Neologism: Easy Vocabulary Publishing

Cosmin Basca[1], Stéphane Corlosquet[1], Richard Cyganiak[1], Sergio Fernández[2] and Thomas Schandl[1]

[1] Digital Enterprise Research Institute
National Univerisity of Ireland, Galway
Galway, Ireland
{firstname.surname}@deri.org
[2] Fundación CTIC
Gijón, Asturias, Spain
sergio.fernandez@fundacionctic.org

Abstract. Creating, documenting, publishing and maintaining an RDF Schema vocabulary is a complex, time-consuming task. This makes vocabulary maintainers reluctant to evolve their creations quickly in response to user feedback; it prevents use of RDF for casual, ad-hoc data publication about niche topics; it leads to poorly documented vocabularies, and contributes to poor compliance of vocabularies with best-practice recommendations. Neologism is a web-based vocabulary editor and publishing system that dramatically reduces the time required to create, publish and modify vocabularies. By removing a lot of pain from this process, Neologism will contribute to a generally more interesting, relevant and standards-compliant Semantic Web.

1 Introduction

Anyone who wants to publish information as RDF on the Semantic Web first faces the choice which RDF Schema vocabulary or OWL ontology to use. Some areas, such as social networks (FOAF), online communities (SIOC) or general document metadata (DC) are covered by established vocabularies. Outside of these domains, registries like SchemaWeb[3] and search services like Falcons Concept Search[4] assist in the task of finding vocabularies for niche topics, but what they find might be of insufficient quality, or might not cover all required terms, and at present many areas of interest are not covered by any vocabulary at all.

In summary, most efforts to publish information on the Semantic Web first require an effort to create, extend or modify an RDF Schema vocabulary or OWL ontology. But this is a complex and time-consuming task in itself. It involves:

- Creating the formal specification of the vocabulary in RDFS or OWL,
- writing documentation that is clear and helpful for users of the ontology,
- keeping both documents in sync as the vocabulary evolves,

[3] `http://www.schemaweb.info/`
[4] `http://iws.seu.edu.cn/services/falcons/conceptsearch/`

– archiving older versions of the documents,
– defining and maintaining mappings to related vocabularies,
– configuring the web server in accordance with W3C best practices [3].

In this paper we present an online vocabulary editor and publishing system based on Drupal[5], implemented in PHP and ActionScript, which will support vocabulary authors in the tasks above and thereby dramatically reduce the time required to create, publish and modify vocabularies. The work presented in this paper is in progress.

2 The value of vocabularies

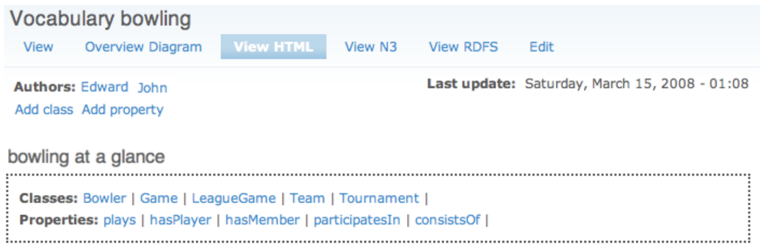
We define vocabularies as simple, "lightweight" ontologies, such as FOAF, DC, SIOC and SKOS. They usually comprise less than 50 terms. Expressivity is limited to RDF Schema plus selected OWL features, e.g. inverse functional properties and class disjointness. Their value is in providing common terminology for exchanging information between programs. The actual information is in the RDF instance data that is expressed with the vocabulary's terms, while in more complex ontologies, the actual information lies in the definitions of the classes and properties. A vocabulary is created by publishing a description of its terms in natural using HTML or formal using RDFS/OWL language. Since classes and properties are identified by URIs, it is considered a good practice to make these URIs resolvable [2, 3]. This enables clients to look up definitions of the vocabulary terms, with the following benefits:

– Information publishers can refer to a specification. This is important to create interoperability around a vocabulary. The top ten most popular vocabularies of 2006[6] all have a such a specification.
– RDF-aware tools such as data browsers (e.g. Tabulator [2]), SPARQL query builders and RDF instance editors can use the formal specification to improve the user experience, e.g. by showing friendlier labels and comments, listing available terms and providing widgets appropriate to a property's data type.
– Inference can be performed to increase recall when performing queries or lookups against RDF data, which is especially useful when terms are mapped to other vocabularies. Systems that use such techniques are the Tabulator data browser [2] and the Sindice semantic lookup index [6].

3 Current approaches to vocabulary publishing

Vocabulary maintenance with text editors and custom scripts. Many popular vocabularies such as FOAF and SIOC are maintained by a process involving hand-authoring of RDF and HTML files and custom scripts, e.g. SpecGen[7]. Often, complex custom Apache configurations are employed to follow best practices regarding content negotiation, MIME types and resolvable URIs [3].

