Submitted on: 6/19/2014

Versioning Vocabularies in a Linked Data World

Diane I. Hillmann
Metadata Management Associates LLC), Jacksonville, NY, USA.
E-mail address: metadata.maven@gmail.com

Gordon Dunsire
Independent Consultant, Edinburgh, Scotland.
E-mail address: gordon@gordondunsire.com

Jon Phipps
Metadata Management Associates LLC, Jacksonville, NY, USA.
E-mail address: jonphipps@gmail.com

Copyright © 2014 by Diane Hillmann, Gordon Dunsire & Jon Phipps. This work is made available under the terms of the Creative Commons Attribution 3.0 Unported License: http://creativecommons.org/licenses/by/3.0/

Abstract:

Policies regarding change management in open or public vocabularies used in the context of Linked Open Data have lagged behind those driving other web-based communities of practice. A fresh emphasis on vocabulary management and maintenance has begun to emerge, as the reliance on potentially volatile vocabularies, and the implications of their ongoing growth and change, has begun to permeate the conversation.

Particularly in libraries, where management of commonly used vocabularies has long been a community-wide activity, management of vocabularies has been seen as the realm of larger institutions and organizations. This centralized control has been workable (if slow to evolve to incorporate new needs) so long as data distribution has also been centralized, but this pattern of distribution has become more questionable as a transition to the more open world of linked data begins to demonstrate the inflexibility of traditional practices. As more attention shifts to new vocabulary standards and usages outside libraries, researchers and innovative organizations have sought to take advantage of this boom in interest, but unlike librarians, they have little experience in implementation over time.

Merging the technology of the Semantic Web with the information management experience of libraries seems a reasonable strategy, but better understanding by all of where practices must change is critical.

Keywords: Vocabulary management, semantic versioning, RDA, change management, ontologies.

Preamble

This paper uses the term "vocabularies" for structured sets of information typically used in cultural heritage resource discovery services. They are categorized in the semantic Web environment as element sets, value vocabularies, and datasets, and represented as data in Resource Description Framework (RDF) for use in linked data applications. The paper uses a weak definition of "ontology" as representing intrinsic and extrinsic meaning – information minus the data – applicable to contextualized datasets and knowledge organization system vocabularies as well as element sets.

Background and introduction

The application of digital technology to library services since the 1960s has driven the evolution of centralized models for library metadata creation and distribution. These have been effective in allowing additions and changes to bibliographic schema and terminologies to be integrated into the entire body of legacy data in a predictable way, enabling libraries and their system vendors to effectively maintain the critical stability of their systems and data. The openness of processes has varied, for example from community participation in the management of change over time in the semantics of the predominate standard, MARC 21, to the relatively opaque curtain behind which, until recently, LCSH was managed by the Library of Congress (LC). Nonetheless, the change process was well-documented, relatively glacial, and effectively communicated to soften the impact.

The majority of changes managed via this approach have been easily implemented by the major data distribution entities and most of the system vendors that libraries rely upon. This centralized architecture has its functional gaps--some changes in LCSH practice, for instance, required extensive human intervention. One example of this gap was the separation of the LCSH heading "Nurses and Nursing" to "Nurses" and "Nursing." A semantic refinement such as this cannot be handled by machine; it was necessary to examine the record or the item to determine which term was appropriate, but the tradition of collaboration and collective activity in libraries pulled in many hands to lighten the workload.

There are other issues with the traditional model, perhaps most obviously the economic impact on smaller institutions with fewer staff to interact with pull processes for synchronizing the local catalog with changes made to the common, "union" database. Libraries who purchase data from vendors sometimes incur additional charges--and always additional work--to replace changed records. There are also problems with timeliness: the process of approving changes in semantics or encoding practice is often measured in years, and implementation can take considerable time to permeate the environment of closely tied central databases and local library catalogs dependent on those databases.

