Lawrence Berkeley National Laboratory

LBL Publications

Title

Dead simple OWL design patterns

Permalink

https://escholarship.org/uc/item/7dv1h5w4

Journal

Journal of Biomedical Semantics, 8(1)

ISSN

2041-1480

Authors

Osumi-Sutherland, David

Courtot, Melanie

Balhoff, James P

et al.

Publication Date

2017-12-01

DOI

10.1186/s13326-017-0126-0

Peer reviewed

eScholarship.org Powered by the California Digital Library University of California

Osumi-Sutherland *et al. Journal of Biomedical Semantics* (2017) 8:18
DOI 10.1186/s13326-017-0126-0

Journal of Biomedical Semantics

SOFTWARE

Open Access

# Dead simple OWL design patterns

David Osumi-Sutherland[1*], Melanie Courtot[1], James P. Balhoff[2] and Christopher Mungall[3]

**Abstract**

**Background:** Bio-ontologies typically require multiple axes of classification to support the needs of their users. Development of such ontologies can only be made scalable and sustainable by the use of inference to automate classification via consistent patterns of axiomatization. Many bio-ontologies originating in OBO or OWL follow this approach. These patterns need to be documented in a form that requires minimal expertise to understand and edit and that can be validated and applied using any of the various programmatic approaches to working with OWL ontologies.

**Results:** Here we describe a system, Dead Simple OWL Design Patterns (DOS-DPs), which fulfills these requirements, illustrating the system with examples from the Gene Ontology.

**Conclusions:** The rapid adoption of DOS-DPs by multiple ontology development projects illustrates both the ease-of use and the pressing need for the simple design pattern system we have developed.

**Keywords:** OWL, OBO, Design pattern

## Background

Biologists classify biological entities in many different ways. A single neuron may be classified by structure (pseudo-bipolar), electrophysiology (spiking), neurotransmitter (glutamatergic), sensory modality (secondary olfactory neuron), location(s) within the brain (antennal lobe projection neuron, mushroom body extrinsic neuron), etc. A transport process occurring in a cell may be classified by the type of chemical transported, where transport starts and ends, and by what membranes are crossed. Bio-ontologies provide a widely used method for documenting such classifications and the relationships that apply between members of classes, such as partonomy. These classifications and relationships are central to the successful use of bio-ontologies in helping biologists make sense of the ever increasing volumes of data they work with. They are critical to the use of the Gene Ontology (GO) [1] and its associated annotations in interpreting genomic data via its application in enrichment analysis [2]. They are critical to the functioning of Virtual Fly Brain in grouping and querying neuroanatomical data [3].

*Correspondence: davidos@ebi.ac.uk
[1]European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, CB10 1SD Cambridge, UK
Full list of author information is available at the end of the article

To be successful in this role, bio-ontologies need to capture all of the many forms of classification that are important to biologists; but maintaining this manually becomes impractical as ontologies grow. Without formalization, the reasons for existing classifications are often opaque. The larger an ontology, the harder it is for human editors to find all valid classifications when adding a term, or to work out how to re-arrange the hierarchy when new intermediate classes are added.

The alternative to manually asserting classification is to use OWL inference to automate it. OWL equivalence axioms can be used to specify necessary and sufficient conditions for class membership. Standard reasoning software can then build a class hierarchy by finding classes that fulfill these conditions.

Many bio-ontologies now follow this approach, including the Uber Anatomy Ontology (Uberon) [4], the GO [5], the Ontology of Biomedical Investigations (OBI) [6], the Drosophila Anatomy Ontology (DAO) [7], the Cell Ontology (CL) [8] and the Ontology of Biological Attributes (Ontology of Biological Attributes (OBA) [9]. In the GO, over 52% of the classification is automated. Much of this classification leverages the structure of imported ontologies; for example, classification of transport processes in the GO relies on a classification of chemicals provided by the chemical ontology ChEBI [10]

BioMed Central

© The Author(s). 2017 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated.

and on object property axioms specified in the OBO relations ontology.

