A document-centric approach for developing the tolAPC Ontology

Aisha Blfgeh [1,2*], Jennifer D. Warrender [1], Catharien M. U. Hilkens [3] and Phillip Lord [1]

[1]School of Computing Science, Newcastle University.
[2]School of Computer Science and Information Technology, King Abdulaziz University, Jeddah, Saudia Arabia.
[3]Institute of Cellular Medicine, Newcastle University.

ABSTRACT

Motivation: There are many challenges associated with ontology building, as the process often touches on many different subject areas; it needs knowledge of the problem domain, an understanding of the ontology formalism, software in use and, sometimes, an understanding of the philosophical background. In practice, it is very rare that an ontology can be completed by a single person, as they are unlikely to combine all of these skills.

So people with these skills must collaborate. One solution to this is to use face-to-face meetings, but these can be expensive and time-consuming for teams that are not co-located. Remote collaboration is possible, of course, but one difficulty here is that domain specialists use a wide-variety of different "formalisms" to represent and share their data – by the far most common, however, is the "office file" either in the form of a word-processor document or a spreadsheet.

We consider here, how to use a highly-programmatic and pattern driven development to enable direct incorporation of office technology into the ontology pipeline. Specifically, we are developing an ontology of immunological cell types using spreadsheets. We also consider how this could be extended to other tools and environments.

Results: We have developed a new ontology, built by instantiating ontology design patterns written programmatically, using values from a spreadsheet catalogue of cell types. The spreadsheet was developed by domain experts, is unconstrained in its usage and can be freely updated, resulting in a new ontology. This provides a general methodology for working with data generated by domain specialists.

Availability:

The software tool, Tawny-OWL, is available from `http://github.com/phillord/tawny-owl`. The tolAPC ontology is currently not available.

*** Contact:** a.blfgeh1@newcastle.ac.uk, abelfaqeeh@kau.edu.sa

1 INTRODUCTION

Ontologies have been used extensively to describe many parts of biology. They have two key features which make their usage attractive: first, they can provide a mechanism for standardizing and sharing the terms used in descriptions, and secondly, they provide a computationally amenable semantics to these descriptions, making it possible to draw conclusions which are not explicitly stated.

Ontologies are increasingly used to facilitate the management of knowledge and the integration of information as in the *Semantic* Web [4]. Biological data is not only heterogeneous but requires special knowledge to deal with and can be large [22]. Ontologies are good for representing complex and potentially changeable knowledge. Therefore, they are widely used in biomedicine with examples such as the Gene Ontology [3], ICD-10 [1] or SNOMED [10] being the best known.

However, building an ontology is a challenging task [14]. Ontologies use a complex formalism (such as OWL for instance) especially when modelling complex domain area such as biology or medicine. Moreover, normally ontology building is a collaboration between domain specialists and ontology developers. However, any form of multi-disciplinary collaboration is difficult. In the case, for example, of the Gene Ontology, these challenges were addressed through explicit community involvement using meetings, focus groups and the like [2]. Other methodologies have adopted a more distributed approach [6].

It is, perhaps, because of these challenges that, despite the computational advantages of ontologies, the oldest and most common form of description in biology is free text, or a semi-structured representation through the use of a standardised fill-in form. These representations have numerous advantages compared to ontologies: they are richly expressive, widely supported by tooling and while the form of language used in science ("Bad English" [31]) may not be easy to use, understand or learn, it is widely taught and most scientists are familiar with it. Similarly, most biologists are familiar with the tools used for producing free-text and forms, either a word-processor document or a spreadsheet. Tools for producing this form of knowledge are wide-spread, richly functional both in application and cloud-delivered form, and support highly collaborative development.