The most recent digital technology to emerge for library applications is the Semantic Web and linked data. There is significant investment in the publication of library datasets such as the catalogues of national libraries, element sets such as ISBD and BIBFRAME, and value vocabularies such as UNIMARC code lists and RDA terms. However, efforts by some large vocabulary owners to begin implementing more linked data friendly processes within legacy vocabularies, though welcomed, have too often fallen short because of reliance on older practices as well as a lack of understanding of semantic Web requirements. This is understandable: best practices are not yet available in either the library community or the semantic Web to ensure that vocabulary changes are managed efficiently and effectively.

As the importance of vocabularies for linked data distribution is becoming more recognized, the growing decentralization this implies for libraries can be intimidating to many practitioners. The notion that in this new environment, each user must figure out how to manage updating, in too many cases without services providing notification of changes, much less automated updating, induces panic. The situation as it stands discourages the use of any linked data vocabularies, much less the variety of vocabularies that a project or institution might require. Without a model for change management that can be used broadly by both data providers and users, adoption of appropriate vocabularies will necessarily be slow, expensive, and frustrating.

Evolving change management models

Most of us old enough to have witnessed the personal computer revolution and subsequent growth of mobile devices have lived through several stages of evolution as developers of applications (not to mention "apps") coped with the necessity of updating their products as operating systems changed, competition for users grew, and functionality sought by customers became more sophisticated. Operating systems, software applications, and open standards such as HTML and JavaScript are increasingly interdependent and a seemingly minor change can have a devastating ripple effect. Current practices for updating software optimize fast distribution of changes and are increasingly automatic, despite past emphasis on user control in an effort to avoid malware.

Software updates in general use version numbering to identify for users, and updating software, the version of individual software packages on a computer. Over time the software industry has refined their practices to be able to indicate via the version number the extent of change represented in an update. The software development community has recently begun to move toward a formal specification of version management known as "Semantic Versioning".

> "This is not a new or revolutionary idea. In fact, you probably do something close to this already. The problem is that "close" isn't good enough. Without compliance to some sort of formal specification, version numbers are essentially useless for dependency management. By giving a name and clear definition to the above ideas, it becomes easy to communicate your intentions to the users of your software. Once these intentions are clear, flexible (but not too flexible) dependency specifications can finally be made." [1]

The semantic versioning proposal for best practices noted above focuses on the problem of "dependencies", recognizing that in the software realm, coping with change is complicated by the practice of using applications in combination to accomplish specific goals, where each part of the "package" might be dependent on different operating systems or versions of other parts of the package.

> "A simple example will demonstrate how Semantic Versioning can make dependency hell a thing of the past. Consider a library called "Firetruck." It requires a Semantically Versioned package named "Ladder." At the time that Firetruck is created, Ladder is at version 3.1.0. Since Firetruck uses some functionality that was first introduced in 3.1.0, you can safely specify the Ladder dependency as greater than or equal to 3.1.0 but less than 4.0.0. Now, when Ladder version 3.1.1 and 3.2.0

> become available, you can release them to your package management system and know that they will be compatible with existing dependent software."