A critical requirement for ongoing development of these ontologies is the specification of design patterns to guide the consistent OWL axiomatization required for automated classification. In many of these ontologies, classes are annotated with textual descriptions that follow standard patterns which also need to be documented. Where formal, machine-readable design patterns are sufficiently detailed, they can be used to quickly generate new classes, update old ones when a pattern changes, and automatically generate user-facing documentation.

### OWL design pattern systems

There is an extensive literature on ontology design patterns in OWL [11, 12]. Much of this is based on an approach known as Content Ontology Design Patterns (CODPs; see [12]) for an overview). CODPs are small, autonomous ontologies that specify multiple classes and properties. CODPs are typically re-used by one of two methods. Either the pattern is imported and new subclasses and sub-properties of pattern entities are instantiated in the target ontology, or it is used as a template, with entities in the pattern being given new identifiers in the namespace of the target ontology.

The GO and several other ontologies including CL and OBA already use standard patterns to generate new class terms via the TermGenie tool [13]. In GO, around 80% of new class terms are added via this route. This tool allows new terms to be added by specifying a desgin pattern and a set of fillers for variable slots. Unlike CODPs, these design patterns are not autonomous: they import classes and object properties from various ontologies. This means that their semantics are dependent on those of the ontologies they import from. This is by design: the patterns are intended to leverage classification and axiomatization from external ontologies to drive classification in the target ontology.

Design patterns in TermGenie are specified directly in Javascript. This specification is opaque to most human editors and is not easily reusable outside the context of TermGenie. The other major mechanisms for specifying design patterns for programmatic use are the languages Tawny OWL [14] and Ontology PreProcessing Language (OPPL) [15]. These are very powerful tools for generating and manipulating ontologies, but are not easy for ontology editors without strong technical backgrounds to write. They are also tied to specific languages and implementations, limiting their use.

Many editors of bio-ontologies are biologists with limited computational expertise beyond a basic understanding of some subset of OWL (typically limited to the subset of OWL that can be encoded in OBO 1.4 [16]), which they interact with via Manchester Syntax rendering and graphs in graphical editing tools such such as Protégé [17]. A simple, lightweight standard for specifying design patterns is needed in order to make their development and use accessible to these editors. This standard should be readable and editable by anyone with a basic knowledge of OWL. It must also be easy to use programmatically without the need for custom parsers – i.e. it should follow some existing data exchange standard that can be consumed by any modern programming language. Based on these requirements, we have defined a lightweight, YAML Ain't Markup Language (YAML)-based syntax for specifying design patterns, called Dead Simple OWL Design Patterns, or DOS-DPs (inversion of two letters is an homage to the Web Ontology Language, OWL, on which it is based).

## Implementation

We have developed a formal specification of DOS-DPs using JSON-schema [18] draft 4 for use in validation and documentation. This is available from the DOS-DP repository [19], which also lists recommendations for additional validation steps. Description fields in the schema document intended usage. Where appropriate, the schema document also includes fields that document mappings to relevant OWL entities. We use the Python jsonschema package to validate the schema and test it against example patterns. Table 1 contains a summary of schema field types and how they are used.

### Approach

DOS-DPs are designed to be easy to read, edit and parse. We chose YAML because it is relatively easy to read and write compared to other common data exchange formats such as JSON and XML, and can be consumed by a wide range of programming languages. In order to take advantage of JSON-Schema for specification and validation, DOS-DPs are restricted to the JSON compatible subset of YAML [20].

Each design pattern can have an arbitrary number of variables. For ease of reading, writing and parsing, variable interpolation uses `printf`, a standard part of most modern programming languages.

OWL is expressed using Manchester Syntax [21], the most human-readable and editable of the OWL syntaxes, and the one most editors with a basic knowledge of OWL are likely to have encountered. For ease of reading and editing, quoted, human-readable identifiers are used for OWL entities throughout the pattern. These are assumed to be sufficient to uniquely identify any OWL entity within a pattern. Dictionaries are used to map readable identifiers to compact URIs (CURIEs) – prefixed short form identifiers. A JSON-LD context is used to map these to full IRIs. The entity IRIs recorded in this way can be used to check reference ontologies to find the