[5] `http://drupal.org/`
[6] `http://ebiquity.umbc.edu/resource/html/id/196/`
[7] `http://sioc-project.org/specgen`



Authors: Edward John

Last update: Saturday, March 15, 2008 - 01:08

bowling at a glance

Classes: Bowler | Game | LeagueGame | Team | Tournament |
Properties: plays | hasPlayer | hasMember | participatesIn | consistsOf |

Fig. 1. A vocabulary page in Neologism, as it appears to an authenticated user.

Offline ontology editors. OWL ontology editors such as Protégé [5], TopBraid Composer[8] and SWOOP[9] can be used to create the formal specification of a vocabulary. While being great tools for knowledge engineering professionals, these applications have a steep learning curve and they intimidate casual users. They use a file-based, offline model, where ontology files are stored on the local user's computer. Remote publishing, if supported at all, is an after-thought.

Web-based systems. OntoWiki [1] provides basic ontology editing, but its main focus is the display and editing of RDF instance data. MyOntology [7] focuses on collaborative editing in a larger community, in the hope of creating rich knowledge bases, while creation of simple vocabularies typically does not involve many collaborating users. Knoodl[10] is a hosted service with strong community features and an easy-to-use vocabulary editor, but it does not publish created vocabularies with resolvable URIs or according to best-practice guidelines.

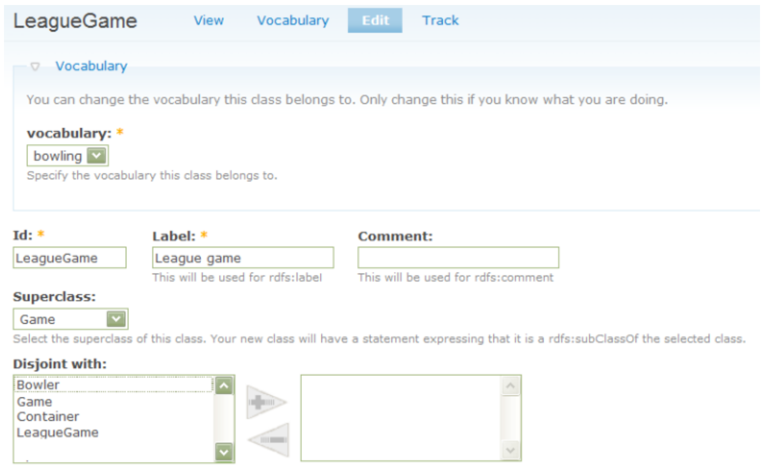
Areas for improvement. We identify four points where we can simplify the process: (i) Instant web-based publishing instead of file-based offline editing. (ii) Focus on a limited subset of RDFS and OWL. (iii) No instance editing or browsing. (iv) Handling of HTTP details like URI management, content negotiation and redirects within the web-based application.

4 Easier vocabulary publishing with Neologism

Neologism[11] is a web-based vocabulary editor and publishing platform designed to address these issues. It is currently being implemented and will soon be released as an open-source project. This section presents Neologism's current state.

Public interface. To non-authenticated users on the Web, Neologism presents a very simple interface: a homepage that lists one or more vocabularies, and for each of them a *vocabulary page* containing some general information about the vocabulary (Figure 1), followed by the descriptions of all its classes and properties.

[8] `http://www.topbraidcomposer.com/`
[9] `http://www.mindswap.org/2004/SWOOP/`
[10] `http://knoodl.com/`
[11] `http://neologism.deri.ie/`


LeagueGame View Vocabulary Edit Track

Vocabulary

You can change the vocabulary this class belongs to. Only change this if you know what you are doing.

vocabulary: *

bowling

Specify the vocabulary this class belongs to.

Id: *

LeagueGame

Label: *

League game

This will be used for rdfs:label

Comment:

This will be used for rdfs:comment

Superclass:

Game

Select the superclass of this class. Your new class will have a statement expressing that it is a rdfs:subClassOf the selected class.

Disjoint with:

Bowler
Game
Container
LeagueGame

Fig. 2. A form for editing a class.

Editor. After a vocabulary maintainer logs in, additional links become visible on the vocabulary page and allow adding new terms, as well as editing of existing terms. Terms are created and edited through a web form (Figure 2). The form allows entry of an ID (to become part of the term's URI), label, comment, subclasses, subproperties, domain, range, disjoint classes, inverse properties, and marking a property as inverse functional. Authenticated users can also create new vocabularies and modify the vocabulary metadata.

Overview diagram. The vocabulary page provides access to a diagram that shows the vocabulary's classes and their relationships (Figure 3). The vocabulary maintainer can arrange the diagram into a sensible layout and then save its current state which will henceforth be shown to all users.

RDFS output, URIs and content negotiation. The URIs identifying classes and properties are always generated by appending the hash character and the term's ID to the URI of the vocabulary page. This makes sure that the vocabulary page is returned when these URIs are resolved. HTTP requests to the vocabulary page are subject to content negotiation. Web browsers will see the HTML variant shown in Figure 1. RDF-aware clients will receive the RDFS/OWL specification, either in RDF/XML or N3 syntax. In a nutshell, Neologism publishes standards-compliant vocabularies on the Web without requiring any additional effort on the part of vocabulary maintainers.

