Ontology Development by Domain Experts (Without Using the "O" Word)

Andrea Westerinen* and Rebecca Tauber
Nine Points Solutions, LLC, Potomac, MD, 20854, USA

* Corresponding author: Andrea Westerinen, Nine Points Solutions, LLC, Potomac, MD, 20854, USA, Email: andreaw@ninepts.com

Abstract: Ontologies are created to describe and reason over the knowledge of a domain of interest. This requires deep understanding of the domain, and therefore, the input and collaboration of the domain's experts. But, the individuals with the domain knowledge are rarely versed in model or ontology development, and do not know the formal languages or logic that express ontological concepts. What is needed is to create renderings of the ontologies that fit how the experts work and make it easy for them to create, review and evolve the domain concepts. This paper presents thoughts on how to bridge the gap between ontology and domain experts, and how to create effective and usable ontologies without ever using the "O" ("ontology") word. In addition, the paper is intended to stimulate discussion on future directions for the techniques and technologies described here.

Keywords: Ontology Review, Ontology Visualization, Spreadsheet Data

1. Introduction

Ontologies are rapidly growing in popularity for use in big and linked data applications to capture the knowledge of a domain and integrate its data sets. The methods have already been adopted across many domains, including planetary science at NASA. NASA's ontology defines the semantics of planetary science data to aid in encoding the specific data of its programs and integrating it with a big data processing system (Earley, 2016). As in many domains, an interdisciplinary team developed NASA's ontology, first working with the scientists to understand what knowledge should be captured. After the knowledge engineering and domain modeling were complete, the ontology was developed through use of formal languages and tooling, such as Protégé (http://protégé.stanford.edu).

It is unlikely that the NASA scientists would have developed the planetary science ontology on their own (or would have had the time and desire to learn how to create it). For many non-ontologists, the utility and development of ontologies is not entirely clear. Even the use of the word, "ontology", conveys complexity and the need to learn new ways of expressing and representing concepts. Description or common logic languages, modeling methodologies, and ontology development tools, while highly useful, take time to comprehend and require experience to use effectively.

The NASA scientists were fortunate to have a supporting team that included ontology experts. But, even for ontology experts, an important step in ontology development is validation. Domain experts need to ensure that the entities, the relationships between them, and all the definitions and semantics are accurate. Asking a domain expert to use an ontology-authoring tool or to understand the complexities of a description logic language (such as OWL) may result in errors or omissions, or in the expert becoming frustrated and losing interest entirely.

The role and engineering of ontologies for the Big Data and Linked Data communities were two of the basic problems addressed in the Ontology Summit 2014 Communiqué (Gruninger, Obrst et al., 2014). However, in order to use ontologies, they must be understandable and accessible to the members of the communities, and correctly reflect the necessary domain concepts. This requires that the concepts and relationships in an ontology be presented in a way that is familiar to the users and the experts.

The following paper reviews development efforts in this space. Related work is reviewed in Section 2. We describe our work and experiences with a custom graphing tool (OntoGraph) and supporting textual documentation in Section 3. Then, in Section 4, we discuss how the OntoGraph and spreadsheet tooling has evolved, and areas of investigation for future development. One of the goals in presenting this work is to stimulate discussion on the requirements, technologies and techniques involved in working on ontologies with domain experts.

2. Related Work

Although there are many ontology engineering methodologies and tools on the market, using them typically requires a high degree of training and ontological expertise. This section provides an overview of efforts targeted at users who are not ontology experts.

2.1. Developing Ontologies

Insight into ontology development and curation by domain experts can be seen in the success of the Gene Ontology (GO, Bada et al., 2004, and "Gene Ontology", n.d.). Bada et al. attributed this to the following characteristics:

- Community involvement – GO "originated from within the biological community rather than being created and subsequently imposed by external knowledge engineers. Terms were created by those who had expertise in the domain."
- Clear goals and limited scope – GO's goal was specific: "to provide a common vocabulary for describing gene products, in terms of three … attributes [cellular components, molecular function and biological process], for the primary purpose of consistently annotating entries in biological databases".
- Simple structure – GO is defined using a directed acyclic graph where "each node in the graph is a natural-language term with a … natural-language definition, while each edge is either an is-a or part-of relationship". A separate graph is maintained for each of the three attributes in GO's scope.
- Continuous evolution and active curation – GO was not meant to be complete, but is evolving as the biomedical domain of knowledge is expanding and being revised.

The goals of community involvement by domain experts with simple, targeted concepts and structure are discussed again in Section 3.

Beyond concepts and structure, choice of an ontology development methodology is extremely important and will impact the entire scope of a project. Many methodologies have been proposed, catalogued and extended over the years (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003, and Sure, Tempich & Vrandecic, 2006). One such methodology, UPON Lite (De Nicola & Missikoff, 2016), extends the Unified Process for Ontology building (UPON) and directly relates to the problems described above. UPON is a cyclical ontology-building method that utilizes the Unified Process (UP) and the Unified Modeling Language ("UML", n.d.) for iterative, incremental, and use-case driven development (De Nicola, Missikoff, & Navigli, 2005). Even so, UPON is still dependent on ontology engineers and requires extensive knowledge of modeling techniques. UPON Lite was developed to address the "growing need for simpler, easy-to-use methods for ontology building and maintenance, conceived and designed for end users, … reducing the role of (and dependence on) ontology engineers".

UPON Lite defines a six-step process to capture domain concepts as an ontology. Each step of the process creates a "self-contained artifact readily available to end users", and able to be enriched/extended by the next step. The process begins with the creation of a domain lexicon, creating a list of the terms (concepts and properties) that are relevant in the domain. After the terms are collected, natural-language descriptions are associated with each one. This is an important step since it often occurs that different communities or business scenarios assign different meanings to the same term. Understanding when this occurs and the underlying semantic differences is very valuable.