**Table 1** DOSDP JSON schema fields

| Field type | Used to | Mandatory subfields | Optional subfields | Used in |
|---|---|---|---|---|
| Printf_owl | Specify a logical OWL axiom using printf to substitute variable values | axiom_type, text, vars | annotations | logical axioms |
| Printf_annotation | Specify an annotation using printf to substitute variable values | annotationProperty, text, vars | annotations | annotations |
| List annotation | Specify a list of annotation property axioms of a single type using a list of values specified by a data list variable | annotationProperty, value | - | annotations |
| Printf_owl_convenience | Specify a logical OWL axiom of a prespecified type, using printf to substitute variable values. | text, vars | annotations | equivalentTo, subClassOf, disjointWith, GCI |
| Printf annotation obo | Specify an annotation axiom of a prespecified type using a list of values specified by a data list variable | text, vars | annotations, xrefs | def, name, comment, |
| List_annotation_obo | Specify a list of annotation property axioms of a single type, pre-specified type. using a list of values specified by a data list variable | value | - | xrefs, exact_synonyms, … |

*Field type*: Name of schema field type (JSON schema definition). *Used to*: Description of field usage. *Used in*: Schema Fields in which this field type is used

current validity and status of all entities referenced in a pattern.

While the full specification of DOS-DPs is intended to be generic and expressive, a major aim is to hide complexity from editors wherever possible. To this end, we define convenience fields that are suitable for use in common, simple design patterns. We also allow extensions that import and extend the core JSON schema and that specify default values for high level fields. For example, we define an extension to support the OBO standard. This defines convenience fields for expressing OBO standard annotations and specifies a default annotation property for readable identifiers and an OBO standard base URI pattern.

Figure 1 shows an example design pattern for generating classes of transport across a membrane defined by cargo type and membrane type. Figure 1a shows a pattern following the OBO extension. Figure 1b shows the same pattern expressed using the more verbose DOSDP core-specification. Figure 2 shows an example class generated using this pattern.

## Details

### *Pattern metadata*

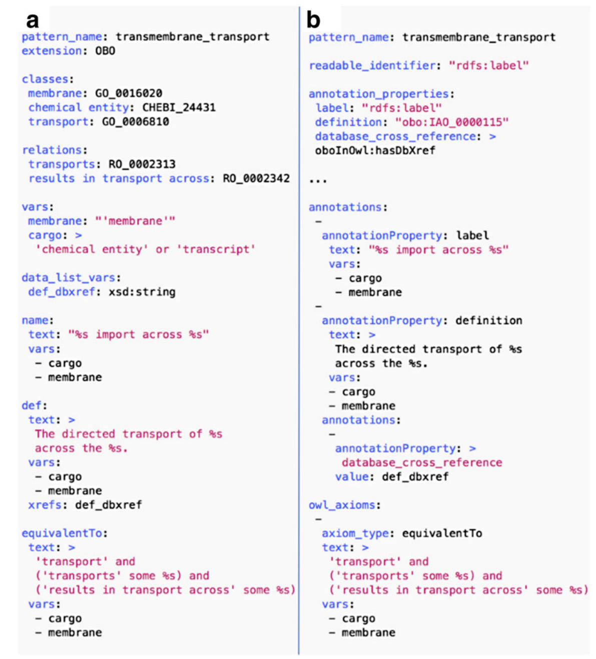
Each pattern is identified by an IRI. The short form of this IRI is recorded in a **pattern_name** field, and, by convention, is used for the file name. Each pattern optionally includes an extension specification, indicating the extension to be used in interpreting the pattern document. In 1a this is set to OBO.

### *Dictionaries*

In both versions of the pattern, the fields **classes** and **relations** serve as dictionaries for the OWL classes and object properties respectively used in the pattern, mapping human readable identifiers (keys) to short_form identifiers (values). The core pattern specifies an annotation property to use as a source of readable identifiers via the **readable_identifier** field. This is not required in the OBO extension version, as the extension specifies a default value of `rdfs:label` for this. The full pattern also contains an additional dictionary of OWL annotation properties. These are not required in the OBO extension, which specifies dedicated fields for annotation properties used in the OBO standard. The core DOSDP specification also defines a dictionary field for OWL data properties.