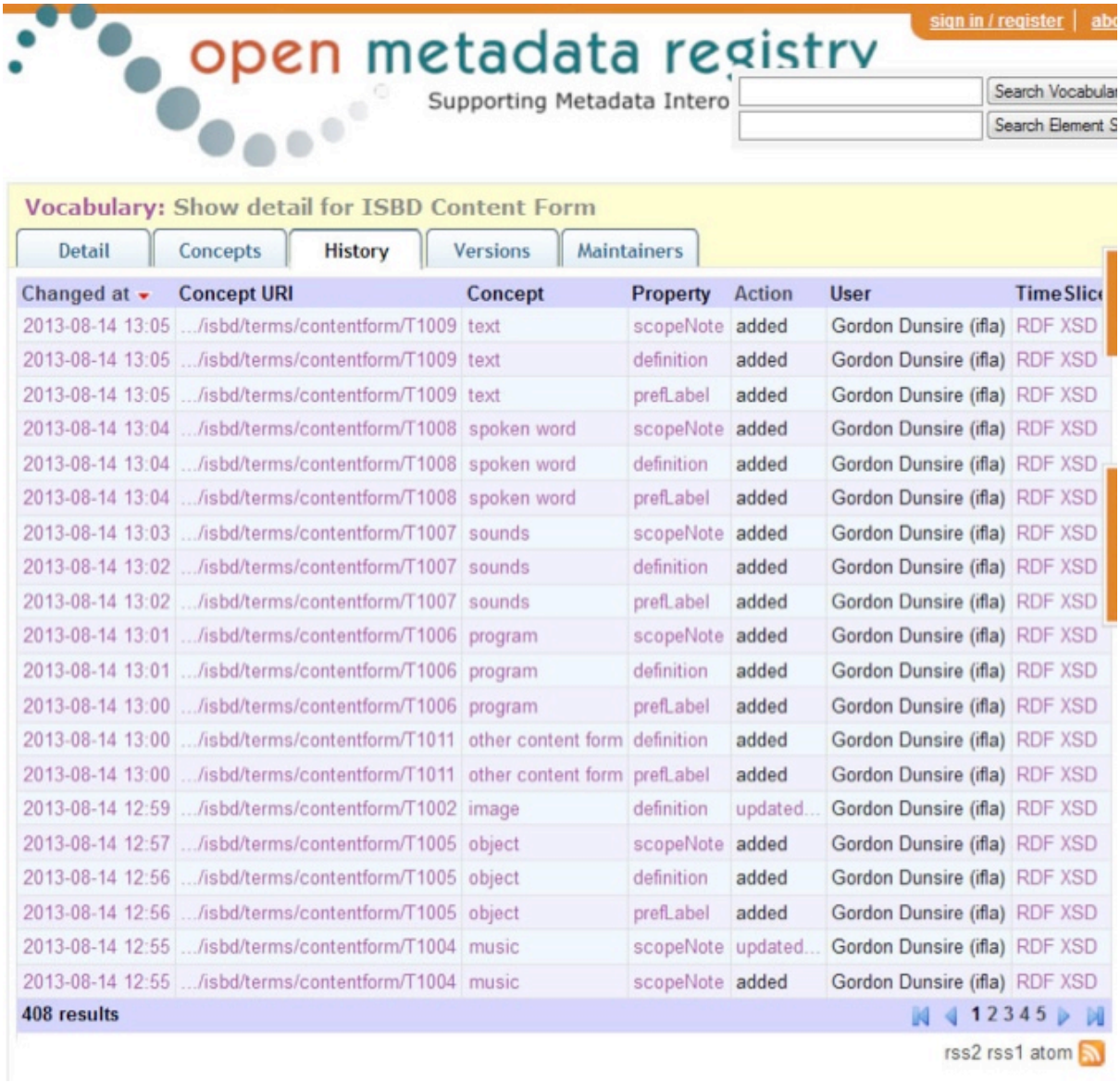
There have been some attempts to apply semantic versioning principles to ontologies, making the point that there are more similarities with the requirements for software than differences, as well as some general similarities to the management of application programming interfaces:

> "OWL ontologies should be semantically versioned, which means two things:
> * make the ontology's version identifier structured & meaningful, i.e., encode some meaning in the string of characters that makes up the version identifier; and
> * change the version identifier according to well-understood, public, and reasonable rules.
> Which suggests, of course, that a version identifier, plus a strategy for changing version identifiers, is a simple signaling mechanism intended to make multi-party coordination games cheaper and less disruptive for the participants. Consumers and producers of an ontology, no less and no more than of an API, are engaging in a multi-party coordination game in which costs should be kept as low as possible. Semantic versioning is one such cost control mechanism." [2]

It seems clear that in order to use a semantic versioning model to manage a similar level of complexity across the web itself requires that vocabulary managers and management systems pay better attention to the way they capture and describe change, focusing their effort at a very granular level, not necessarily at the traditional "record" level so ingrained in current library authority control systems.