The next steps in the UPON Lite process are to organize the nouns into a generalization/specialization (IS-A) hierarchy, and to define the scope of the properties and relationships. The latter step defines the domains and ranges of the properties, restricting them to the specific concepts or atomic datatypes to which they apply. Concepts are also analyzed regarding their structure and components to create a meronymic (whole-part) organization. Finally, an ontologist expresses all the information gathered from the domain experts using a formal language such as OWL.

The steps in the UPON Lite methodology are discussed in more detail in Section 4.

2.2. Visualizing Ontologies

There are various techniques to define ontologies in an unambiguous, computer-understandable way. Here, we focus on the use of the Resource Description Framework ("RDF", 2014) and Web Ontology Language ("OWL", 2012) due to the ubiquity of the OWL ecosystem for tooling and development. This means that RDF and OWL visualization tools are of the most immediate interest for collaborative work on ontologies.

There exist a wide variety of approaches for the visualization of OWL ontologies (Katifori, Halatsis, Lepouras, Vassilakis & Giannopoulou, 2003, and Lanzenberger, Sampson & Rester, 2010). In their survey, Katifori et al. described six categories of visualization distinguishing between whether the ontology is represented as an indented text list, a graph or a landscape with interacting nodes, whether it is shown in two or three dimensions, and the user-interaction options that are available (for example, having the ability to zoom or move focus to different nodes). For basic visualization, the simplest approaches are the indented list, such as the class browser in Protégé, and node-edge graphs. Katifori et al. also highlighted that all aspects of an ontology (i.e., its classes, inheritance hierarchies, instances, relationships and properties) should be visualized in order to have a complete understanding of it.

One scheme for a 2D node-edge visualization of OWL ontologies is the Visual Notation for OWL Ontologies ("VOWL", n.d.). VOWL was designed for "casual ontology users with only little training" (Lohmann, Negru, Haag & Ertl, 2014). This makes it an attractive option for ontology visualization for domain experts. VOWL and other graph representations are discussed further in Section 3.

Also relevant in this space is a study comparing indented tree visualizations and graphs (Fu, Noy & Storey, 2013). Fu et al. highlighted that multiple approaches present different viewpoints that can complement each other and better engage users. In the same study, the authors found that participants believed that graphs "held their attention better", and were better suited to obtain an overview of a domain or to show multiple inheritance. Important factors in the usability of a graph are its ability to be consumed in manageable fragments, and to be customized based on personal preference or style.

2.3. Ontologies and Spreadsheets

Spreadsheets are another popular approach to representing ontologies; they are "familiar" tools used in almost all domains. The author of one such tool, Populous (Stevens, 2012), stated that spreadsheet use in the sciences (especially the life sciences) is "almost ubiquitous". Additionally, the tooling that supports UPON Lite (discussed in Section 2.1) is based on a spreadsheet. De Nicola and Missikof (2016) "experimented with shared Google Sheets [for UPON Lite] for ontology engineering, plus Google Forms and Google+ for other functions (such as debating and voting on contentious issues)". Their spreadsheets had specific columns that captured a term and its synonyms, object or data type, description, etc.

There are many spreadsheet-based tools to aid in the development and use of ontologies. The work described below lays a foundation for collaboration through use of spreadsheets, but is not an exhaustive overview.

A simple approach to importing or exporting data to a spreadsheet is as a set of comma-separated values (CSVs). TARQL ("TARQL", n.d.) is an open-source tool that converts data in CSV files to a Resource Description File (RDF) format. Using TARQL, it is possible to create ontology class, property and instance data. However, this is not a tool for domain experts, since the conversion is defined using a SPARQL query.

A similar tool for comma- or tab-separated value conversion is ROBOT (Overton, Dietze, Essaid, Osumi-Sutherland & Mungall, 2015, and "ROBOT", n.d.). While ROBOT was developed for working with the Open Biomedical Ontologies (http://www.obofoundry.org), it can be used with any OWL ontology. More than just manipulating CSVs, ROBOT goes beyond TARQL to include development features such as merging, subsetting, reasoning with, and comparison of ontologies.

ROBOT uses a "template" string to define the meaning of each column in a spreadsheet. For example, a column may be defined as holding an "ID" or "LABEL", or identify that the values in the subsequent rows define specific property annotations (if the template string starts with an "A") or class expressions (if it starts with a "C"). Template strings are specified in the second row of a spreadsheet, while the first row defines a column name. Although ROBOT is quite versatile, its use mandates a basic understanding of OWL, ontologies and the template syntax.

Tooling such as ROBOT is not unique in the biomedical field. Necessitated by the huge growth of data, several tools have been created to aid in the curation and annotation of data by

biomedical domain experts. Structured vocabularies and ontologies are used to promote consistency in definition and categorization, and to enable reuse (Howe et al., 2008). Unfortunately, while there is some automation, much of the data curation and annotation has to be done manually.

RightField (Wolstencroft et al., 2011, and "RightField", n.d.) is an application targeted at improving data annotation by removing the need for experts to understand the necessary vocabularies, ontologies or mechanisms of metadata/annotation management. It provides a means to restrict the values of specific cells, columns or rows of a spreadsheet to the classes or instances of an existing ontology.

One problem, however, is that an ontology or vocabulary may not be complete and new terms may need to be added. For this reason, Populous (Stevens, 2012, and "Populous", n.d.) was created as an extension of RightField, allowing both the use and addition of terms to a vocabulary/ontology. Populous was used in the development of the Kidney and Urinary Pathway Ontology (KUPO) (Jupp et al., 2012).

Another interesting application of spreadsheets is Owlifer (Bowers, Madin, & Schildhauer, 2010). This tool was created to allow the import and definition of ontology concepts, subclasses, synonyms, properties and comments/descriptions using spreadsheet templates, and then to output the information in OWL.