### *Input fields*

All patterns contain one or more variable specification fields. These are simple objects in which the keys are variable names and the values specify variable range. The **vars** field specifies variables that range over OWL classes, specified as Manchester syntax expressions. For example, the value of the cargo variable in Fig. 1 is specified by

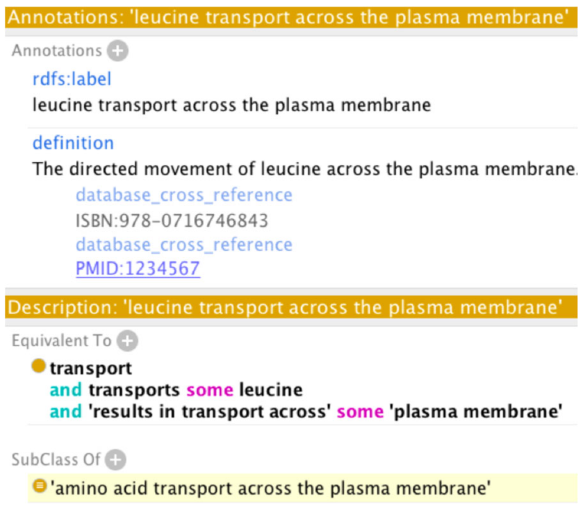

a

```
pattern_name: transmembrane_transport
extension: OBO

classes:
 membrane: GO_0016020
 chemical entity: CHEBI_24431
 transport: GO_0006810

relations:
 transports: RO_0002313
 results in transport across: RO_0002342

vars:
 membrane: "'membrane'"
 cargo: >
  'chemical entity' or 'transcript'

data_list_vars:
 def_dbxref: xsd:string

name:
 text: "%s import across %s"
 vars:
  - cargo
  - membrane

def:
 text: >
  The directed transport of %s
  across the %s.
 vars:
  - cargo
  - membrane
 xrefs: def_dbxref

equivalentTo:
 text: >
  'transport' and
  ('transports' some %s) and
  ('results in transport across' some %s)
 vars:
  - cargo
  - membrane
```

b

```
pattern_name: transmembrane_transport

readable_identifier: "rdfs:label"

annotation_properties:
 label: "rdfs:label"
 definition: "obo:IAO_0000115"
 database_cross_reference: >
 oboInOwl:hasDbXref

...

annotations:
 -
  annotationProperty: label
   text: "%s import across %s"
   vars:
    - cargo
    - membrane
 -
  annotationProperty: definition
   text: >
    The directed transport of %s
    across the %s.
   vars:
   - cargo
   - membrane
  annotations:
   -
    annotationProperty: >
     database_cross_reference
    value: def_dbxref

owl_axioms:
 -
  axiom_type: equivalentTo
  text: >
   'transport' and
   ('transports' some %s) and
   ('results in transport across' some %s)
  vars:
   - cargo
   - membrane
```

**Fig. 1** DOS-DP for defining classes of transmembrane import (based on an example from the GO.) *Panel A* shows the DOS-DP using the OBO extension. *Panel B* shows the same pattern expressed using the core specification (*classes*, *relations* and *vars* fields omitted from *panel B* for brevity). In *Panel A*, annotations are specified using dedicated fields (**def**, **name**, **xrefs**). The mapping from these to OWL annotation properties is specified in the OBO extension schema. This mapping is made explicit in *Panel B*, using an **annotation_property** dictionary and the **annotationProperty** field in axiom specifications under **annotations**. Throughout both versions of the pattern, paired fields **text** and **vars** specify `printf` text and fillers respectively. The value field is used with the **data_list_var** def_xrefs to specify a list database_cross_reference annotations on the definition


Annotations: 'leucine transport across the plasma membrane'

Annotations

rdfs:label
leucine transport across the plasma membrane

definition
The directed movement of leucine across the plasma membrane.
database_cross_reference
ISBN:978-0716746843
database_cross_reference
PMID:1234567

Description: 'leucine transport across the plasma membrane'

Equivalent To
transport
and transports some leucine
and 'results in transport across' some 'plasma membrane'

SubClass Of
'amino acid transport across the plasma membrane'

**Fig. 2** Example pattern implementation. An example of a term, 'leucine transport across the plasma membrane', generated using the pattern in Fig. 1. Note the automated classification under 'amino acid transport across the plasma membrane', specified using the same pattern

the class expression: "'chemical entity' or 'transcript'". The quoted OWL entity names in this expression are specified in the dictionaries. Both patterns also include an example of a variable that takes a data type as an input. The **data_list_vars** field specifies variables whose values are lists in which all elements share an OWL data type, specified in the value of the variable field. For example def_dbxref in Fig. 1 is specified to be a list of (XSD) strings.