The Open Metadata Registry (OMR) has been using the detailed history information it collects since 2006 to enable time-defined snapshots and named versions. The process is described in detail in a Registry Blog post. [3]

The image below, showing a page from the History tab for ISBD Content Form vocabulary, shows that all changes to that vocabulary can be viewed, including which authorized administrator or maintainer made the change and whether the change was an addition or an update. The last column, which supports the view of a time-delimited "slice" of the vocabulary itself, is the basis for the creation of named versions, accessible behind the Versions tab.

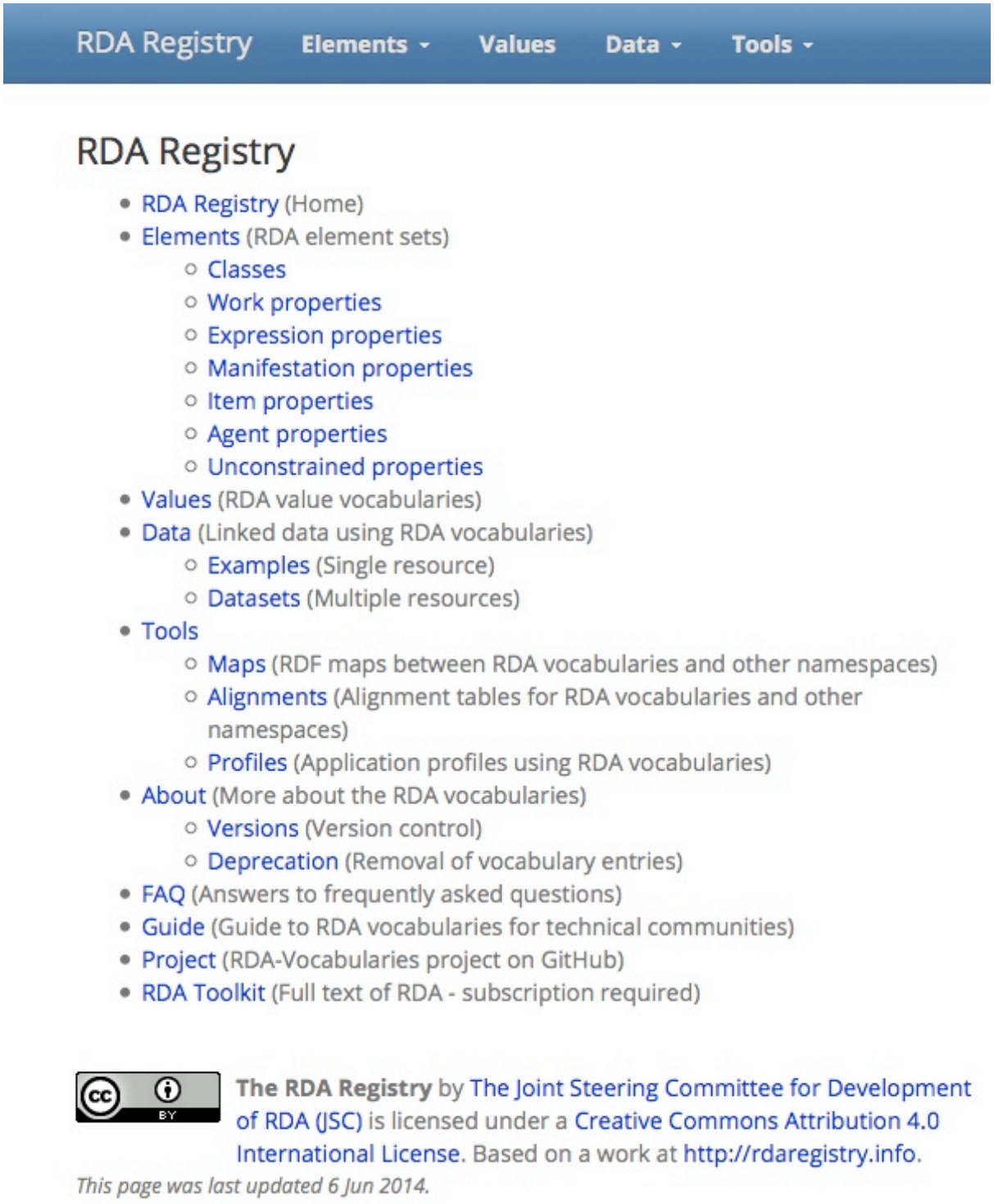


The expectation was that as value vocabulary usage began moving beyond the human-readable “heading” function used in library systems towards a more sophisticated, automated environment, such services would be welcomed. In fact, the capability has been little used, and was never implemented in