The ontology community, conversely, has largely built its own tool-chain for development. Tools such as Protégé are highly functional in their own right, but have a user interface which is far removed from those that biologists are used to. There have been several responses to this problem. First, it is possible to take existing ontology tools and customize them for use within a specific community, so that they have a familiar look and feel; this is the approach taken by iCAT – a version of WebProtégé built explicitly for the ICD-11 community [25]. A second approach is to enable existing ontology tools to ingest office documents; for example, Cellfie is a Protégé plugin which can transform a spreadsheet into an OWL ontology, which can then be developed further; however this is a one-off process – once ingested, the data in the spreadsheet is converted into OWL; further updates cannot be made using the original spreadsheet formalism. Finally, tools such as RightField [30] and Populous [12] add ontological features to office documents, by allowing selection of spreadsheet cells from a

controlled vocabulary, followed by export to OWL using OPPL [5] to express the patterns used in the transformation [13].

These tools, however much they support the use of office software, at some point require leaving this software and moving into an ontology specific environment. We have developed a new, highly-programmatic environment for ontology development called Tawny-OWL [14]. With this approach the ontology is developed as programmatic *source code*, which is then evaluated to generate the final ontology, either in memory or as an OWL file. This offers a new methodology. In this research, we developed a document-centric workflow centred on the use of office tooling to construct the ontology; biologists generate and maintain their dataset in an unconstrained Excel spreadsheet; we then use this spreadsheet directly as part of our source code, driven by Tawny-OWL. In this model, we can apply arbitrary validation and transformation of the data held in the spreadsheet, into an ontological form. As the spreadsheet is now part of the source code, rather than being used as knowledge capture interface, it can be freely updated and the final ontology regenerated.

In this paper, we describe the application of this methodology to the generation of a catalogue of immunological cell types, called the tolAPC catalogue. We discuss the background technology, the design decisions that we have faced and the general implications that this approach has for ontology development.

2 BACKGROUND

The tolAPC catalogue is a list of immunological cell types. It has been captured as part of the EU Cost Action BM1305 (A-FACTT) which is aimed at increasing data sharing and collaborative working across the community [24]. These cell types have been "tolerized" – that is treated so that they suppress the immune response – and have been created with the intention that they will be used therapeutically in a variety of situations including: the treatment of auto-immune disease such as rheumatoid arthritis; or to reduce rejection following transplantation [7]. Information about these cells is, therefore, high value. The tolAPC catalogue contains extensive details about these cell lines, including 9 "sheets" of data. The catalogue has been created as an Excel spreadsheet, although it uses the spreadsheet only to represent tabular information (i.e. there is no use of equations or calculation in the spreadsheet). The spreadsheet has been created by individual scientists freely; that is, there is no formal constraint on the legal set of values in each cell, just the social convention of copying previous cells.

Tawny-OWL is a fully programmatic development environment for OWL. It has been implemented in Clojure, which is a dialect of lisp, running on the Java Virtual Machine. It wraps the OWL-API [8] which performs much of the actual work, including interaction with reasoners, serialisation and so forth. Tawny-OWL has a simple syntax which was originally modelled on the Manchester OWL notation [9], modified to conform to standard Clojure syntax and to increase regularity [15]. For example, we can create a new class with an existential restriction as follows:

```
(defclass A :super (some r B))
```

Or, we can define a new individual with a property assertion:

```
(defindividual i :fact (is r j))
```

As a domain specific language embedded in a full programming language, we also gain all the features of that environment; for instance, we can create arbitrary patterns simply by using a Clojure function. Consider for example:

```
(defn some-only [property & classes]
 (list (some property classes)
       (only property
             (or classes))))
```

Here `defn` introduces a new function, `property & classes` are the arguments, and `list` packages the return values as a list. `some`, `only` and `or`[1] are defined by Tawny-OWL as the appropriate OWL class constructors. This allows a definition specifying an existential relationship with a closure axiom as follows:

```
(defclass D :super (some-only r A B))
```

We also gain access to the full Clojure infrastructure: we can edit and evaluate terms in a power editor or IDE [2]; write unit tests and run them through a build tool, publish and version using git, and continuously integrate our work with other ontologies.

We have previously used this functionality to create the karyotype ontology which is generated from a series of interlocking patterns [28], parameterised using literal data structures in the source code. The karyotype ontology is highly patternised, with almost all of the classes coming from a single large pattern.