### *Output fields*

The core schema has just two output fields: **annotations** for annotation property axioms and **logical_axioms** for logical owl axioms. The value of both of these fields is a list of axiom specifications. Each axiom specification includes a specification of axiom type (logical type or annotation property). Content is either specified using printf substitution of variable values into a text string (field type *printf_annotation* or *printf_owl* in Table 1 or by specifying a list of values to be used to generate multiple axioms of the same type (e.g. field type list_annotation in Table 1. Where OWL entities (specified as vars) are used to specify `Printf` substitution, the readable label of the entity is used. Axiom specifications can also be used to specify annotations of the specified axiom.

In our example, the annotations field is used to specify an `rdfs:label` axiom and a definition axiom. In both cases a text output is specified using a **text** field to specify a `printf` statement and a **vars** field to specify an ordered list of fillers. The definition axiom specification specifies a set of axiom annotations using a database_cross_reference annotation property. These axioms will be generated using a list of strings provided in the **data_list_var def_dbxref**. The results can be seen in Fig. 2.

The OBO version (1) encodes the same information using named fields: **name**, **def**, and **xrefs**. These fields follow the tag names used in OBO format [16]. The field specifications (in the OBO JSON schema doc) map these fields to the relevant OWL annotation properties, removing the need for ontology pattern developers to specify these mappings in an annotation property dictionary.

The **logical_axioms** field in Fig. 1b specifies just one equivalence axiom. This is a very common pattern for defining classes. To make specifying this type of pattern easier, we define convenience fields that can be used whenever there is only one axiom of a given type per pattern. The pattern in 1a uses the convenience field for **equivalentTo** to concisely capture the single logical axiom in this pattern.

## Discussion

### Limitations

DOS-DPs are designed to be simple and clear. There are a number of obvious ways that they could be made more powerful but which we have avoided in order to retain simplicity and clarity.

By design, DOS-DPs lack a mechanism for relating patterns to each other via inheritance or composition. Such mechanisms would add a technical burden to their, use requiring additional tooling, and so be a barrier to their adoption. Manual maintenance of design pattern hierarchies also risks re-creating the maintenance problem that these patterns are meant to solve.

For the sake of simplicity, DOS-DPs also lack a system for specifying optional clauses. This places some burden on the development of patterns that naturally form a subsumption hierarchy. However, the relationships between patterns can easily be derived by generating a set of OWL classes using default fillers (variable ranges) and classifying the results using a reasoner. This classification can then be used as a way of testing sets of DOS-DPs and to generate a browsable hierarchy of related patterns.

### Adoption

DOS-DPs are used both as formal documentation, and as part of the ontology-engineering pipelines in the GO, OBA, the Environmental Ontology (ENVO) [22], the Plant Trait Ontology [23], the Plant Stress and Disease Ontology [24], the Agriculture Ontology, and the Environmental Conditions and Exposures Ontology [25]; the central DOS-DP GitHub repo has a list of all adopters. See Figs. 1 and 2 for an example of a pattern used extensively in the GO.

One heavy user of (OPPL) patterns is Webulous, an application that allows specification of OWL classes using templates loaded into Google spreadsheets. It should be straightforward to develop a version of Webulous that supports design patterns specified as DOS-DPs, removing the need for expertise in OPPL to specify new patterns. Similarly, it should be possible to extend Tawny-OWL to support DOS-DPs. This could prove to be a very effective combination of accessible design pattern specification with a computationally powerful language for writing and manipulating OWL ontologies.

Patterns inevitably evolve as use-cases evolve. Changing all uses of an existing pattern by hand is impractical unless the number of uses is relatively low. For branches of ontologies where all terms follow a completely stereotyped pattern, we can specify whole branches simply by specifying a DOS-DP together with a URI and set of variable fillers for each term. We plan to use this to programmatically generate suitable branches of the GO at each release.