The tools discussed above demonstrate the value of spreadsheets in ontology development and use, and there are many more (Kovalenko, Serral & Biffl, 2013). The key aspect in the use of spreadsheets is that they require information be provided using a specific format or template. We return to this discussion in Section 3, where we document one, specific spreadsheet format that was used with several of our customers.

3. Visualization and Spreadsheet Tooling for Domain Experts

With the goal of presenting an ontology to a community of experts and based on the work described in Section 2, we developed several tools to provide both visual and written information, using formats with which experts were comfortable (graphs and spreadsheets). Both visual and written outputs are generated, as studies have indicated the value in using multiple techniques (Katifori et al, 2003, and Fu et al., 2013).

3.1. OntoGraph

For visualization, we created the OntoGraph program ("OntoGraph", 2016). It was designed to provide documentation on existing OWL ontologies, developed for our consulting customers. OntoGraph is architected to create separate graphs for the classes, object and data properties, and individuals of an ontology. Separate graphs are generated to reduce the number of nodes and edges on any single graph, and thereby reduce crowding and help focus the semantics. It is also possible to generate a single, UML-style[1] class diagram.

The type of graph to be generated (class and inheritance information, a property diagram, instance/enumeration details, or a UML-style class diagram) is specified using a "Graph Type" selection (shown in the Fig. 1 below). It is also possible to output a graph with both class and property information.

[1] The text, "UML-style class diagram", is used to indicate that a single diagram is generated which uses the visual notations of UML.

OntoGraph visualizations can take many different formats. The program supports "standard" representations (such as Graffoo[2], VOWL, and UML-style class diagrams) or a custom format. By selecting a "custom" visualization, the graphical representation can be tailored. Node shape, node color, property line color, and source and target property line arrow shapes can all be defined. And, these can be varied for classes versus individuals, and object versus data properties. The flexibility is intended to align the graph with business or industry conventions, or to allow the output to be tailored to a user's specific needs or preferences. Fig. 1 shows the initial interface of OntoGraph, where the ontology file, visualization and other configuration options are identified. Fig. 2 shows the property-related, customizable features.