As a full programming environment, Clojure can also read and parse arbitrary data formats, which can operate as additional source during the generation of the ontology. We have previously used this to *scaffold* a mitochondrial ontology from a varied set of input files [27], or to add multi-lingual annotation using `key=value` properties files to the pizza ontology. We have also used this technology with a spreadsheet to specify a set of ontological unit tests for the karyotype ontology [29]. In this case, the values in the spreadsheet are used to generate a set of OWL classes which are then checked for correct subsumption using a reasoner. In this case, however, these ontological statements are used only as part of a test suite, rather than intended for downstream usage, and the spreadsheet was created specifically for this purpose.

3 BUILDING THE TOLAPC ONTOLOGY

The data for the tolAPC catalogue was captured directly in a spreadsheet largely co-ordinately through email. As a pre-existing resource, it made little sense to rewrite directly in OWL either using Protégé or Tawny-OWL– to do so would have resulted in transcription errors, and made updates more complex. However, as described in Section 2, we have all the components that we need to build an ontology directly from a spreadsheet.

In this section, we describe the issues that have arisen during the process which can conceptually be split into three phases [3]:

[1] We have elided namespaces: `or` and `some` are also core Clojure functions.

[2] We use Emacs but there is rich support in Vim, Eclipse, IntelliJ, or LightTable

[3] In practice, the tolAPC ontology is developed iteratively.

1. Extraction
2. Validation
3. Ontologisation

The extraction phase is straight-forward. Clojure offers a number of libraries capable of reading a spreadsheet. In the case of the tolAPC catalogue, we read the spreadsheet using the Docjure library [4], accessed directly from the file system. It would be simple and straight-forward to read from a network which would support building ontologies from cloud hosted spreadsheets. Previously, for performance reasons, we have read and then cached the results of tests generated from a spreadsheet [29]; however, for the tolAPC catalogue performance is such that the spreadsheet can be read in full every time the environment is initialised, significantly simplifying the development.

In the second phase, values extracted are validated against a set of constraints specifying those which are legal. For many of the fields, values are highly stereotyped having only a few different options; for example, cells can either be `Autologous` or `Allogeneic`, while expression levels can either be + or −. Currently, validation is performed through the use of *ad hoc* testing – we expect to move to a more formal data constraint language in future. The choice of validation depends on the requirements and modelling choices made, which will be discussed later.

In the third phase, values are "ontologised". The top level of the ontology which provides what we describe as *schema terms* is written by hand using Tawny-OWL. In the case of the tolAPC catalogue, this includes terms such as `CellType`, `Species` and `AntigenLoad`. Next, a set of patterns is defined using these schema terms. Finally, these patterns are instantiated using the values from the second phase, generating entities that we call *patternised terms*.

During the development process, both reasoning and manual inspection of the created ontology is used to ensure that the process is happening as expected; for the latter process, the ontology is saved to file and examined, either in the Clojure development environment or within Protégé, as shown in Figure 1.

We next discuss the modelling issues that have arisen.

4 MODELLING IN THE TOLAPC ONTOLOGY

All entities in the ontology need to be represented by an IRI. Two broad schemes are used to generate IRIs: semantics free identifiers which are generally numeric; and semantically meaningful identifiers which are normally derived from the common name for the entities. Generally, the latter are easier to work with, while the former are easier to keep stable over releases.

Currently, for the tolAPC ontology, schema terms have IRIs which reflect their names (`CellType` uses an IRI with a fragment of "CellType"), while patternised terms use an *ad hoc* schema based on several of their properties (a single property is not enough to ensure uniqueness). If we wish to re-evaluate this situation at a later date, however, Tawny-OWL simplifies the situation; we can easily allocate IRIs to entities according to any scheme that we choose, by changing a single function.