Where more flexibility is required, DOS-DPs could be used to update existing terms that are part of a human-edited ontology file. A system of tagging terms by the pattern they implement would allow all relevant terms to be identified. DOSDP-scala [26] can be used to identify existing classes within an ontology that follow a specified pattern, returning the fillers populating each variable in the pattern. If an ontology pattern changes then DOSDP-scala can also be used to test whether tagged terms conform to the old pattern, flagging those that do for automated update and those that do not for manual inspection.

## Conclusions

As can be seen from Fig. 1, which shows a pattern for defining terms in the GO, DOS-DPs are easy to read and write. The choice of YAML limits the need for balancing brackets and commas. The use of `printf`, Manchester syntax, and labels for OWL entities makes the pattern easy to read. Figure 2, which shows an application of the pattern specified in Fig. 1, illustrates how similar the pattern is to the way human editors interact with ontology classes in a GUI editor like Protégé [17]. As well as ease of reading and writing, our other aim is language independence. Currently there are partial (OBO-specific) implementations in Python [27] and Jython [28, 29], along with the Scala-based pattern matcher [26]. TermGenie is being extended to consume DOS-DPs. These implementations cover pattern validation and the addition of new classes. They also allow for generation of markdown format documentation from design patterns.

## Availability and requirements

**Project name:** Dead Simple OWL Design Patterns (DOS-DP). The specification and recommendations for validation are available from [29] under the GNU General Public License v3.0.

**Programming language and requirements:** The schema is specified using JSON-schema [18]. This specification can be consumed by any language for which a schema checker exists (see [18]).

#### Abbreviations

ChEBI: Chemical entities of biological interest; CL: Cell ontology; CODP: content ontology design pattern; CURIE: Compact URI; DOS-DP: Dead simple OWL design pattern; GO: Gene ontology; GUI: Graphical user interface; IRI: Internationalized resource identifier; JSON: JavaScript object notation; OBA: Ontology of biological attributes; OBO: Open biomedical ontologies; OPPL: Ontology preprocessing language; OWL: Web ontology language; XML: Extensible markup language; XSD: XML schema description; YAML: YAML ain't markup language

#### Acknowledgements

We sincerely thank Helen Parkinson, the Gene Ontology Consortium and the Sample, Phenotypes and Ontologies Group at the EBI for providing a fertile intellectual environment for the development of DOS-DPs.

**Funding**
This work was funded by the Gene Ontology Consortium P41 grant from the National Human Genome Research Institute (NHGRI) [grant 5U41HG002273- 14] and by BD2K [grant U01 HG009453]. CJM acknowledges the support of the Director, Office of Science, Office of Basic Energy Sciences, of the US Department of Energy (DE-AC02- 05CH11231). MC was funded by the Gene Ontology Consortium P41 grant from the National Human Genome Research Institute (NHGRI) [grant 5U41HG002273- 14] and EMBL-EBI core funds.

**Availability of data and materials**
The DOS-DP specification and recommendations for validation are available from [29].

**Authors' contributions**
DOS-DPs were developed by DOS in with suggestions from CJM and JPB. Many of the uses of DOS-DPs described here were implemented by CJM. MC supervised some of this work and contributed to drafting this paper. JPB developed DOSDP-scala. All authors read and approved the final manuscript.

**Competing interests**
The authors declare that they have no competing interests.

**Consent for publication**
Not applicable.

**Ethics approval and consent to participate**
Not applicable.

## Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Author details**
[1]European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, CB10 1SD Cambridge, UK. [2]RTI International, 27709 Research Triangle Park, NC, USA. [3]Genomics Division, Lawrence Berkeley National Laboratory, 94720 Berkeley, CA, USA.