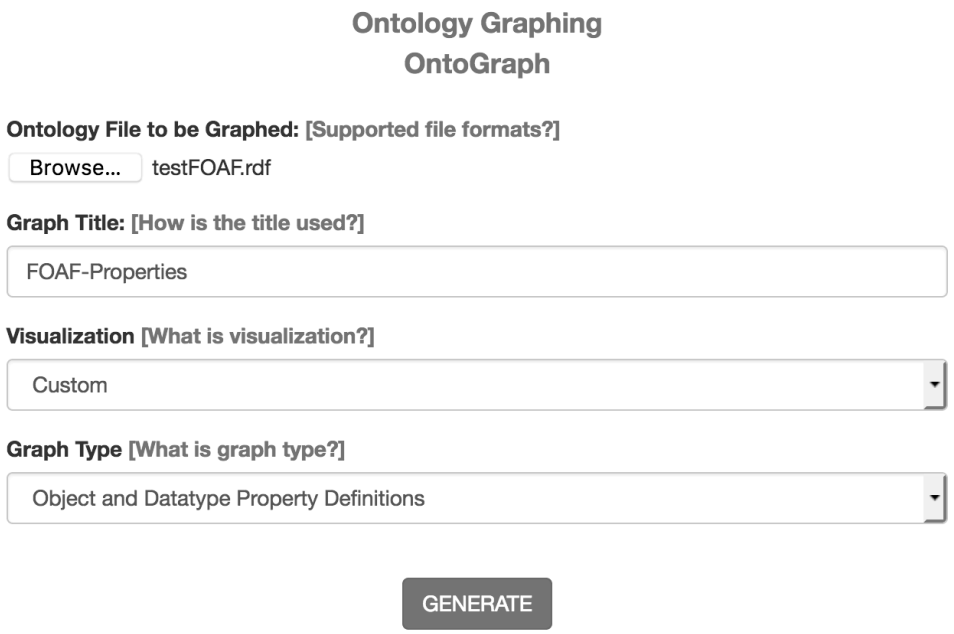

Fig. 1. OntoGraph Interface for a Custom Visualization

[2] http://www.essepuntato.it/graffoo/specification/current.html

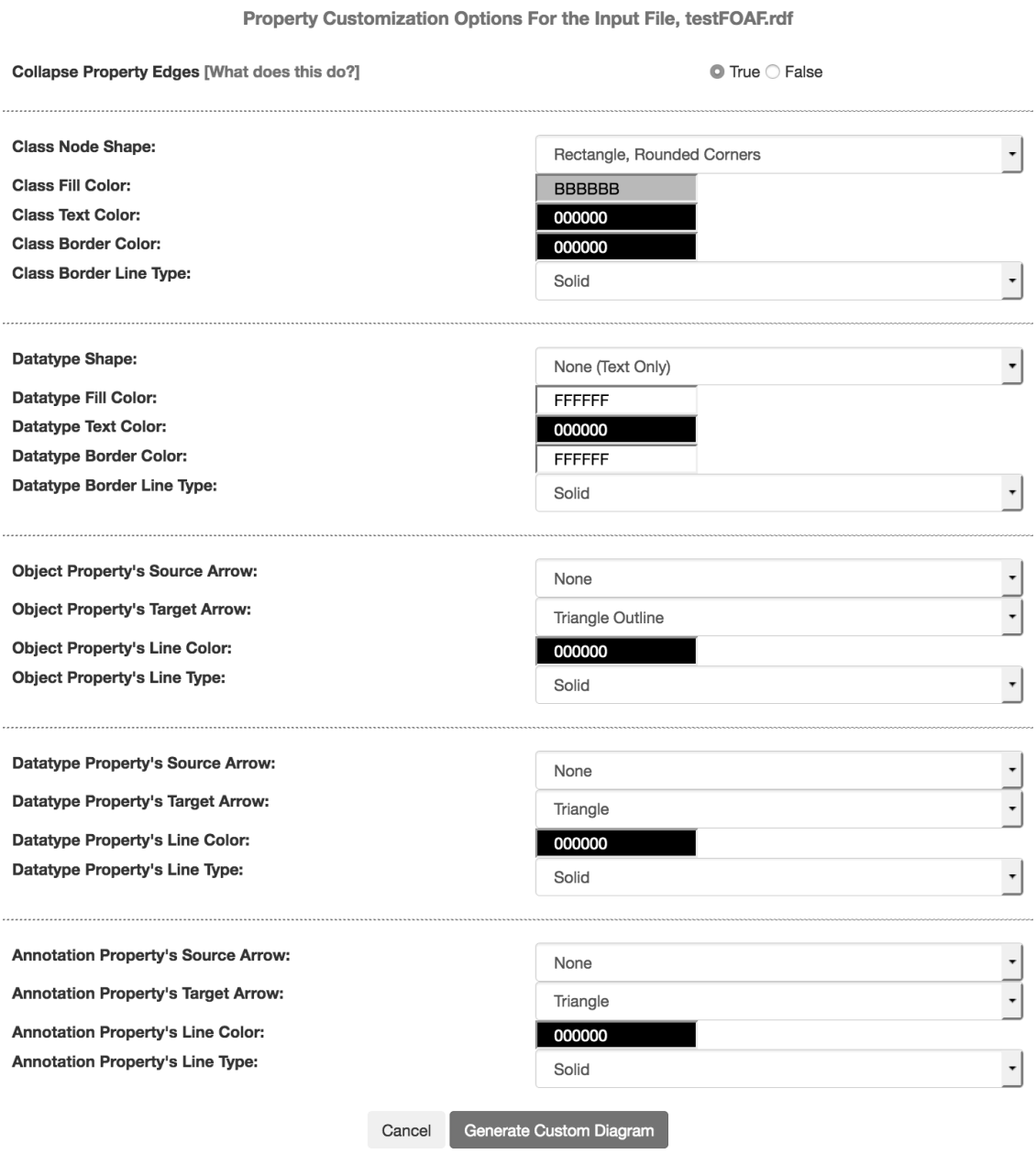

Fig. 2. Customization Options for Property Visualizations

Examples of two of the OntoGraph outputs are shown in the figures below. Fig. 3 shows most of the rendering of the Friend of a Friend ("FOAF", n.d.) class information (details were omitted to make the diagram more readable). This is a custom visualization of a "Class Definitions" graph type. The "Class Definitions" customization options were modified to select a "rectangle, rounded corners" class node shape, with no source arrow and a "triangle" target arrow. Fig. 4 shows the property graph for the same ontology. It was generated using the "Property" customization options shown in Fig. 2.