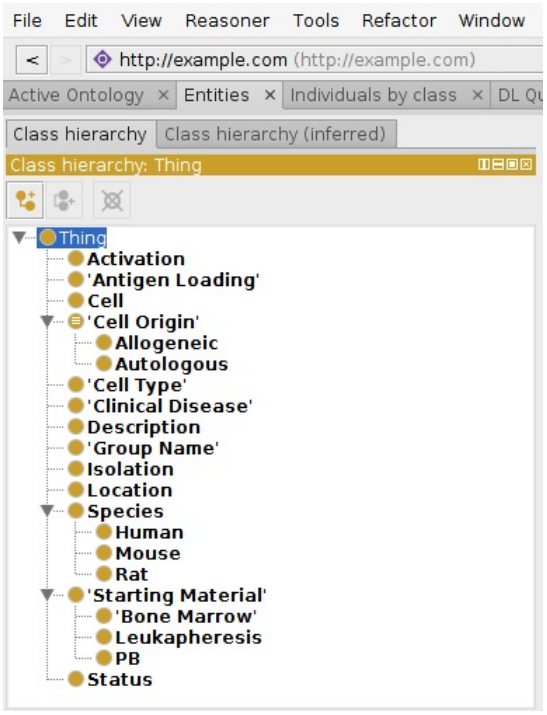

Fig. 1. tolAPC ontolgy displayed from Protégé screen.

A recurrent issue in ontology modelling is whether to use classes or individuals; within the tolAPC ontology, we faced this question for cell types. There are a number of different criteria for making this decision [23]. We considered briefly a "realist" perspective: modelled as a single entity, cell types are probably best represented as a metaclass, akin to a taxonomic species [21]; modelling as multiple entities (differentiating between the protocol and the cell type produced) would also be possible. However, there appears to be no clear principle to distinguish between these options. Similar problems also arise for proteins/cell-surface markers which are described in the ontology. As an additional problem, these representations introduce considerable unnecessary complexity [17].

We considered therefore the needs of our application: it seems unlikely that we will ever need subclasses of a cell type, but might reasonably wish for cell types to be unique – to state that two cell types are necessarily the same individual. For these reasons, we model cell types as individuals.

The tolAPC ontology largely models a set of cell types, with the rest of the ontology designed to support these descriptions. The ontology, as a result, contains very little hierarchy, and is at the extreme end of a normalised ontology [19]. Cell types are defined as individuals with a large set of different property assertions, as can be seen from the following definition:

[4] https://github.com/mjul/docjure

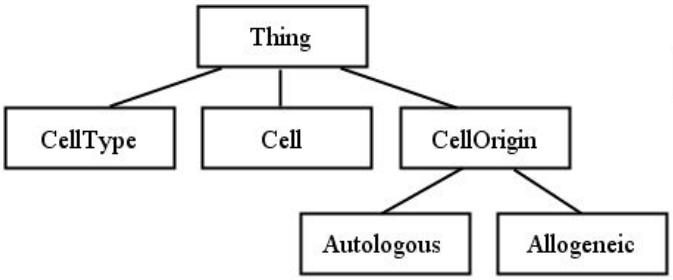

Fig. 2. Sample of Class Structure.

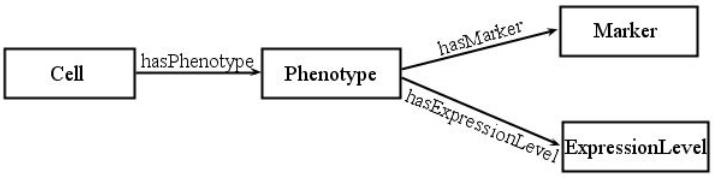

Fig. 3. N-ary Relation in tolAPC ontology.

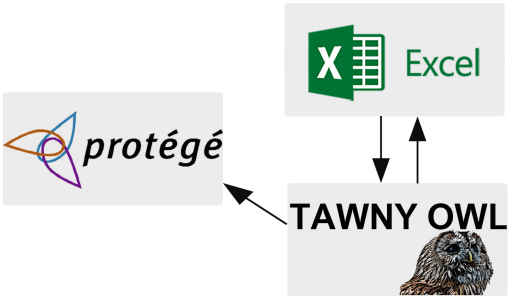

Fig. 4. Workflow using Excel spreadsheet and Tawny OWL Patterns.

```
(individual cell-name
   :fact (is fromGroup group)
         (is hasLocation loc)
         (is fromClinicalDisease clinic-disease)
         (is fromSpecies from-species)
         (is hasStatus stat)
         (is hasType c-type)
         (is hasDescription desc)
         (is hasActivation active)
         (is hasAntigenLoad anti-load)
         (is itsOrigin cell-org)
         (is withStartMaterial start-material)
         (is hasIsolation isol))
```

Here, `cell-line`, `group`, `loc` and others are variables, therefore, this definition describes a pattern. `fromGroup`, `hasLocation` and others are specific object properties from the *schema* terms of the ontology. `individual`, `:fact:` and `is` are part of Tawny-OWL syntax. The whole definition defines a new cell type, and its association with a set of individuals.

