. Studer. Ontology evolution as reconfiguration-design problem solving. In *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*, pages 162–171, New York, NY, USA, 2003. ACM.

[24] Y. Tzitzikas and D. Kotzinos. (semantic web) evolution through change logs: Problems and solutions. In *AIAP'07: Proceedings of the 25th IASTED International Multi-Conference*, pages 610–615, Anaheim, CA, USA, 2007. ACTA Press.

[25] RDF Semantics. W3C Recommendation, 02 2004. `http://www.w3.org/TR/rdf-mt/` (2008-10-09).