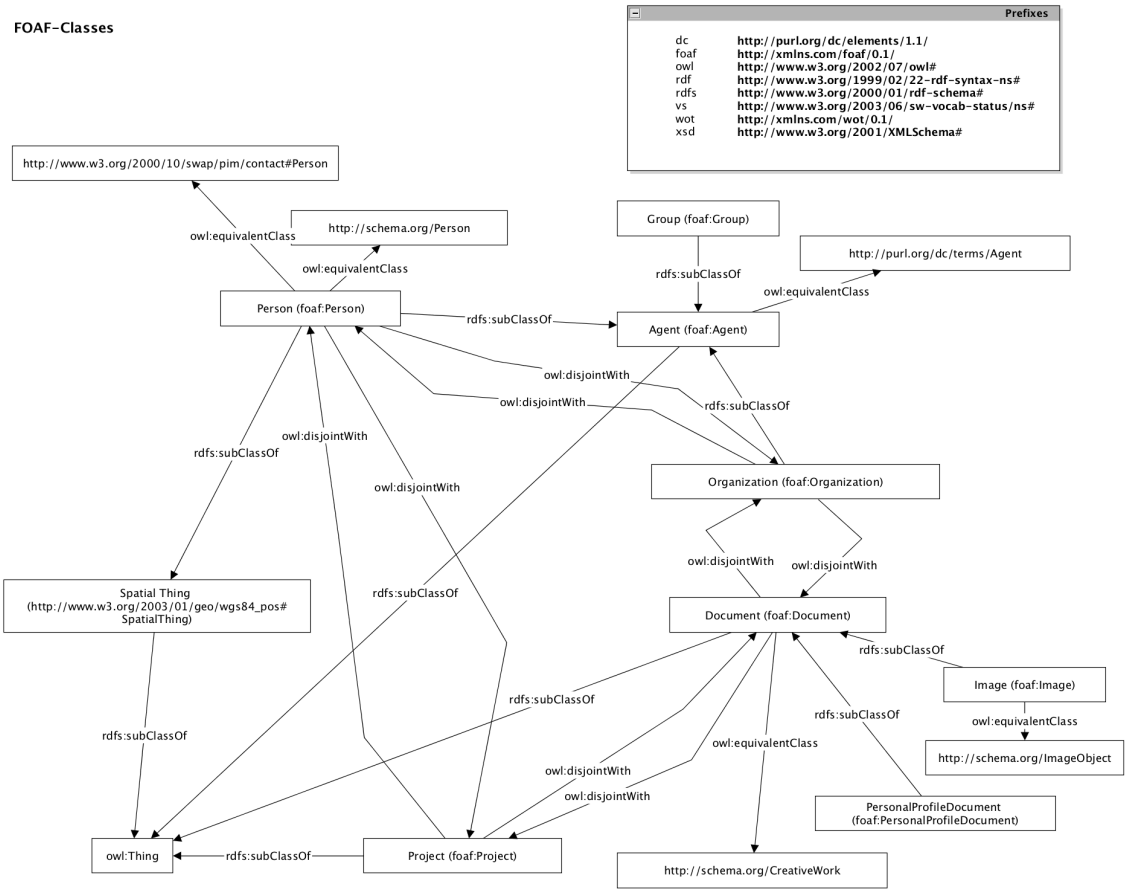

Fig. 3. Visualization of the FOAF Class Definitions

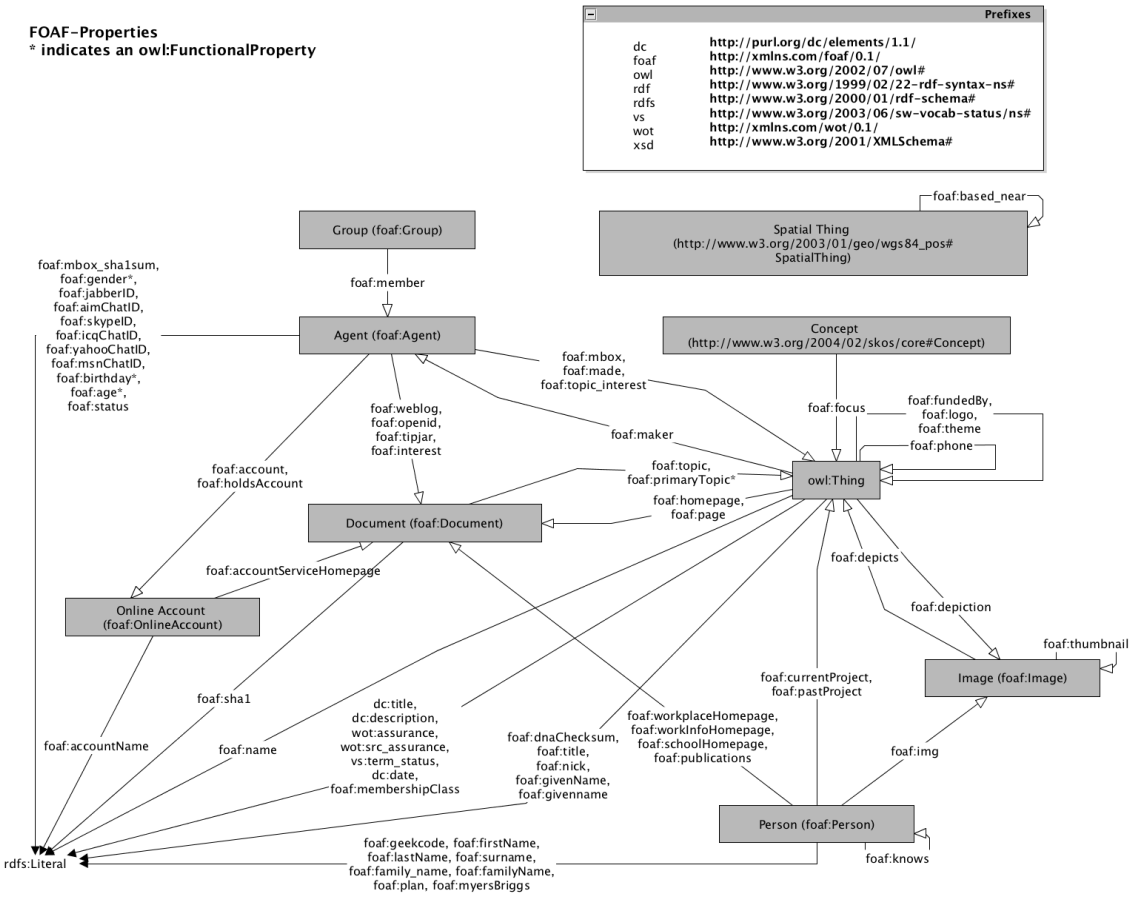

Fig. 4. Visualization of the FOAF Properties

It is valuable to consider how complex Fig. 4 would be, if the "Collapse property edges" option was not chosen. Most visualization tools draw individual lines for each edge. When this is done for an ontology such as Friend-Of-A-Friend, the result is as shown in Fig. 5 – where many of the property names are not readable. To address this without removing edges would require a very large graph. Instead, OntoGraph allows the option of collapsing all the edges connecting the same source and target to one line, where the text lists the combined labels of each of the individual lines.