The values of the property assertions fall into one of three main categories.

Open but Limited: Many properties support a very limited, but nonetheless open, range of values. Examples of these are `withStartMaterial` which describes the tissue or part of the tissue from which the cells are derived. These values are modelled as disjoint classes, explicitly stated in the ontology. Although, we could have used an external ontology at this point, as only a few options are actually used, we have not imported one. The same principle applies to `Species`, as the cells are extracted from only three species.

Constrained Partition: Many properties support an exact number of options. These are modelled using a Value Partition [20]. Fortunately, Tawny-OWL provides explicit support for this design pattern, which allows a relatively succinct definition. An example of this is `CellOrigin` which is defined as follows:

```
(defpartition CellOrigin
  [[Allogeneic
    :comment "Allogeneic is a cell from
              a donor blood"]
   [Autologous
    :comment "Autologous is a cell from
              a patient own blood"]])
```

Unconstrained Values: Some properties have unconstrained values such as `Location`, `Group` (i.e. the people responsible for the cell type) or `AntigenLoad`. These are currently modelled as individuals, created on-demand.

In addition to these three main categories, we are adding phenotype descriptors to the cell types, in terms of raised or lowered expression levels. For these, we are modelling the expression levels as a value partition, while the overall phenotype is modelled using the N-ary relationship pattern [18], as shown in Figure 3.

5 DISCUSSION

In this paper, we have described the development of the tolAPC ontology, describing data about immunological cell types. This ontology is unusual in that it is derived directly from another data resource, the tolAPC catalogue which is maintained as an Excel spreadsheet. Essentially, the ontology provides context and semantics to data which is available in another form.

The value of recasting a spreadsheet into a form with strong machine interpretable semantics is obvious, but there are less apparent virtues arising from the process. In the initial validation step, for example, we have had to clarify parts of the tolAPC catalogue which are otherwise unclear. For example, one cell-line is described as “Autologous/Allogeneic”. The original author intent here is unclear: this could be intended to mean either autologous or allogeneic (possible), both (probably inconsistent) or just the absence of knowledge. Similarly the process of ontologisation forces us to clarify some areas of the biology; including questions about whether cell types produced by the same protocol at different times are “the same” or otherwise, which touches on issues of reproducibility. Where these issues have arisen, either the ontology schema, patterns or the spreadsheet can be modified accordingly. As shown in Figure 4, information flows in both directions between the speadsheet and ontology.

The development of the tolAPC ontology is a work in progress. As can be seen from Table 1, while parts of the tolAPC catalogue have been recast, there are significantly more spreadsheet cells which need to be converted. However, while including machine interpretable semantics is useful in its own right, we have not yet addressed the issue of interoperability with other ontologies. For example, concepts such as "species" and "clinical disease" have clearly be described in (many) other ontologies. Currently, these terms are not reused inside the tolAPC ontology. At the current time, we have not prioritized this process because confidentiality restrictions on the tolAPC catalogue limit our ability to share the results anyway. We do not expect the addition of this interoperability to be complex though, as we can reuse the "scaffolding" process described previously [27].

tolAPC Catalogue	Number of Sheets	9
	Number of Cells	1181
	Number of Cell types	15
tolAPC Ontology	Number of Classes	21
	Number of Individuals	101
	Number of relations	13

Table 1. Current Statistics of Excel sheet and tolAPC ontolgy.

This work is a further demonstration of the value of programmatic and pattern-driven ontology development using the Tawny-OWL library; it builds on earlier work on a karyotype ontology where patterns are instantiated using in-code literal data structures, the mitochondrial ontology which is *scaffolded* using a variety of different input formats, or our reworking of SIO which patternises a pre-existing ontology [26]. Patternization allows the development of an ontology to be performed rapidly and repeatedly.