Received: 23 November 2016 Accepted: 29 March 2017
Published online: 05 June 2017

**References**
1. Blake JA, Christie KR, Dolan ME, Drabkin HJ, Hill DP, Ni L, Sitnikov D, Burgess S, Buza T, Gresham C, McCarthy F, Pillai L, Wang H, Carbon S, Dietze H, Lewis SE, Mungall CJ, Munoz-Torres MC, Feuermann M, Gaudet P, Basu S, Chisholm RL, Dodson RJ, Fey P, Mi H, Thomas PD, Muruganujan A, Poudel S, Hu JC, Aleksander SA, McIntosh BK, Renfro DP, Siegele DA, Attrill H, Brown NH, Tweedie S, Lomax J, Osumi-Sutherland D, Parkinson H, Roncaglia P, Lovering RC, Talmud PJ, Humphries SE, Denny P, Campbell NH, Foulger RE, Chibucos MC, Gwinn Giglio M, Chang HY, Finn R, Fraser M, Mitchell A, Nuka G, Pesseat S, Sangrador A, Scheremetjew M, Young SY, Stephan R, Harris MA, Oliver SG, Rutherford K, Wood V, Bahler J, Lock A, Kersey PJ, McDowall MD, Staines DM, Dwinell M, Shimoyama M, Laulederkind S, Hayman GT, Wang SJ, Petri V, D'Eustachio P, Matthews L, Balakrishnan R, Binkley G, Cherry JM, Costanzo MC, Demeter J, Dwight SS, Engel SR, Hitz BC, Inglis DO, Lloyd P, Miyasato SR, Paskov K, Roe G, Simison M, Nash RS, Skrzypek MS, Weng S, Wong ED, Berardini TZ, Li D, Huala E, Argasinska J, Arighi C, Auchincloss A, Axelsen K, Argoud-Puy G, Bateman A, Bely B, Blatter MC, Bonilla C, Bougueleret L, Boutet E, Breuza L, Bridge A, Britto R, Casals C, Cibrian-Uhalte E, Coudert E, Cusin I, Duek-Roggli P, Estreicher A, Famiglietti L, Gane P, Garmiri P, Gos A, Gruaz-Gumowski N, Hatton-Ellis E, Hinz U, Hulo C, Huntley R, Jungo F, Keller G, Laiho K, Lemercier P, Lieberherr D, MacDougall A, Magrane M, Martin M, Masson P, Mutowo P, O'Donovan C, Pedruzzi I, Pichler K, Poggioli D, Poux S, Rivoire C, Roechert B, Sawford T, Schneider M, Shypitsyna A, Stutz A, Sundaram S, Tognolli M, Wu C, Xenarios I, Chan J, Kishore R, Sternberg PW, Van Auken K, Muller HM, Done J, Li Y, Howe D, Westerfield M. Gene Ontology Consortium: going forward. Nucleic Acids Res. 2015;43(Database issue):1049–56.
2. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci. 2005;102(43): 15545–50. doi:10.1073/pnas.0506580102, http://www.pnas.org/content/102/43/15545.full.pdf.
3. Milyaev N, Osumi-Sutherland D, Reeve S, Burton N, Baldock RA, Armstrong JD. The Virtual Fly Brain browser and query interface. Bioinformatics. 2012;28(3):411–5.
4. Haendel MA, Balhoff JP, Bastian FB, Blackburn DC, Blake JA, Bradford Y, Comte A, Dahdul WM, Dececchi TA, Druzinsky RE, Hayamizu TF, Ibrahim N, Lewis SE, Mabee PM, Niknejad A, Robinson-Rechavi M, Sereno PC, Mungall CJ. Unification of multi-species vertebrate anatomy ontologies for comparative biology in Uberon. J Biomed Semantics. 2014;5:21.
5. Mungall C, Deitze H, Osumi-Sutherland D. Use of OWL within the Gene Ontology In: Maria Keet C, Tamma V, editors. Proceedings of the 11th International Workshop on OWL: Experiences and Directions - OWLED. CEUR Workshop Proceedings. Volume 1265. Aachen; 2014. p. 25–36. ceur-ws.org/Vol-1265.
6. Brinkman RR, Courtot M, Derom D, Fostel JM, He Y, Lord P, Malone J, Parkinson H, Peters B, Rocca-Serra P, Ruttenberg A, Sansone SA, Soldatova LN, Stoeckert CJ, Turner JA, Zheng J. Modeling biomedical experimental processes with OBI. J Biomed Semantics. 2010;1 Suppl 1:7.
7. Costa M, Reeve S, Grumbling G, Osumi-Sutherland D. The Drosophila anatomy ontology. J Biomed Semantics. 2013;4(1):32.
8. Meehan TF, Masci AM, Abdulla A, Cowell LG, Blake JA, Mungall CJ, Diehl AD. Logical development of the cell ontology. BMC Bioinforma. 2011;12:6.