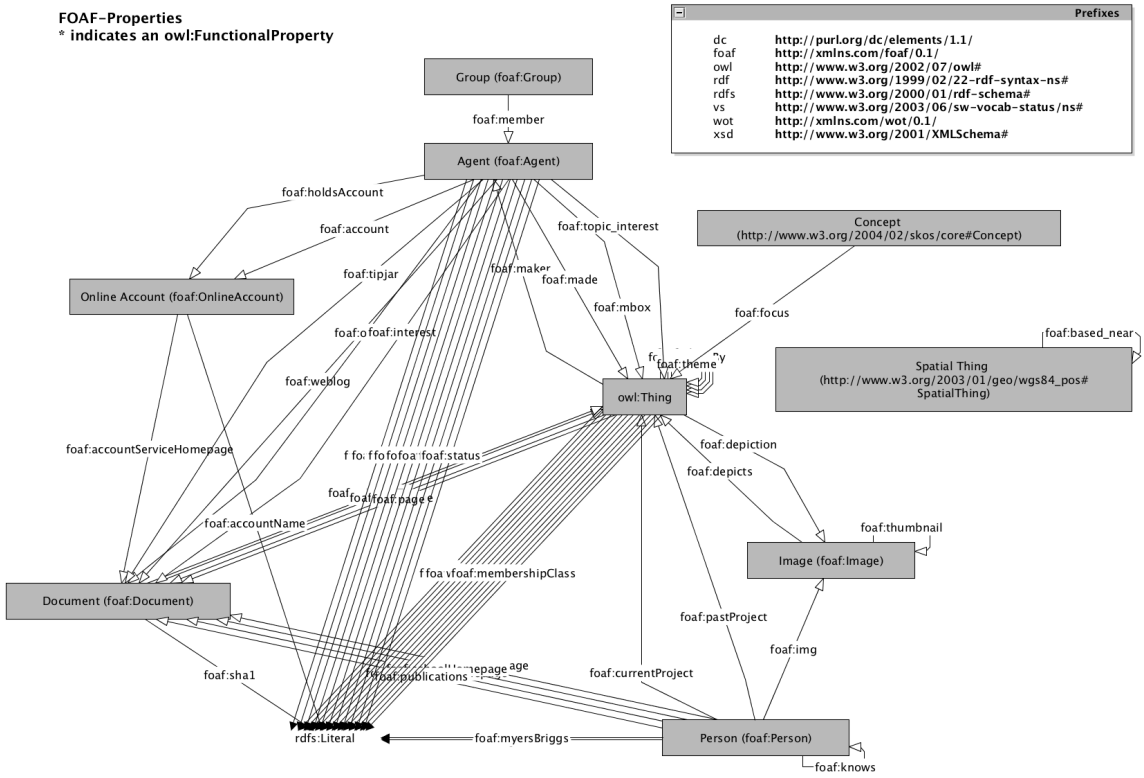

Fig. 5. Visualization of FOAF Properties without Reducing the Number of Edges

Figs. 3, 4 and 5 were generated taking the output of OntoGraph and importing it into the yEd graph editor ("yEd", n.d.). The class diagram, Fig. 3, was generated using a circular layout, with some minor manipulation of the layout result to move nodes closer together, to reduce edge overlap and to split long node labels to multiple lines. The property diagrams, Figs. 4 and 5, were also generated using a circular layout (again, with some manipulation of the layout result).

OntoGraph does not produce an automatic layout of the generated graph, since a user will typically want to tweak, reposition and/or annotate the elements. For this reason, OntoGraph simply creates a GraphML ("GraphML", n.d.) declaration of the rendering. This is then imported into a graphical layout tool. Using the layout tool, a user can select any appropriate layout. Separating functionality between OntoGraph and a layout/graph editing tool enables each application to focus on its specific user requirements.

3.2. Spreadsheet Tooling

Beyond graphs, we also generate domain documentation for existing OWL ontologies using a simple spreadsheet format. The spreadsheet consists of four pages (worksheets) describing the concepts (classes) of a domain, their relationships (object properties), data properties and instances. The layout of each of the worksheets is as follows:

- All Worksheet pages
  - Columns are defined for the concept's/relationship's/property's or instance's name and definition
  - Also, a column with a comma-separated list of synonym names is provided

- Concepts (Classes) Worksheet
    - A column is added for the definition of a comma-separated list of "more general" concepts (and allows for multiple inheritance)
- Relationships (Object Properties) Worksheet
    - Relationships are assumed to be directed and columns are provided to reference a comma-separated list of the source and target concepts (from the Concepts Worksheet)
    - Columns also indicate if the relationship is single-valued or transitive
- Properties (Data Properties) Worksheet
    - Columns are provided to reference a comma-separated list of the concepts (from the Concepts Worksheet) to which the property is applicable, and the type of data (boolean, integer, string, etc.)
    - Columns also indicate if the relationship is single-valued
- Instance Worksheet
    - A column is provided to reference a comma-separated list of concepts (from the Concepts Worksheet) which define the type of the individual
    - At this time, instances are treated as enumerated values – their relationships, properties and their values cannot currently be defined (although this functionality is intended for a future release)

The spreadsheet is generated by loading an OWL ontology to an RDF triple store and then querying the following fields[3]:

- rdf:type to distinguish between concepts, object and data properties and instances, to determine the type of instances, to determine if a relationship is single-valued (functional) or transitive, and to determine if a property is single-valued
- rdfs:subClassOf to define the inheritance hierarchy
- rdfs:label, SKOS ("SKOS", 2009) prefLabel or a custom property for the name
- SKOS definition, Dublin Core ("Dublin Core", n.d.) description, rdfs:comment or a custom property for the definition
- owl:equivalentClass for conceptual "synonyms", SKOS altLabel or a custom property for label synonyms
- owl:disjointWith for conceptual "antonyms"
- rdfs:domain to define the source concepts of a relationship or the concepts where a property is applicable
- rdfs:range to define the target concepts of a relationship or the data type of a property

Fig. 6 shows the query that generates the Concepts Worksheet CSV.

[3] The fields discussed in the bullets are the customary ones defined in OWL and several widely used ontologies, but queries can be customized to use any convention.

```
select distinct ?label ?description ?subClassOf ?conceptSynonym
                ?labelSynonym ?conceptAntonym
where { ?s ?p ?o .
        { { ?s rdf:type owl:Class } UNION { ?s rdf:type rdfs:Class } }
        { { ?s rdfs:label ?label } UNION { ?s skos:prefLabel ?label } }
        OPTIONAL {
          { { ?s rdfs:comment ?description } UNION { ?s dc:description ?description }
            UNION { ?s skos:definition ?description } } }
        OPTIONAL { ?s rdfs:subClassOf ?subClassOf }
        OPTIONAL { ?s owl:equivalentClass ?conceptSynonym }
        OPTIONAL { ?s skos:altLabel ?labelSynonym }
        OPTIONAL { ?s owl:disjointWith ?conceptAntonym }
}
```

Fig. 6. Query of FOAF Concepts

Fig. 7 shows the results from the query executed over the FOAF ontology (and then modified by our program), and imported into a spreadsheet program. The following manipulations were performed to generate the worksheet output shown in Fig. 7:

- Collapsed the information in all columns for any rows referencing the same subject, ?s (since the query returns each combination of variables as an individual result)
- Replaced the query variable names in the first row with "friendly" identifiers (for example, replacing ?subClassOf with "More General Concept")
- Simplified the URLs (using prefixes) for any references outside of the FOAF ontology

A	B	C	D	E	F
Label	**Description**	**More General Concept**	**Conceptual Synonym**	**Label Synonym**	**Conceptual Antonym**
Label Property	A foaf:LabelProperty is any RDF property with texual values that serve as labels.				
Person	A person.	Agent, W3C Basic Geo SpatialThing	schema.org Person, W3C Contact Person		Organization, Project
Agent	An agent (eg. person, group, software or physical artifact).		Dublin Core Agent		
Spatial Thing					
Organization	An organization.	Agent			Person, Document
Project	A project (a collective endeavour of some kind).				Person, Document
Document	A document.		schema.org CreativeWork		Organization, Project
Group	A class of Agents.	Agent			
Image	An image.	Document	schema.org ImageObject		
PersonalProfileDocument	A personal profile RDF document.	Document			
Online Account	An online account.	owl Thing			
Online Gaming Account	An online gaming account.	OnlineAccount			
Online E-commerce Account	An online e-commerce account.	OnlineAccount			
Online Chat Account	An online chat account.	OnlineAccount			

Fig. 7. Worksheet of FOAF Classes

4. Conclusions and Future Work

Section 3 focused on tooling to document existing OWL ontologies and share their content beyond ontology experts. The tooling is unique from that discussed in Section 2 in that it:

- Combines both visual and spreadsheet (textual) output, versus being focused only on visualization or only using spreadsheet data
- Provides visual output that can be customized to business or industry conventions, versus mandating a specific visualization format
- Simplifies visualizations by collapsing multiple edges between two nodes to a single edge
- Is general purpose, versus targeted at a particular industry or domain

- Creates flexible spreadsheet outputs, versus restricting spreadsheet cells, columns or rows using templates and hard-coded conventions
- Supports the current version of OWL (OWL 2)
- Is available and maintained as open-source (https://github.com/NinePts)

The tooling is released as open-source to encourage broader development and usage. Feedback, bug reports, and information on improvements and new requirements are very important. The remainder of the paper discusses current areas of work.

The OntoGraph and worksheet tooling assume that an appropriate, domain ontology exists and is defined using an OWL syntax. We are working on extending the tooling to capture new concepts, properties and other details (such as synonyms and functional properties) which are added to the diagrams and worksheets, as well as any review comments defined for existing concepts and properties. (Also important in this approach is to capture the provenance of the additions and changes.) The intent is to follow the workflow defined by UPON Lite, with the goals of:

- Defining and curating a set of nouns, relationships (verbs and objects), properties (adjectives), and instances/enumerated values (proper nouns and restricted values) that address a specific aspect or area of a domain
- Understanding how the nouns are connected in a taxonomy (generalization/specialization hierarchy)
- Restricting the concepts/nouns to which the relationships and properties apply

As part of capturing changes to the diagrams, we are working to maintain the layout of nodes, notes and edges. This will then reduce the overhead when updating and re-generating a graph.

The problem of crowded images is also being examined. Simply separating class, instance and property diagrams and reducing the number of edges cannot address the full issue. If an ontology is large, its visualization can be confusing and unreadable, too large to be viewed without scrolling. We are working to define a straightforward mechanism to select (limit) specific concepts to display on a single graph. Until this work is complete, OntoGraph is best suited for small to mid-sized, modular ontologies and ontology design patterns.

Another area of investigation is to transform the graphical and spreadsheet output based on the community of domain experts (their context and preferred vocabulary). Many ontologies utilize unique IDs to identify entities, and displaying these on a graph is meaningless to domain experts unless the text label is present. Currently, OntoGraph displays the rdfs:label defined for a class, along with the class URI in parentheses. The goal is to customize this further, translating a class name or label to the terms appropriate for a particular community. In this way, information is available to the experts using the terminology with which they are most familiar, instead of asking them to think in terms of a "standard" vocabulary. This is particularly important when there are overlapping or widely divergent semantics for a term. In these cases, it may not be possible to get the experts to agree on a single phrase or term. If this obstacle is removed, and the experts can agree on the semantics of a concept, the path to broader acceptance of ontology use will be cleared.