The fully programmatic environment also demonstrates its value, as we have been able to add a new input format, even a very complex format such as an Excel spreadsheet with relative ease, building on tools provided by other people. This validates our earlier experiences with Tawny-OWL, where we have easily adapted a complete software development environment to the task of ontology development.

The use of Excel spreadsheets to drive ontology patterns is not new of course; it is directly supported with Protégé plugins as well as with tools such as RightField and Populous. The key addition of our methodology is to incorporate the spreadsheet as a part of the ontology source code. The spreadsheet can be updated, changed and consulted with by the domain specialists who created it, and still remain part of the ontology development process. The importance of the right format should not be under-estimated; for example, early versions of the Gene Ontology were developed in their own bespoke syntax (later to evolve into OBO Format), something which persisted for a considerable time after the development and release of OWL. The reasons for this were simple: OBO Format behaved well in a version control system, and can be easily created, edited and manipulated in a text editor, something not true of RDF serialisations of OWL. We wish to build on these lessons: ontologists should seek to interact and build on the tools that domain specialists already use, if they hope to describe the knowledge that these specialists have.

The tolAPC ontology and the document-centric approach it embodies is a first step toward establishing a richer methodology, where we interact with domain specialists using their own tool chain to capture knowledge. While spreadsheets are a first useful step, we plan to next focus on word processors and, more generally, the narrative document. It is the most common format for representing and transferring knowledge – indeed, it is what you are reading now. Currently, there has been significant interest in recasting scientific papers into semantic documents [16, 11], so that the semantics can add value to the papers. Conversely, we wish to incorporate these as part of our ontology development methodology, so that the papers can add value to our semantics.

ACKNOWLEDGEMENT

We thank Dr Paloma Riquelme (University Hospital Regensburg) for help in generating the tolAPC catalogue and participants of the AFACTT network for providing data for the tolAPC catalogue.

Funding: COST is part of the EU Framework Programme Horizon 2020 (action to focus and accelerate cell-based tolerance-inducing therapies, BM1305, http://www.afactt.eu).

Also, thanks to King Abdulaziz University, Jeddah, Saudia Arabia for granting a scholarship and supporting the study.

REFERENCES

[1]WHO International Classification of Diseases (ICD), 2016 (accessed June 20, 2016). `http://www.who.int/classifications/icd/en/`.

[2]M. Bada, R. Stevens, C. Goble, Y. Gil, M. Ashburner, J. Blake, M. Cherry, M. Harris, and S. Lewis. A Short Study on the Success of Gene Ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2):235–240, 2004.

[3]M. Bada, R. Stevens, C. Goble, Y. Gil, M. Ashburner, J. A. Blake, J. M. Cherry, M. Harris, and S. Lewis. The Gene Ontology: What It Is (and What It Isn't) - go-web-semantics.pdf, 2015.

[4]P. Bernus and M. J. Shaw. *International Handbooks on Information Systems*. 2007.

[5]M. Egaña, R. Stevens, and E. Antezana. Transforming the Axiomisation of Ontologies: The Ontology Pre-Processor Language. *Proceedigns of OWLED*, 2009.

[6]A. Garcia, K. O'Neill, L. J. Garcia, P. Lord, R. Stevens, O. Corcho, and F. Gibson. Developing ontologies within decentralised settings. In H. Chen, Y. Wang, and K.-H. Cheung, editors, *Semantic e-Science*, pages 99–140. Springer, 2010.

[7]C. Hilkens. Cell therapies: From the laboratory to the clinic. `https://www.youtube.com/watch?v=Y1ESfJBxvqY`, 2016.

[8]M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal*, 2, 2011.

[9]M. Horridge and P. Patel-Schneider. Owl 2 web ontology language manchester syntax. `http://www.w3.org/TR/owl2-manchester-syntax/`, 2012.

[10]IHTSDO. International health terminology standards development organisation, 2016 (accessed June 17, 2016). `http://www.ihtsdo.org/snomed-ct/`.