the element set portion of the OMR (though detailed history is maintained there as well). The OMR is currently building new change management capability using GitHub, which is a better known and understood set of services and more likely to be the basis for developing change control in the OMR in future.

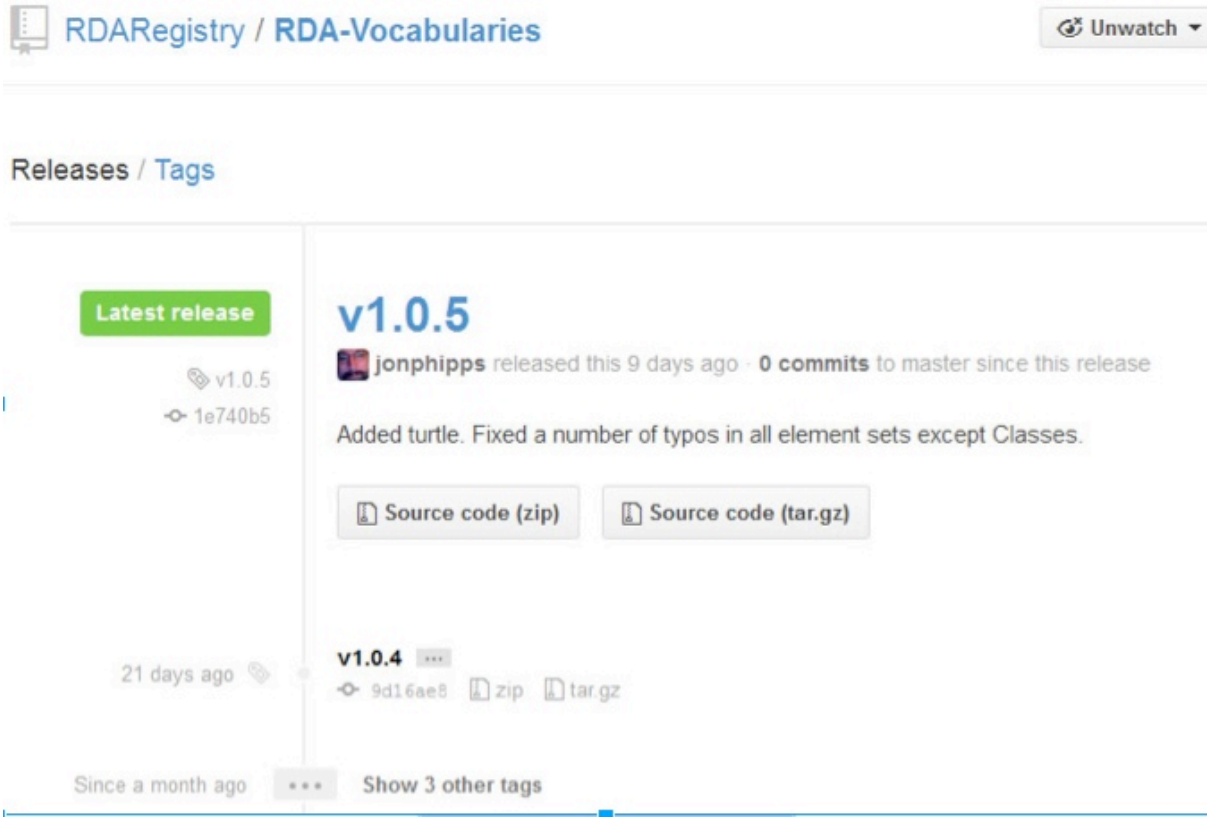
The basis for GitHub is Git, a distributed version control system distinguished from its predecessors by its view of data, rather than its user interface [4]. GitHub builds upon this software, bringing in a web-based hosting service, a more standardized, stable and secure workflow plus additional services supporting community development and social networking. Because the GitHub user interface is relatively simple and optimized for groups and projects, it has become ubiquitous in software development, and increasingly in vocabulary management. Despite its origins in the software community, software experience isn’t required to usefully participate.