References

1. Bada, M., Stevens, R., Goble, C., Gil, Y., Ashburner, M., Blake, J., Cherry, M., Harris, M., & Lewis, S. (2004). A Short Study on the Success of the Gene Ontology. Web Semantics: Science, Services and Agents on the World Wide Web, Volume 1, Issue 2, pp. 235-240. doi: 10.1016/j.websem.2003.12.003.
2. Bowers, S., Madin, J. , & Schildhauer, M. (2010). Owlifier: Creating OWL-DL Ontologies from Simple Spreadsheet-Based Knowledge Descriptions. Ecological Informatics 5(1), pp. 19-25. doi: 10.1016/j.ecoinf.2009.08.010. Retrieved from http://www.cs.gonzaga.edu/~bowers/papers/ecoinf-2010.pdf
3. Corcho, O., Fernandez-Lopez, M., & Gomez-Perez, A. (2003). Methodologies, Tools and Languages for Building Ontologies. Where is Their Meeting Point? Data & Knowledge Engineering (46), pp. 41-64. doi: 10.1016/S0169-023X(02)00195-7. Retrieved from http://oa.upm.es/2637/1/JCR02.pdf
4. De Nicola, A., & Missikoff, M. (2016). A Lightweight Methodology for Rapid Ontology Engineering. Communications of the ACM, Volume 59, Issue 3, pp. 79-86. doi: 10.1145/2818359.
5. De Nicola, A., Missikoff, M., & Navigli, R. (2005). A Proposal for a Unified Process for Ontology Building: UPON. DEXA 2005, Lecture Notes in Computer Science, Volume 3588, pp. 655-644. doi: 10.1007/11546924_64.
6. Dublin Core (n.d.). DCMI Specifications. Retrieved from http://www.dublincore.org/specifications/
7. Earley, S. (2016). Really, Really Big Data: NASA at the Forefront of Analytics. IT Professional, Volume 18, Issue 1, pp 58-61. doi: 10.1109/MITP.2016.10.
8. FOAF (2000). The FOAF Project. Retrieved from http://www.foaf-project.org/
9. Fu, B., Noy, N., & Storey, M. (2013). Indented Tree or Graph? A Usability Study of Ontology Visualization Techniques in the Context of Class Mapping Evaluation. International Semantic Web Conference 2013, Lecture Notes in Computer Science, Volume 8218, pp 117-134. doi: 10.1007/978-3-642-41335-3_8.
10. Gene Ontology (n.d.). Gene Ontology Consortium. Retrieved from http://www.geneontology.org/
11. GraphML (n.d.). The GraphML File Format. Retrieved from http://graphml.graphdrawing.org/
12. Obrst, L., Gruninger, M., Baclawski, K., Bennett, M., Brickley, D., Berg-Cross, G., Hitzler, P., Janowicz, K., Kapp, C., Kutz, O., et al. (2014). Semantic Web and Big Data Meets Applied Ontology: The OntologySummit 2014. Applied Ontology, 9(2), pp 155-170. Retrieved from http://ontolog.cim3.net/file/work/OntologySummit2014/OntologySummit2014_Communique/OntologySummit2014_Communique_v1-0-0_20140429-1045.pdf
13. Howe, D., Costanzo, M., Fey, P., Gojobori, T., Hannick, L., Hide, W., Hill, D., Kania, R., Schaeffer, M., St Pierre, S., Twigger, S., White, O., & Rhee, S. Y. (2008). Big data: The future of biocuration. Nature 455(7209), pp. 47-50. doi: 10.1038/455047a.
14. Jupp, S., Horridge, M., Iannone, L., Klein, J., Owen, S., Schanstra, J., Stevens, R., & Wolstencroft, K. (2012). Populous: a tool for building OWL ontologies from templates. BMC Bioinformatics, Volume 13, Supplement 1. doi: 10.1186/1471-2105-13-S1-S5. Retrieved from http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-S1-S5
15. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2003). Ontology Visualization Methods – A Survey. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.7581&rep=rep1&type=pdf
16. Kovalenko, o., Serral, E., & Biffl, S. (2013). Towards Evaluation and Comparison of

Tools for Ontology Population from Spreadsheet Data. Proceedings of the 9th International Conference on Semantic Systems, pp. 57-64. doi: 10.1145/2506182.2506190.

17. Lanzenberger, M., Sampson, J., & Rester, M. (2010). Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data. Journal of Universal Computer Science, Volume 16, Number 7, pp. 1036-1054. Retrieved from http://pdfs.semanticscholar.org/dc94/73af090e61dd758b7728aa8cccc31874962e.pdf
18. Lohmann, S., Negru, S., Haag, F., & Ertl, T. (2014). VOWL 2: User-oriented visualization of ontologies. Retrieved from http://www.semantic-web-journal.net/system/files/swj750.pdf
19. OntoGraph (2016). Available from https://github.com/NinePts/ontograph
20. OWL 2 (2012). OWL – Semantic Web Standards. Retrieved from http://www.w3.org/OWL/
21. Overton, J., Dietze, H., Essaid, S., Osumi-Sutherland, D., & Mungall, C. (2015). ROBOT: A command-line tool for ontology development. Retrieved from http://ceur-ws.org/Vol-1515/demo6.pdf
22. Populous (n.d.). Google Code Archive. Retrieved from http://code.google.com/archive/p/owlpopulous/
23. RDF 1.1 (2014). RDF – Resource Description Framework. Retrieved from http://www.w3.org/RDF/
24. RightField (n.d.). RightField, Semantic Annotation by Stealth. Retrieved from http://www.rightfield.org.uk/
25. ROBOT (n.d.). GitHub – ontodev/robot. Retrieved from https://github.com/ontodev/robot
26. SKOS (2009). SKOS Simple Knowledge Organization System Namespace Document, W3C Recommendation. Retrieved from https://www.w3.org/2009/08/skos-reference/skos.html
27. Stevens, R. (2012). Easing the pain of ontology building with Populous. Retrieved from http://robertdavidstevens.wordpress.com/2012/11/13/easing-the-pain-of-ontology-building-with-populous/
28. Sure, Y., Tempich, C., & Vrandecic, D. (2006). Ontology Engineering Methodologies. In J. Davies, R. Studer, R. & P. Warren (Eds.), Semantic Web Technologies: Trends and Research in Ontology-Based Systems (pp. 171-190). West Sussex, England: John Wiley & Sons Ltd.
29. UML (n.d.). Welcome to UML Web Site! Retrieved from http://www.uml.org/index.htm
30. TARQL (n.d.). TARQL: SPARQL for Tables. Retrieved from http://tarql.github.io/docs/
31. VOWL (n.d.). VOWL: Visual Notation for OWL Ontologies. Retrieved from http://vowl.visualdataweb.org/
32. Wolstencroft, K., Owen, S., Horridge, M., Krebs, O., Mueller, W., Snoep, J., du Preez, F., & Goble, C. (2011). RightField: Embedding Ontology Annotation in Spreadsheets. Bioinformatics, 27(14), pp. 2021-2022. doi: 10.1093/bioinformatics/btr312. Retrieved from http://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr312
33. yEd (n.d.). yEd – Graph Editor. Retrieved from http://www.yworks.com/products/yed