9. Ontology of Biological Attributes. 2016. Available at https://github.com/obophenotype/bio-attribute-ontology.
10. Hastings J, de Matos P, Dekker A, Ennis M, Harsha B, Kale N, Muthukrishnan V, Owen G, Turner S, Williams M, Steinbeck C. The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Res. 2013;41(Database issue): 456–63.
11. Gangemi A. Ontology Design Patterns for Semantic Web Content In: Gil Y, Motta E, Benjamins VR, Musen MA, editors. The, Semantic Web – ISWC Lecture Notes in Computer Science 2005, Volume 3729. Berlin: Springer; 2005.
12. Hammar K, Presutti V. Template-based content odp instantiation. In: Proceedings of the Workshop on Ontology and Semantic Web Patterns (7th Edition) - WOP2016; 2016. http://ontologydesignpatterns.org/wiki/WOP:2016%23Proceedings.
13. Dietze H, Berardini TZ, Foulger RE, Hill DP, Lomax J, Osumi-Sutherland D, Roncaglia P, Mungall CJ. TermGenie - a web-application for pattern-based ontology class generation. J Biomed Semant. 2014;5:48. doi:10.1186/2041-1480-5-48.
14. Lord P. The Semantic Web takes wing: Programming ontologies with Tawny-OWL In: Rodriguez-Muro M, Jupp S, Srinivas K, editors. Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) CEUR Workshop Proceedings. Volume 1080. Aachen; 2013. ceur-ws.org/Vol-1080.
15. Egana M, Stevens R, Antezana E. Transforming the axiomisation of ontologies: The ontology pre-processor language. In: CEUR Workshop Proceedings. vol. 496; 2009. doi:10.1038/npre.2009.4006.1.
16. Antezana E, Balhoff J, Day-Richter J, Ireland A, Manzoor S, Ruttenberg A, Osumi-Sutherland D, Hamid Tirmizi S, Mungall C. OBO 1.4 - Available at http://owlcollab.github.io/oboformat/doc/GO.format.obo-1_4.html.
17. Musen MA. The protégé project: A look back and a look forward. AI Matters. 2015;1(4):4–12. doi:10.1145/2757001.2757003.
18. JSON Schema -http://json-schema.org/.
19. Osumi-Sutherland DJ. Dead Simple OWL Design Patterns - Specification. 2016. Available at https://github.com/dosumis/dead_simple_owl_design_patterns.
20. YAML - Relation to JSON - http://www.yaml.org/spec/1.2/spec.html#id2759572.
21. In: Hitzler P, Krötzsch M, Parsia B, Patel-Schneider PF, Rudolph S, editors. OWL2 Web Ontology Language: Primer: W3C Recommendation; 2009. Available at http://www.w3.org/TR/owl2-primer/.
22. Buttigieg P, Morrison N, Smith B, Mungall CJ, Lewis SE. The environment ontology: contextualising biological and biomedical entities. J Biomed Semant. 2013;4(1):43. doi:10.1186/2041-1480-4-43.

23. Plant Trait Ontology. 2016. Available at https://github.com/Planteome/plant-trait-ontology.
24. Plant Stress Ontology. 2016. Available at https://github.com/Planteome/plant-stress-ontology.
25. Environmental Conditions Treatments and Exposures Ontology. 2016. Available at https://github.com/cmungall/environmental-conditions.
26. Balhoff JP. dosdp-scala . 2016. Available at https://github.com/balhoff/dosdp-scala.
27. Mungall CJ. Pattern2OWL - https://github.com/cmungall/pattern2owl.
28. Jython maintainers: The Jython project. 2016. Available at http://www.jython.org.
29. Osumi-Sutherland DJ. Dead Simple OWL Design Patterns. 2016. Available at https://github.com/dosumis/dead_simple_owl_design_patterns.

Submit your next manuscript to BioMed Central and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at www.biomedcentral.com/submit

BioMed Central