Implementation. Neologism is implemented in PHP as a Drupal module. Drupal reduces development time by providing many features for free, such as account management. It also makes integration with a larger Drupal-based site very easy, for example to provide a news blog and discussion forum for each vocabulary. All

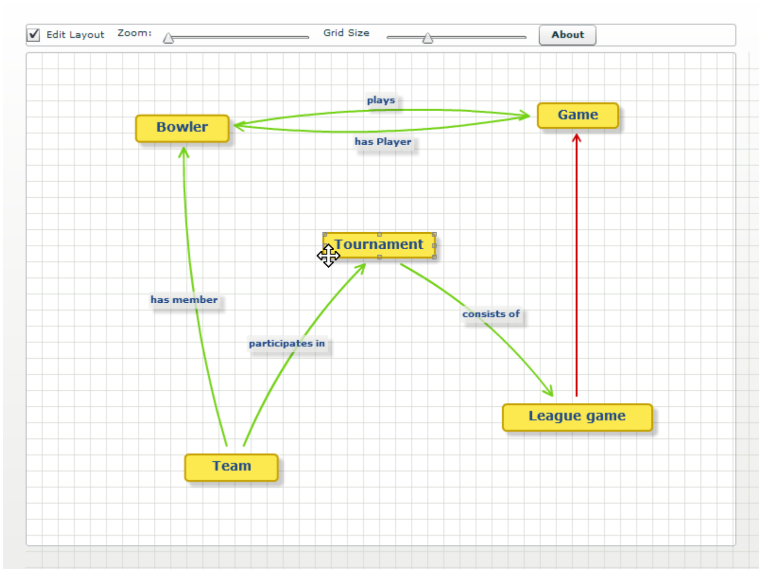


Fig. 3. The vocabulary overview diagram.

data is stored in a MySQL database. RAP[12] is used to serialize RDF/XML and N3. The PHP Content Negotiation library[13] is used instead of the usual Apache rules to implement content negotiation, and Vapour[14] was used to validate its correctness. The overview diagram is implemented using Adobe Flex and coded in ActionScript; the ObjectHandles and Tweener libraries are used for animation and object handling.

5 Future Work

Hosted Neologism service. Currently, vocabulary maintainers must install Neologism on their own webspace. A central hosted service, which could be easily built on the Drupal platform, would remove this barrier.

Branching and revision tracking. Neologism does not yet offer revision control. Some desirable features for vocabulary revision control are: archival of all prior versions; grouping of several small edits into a single version to avoid putting the vocabulary into an inconsistent intermediate state; publishing changes as a draft before accepting them as a new version.

[12] `http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/`
[13] `http://ptlis.net/source/php-content-negotiation/`
[14] `http://vapour.sourceforge.net/`

Plugin system. We intentionally kept the set of supported class and property annotations small to simplify the user experience, and don't support many possible further annotations, such as OWL cardinality constraints, plural and inverse labels[15], multilingual labels or associating Fresnel lenses [4] with classes and properties. Such additional annotations could be supported through plugins that are installed by vocabulary maintainers.

Consistency checking. Neologism doesn't check the created vocabulary for consistency. This can become an issue when a vocabulary is integrated with several external vocabularies. A solution could be the integration of an external reasoning service that performs consistency checks and is invoked through an API over the Web.

6 Conclusion

We have shown a web-based vocabulary publishing system that simplifies the process of creating, publishing and maintaining RDF vocabularies by (i) instant web-based publishing, (ii) focus on a limited subset of RDFS and OWL, (iii) avoiding instance editing or browsing, and (iv) handling URI management and HTTP content negotiation. We hope that the presented system will encourage the creation of new vocabularies and thereby contribute to a generally more interesting, relevant and standards-compliant Semantic Web.

References

1. S. Auer, S. Dietzold, and T. Riechert. OntoWiki, a tool for social, semantic collaboration. *The Semantic Web - ISWC 2006*, 4273/2006:736–749, 2006.
2. T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, , and D. Sheets. Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. In *The 3rd International Semantic Web User Interaction Workshop (SWUI06)*, 2006.
3. D. Berrueta and J. Phipps. Best Practice Recipes for Publishing RDF Vocabularies. Working Draft, W3C, 2008.
4. C. Bizer, R. Lee, and E. Pietriga. Fresnel, a Browser-Independent Presentation Vocabulary for RDF. In *International Semantic Web Conference 2006*, 2006.
5. H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The Protégé OWL Plugin: An open development environment for semantic web applications. *The Semantic Web ISWC 2004*, 3298/2004:229–243, 2004.
6. E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1), 2008.
7. K. Siorpaes and M. Hepp. myOntology: The marriage of ontology engineering and collective intelligence. In *ESWC 2007 Workshop Bridging the Gap between Semantic Web and Web 2.0*, 2007.

[15] `http://www.wasab.dk/morten/2004/03/label`