Because projects are built up from a hosted platform, Github supports services largely missing from vocabulary development projects, in particular the documentation for the vocabulary and the provision of multiple flavors of output. The image below shows part of a Github webpage under development for the RDA element sets, giving access to many different formats of the RDA properties for manifestations.

The image below is the left half of the home page, showing the range of information available.



The image below shows another Github page under development for RDA with the new change management features for the OMR data.



Envisioning a well maintained future

What kind of a world can we envision using well-supported change management? Returning to the model of personal computing software, it's clear that there are good financial reasons for developers to invest in building their software to interact smoothly with the variety of customers. Their goal is to entice their users to continue to use their software, to purchase or download (if the software is free) new versions and bug fixes, and, importantly, to review the software in ways designed to attract even more users. The shift of some software sales from bespoke websites to platforms like iTunes and Google Play have provided marketing opportunities, easy discovery, reviewing and rating, and, for better or worse, standards with which software must comply to participate.

The aspects that make the environment work are few but critical, starting with a shared model and vocabulary for describing change, primarily for communicating with machines. The builders of operating systems and hardware have a parallel need to support these software applications, since they are also evaluated on the ease with which their products can interact with applications of all kinds.

In the vocabulary development community adoption of modern technology-focused maintenance models has been held back for several reasons, perhaps chief among them the lack of understanding of how vocabularies should work in a semantic Web environment. In some specific communities, like libraries, despite the strong tradition of use and development of shared descriptive vocabularies, the transition from a well-understood but closed environment to an open one with different technical requirements has been slow and badly

supported by existing institutions and organizations. The focus in the library community is still largely on vocabularies as human-readable and interpretable text, and the models for maintenance, while nicely collaborative, remain human centered and expensive.

In a recent blog post on some changes in vocabularies managed by the Library of Congress, one of the authors of this paper noted:

> "Large public vocabularies have tended to make an incomplete transition from print to online, getting stuck, like LC, attempting to use the file management processes of the print era to manage change behind a 'service' front end that isn't really designed to do the job it's being asked to do. What needs to be examined, soon and in public, is what the relationship is between these files and the legacy data which hangs over our heads like a boulder of Damocles. Clearly, we're not just in need of access to files (whether one at a time or in batches) but require more of the kinds of services that support libraries in managing and improving their data. These needs are especially critical to those organizations engaged in the important work of integrating legacy and project data, and trying to figure out a workflow that allows them to make full use of the legacy public vocabularies." [5]

Even recent cross-community discussions of vocabulary issues, such as those under the aegis of the Dublin Core Metadata Initiative (DCMI), have focused on problems around discovery of vocabularies, models of acceptable re-use, governance, and documentation. [6]

Vocabulary maintenance is seen as a general good, but how it could best be accomplished is seldom discussed.

Looking ahead

What could a maintenance model based on Semantic Versioning provide? Perhaps most importantly, it must provide a proven method for defining several levels of semantic interoperability and stability that could be the basis of automated notifications and updating for users of open vocabularies. Numeric version numbers, as used by software, could fairly easily be adapted for vocabulary versions, although there are a few areas where terms of practice used for software changes, like "patch" have no simple equivalent in the vocabulary world.

Clark recognizes that there are differences between software and ontologies that must be addressed:

> "The last three are the hardest: what conditions constitute changes to major, minor, and patch fields? These are harder because OWL ontologies are in some sense quite different from programming language APIs. We've so far been riding the high of their similarity, but now we have to deal substantively with their dissimilarity.
> We want to end up with a versioning scheme that sends this set of signals:
> * if there's a patch change, consumers can safely ignore that version
> * if there's a major change, consumers should not ignore that version
> * if there's a minor change, consumers need to investigate further
> Admittedly, the minor change ambiguity is not ideal, but for now we can't seem to do any better.