[11]A. D. Iorio, A. Gonzalez-Beltran, F. Osbourne, S. Peroni, F. Poggi, and F. Vitali. It rocs! – the rash online conversion

service. https://rawgit.com/essepuntato/rash/master/papers/rash-poster-www2016.html.

[12]S. Jupp, M. Horridge, L. Iannone, J. Klein, S. Owen, J. Schanstra, K. Wolstencroft, and R. Stevens. Populous: a tool for building OWL ontologies from templates. *Semantic Web Applications and Tools for Life Sciences*, 13(Suplement1):55, 2010.

[13]S. Jupp, M. Horridge, L. Iannone, J. Klein, S. Owen, J. Schanstra, K. Wolstencroft, and R. Stevens. Populous: a tool for building owl ontologies from templates. *BMC Bioinformatics*, 13(Suppl 1):S5, 2011.

[14]P. Lord. The Semantic Web takes Wing: Programming Ontologies with Tawny-OWL. *OWLED 2013*, Mar. 2013.

[15]P. Lord. Manchester syntax is a bit backward. http://www.russet.org.uk/blog/2985, 2014.

[16]P. Lord, S. Cockell, and R. Stevens. Three steps to heaven: Semantic publishing in a real world workflow. *Future Internet*, 4(4):1004–1015, 2012.

[17]P. Lord and R. Stevens. Adding a little reality to building ontologies for biology. *PLoS One*, 2010.

[18]N. Noy and A. Rector. Defining n-ary relations on the semantic web. https://www.w3.org/TR/swbp-n-aryRelations/, 2006.

[19]A. Rector. Normalisation of ontology implementations: Towards modularity, re-use, and maintainability. Proceedings Workshop on Ontologies for Multiagent Systems (OMAS) in conjunction with European Knowledge Acquisition Workshop, 2002. Siguenza, Spain.

[20]A. Rector. Representing specified values in owl: "value partitions" and "value sets". W3C Working Group Note, 2005.

[21]S. Schulz, H. Stenzhorn, and M. Boeker. The ontology of biological taxa. *Bioinformatics*, 24(13):i313–i321, Jul 2008.

[22]R. Stevens, C. a. Goble, and S. Bechhofer. Ontology-based knowledge representation for bioinformatics. *Briefings in bioinformatics*, 1(4):398–414, 2000.

[23]R. Stevens and U. Sattler. An object lesson in choosing between a class and an object. http://ontogenesis.knowledgeblog.org/1418, 2013.

[24]A. Ten Brinke, C. M. U. Hilkens, N. Cools, E. K. Geissler, J. A. Hutchinson, G. Lombardi, P. Lord, B. Sawitzki, P. Trzonkowski, S. M. Van Ham, and et al. Clinical use of tolerogenic dendritic cells-harmonization approach in european collaborative effort. *Mediators of Inflammation*, 2015:18, 2015.

[25]T. Tudorache, C. I. Nyulas, N. F. Noy, T. Redmond, and M. Musen. icat: A collaborative authoring tool for icd-11. http://ceur-ws.org/Vol-809/paper-09.pdf, 2011.

[26]J. Warrender. *The Consistent Representation of Scientific Knowledge: Investigations into the Ontology of Karyotypes and Mitochondria.* PhD thesis, School of Computing Science, Newcastle University, 2015.

[27]J. Warrender and P. Lord. Scaffolding the mitochondrial disease ontology from extant knowledge sources, 2015.

[28]J. D. Warrender and P. Lord. The Karyotype Ontology: a computational representation for human cytogenetic patterns. *Bio-Ontologies 2013*, 2013.

[29]J. D. Warrender and P. Lord. How, What and Why to test an ontology, 2015.

[30]K. Wolstencroft, S. Owen, M. Horridge, O. Krebs, W. Mueller, J. L. Snoep, F. du Preez, and C. Goble. RightField: embedding ontology annotation in spreadsheets. *Bioinformatics (Oxford, England)*, 27(14):2021–2, jul 2011.

[31]A. Wood, J. Flowerdew, and M. Peacock. International scientific english: The language of research scientists around the world. *Research Perspectives on English for Academic Purposes*, pages 71–83, 2001.