> This is a kind of Goldilocks or binning problem: what counts as a big, medium, and minor change to an OWL ontology? Someone's always unhappy, no matter what solution one offers to this kind of problem."

Development of best practices and policy development in this environment is still emerging from the software world, but a few efforts should be mentioned. The community around GitHub is still working this ground, and the GitFlow policy statements are succinct and simple. [7] The Ruby community is a bit more tentative in explaining their rules, but the categorizations are still very similar. [8]

The OMR team, in developing the RDA Vocabularies has attempted some policy statements, based on the core principles of http://semver.org, but generalized to the needs of vocabulary managers:

- All public vocabularies *MUST* declare a version number, including those still in development and not yet "published" or "released" (typically this would be a 'pre-release' version of "0.X.X".
- The version number *MUST* be declared as a meaningful (semantic) 3 segment number (1.2.3), with segment meanings defined as follows:
    1. MAJOR - changes in semantics that break backward compatibility.
    2. MINOR - refinements to existing semantics and additional elements (including things like additional scope notes).
    3. PATCH - typos, changes to existing elements that don't alter or refine existing semantics (e.g. minor rewording of an existing definition).
- Ongoing development work of published vocabularies *MUST* proceed on unpublished copies of the vocabulary. In the case of a git-based workflow this will occur in any branch not labeled "master".
- Whenever a change to an existing vocabulary is published, however minor, the version number *MUST* be incremented using the above rules. In the case of a git-based workflow "publish" or "release" will mean merging the changes in the development branch into the branch labeled "master".
- A published vocabulary *SHOULD* maintain a changelog for each version number, however minor, indicating what specifically was changed in that release.

Conclusion

A system for supporting the management of change of semantic data, based on successful techniques used for software packages, is an essential requirement for encouraging wider participation by the library and cultural heritage communities in the semantic Web. Many of the element sets and value vocabularies of use in bibliographic metadata are in (perpetual) developmental phases, at the same time as the pressure to publish datasets for linked data applications is increasing. The decentralized nature of the semantic Web, where incompleteness and contradiction are assumed, is unfamiliar to the traditional paradigm of "perfect" records created according to complex sets of rules. Semantic versioning offers a simple, low-cost method to meet the needs of linked data contributors and consumers, provided the community can agree on the meta-semantics of major and minor. Similar versioning techniques may also support the publication of datasets as continuing resources, with scope and context changing over time.

References

[1] Preston-Werner, Tom. “Semantic Versioning 2.0.0.” Available at: http://semver.org/
[2] Clark, Kendall. Clark & Parsia blog posting “Semantic Versioning and OWL Ontologies”. Available at: http://weblog.clarkparsia.com/2011/09/19/semantic-versioning-and-owl-ontologies/
[3] Phipps, Jon. “Timeslices and Versions. Available at: http://metadataregistry.org/blog/2008/03/26/timeslices-and-versions/
[4] Chacon, Scott. “Getting Started: Git Basics”. Available at: http://www.git-scm.com/book/en/Getting-Started-Git-Basics
[5] Hillmann, Diane. Blog post on Metadata Matters “Versions and Services, pt. 2”. Available at: http://managemetadata.com/blog/2013/07/23/versions-and-services-pt-2/
[6] Dublin Core Metadata Initiative, Vocabulary Management Community. “Vocabulary Special Session Meeting Report”. Available at: http://wiki.dublincore.org/index.php/DC-2011_Vocabulary_Special_Session/Meeting_Report
[7] DataSift: Open Source Projects. “Versioning”. Available at: http://datasift.github.io/gitflow/Versioning.html
[8] “Semantic Versioning starting with Ruby 2.1.0”. Available at: https://www.ruby-lang.org/en/news/2013/12/21/semantic-versioning-after-2-1-0/